Fundamental of Computer

(MCA/PGDCA - 101)



Jaipur National University
Directorate of Distance Education

Established by Government of Rajasthan
Approved by UGC under Sec 2(f) of UGC ACT 1956
(**Recognised by Joint Committee of UGC-AICTE-DEC, Govt. of India**)

1

Introduction to Computer

STRUCTURE

1.0 Objectives

1.1 Introduction

1.2 Strengths of Computers

1.3 Limitations of Computers

1.4 Fundamental Uses of Computers

1.5 Development of Computers

1.6 Types of Computers

1.7 Generations of Computers

1.8 Summary

1.9 Keywords

1.10 Review Questions

1.11 Further Studies

1.0 Objectives

After studying this chapter, you will be able to:

- Discuss the strengths of computers
- Explain the limitations of computers
- Discuss the fundamental uses of computers
- Explain the developments of computers
- Define the generations of computers

1.1 Introduction

The word "computer" comes from the word "compute", which means, "to calculate". Hence, people usually a computer to be a calculating device that can perform arithmetic operations at high speed. In fact the original objective for inventing a computer was to create a fast calculating machine. However, more than 80% of work done by computers today is of non-mathematical or non-numerical nature. Hence, to define computer merely as a calculating device is to ignore over 80% of its functions.

1.2 Strengths of Computers

Increasing popularity of computers has proved that it is a very powerful and useful tool. The power and usefulness of this popular tool are mainly due to its following characteristics:

1. Automatic. An automatic machine works by itself without human intervention. Computers are automatic machines because once started on a job, they carry out the job (normally without am human assistance) until it is finished. However, computers being machines cannot start themselves and cannot go out and find their own problems and solutions. We need to instruct a computer using coded instructions that specify exactly how it will do a particular job. Some of the other characteristics of computers (such as speed and accuracy) are because they are automatic and work on a problem without any human intervention.

2. ***Speed.*** A computer is a very fast device. It can perform in a few seconds, the amount of work that a human being can do in an entire year, if he/she worked day and night and did nothing else. In other words, a computer can do in a few minutes what would take a man his entire lifetime.
While talking about the speed of a computer we do not talk in terms of seconds or even milliseconds (10^{-3}) but in terms of microseconds (10^{-6}), nanoseconds (10^{-9}), and even picoseconds (10^{-12}). A powerful computer is capable of performing several billion (10^{9}) simple arithmetic operations per second.

3. Accuracy. In addition to being very fast, computers are very accurate. Accuracy of a computer is consistently high and the degree of its accuracy depends upon its design. A computer performs even calculation with the same accuracy.
However, errors can occur in a computer. These errors are mainly due to human rather than technological weaknesses. For example, errors may occur due to imprecise thinking by a programmer (a person who writes instructions for a computer to solve a particular problem) or incorrect input data. We often refer to computer errors caused due to incorrect input data or unreliable programs as garbage-in-garbage-out (GIGO).

4. Diligence. Unlike human beings, a computer is free from monotony, tiredness, and lack of concentration. It can continuously work for hours without creating any error and without grumbling. Hence, computers score over human beings in doing routine type of jobs that require great accuracy. If ten million calculations have to be performed, a computer will perform the last one with exactly the same accuracy and speed as the first one.

5. Versatility. Versatility is one of the most wonderful things about a computer. One moment it is preparing results of an examination, next moment it is busy preparing electricity bills, and in between. It may be helping an office secretary to trace an important letter in seconds. All that is required to change its talent is to slip in a new program (a sequence of instructions for the computer) into it. In brief, a computer is capable of performing almost any task, if the task can be reduced to a finite series of logical steps.

6. Power of Remembering. As a human being acquires new knowledge, his/her brain subconsciously selects what it feels to be important and worth retaining in memory. The brain relegates unimportant details to back of mind or just forgets them. This is not the case with computers. A computer can store and recall any amount of information because of its secondary storage (a type of detachable memory) capability. It can retain a piece of information as long as a user desires and the user can recall the information whenever required. Even after several years, a user can recall exactly the same information that he/she had stored in the computer several years ago. A computer forgets or looses certain information only when a user asks it to do so. Hence, it is entirely up to the user to make a computer retain or forget some information.

7. No I Q. A computer is not a magical device. It possesses no intelligence of its own. Its I. Q. is zero at least until today. It has to be told what to do and in what sequence. Hence, only users determine what tasks a computer will perform. A computer cannot take its own decision in this regard.

8. No Feelings. Computers are devoid of emotions. They have no feelings and no instincts because they are machines. Although men have succeeded in building a memory for computer, but no computer

possesses the equivalent of a human heart and soul. Based on our feelings, taste, knowledge, and experience we often make certain judgements in our day-to-day life whereas, computers cannot make such judgements on their own. They make judgements based on the instructions given to them in the form of programs that are written by us (human beings).

1.3 Limitations of Computers

Computer has done this and that. Actually this is not the computer which has done this but the user. Despite its various features, a computer does have the following limitations:

No Self Intelligence

Today, a computer is able to do a work which is impossible for man. Computers are used to do risky and dangerous work and where sharp actually is needed. But it does not have any intelligence of its own. It works according to the instruction only.

No Decision-Making power

Computer cannot take any decision of its own. It does only those tasks which are already instructed to it.

No learning power

Computer has no learning power. Once you give instructions to a computer how to perform a task, the very task is cannot perform if you do not give it any instructions for the next time. For example, when you are taught how to solve a problem and if same type of problem is given you to solve, then you can do it.

Emotionless

Computers are emotionless. They do not have emotion, like dislike feelings. They are simply machines which work as per the instruction given to them.

Curtail human Capabilities

Although computers are great help to human beings, it is commonly felt that we people have become so dependent on calculator that we cannot make very simple calculation without calculator. Sometimes, find it difficult to instantly speak out even that telephone number which they use every now and then as they have got the habit of using them by retrieving the storage.

Caution

Excessive use of computers is causing various type of health injuries such as cervical and back pain, pain in eye, headache.

1.4 Fundamental Uses of Computers

With technology continuously evolving and intertwining into everyday life, people are using computers for more and more simple tasks. This is possible thanks in no small part to the continued efforts of computer manufacturers as well as software designers to simplify computing so that anyone can sit down, power on the machine and jump right in.

Communication

Thanks to computers and the Internet, the world has gotten much smaller in recent years. Many people use their computers to keep in touch with friends and family using instant messenger programs as well as email. A growing communication tool is social networking, with sites like Facebook and Twitter becoming incredibly popular.

Games

PCs have long served as recreational devices with hundreds of games available each year. Gaming on a PC can be an expensive hobby, with video cards ranging in price from INR 3,000 to more than 20,000 and fully equipped gaming PCs costing in excess of INR 75,000 in many cases. For all the top-tier AAA titles, there are other games that users can find both pre-installed on PCs as well as online.

Entertainment

Almost all computers come with CD or DVD disk drives, which allow you to use the computer as a CD player or DVD player. Some computers are also capable, with the proper hardware, of viewing and recording television onto the machine's hard drives. With an Internet connection, users have a nearly limitless of videos and music available online as well.

Work

Almost every working environment uses computers in one capacity or another. Office buildings use computers to keep track of everything from pay wages to hours logged, retail stores use computers as cash registers and industries such as construction and architecture use computers to help design buildings.

1.5 Development of Computers

Necessity is the mother of invention. The saying holds true for computers too. Computers were invented because of man's search for fast and accurate calculating devices.

Basic Pascal invented the first mechanical adding machine in 1642. Later, in the year 1671, Keyboard machines originated States around 1880 and we use them even today. Around the same period, Herman Hollerith came up with concept of punched cards that were extensively used as input medium in computers even in late 1970s. Machines and calculators made their appearance in Europe and America towards the end of the century.

Charles Babbage, a nineteenth century Professor at Cambridge University, is considered the father of modern digital computers. In order to have a better idea of the evolution of computers it is worthwhile to discuss of the well-known early computers. These are as follows:

1. **The Mark-I Computer (1937-44).** Also known as Automatic Sequence Controlled calculator, this was the first fully automatic calculating machine designed by Howard A. Aiken of Harvard University in collaboration with IBM (International Business Machines) Corporation. It was an electro-mechanical device (used both electronic and mechanical components) based on the techniques already developed for punched card machines.

2. **The Atanasoff-Berry Computer (1939-42).** Dr. John Atanasoff developed an electronic machine to solve certain mathematical equations. The machine was called the Atanasoff-Berry Computer, or ABC, after its inventor's name and his assistant, Clifford Berry. It used 45 vacuum tubes for internal logic and capacitors for storage.

3. **The ENIAC (1943-46).** The Electronic Numerical Integrator And Calculator (ENIAC) was the first all electronic computer. It was constructed at the Moore School of Engineering of the University of Pennsylvania, U.S.A. by a design team led by Professors J. Presper Eckert and John Mauchly.

The team developed ENIAC because of military needs. It was used for many years to solve ballistic related problems. It took up wall space in a 20 x 40 square feet room and used 18,000 vacuum tubes it could add two numbers in 200 microseconds and multiply them in 2000 microseconds.

Did You Know?

Baron Gottfried Wilhelm von Leibniz of Germany invented the first calculator for multiplication.

4. **The EDVAC (1946-52).** A major drawback of ENIAC was that its programs were wired on boards that made it difficult to change the programs. Dr. John Von Neumann later introduced the "stored program" concept that helped in overcoming this problem. The basic idea behind this concept is that a sequence of instructions and data can be stored in the memory of a computer for automatically directing the flow of operations. This feature considerably influenced the development of modern digital computers because of the ease with which different programs can be loaded and executed on the same computer. Due to this feature, we often refer to modern digital computers as stored program digital computers. The Electronic Discrete Variable Automatic Computer (EDVAC) used the stored' program concept in its

design. Von Neumann also has a share of the credit for introducing the idea of storing both instructions and data in binary form (a system that uses only two digits - 0 and 1 to represent all characters), instead of decimal numbers or human readable words.

5. **The EDSAC (1947-49).** Almost simultaneously with EDVAC of U.S.A., the Britishers developed the Electronic Delay Storage Automatic Calculator (EDSAC). The machine executed its first program in May 1949. In this machine, addition operations took 1500 microseconds and multiplication operation: took 4000 microseconds. A group of scientists headed by Professor Maurice Wilkes at the Cambridge University Mathematical Laboratory developed this machine.

6. The UNIVAC I (1951). The Universal Automatic Computer (UNIVAC) was the first digital computer that was not "one of a kind". Many UNIVAC machines were produced, the first of which was installed in the Census Bureau in 1951 and was used continuously for 10 years. In 1952, the International Business Machines (IBM) Corporation introduced the IBM-701 commercial computer. In rapid succession, improved models of the UNIVAC I and other 700-series machines were introduced. In 1953, IBM produced the IBM-650, and sold over 1000 of these computers.

Did You Know?
UNIVAC marked the arrival of commercially available digital computers for business and scientific applications and was developed by General Electric Corporation in 1954.

Self Assessment Questions

1. The value of picoseconds is:
(a) 10^{-6} (b) 10^{-12}
(c) 10^{-9} (d) None of these.

2. The term GIGO stands for:
(a) garbage-in-garbage-out (b) global-in-global-out
(c) go-in-go-out (d) None of these.

3. Which one is not the strength of the computer?
(a) Automatic machine (b) Fast speed
(c) No learning power (d) Accuracy

4. The average speed of first analytical engine was:
(a) 60 additions per minute (b) 40 additions per minute
(c) 70 additions per minute (d) 80 additions per minute

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: Search more about the history of computers.

……..……

……..……

……

1.6 Types of Computers

A computer is one of the most brilliant inventions of mankind. Thanks to computer technology, we were able to achieve storage and processing of huge amounts of data; we could rest our brains by employing

computer memory capacities for storage of information. Owing to computers, we have been able speed up daily work, carry out critical transactions and achieve accuracy and precision at work. Computers of the earlier times were of the size of a large room and were required to consume huge amounts of electric power. However, with the advancing technology, computers have shrunk to the size of a small watch. Depending on the processing power and size of computers, they have been classified under various types. Let us look at the classification of computers.

1.6.1 Operational Based

Based on the operational principle of computers, they are categorized as analog, digital and hybrid computers.

Analog Computers*:* These are almost extinct today. These are different from a digital computer because an analog computer can perform several mathematical operations simultaneously. It uses continuous variables for mathematical operations and utilizes mechanical or electrical energy.

Digital Computers**:** They use digital circuits and are designed to operate on two states, namely bits 0 and 1. They are analogous to states ON and OFF. Data on these computers is represented as a series of 0s and 1s. Digital computers are suitable for complex computation and have higher processing speeds. They are programmable. Digital computers are either general purpose computers or special purpose ones. General purpose computers, as their name suggests, are designed for specific types of data processing while general purpose computers are meant for general use.

Hybrid Computers: These computers are a combination of both digital and analog computers. In this type of computers, the digital segments perform process control by conversion of analog signals to digital ones.

1.6.2 Sizes and Processing Powers

This was the classification of computers based on their style of functioning. Following is a classification of the different types of computers based on their sizes and processing powers.

Mainframe Computers: Large organizations use mainframes for highly critical applications such as bulk data processing and ERP. Most of the mainframe computers have capacities to host multiple operating systems and operate as a number of virtual machines. They can substitute for several small servers.

Microcomputers**:** A computer with a microprocessor and its central processing unit is known as a microcomputer. They do not occupy space as much as mainframes do. When supplemented with a keyboard and a mouse, microcomputers can be called personal computers. A monitor, a keyboard and other similar input-output devices, computer memory in the form of RAM and a power supply unit come packaged in a microcomputer. These computers can fit on desks or tables and prove to be the best choice for single-user tasks.

1.6.3 Personal computers

Personal computers come in different forms such as desktops, laptops and personal digital assistants. Let us look at each of these types of computers.

Desktops**:** A desktop is intended to be used on a single location. The spare parts of a desktop computer are readily available at relatively lower costs. Power consumption is not as critical as that in laptops. Desktops are widely popular for daily use in the workplace and households.

Laptops**:** Similar in operation to desktops, laptop computers are miniaturized and optimized for mobile use. Laptops run on a single battery or an external adapter that charges the computer batteries. They are enabled with an inbuilt keyboard, touch pad acting as a mouse and a liquid crystal display. Their portability and capacity to operate on battery power have proven to be of great help to mobile users.

Notebooks: They fall in the category of laptops, but are inexpensive and relatively smaller in size. They had a smaller feature set and lesser capacities in comparison to regular laptops, at the time they came into

the market. But with passing time, netbooks too began featuring almost everything that notebooks had. By the end of 2008, netbooks had begun to overtake notebooks in terms of market share and sales.

Personal Digital Assistants (PDAs): It is a handheld computer and popularly known as a palmtop. It has a touch screen and a memory card for storage of data. PDAs can also be used as portable audio players, web browsers and smartphones. Most of them can access the Internet by means of Bluetooth or Wi-Fi communication.

Minicomputers: In terms of size and processing capacity, minicomputers lie in between mainframes and microcomputers. Minicomputers are also called mid-range systems or workstations. The term began to be popularly used in the 1960s to refer to relatively smaller third generation computers. They took up the space that would be needed for a refrigerator or two and used transistor and core memory technologies. The 12-bit PDP-8 minicomputer of the Digital Equipment Corporation was the first successful minicomputer.

Servers: They are computers designed to provide services to client machines in a computer network. They have larger storage capacities and powerful processors. Running on them are programs that serve client requests and allocate resources like memory and time to client machines. Usually they are very large in size, as they have large processors and many hard drives. They are designed to be fail-safe and resistant to crash.

Supercomputers: The highly calculation-intensive tasks can be effectively performed by means of supercomputers. Quantum physics, mechanics, weather forecasting, molecular theory are best studied by means of supercomputers. Their ability of parallel processing and their well-designed memory hierarchy give the supercomputers, large transaction processing powers.

***Wearable Computers*:** A record-setting step in the evolution of computers was the creation of wearable computers. These computers can be worn on the body and are often used in the study of behaviour modelling and human health. Military and health professionals have incorporated wearable computers into their daily routine, as a part of such studies. When the users' hands and sensory organs are engaged in other activities, wearable computers are of great help in tracking human actions. Wearable computers do not have to be turned on and off and remain in operation without user intervention.

Tablet Computers: Tablets are mobile computers that are very handy to use. They use the touch screen technology. Tablets come with an onscreen keyboard or use a stylus or a digital pen. Apple's iPod redefined the class of tablet computers.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Discuss about some advance computer generations such as iPod and tablet PCs.

……..……

……..……

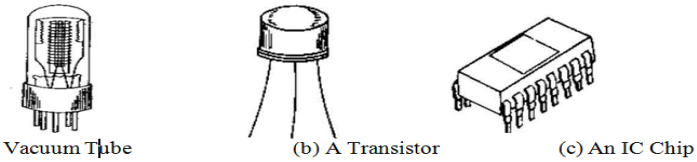
……

1.7 Generations of Computers

Generation in computer talk is a step in technology. It provides a framework for the growth of computer industry. Originally, the term "generation" was used to distinguish between varying hardware technologies but it has now been extended to include both hardware and software that together make up a computer system. The custom of referring to computer era in terms of generations came into wide use only after

1964. There are totally five computer generations known till today. Below we describe each generation along with its identifying characteristics. Although there is a certain amount of overlap between different generations, the approximate period shown against each are normally accepted. During description of various computer generations, you will come across several new terminologies and computer jargons that you may not be able to understand properly. There are Five Generations of Computer given below:

1 First Generation (1942-1955)
2 Second Generation (1955-1964)



(a) A Vacuum Tube (b) A Transistor (c) An IC Chip

Figure 1.1: Electronic devices used for manufacturing computers of different generations.

3 Third Generation (1964-1975)
4 Fourth Generation (1975-1989)
5 Fifth Generation (1989-Present)

Exercise: Check Your Progress 3

Note: i) Use the space below for your answer.

Ex1: Search more about Computer Generation.

……..……

……..……

……

Did You Know?

An integrated circuit (IC) is a small electronic device made out of a semiconductor material. The first integrated circuit was developed in the 1950s by Jack Kilby of Texas Instruments and Robert Noyce of Fairchild Semiconductor.

Exercise: Check Your Progress 4

Case Study-Apple Computer

Apple Computer is an American Multinational corporation with a focus on designing and manufacturing consumer electronics and even develops software products. It was co-founded by Steve Wozniak and Steve Jobs. Steve Wozniak met Steve Jobs while he was working at Hewlett-Packard. Steve Jobs worked part time, where he would finish up games that they designed in Grass Valley.

In 1975, the first personal computer kit, the Alistair 8800 was announced. Since Steve Wozniak could not afford an Alistair 8800 he decided to build his own personal computer by using cheaper chips. As circuit board alone, it could do more than Alistair. He and Steve Jobs called it Apple I, Jobs handled on marketing it while Wozniak continued to improve it. By 1977, Wozniak had built Apple II, then he and Jobs decided to form Apple Computer Inc. when it went public on 1980, its stock value was $117 million, three years later it was $985 million.

This story of how Apple started is one of my favourite. We watched a film titled “Pirates of the Silicon Valley” last year that’s why Apple story is a little familiar for me. We can see that Steve Wozniak is determined that he will develop a computer even if he was still a child. I can say that if we really put out hard work and commitment to do something then we can be successful. According to Wozniak, if we try to start our own company, we must have the highest ethics and be open and truthful about things, not hide. We must not lead people. Know in your heart that you are a good person with good goals because it will carry over to our own self-confidence: make our own product better that the average person would.

Questions

1. Who was the founder of Apple Computer?
2. Write the year of first Apple computer.

1.8 Summary

- The word computer comes from the word 'compute', which means, to calculate.
- Computer generation like First Generation, Second Generation, Third Generation, Fourth Generation and Fifth Generation.
- Computers are emotionless. They do not have emotion, like dislike feelings.
- Basic Pascal invented the first mechanical adding machine in 1642.
- Charles Babbage, a nineteenth century Professor at Cambridge University, is considered the father of modern digital computers.

1.9 Keywords

Generation: Originally, the term .generation was used to distinguish between varying hardware technologies but it has now been extended to include both hardware and software that together make up a computer system.

Graphical user interface (GUI): It enables new users to quickly learn how to use computers.

Integrated Circuits: They are usually called ICs or chips. They are complex circuits which have been etched onto tiny chips of semiconductor (silicon). The chip is packaged in a plastic holder with pins spaced on a 0.1"(2.54 mm) grid which will fit the holes on strip board and breadboards. Very fine wires inside the package link the chip to the pins.

***Medium scale integration* (*MSI*):** Medium-Scale Integration is a term used in electronic chip manufacturing industry. An integrated circuit which contained hundreds of transistors on each chip, called Medium-Scale Integration (MSI).

Small-Scale Integration (SSI): The first integrated circuits contained only a few transistors. Called Small-Scale Integration (SSI), they used circuits containing transistors numbering in the tens.

1.10 Review Questions

1. Define a computer? It is also known as a data processor. Explain.
2. Differentiate between data and information. Which is more useful to the people and why?
3. List and explain some important characteristics of a computer.
4. Who is known as the father of modern digital computers?
5. What are the limitations to use the computer?
6. What are the basic uses of computers?
7. Define the types of computers.
8. Which was the first commercially produced computer? When and where it was first installed?
9. Define: IBM, ENIAC, EDVAC, EDSAC, and UNIVAC.
10. Discuss the different generations of computers.

Answers for Self Assessment Questions

1. (b) 2. (a) 3. (c) 4. (a)

1.11 Further Studies

1. Computer Fundamentals by P. K. Sinha

2

Personal Computer (PC)

STRUCTURE

2.0 Objectives

2.1 Introduction

2.2 Algorithm

2.3 Personal Computer

2.4 Uses of a Personal Computer

2.5 Components of Personal Computers

2.6 Evolution of PCs

2.7 Developments of Processors

2.8 Architecture of Pentium IV

2.9 Configuration of PC

2.10 Distributed Computer System

2.11 Parallel Computer System

2.12 Summary

2.13 Keywords

2.14 Review Questions

2.15 Further Studies

2.0 Objectives

After studying this chapter, you will be able to:

- Discuss the algorithm
- Explain about the personal computer
- Discuss the uses of a personal computer
- Define about components of personal computers
- Discuss the evolution of PCs
- Explain the development of processors
- Describe architecture of Pentium IV
- Discuss the configuration of a PC

2.1 Introduction

A PC (personal computer) is a microcomputer. A PC is a single-user system, designed to fit on a desk-top; hence the word Personal. The IBM PC was introduced in the early eighties and since then has been

modified and improved. Subsequent PCs have been designed to run any software written for previous versions of the PC. Many other manufacturers have produced compatible computers; that is, computers which work in the same manner as the IBM PC and use the same software. These are often known as clones.

2.2 Algorithm

This is a problem solving technique. An algorithm can be defined as a step by step procedure to solve a particular problem. It consists of English like statements. Each statement must be precise and well-defined to perform a specific operation. When these statements are executed for a given set of conditions, they will produce the required results. See Example:

Example: Write an algorithm to compute the area of a circle. Algorithm: Area of a circle
Step 1: Read radius
Step 2: [Compute the area]
Area = 3.142 x radius x radius Step 3: [Print the area]
Print 'Area of a circle=', Area Step 4: [End of algorithm]
Stop

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: Write an algorithm to compute the area of a cube.

……..……

……..……

……

2.3 Personal Computer

A personal computer (PC) is any general-purpose computer whose size, capabilities, and original sales price make it useful for individuals, and which is intended to be operated directly by an end-user with no intervening computer operator. In contrast, the batch processing or time-sharing models allowed larger, more expensive minicomputer and mainframe systems to be used by many people, usually at the same time. Large data processing systems require a full-time staff to operate efficiently.

Did You Know?
Through the 1980s, computers were further developed for household use, with software for personal productivity, programming and games. One such machine, the Commodore 64, totaled 17 million units sold, making it the best-selling single personal computer model of all time.

2.4 Uses of a Personal Computer

Personal computers can be instructed to perform a variety of individual functions. A set of instructions that tells a computer what to do is called a program. Today, more than 10,000 application programs are available for use on personal computers. They include such popular programs as word processing programs, spreadsheet programs, database programs, and communication programs.

2.5 Components of Personal Computers

If you use a desktop computer, you might already know that there is not any single part called the "computer." A computer is really a system of many parts working together. The physical parts, which you can see and touch, are collectively called hardware. (Software, on the other hand, refers to the instructions, or programs, that tell the hardware what to do.)
Figure 2.1 the most common hardware in a desktop computer system. Your system may look a little different, but it probably has most of these parts. A laptop computer has similar parts but combines them into a single notebook-sized package.

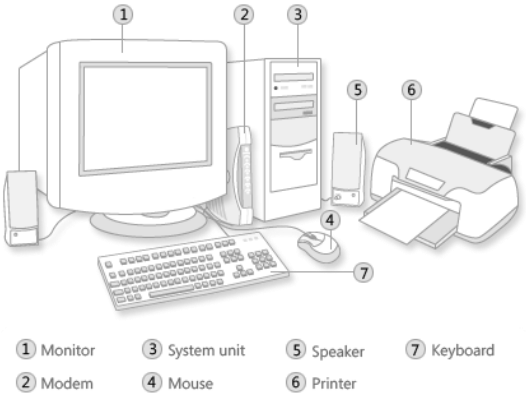


Figure 2.1: Components of personal computers.

2.5.1 System Unit

The system unit is the core of a computer system. Usually it is a rectangular box placed on or underneath your desk. Inside this box are many electronic components that process information. The most important of these components is the central processing unit (CPU), or microprocessor, which acts as the "brain" of your computer. Another component is random access memory (RAM), which temporarily stores information that the CPU uses while the computer is on. The information stored in RAM is erased when the computer is turned off.

Almost every other part of your computer connects to the system unit using cables. The cables plug into specific ports (openings), typically on the back of the system unit. Hardware that is not part of the system unit is sometimes called a peripheral device or device.

2.5.2 Storage

Your computer has one or more disk drives—devices that store information on a metal or plastic disk. The disk preserves the information even when your computer is turned off. There are some types of storage device:-

Hard disk drive

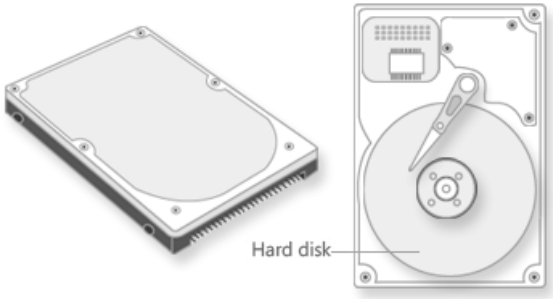


Figure 2.2: Hard disk drive.

CD and DVD drives
Floppy disk drive



Figure 2.3: Floppy disk drives.

Caution
Be careful while writing data in CD/DVD, if power fails data may be loss.

2.5.3 Mouse

A mouse is a small device used to point to and select items on your computer screen. Although mice come in many shapes, the typical mouse does look a bit like an actual mouse. It is small, oblong, and connected to the system unit by a long wire that resembles a tail. Some newer mice are wireless.
A mouse usually has two buttons: a primary button (usually the left button) and a secondary button. Many mice also have a wheel between the two buttons, which allows you to scroll smoothly through screens of information.

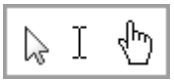


Figure 2.4: Mouse pointer

When you move the mouse with your hand, a pointer (see Figure 2.4) on your screen moves in the same direction. (The pointer's appearance might change depending on where it is positioned on your screen.) When you want to select an item, you point to the item and then click (press and release) the primary button. Pointing and clicking with your mouse is the main way to interact with your computer.

2.5.4 Keyboard

A keyboard (see Figure 2.5) is used mainly for typing text into your computer. Like the keyboard on a typewriter, it has keys for letters and numbers, but it also has special keys:

- The *function keys*, found on the top row, perform different functions depending on where they are used.
- The *numeric keypad*, located on the right side of most keyboards, allows you to enter numbers quickly.
- The *navigation keys*, such as the arrow keys, allow you to move your position within a document or webpage.



Figure 2.5: Keyboard.

You can use your keyboard and mouse to perform many of the same tasks.

2.5.5 Monitor

A monitor (see Figure 2.6) displays information in visual form, using text and graphics. The portion of the monitor that displays the information is called the screen. Like a television screen, a computer screen can show still or moving pictures.There are two basic types of monitors: CRT (cathode ray tube) monitors and

LCD (liquid crystal display) monitors. Both types produce sharp images, but LCD monitors have the advantage of being much thinner and lighter. CRT monitors, however, are generally more affordable.

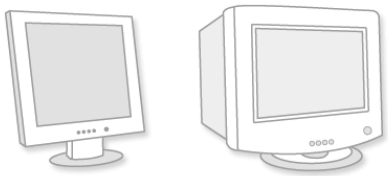


Figure 2.6: Monitor

2.5.6 Printer

A printer (see Figure 2.7) transfers data from a computer onto paper. You do not need a printer to use your computer, but having one allows you to print e-mail, cards, invitations, announcements, and other materials. Many people also like being able to print their own photos at home.

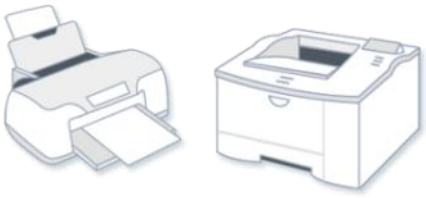


Figure 2.7: Printer

2.5.7 Speakers

Speakers (see Figure 2.8) are used to play sound. They may be built into the system unit or connected with cables. Speakers allow you to listen to music and hear sound effects from your computer.

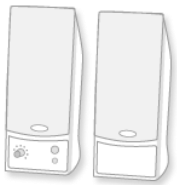


Figure 2.8: Speakers.

2.5.8 Modem

To connect your computer to the Internet, you need a modem. A modem (see Figure 2.9) is a device that sends and receives computer information over a telephone line or high-speed cable. Modems are sometimes built into the system unit, but higher-speed modems are usually separate components.

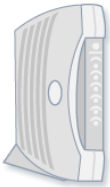


Figure 2.9: Modem.

2.6 Evolution of PCs

John W. Mauchly, who worked with the Moore School of Electrical Engineering, coined the term "personal computer" in the 1960s, according to the University of Pennsylvania. A personal computer is a machine intended for individual use that receives and provides information, calculates and manipulates data. Original computers, called mainframes, were so large that they filled up much of the space in a room. Technology eventually evolved to the point that microprocessors were invented, making it possible for personal computers to become a common business and household item.

1. The Start of the Computer Industry
2. The "Trinity"
3. Home Computers
4. IBM, PC Clones and Apple Computers
5. Multimedia Personal Computers

Exercise: Check Your Progress 2
Note: i) Use the space below for your answer.
Ex1: Search more about Computer Evolution.

……..……

……..……

……

2.7 Developments of Processors

A computer processor is one of the most advanced technologies available in the market today. It is also one of the most defining inventions by mankind, along with television and automobiles. Computer processors grow by leaps and bounds every year, with processors getting faster year in and year out. Processor speeds are increasing all the time and their sophistication is growing with every innovation.
Processors are found everywhere, in many applications such as cars, cameras, and cell phones. A processor is the brain of the computer and it controls other parts of the computer such as the memory, the hard drive and the video card. The processor is found in most high-tech gadgets such as game consoles, digital audio players, and GPS devices. Processors have complicated circuitry and their architecture gets more complex with every improvement in their design.
The Intel 4004 was a 4-bit processor which had a speed of 740 kHz. In 1976, Intel introduced the 16-bit 8086 processor which had a speed of 5 MHz. A later version of the 8086 was used to build the first personal computer by IBM. This was followed by the Intel 486, which was a 32-bit processor which had a speed of 16 MHz. During this time, several improvements in technology were made. For instance, processors could run in both real mode and protected mode, which introduced the concept of multitasking. Power-saving features, such as the System Management Mode (SMM), meant that the computer could power down various components. Computers finally went from command-line interaction to WIMP (Window, Icon, Menu, Pointing device) interaction.

In 1993, Intel introduced the Pentium processor which has a speed of 60 MHz. This was followed by the Pentium II which has a speed of 233 MHz, and the Pentium III which has a speed of 450 MHz, and the Pentium 4 which has a speed of 1.3 GHz. Later, Intel brought out the Celeron processor, which has a speed of 266 MHz and which is used in affordable low-end computers. In 2003, Intel inaugurated the Pentium M processor, which ushered in a new era of mobile computing, under the Centrino platform. The Pentium M is slower, at 900 MHz, so that energy consumption is reduced and the battery of the laptop lasts longer. In 2006, Intel introduced the Core processor which has a speed of 1.6 GHz. It has more than one core, like in the case of Core Duo (which has two cores) and has virtualization capability which allows multiple copies of an operating system to be run on the same computer.

While Intel is the leading company in the manufacturing of processors, there are other companies such as AMD that make processors too. In 1991, AMD had brought out the Am386 processor and its speed is 40 MHz. It is compatible with the Intel 386 processor. In 1999, AMD introduced the Athlon processor which has a speed of 500 MHz. Athlon was a legitimate competitor to Intel Pentium III because it was faster. As a matter of fact, AMD Athlon was the first processor to reach the speed of 1 GHz. The future for the computer processor industry is promising, as processors will continue to get faster and cheaper. According

to Moore's Law, the number of transistors on a chip used to double every year, and from 1975, it used to double every two years.

In the future, processors will have more cores that will be blistering fast and reduce power consumption. Software programmers will have to create multi-threaded applications to utilize the multiple cores. Computers with such processors will be faster for multimedia applications such as graphics software, audio players and video players. There is also a possibility that optical computing will increase processor speeds exponentially. All these signs point to a brighter future for processors, which will be to the benefit of everyone.

Did You Know?

In 1971 Intel made the first processor called as Intel 4004 and was made possible because of integrated circuit technology. An integrated circuit means that the entire CPU (central processing unit) is on a single chip.

2.8 Architecture of Pentium IV

In early 2000, Intel unveiled details of its first new IA-32 core since the Pentium Pro – introduced in 1995. Previously codenamed Willamette – after a river that runs through Oregon – it was announced a few months later that the new generation of microprocessors would be marketed under the brand name Pentium 4 (see Figure 2.10) and be aimed at the advanced desktop market rather than servers.

Representing the biggest change to Intel's 32-bit architecture since the Pentium Pro in 1995, the Pentium 4's increased performance is largely due to architectural changes that allow the device to operate at higher clock speeds and logic changes that allow more instructions to be processed per clock cycle. Foremost amongst these is the Pentium 4 processor's internal pipeline – referred to as Hyper Pipeline – which comprises 20 pipeline stages versus the ten for the P6 micro architecture.

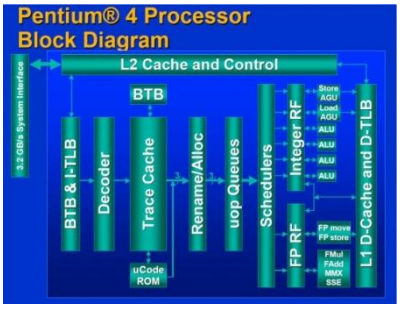


Figure 2.10: Architecture of Pentium IV

A typical pipeline has a fixed amount of work that is required to decode and execute an instruction. This work is performed by individual logical operations called gates. Each logic gate consists of multiple transistors. By increasing the stages in a pipeline, fewer gates are required per stage. Because each gate requires some amount of time (delay) to provide a result, decreasing the number of gates in each stage allows the clock rate to be increased. It allows more instructions to be in flight or at various stages of decode and execution in the pipeline. Although these benefits are offset somewhat by the overhead of additional gates required to manage the added stages, the overall effect of increasing the number of pipeline stages is a reduction in the number of gates per stage, which allows a higher core frequency and enhances scalability.

In absolute terms, the maximum frequency that can be achieved by a pipeline in an equivalent silicon production process can be estimated as:

1/ (pipeline time in ns/number of stages) * 1,000 (to convert to megahertz) = maximum frequency

Accordingly, the maximum frequency achievable by a five-stage, 10-ns pipeline is: 1/ (10/5) * 1,000 = 500MHz

In contrast, a 15-stage, 12-ns pipeline can achieve: 1/ (12/15) * 1,000 = 1,250MHz or 1.25GHz

Additional frequency gains can be achieved by changing the silicon process and/or using smaller transistors to reduce the amount of delay caused by each gate.

Other new features introduced by the Pentium 4′s new micro-architecture – dubbed NetBurst – include:

- An innovative Level 1 cache implementation comprising – in addition to an 8KB data cache – an Execution Trace Cache, that stores up to 12K of decoded x86 instructions (micro-ops), thus removing the latency associated with the instruction decoder from the main execution loops.
- A Rapid Execution Engine that pushes the processor's ALUs to twice the core frequency resulting in higher execution throughput and reduced latency of execution – the chip actually uses three separate clocks: the core frequency, the ALU frequency and the bus frequency.
- A very deep, out-of-order speculative execution engine – referred to as the Advanced Dynamic that avoids stall can occur while instructions are waiting for dependencies resolve by providing a large window of from which units choose.
- A 256KB Level 2 Advanced Transfer Cache that provides a 256-bit (32-byte) interface that transfers data on each core clock, thereby delivering a much higher data throughput channel – 44.8 GBps (32 bytes x 1 data transfer per clock x 1.4 GHz) – for a 1.4GHz Pentium 4 processor.
- SIMD Extensions 2 (SSE2) – the latest iteration of Intel's Single Instruction Multiple Data technology which integrate 76 new SIMD instructions and improvements to 68 integer instructions, allowing chip grab 128-bits at a time in *both* floating-point and integer and thereby accelerate CPU-intensive encoding and decoding operations such as streaming video, speech, 3D rendering and other multimedia procedures.
- The industry's first 400MHz system bus, providing a 3-fold increase in throughput compared with Intel current 133MHz bus.

Based on Intel's ageing 0.18-micron process, the new chip comprised a massive 42 million transistors. Indeed, the chip's original design would have resulted in a significantly larger chip still – and one that was ultimately deemed too large to build economically at 0.18 micron. Features that had to be dropped from the Willamette's original design included a larger 16KB Level 1 cache, two fully functional FPUs and 1MB of external Level 3 cache. What this reveals is that the Pentium 4 really needs to be built on 0.13-micron technology – something that was to finally happen in early 2002.

The first Pentium 4 shipments – at speeds of 1.4GHz and 1.5GHz – occurred in November 2000. Early indications were that the new chip offered the best performance improvements on 3D applications – such as games – and on graphics intensive applications such as video encoding. On everyday office applications – such as word processing, spreadsheets, Web browsing and e-mail – the performance gain appeared much less pronounced.

One of the most controversial aspects of the Pentium 4 was its exclusive support – via its associated chipsets – for Direct Rambus DRAM (DRDRAM). This made Pentium 4 systems considerably more expensive than systems from rival AMD that allowed use of conventional SDRAM, for little apparent performance gain. Indeed, the combination of an AMD Athlon CPU and DDR SDRAM outperformed Pentium 4 systems equipped with DRDRAM at a significantly lower cost.

During the first half of 2001 rival core logic providers SiS and VIA decided to exploit this situation by releasing Pentium 4 chipsets that did support DDR SDRAM. Intel responded in the summer of 2001 with the release of its i845 chipset. However, even this climb down appeared half-hearted, since the i845 supported only PC133 SDRAM and not the faster DDR SDRAM. It was not until the beginning of 2002 that the company finally went the whole hog, re-releasing the i845 chipset to extend support to DDR SDRAM as well as PC133 SDRAM.

During the course of 2001 a number of faster versions of the Pentium 4 CPU were released. The 1.9GHz and 2.0GHz versions released in the summer of 2001 were available in both the original 423-pin Pin Grid Array (PGA) socket interface and a new Socket 478 form factor. The principal difference between the two

is that the newer format socket features a much more densely packed arrangement of pins known as a micro Pin Grid Array (AµPGA) interface. It allows both the size of the CPU itself and the space occupied by the interface socket on the motherboard to be significantly reduced.

Exercise: Check Your Progress 3
Note: i) Use the space below for your answer.
Ex1: Search about some latest versions of processors.

……..……

……..……

……

2.9 Configuration of PC

It can be safely assumed that an average home computer user uses his computer mostly to send/receive mails, browse the net, access online applications, watch movies, listen to music, use some desktop applications and work on documents or spreadsheets, in that order of decreasing frequency of use. But for the avid gamers, most would play some sort of simple computer games like card games, puzzles, chess etc. A very small percentage of the population would use a home computer for programming.
Except for extreme gaming, rendering animations and heavy duty data processing, most of the processing requirements of a typical user can be met by an entry level system. Of these three activities only the 2nd and the 3rd could be considered of any practical value to the system (and to an extent to the user) and these two are required by a very small percentage of the total population. We could possibly even conclude that most of the home users could perform most of their normal computer uses using an entry level PC.
Consider the following configuration

- AMD Sempron 3000
- 512 MB DDRII RAM
- 80 GB Harddisk
- DVD Combo Drive
- 15" CRT Monitor
- Multimedia Keyboard
- Optical Mouse
- Speakers

Self Assessment Questions

1. The Automatic Sequence Controlled Calculator was created in
(a) 1949 (b) 1942
(c) 1944 (d) None of these.

2. The IBM 701 was launched in:
(a) 1955 (b) 1952
(c) 1953 (d) None of these

3. ……can is defined as a step by step procedure to solve a particular problem.
(a) Word (b) Problem
(c) Algorithm (d) None of these

4. Which is not a windows version?
(a) Definiteness (b) Finiteness

(c) Effectiveness (d) All of these

5. ……..stores information on a hard disk, a rigid platter or stack of platters with a magnetic surface.
(a) CD or DVD drive (b) Hard disk drive
(c) Floppy disk drives (d) None of these

6. ……store information on floppy disks
(a) CD or DVD drive (b) Hard disk drive
(c) Floppy disk drives (d) None of these

2.10 Distributed Computer System

A distributed computer system consists of multiple software components that are on multiple computers, but run as a single system. The computers that are in a distributed system can be physically close together and connected by a local network, or they can be geographically distant and connected by a wide area network. A distributed system can consist of any number of possible configurations, such as mainframes, personal computers, workstations, minicomputers, and so on. The goal of distributed computing is to make such a network work as a single computer.
Distributed systems offer many benefits over centralized systems, including the following:

- Scalability
- Redundancy

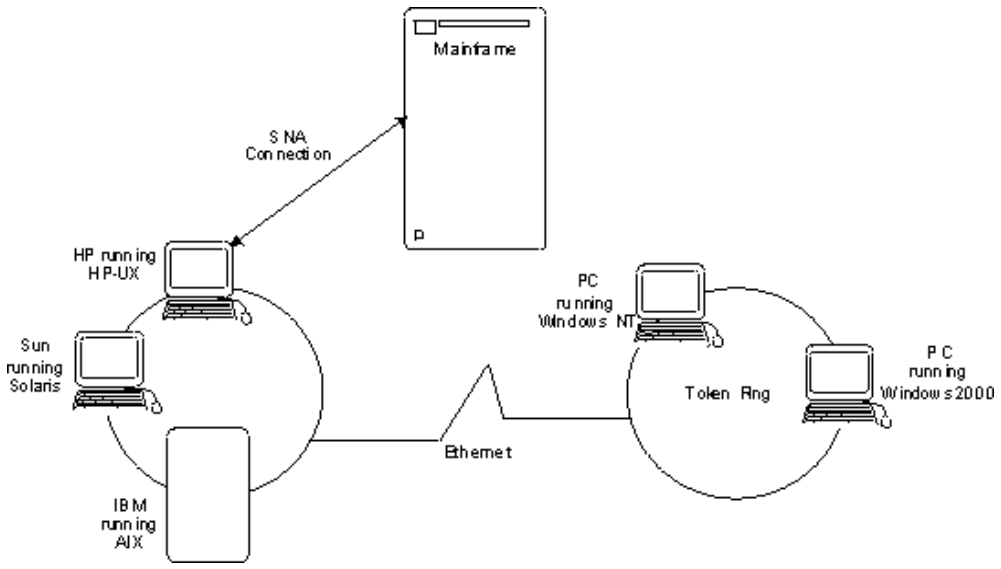


Figure 2.11: Distributed system.

2.11 Parallel Computer System

It is now clear that silicon based processor chips are reaching their physical limits in processing speed, as they are constrained by the speed of electricity, light, and certain thermodynamic laws. A viable solution to overcome this limitation is to connect multiple processors working in coordination with each other to solve grand challenge problems. Hence, high performance computing requires the use of Massively Parallel Processing (MPP) systems (see Figure 2.12) containing thousands of powerful CPUs. A dominant representative computing system (hardware) built using MPP approach is C-DAC.s PARAM supercomputer.

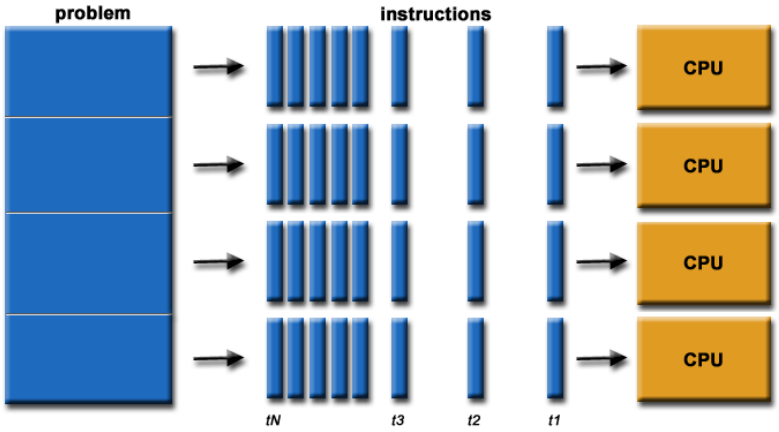


Figure 2.12: Parallel computer system.

All high performance systems will be parallel computer systems. High-end super computers will be the Massively Parallel Processing (MPP) systems having thousands of processors interconnected. To perform well, these parallel systems require an operating system radically different from current ones. Most researchers in the field of operating systems have found that these new operating systems will have to be much smaller than traditional ones to achieve the efficiency and flexibility needed. The solution appears to be to have a new kind of OS that is effectively a compromise between having no OS at all and having a large monolithic OS that does many things that are not needed.

Exercise: Check Your Progress 4

Case Study-IBM PC Server 704

Choosing a server is one of the most important decisions any company can make, it determines its effectiveness, efficiency and productivity of a company

IBM PC Server 704s keep Grammar AG in the driving seat

To keep pace with new and highly demanding applications to better run their (just In Time) JIT production schedules, improve their response rates and better service their customers, Grammer AG turned to the powerful IBM PC Server 704. Grammer AG, based in Amberg, Germany, has implemented applications such as SAP R/3, Oracle and Microsoft Office running on a Microsoft Windows NT Server platform to drive its business intelligence solution.

Leaders working with leaders

Renowned for technological excellence in ergonomic seating solutions, Grammer AG is a worldwide manufacturer in the design industry. Working with leading motor companies, Grammer AG's seating products are used in many areas: cars, trucks, buses, tractors and building machinery - it has even supplied seats for the new German ICE high-speed train. Most major motor and utility vehicle manufacturers have implemented a Just in Time (JIT) production schedule strategy and Grammer AG turned to IBM to compete with its competitors. IBM PC Server 704 systems running Microsoft Windows NT with SAP R/3 were rolled out to meet customer requirements and to improve Grammer's internal operations and processes.

Until recently, a mainframe system sat at the heart of Grammer's IT infrastructure, running a Production Planning System (PPS) application and other applications for accounting and human resources. Although Grammer had been very satisfied with this system, it was impossible to reengineer it to include the new, complex enterprise resource planning applications that were needed to meet the demanding JIT requirements. Additionally, with the need to adapt to the European Monetary Unit (EMU) in 1999 and the Year 2000 issues to be faced, Grammer knew that it was time for a technology overhaul.

Enter IBM PC Server Systems and SAP R/3

Gunnar Blodig, IT manager at Grammer AG, set very high standards for the new hardware. It had to offer not only a high level of performance, power and integration, but it also supports the mission critical SAP R/3 applications essential in meeting the company's objectives. Grammer's complete deployment combined 13 IBM PC Server 704 systems running Microsoft Windows NT Server, SAP R/3 and various software applications.

Reliability and manageability were critical for success. IBM's high performance Serial Storage Architecture (SSA) hard disk storage and the IBM PC Server 704's proven track record were determining factors in Grammer's decision to revamp its technology. The IBM PC Server 704 offered a strong platform for Grammer to build its application-serving environment while SAP R/3 made the most sense when it came to the best plan for its business. "We were looking for something to cover all fields such as commercial applications, human resources, or PPS throughout the whole company. International support for all components was very important to us, because we are a presence in every part of the world," Blodig explains. SAP R/3 will allow Grammer to meet the demanding universe of close cooperation with motor vehicle manufacturers and integrate the flow of data and information throughout its worldwide operating group via datamining, datamarts and datawarehousing.

The best choice

Grammer AG's choice to implement an IBM solution was reinforced by its systems integrator - Memorex Telex. "(The team at) Memorex Telex was very clear in its endorsement of IBM. We compared the hardware against benchmarks that we had derived from the host-centered (mainframe) world, and the IBM PC Servers were unequivocally the best. We asked other vendors for equally powerful hard disks, and some said that such fast hard disks were not available on the market."

In addition, the implementation has been a favourable learning experience for the Grammer AG information technologies team, "We are building the new IT infrastructure using our experienced mainframe people and they show terrific motivation and involvement. They are looking forward to the new world, because it is a new future for them, too. We rely upon IBM as part of that team. We know IBM, we know their people and together we always find a way of getting things done."

After Grammer's complete rollout of the SAP logistics modules, scheduled for 3rd Quarter 1998, nearly 500 IBM personal computers will be connected to the network. At that point, system fine-tuning will be conducted to enhance system resiliency and implement Microsoft Cluster Server software.

Questions

1. What are the advantages of IBM PC Server 704?
2. What was the view of Grammer about IBM personal computers?

2.12 Summary

- A personal computer is a machine intended for individual use that receives and provides information, calculates and manipulates data.
- Algorithm consists of English like statements. Each statement must be precise and well-defined to perform a specific operation.
- A mouse is a small device used to point to and select items on your computer screen.
- A keyboard is used mainly for typing text into your computer. Like the keyboard on a typewriter, it has keys for letters and numbers, but it also has special keys.
- A monitor displays information in visual form, using text and graphics.
- A printer transfers data from a computer onto paper.
- A modem is a device that sends and receives computer information over a telephone line or high-speed cable.

- Distributed computing systems run on hardware that is provided by many vendors, and use a variety of standards-based software components.

2.13 Keywords

Algorithm: This is a problem solving technique, defined as a step by step procedure to solve a particular problem.

Effectiveness: This means that operations must be simple and are carried out in a finne time at one or more levels of complexity. It should be effective whenever traced manua:?, for the results.

Finiteness: It should be a sequence of finite instructions. That is, it should end after: fixed time. It should not enter into an infinite loop.

Hardware: The physical parts of a computer, which you can see and touch, are collectively called hardware.

Input: The value entered by a user to the system is called input.

Output: The reply given by the system as the answer of input is called output.

Personal computer (PC): Any general-purpose computer whose size, capabilities, and original sales price make it useful for individuals, and which is intended to be operated directly by an end-user with no intervening computer operator.

2.14 Review Questions

1. What is algorithm? Discuss its characteristics and notations.
2. What do you mean by personal computer? How many types of personal computers are available?
3. What are the basic uses of the personal computers?
4. What are the main components of a personal computer? Explain in details.
5. Write a brief notes about the advancement of the personal computers.
6. How the processors are developed at different time slots? Explain in detail.
7. Define the architecture of Pentium IV.
8. What is the general configuration of a personal computer? Discuss in brief.
9. What is the distributed computer system? Give the architecture.
10. What is the parallel computer system?

Answers for Self Assessment Questions

1. (c) 2. (b) 3. (c) 4. (d) 5. (b) 6. (c)

2.15 Further Studies

1. A History of the Personal Computer: The People and the Technology by Roy A. Allan
2. Computer Fundamentals by P. K. Sinha

3

Boolean algebra and Logic Gates

STRUCTURE

3.0 Objectives

3.1 Introduction

3.2 Boolean Algebra

3.3 Binary Valued Quantities and Operator

3.4 Basic Postulates of Boolean Algebra

3.5 Theorems of Boolean algebra

3.6 De Morgan's Theorems

3.7 Proving the Equations of Boolean Expressions by Truth Table

3.8 Principle of Duality

3.9 Basic of Logic Gates

3.10 Use of Logic Gates in Circuits

3 11 Karnaugh Maps (K-Map)

3.12 Summary

3.13 Keywords

3.14 Review Questions

3.15 Further Studies

3.0 Objectives

After studying this chapter, you will be able to:

- Understand the Boolean algebra
- Understand the binary valued quantities and operator
- Explain the basic postulates of Boolean algebra
- Explain the theorems of Boolean algebra
- Define the de Morgan's theorems

3.1 Introduction

Boolean logic forms the basis for computation in modern binary computer systems. You can represent any algorithm, or any electronic computer circuit, using a system of Boolean equations. This provides a brief introduction to Boolean algebra, truth tables, canonical representation, of Boolean functions, Boolean function simplification, logic design, combinatorial and sequential circuits, and hardware/software equivalence.

3.2 Boolean Algebra

That framework is Boolean algebra. This document of course provides only and introduction to Boolean algebra, refer to dedicated texts for a detailed discussion of the subject.

All arithmetic operations performed with Boolean quantities have but one of two possible outcomes: either 1 or 0. There is no such thing as "2" or "-1" or "1/2" in the Boolean world. It is a world in which all other possibilities are invalid by fiat. As one might guess, this is not the kind of math you want to use when balancing a checkbook or calculating current through a resistor. However, Claude Shannon of MIT fame recognized how Boolean algebra could be applied to on-and-off circuits, where all signals are characterized as either "high" (1) or "low" (0). His 1938 thesis, titled A Symbolic Analysis of Relay and Switching Circuits, put Boole's theoretical work to use in a way Boole never could have imagined, giving us a powerful mathematical tool for designing and analyzing digital circuits.

Did You Know?

The English mathematician George Boole sought to give symbolic form to Aristotle's system of logic. Boole wrote a treatise on the subject in 1854, titled an Investigation of the Laws of Thought, on Which Are Founded the Mathematical Theories of Logic and Probabilities, which codified several rules of relationship between mathematical quantities limited to one of two possible values: true or false, 1 or 0. His mathematical system became known as Boolean algebra.

Caution

1. Remember that in the world of Boolean algebra, there are only two possible values for any quantity and for any arithmetic operation: 1 or 0.
2. Be careful that "Truth table" must be considered, while designing digital circuits.

Did You Know?

Stone's representation theorem for Boolean algebras states that every Boolean algebra is isomorphic to a field of sets was stated by Marshall Harvey Stone in 1936.

3.3 Binary Valued Quantities and Operator

The singular reason for learning and using the binary numeration system in electronics is to understand how to design, build, and troubleshoot circuits that represent and process numerical quantities in digital form. Since the bivalent (two-valued) system of binary bit numeration lends itself so easily to representation by "on" and "off" transistor states (saturation and cutoff, respectively), it makes sense to design and build circuits leveraging this principle to perform binary calculations.

3.3.1 Logical AND operator &&

The && (logical AND) operator indicates whether both operands are true.

In C, if both operands have nonzero values, the result has the value 1. Otherwise, the result has the value 0. The type of the result is int. Both operands must have a arithmetic or pointer type. The usual arithmetic conversions on each operand are performed.

The following examples show how the expressions that contain the logical AND operators are evaluated:

Expression	Result
1 && 0	False or 0
1 && 4	True or 1
0 && 0	False or 0

Figure: 1

3.3.2 Logical OR operator ||

The || (logical OR) operator indicates whether either operand is true.

In C, if either of the operands has a nonzero value, the result has the value 1. Otherwise, the result has the value 0. The type of the result is int. Both operands must have an arithmetic or pointer type. The usual arithmetic conversions on each operand are performed.

In C++ if either operand has a value of true, the result has the value true. Otherwise, the result has the value false. Both operands are implicitly converted to bool and the result type is bool.

Unlike the | (bitwise inclusive OR) operator, the || operator guarantees left-to-right evaluation of the operands. If the left operand has a nonzero (or true) value, the right operand is not evaluated.

The following examples show how expressions that contain the logical OR operator are evaluated:

Expression	Result
1 || 0	True or 1
1 || 4	True or 1
0 || 0	False or 0

Figure: 2

The following example uses the logical OR operator to conditionally increment y:

```
++x || ++y;
```

The expression ++y is not evaluated when the expression ++x evaluates to a nonzero (or true) quantity.

3.4 Basic Postulates of Boolean Algebra

The operation of ternary rejection in Boolean algebra is the operation ( ) given by (abc) = a'b' + b'c' + c'a'. There are a set of postulates for Boolean algebras in which ternary rejection is taken as the only primitive idea, besides that of class. As a result, all the special Boolean elements are introduced with elegance not possible in any other set known to the author. Thus, the negative of an element is defined in terms of the primitive ideas, and then any two contradictory elements are chosen to represent the zero elements and the universe element of the resulting Boolean algebra.

We prove the sufficiency of the new postulates for Boolean algebra by deriving from them the well known Whitehead-Huntington set;§ the proof of necessaries consists in the converse derivation. Finally, we establish the consistency and independence of the postulates by furnishing proof-systems of the usual type.

The New Postulates The new postulates have as undefined ideas a class K and a ternary operation ( ). The postulates are the propositions A_1-A_5 below. In Postulates A_3-A_5 the condition if the elements involved and their indicated combinations belong to K is to be understood.

Postulate A_1. K contains at least two distinct elements.
Postulate A_2. If a, b, c are elements of K, (abc) is an element of K.\
Postulate A_3. {abc) = (bca).
Definition 1. a' = (aaa).
Postulate A_4. (a'bb') =a.
Postulate A_5.* [ab(cde)'] = [(abc)'(abd)'e].

3.4.1 Theorems

We now list a number of theorems of the resulting algebra.

1. a"= a, where a"= (a')'.
2. (aab)=a'.
3. (abc) = (acb).
4. (abc) = (cba) = (bca) = (acb) = (cab) = (bac).
5. [a'(abc)' (a'b'c')']=a.

6. [a(abc)' (a'b'c')']=a'.
7. (abc) = [(abd)'(abd')'c].
8. [d' (abc)'(a'b'c')']=d.
9. If (a'bc) =a for all a, then c = b'.
10. (abc)' = (a'b'c').

3.4.2 Proofs of the Preceding Theorems

1. a'' = (a"a") = (a'a"a"') =a, by A_4, A_5, A_4.
2. (aab) =[(abb')' (abb')' b]= [ab(b'b'b)'] = (abb')= a', by A_4, 1, A_5, A_4, 1, A_4, 1.
3. Put (acb)'=d, [c(abc)'b]' = e. Then
   (abc) = [ab(ccd)'] = [ab(cdc)'] = [(abc)'(abd)' c] = [c(abc)'(abd)']
= {[c(abc)'a]' [c(abc)'b]'d} = {[c(abc)'a]ed) = {[ac{abc)']'ed)
= {[(aca)' (acb)'c]'ed) = {[(aac)' (acb)'c}'ed) = { [a(acb)'c]'ed
= {[ca(acb)']'ed} = {[(caa)'(cac)'b]'ed} = {[(aac)' (cac)'b]'ed}
= {[a(cac)'b]'ed} = {[a(cca)'b]'ed} = [(acb)'ed] = [d(acb)'e]
= { (acb)'(acb)'[c(abc)']'} = (acb), by 2, 1, A_3, A_5, A_3, A_5, A_3, A_5, A_3, 2, 1, A_3, A_5, A_3, 2, 1, A_3, 2, 1, A_3, 2, 1.
4. A_3, 3.
   In the rest of the proofs implicit use will be made of 1 and 4.
5. [a'(abc)'(a'b'c')'] = { [a'(abc)'a']'[a'(abc)'b']c'}
   ={a'[a'b'(abc)']'c') = {a'[(a'b'a)'(a'b'b)'c]'c'} = [a'c'(b'a'c)']
   = [(a'c'a')'(a'c'c)'b'] = [a'(a'c'c)'b'] = (a'a'b') = a, by A_5, 2, A_5, A_4, A_5, 2, A_4, 2.
6. [a(abc)'(ab'c')'] = {[a(abc)'a]'[a(abc)'b']'c'}
   = {a[ab'(abc']'c') = {ac'[(ab'a)'(ab'b)'c]'} = {ac'[a(ab'b)'c]'}
   = [ac'(aac)'] = {ac'a) = a', by A_5, 2, A_5, 2, A_4, 2, 2.
7. (abc) = [ab(dd'c)'] = [(abd)'(abd')'c], by A_4, A_5.
8. Put (abd)' = p, (abd')' = q, (a'b'd)' = r, (a'b'd') = s.
   Then
   [d'(abc)' (a'b'c')'] = [d'(pqc)'(rsc')']= { [d'r(pqc)']' [d's(pqc)']'c'}
   = { [(d'rp)'(d'rq)'c [(d'sp)'(d'sq)']'c'},by 7, A_5, A_5. But (d'rq) =d, (d'sp) =d by 5, and (d'sq) =d by 6.
   Hence the last expression reduces to
   { [cd'(d'rp)'(cd'd)'c') = {c'd'[cd'(d'rp)']'} = [(c'd'c)'(c'd'd')'(d'rp)']
   = [d'(c'd'd')'(d'rp)'] = [d'd'(d'rp)'=d, by 2, A_5, A_4, 2, 2.
9. Suppose that for two fixed elements b, c, (a'bc) =a for every element a. Then for a = c, c = (c'bc) = b' by A_4.
10. By 8, 9.

Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: The following example uses the logical AND operator to avoid division by zero:

(y != 0) && (x / y)

The expression x / y is not evaluated when y != 0 evaluates to 0 (or false).

……..……

……..……

……

3.5 Theorems of Boolean algebra

We have seen how Boolean algebra can be used to help analyze a logic circuit and express its operation mathematically. We will continue our study of Boolean algebra by investigating the various Boolean theorems (rules) that can help us to simplify logic expressions and logic circuits. The first group of theorems is given in Figure 3. In each theorem, x is a logic variable that can be either a 0 or a 1. Each theorem is accompanied by a logic-circuit diagram that demonstrates its validity.

Theorem (1) states that if any variable is ANDed with 0, the result has to be 0. This is easy to remember because the AND operation is just like ordinary multiplication, where we know that anything multiplied by 0 is 0. We also know that the output of an AND gate will be 0 whenever any input is 0, regardless of the level on the other input.

Theorem (2) is also obvious by comparison with ordinary multiplication.

Theorem (3) can be proved by trying each case. If $x = 0$, then $0 \cdot 0 = 0$; if $x=1$, then $1 \cdot 1=1$. Thus, $x \cdot x=x$.

Theorem (4) can be proved in the same manner. However, it can also be reasoned that at any time either x or its inverse x has to be at the 0 level, and so there AND product always has to be 0.

Theorem (5) is straightforward, since 0 added to anything does not affect its value, either in regular addition or in OR addition.

Theorem (6) states that if any variable is ORed with 1, the result will always be 1. Checking this for both values of x: $0 + 1 = 1$ and $1 + 1 = 1$. Equivalently, we can remember that an OR gate output will be 1 when any input is 1, regardless of the value of the other input.

Theorem (7) can be proved by checking for both values of x: $0 + 0 = 0$ and $1+1=1$.

Theorem (8) can be proved similarly, or we can just reason that at any time either x or $\overline{x}$ has to be at the 1 level so that we are always ORing a 0 and a 1, which always results in 1.

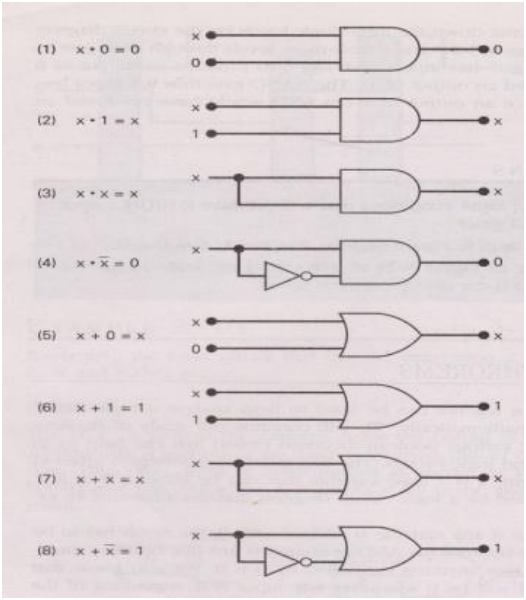


Figure 3: Single variable theorems.

Before introducing any more theorems, it should be pointed out that in applying theorems (1) through (8) the variable x may actually represent an expression containing more than one variable. For example, if we have $A\overline{B}$ $(\overline{A\overline{B}})$, we can invoke theorem (4) by letting $x=\bullet A\overline{B}$. Thus, we can say that $A\ \overline{B}\ (\overline{A\overline{B}} = 0$. The same idea can be applied to the use of any of these theorems.

3.5.1 Multivariable Theorems

The theorems presented below involve more than one variable:

Theorem (9) $x+ y= y+ x$

Theorem (10) $x \bullet y= y\bullet x$

Theorem (11) $x+(y+z) =(x+y)+z=x+y+z$

Theorem (12) $x(yz) = (xy)z= xyz$

Theorem (13a)	x(y + z) = xy + xz
Theorem (13b)	(w + x)(y + z) = wy + xy + wz + xz
Theorem (14)	x+ xy= x
Theorem (15)	x+ $\bar{x}$y= x+ y

Theorems (9) and (10) are called the commutative laws. These laws indicate that the order in which we OR Or AND two variables is unimportant; the result is the same.

Theorems (11) and (12) are the associative laws, which state that we can group the variables in an AND expression or OR expression any way we want.

Theorem (13) is the distributive law, which states that an expression can be expanded by multiplying term by term just the same as in ordinary algebra. This theorem also indicates that we can factor an expression. That is, if we have a sum of two (or more) terms, each of which contains a common variable, the common variable can be factored out just like in ordinary algebra. For example, if we have the expression $A\bar{B}C + \bar{A}\bar{B}\bar{C}$, we can factor out the $\bar{B}$ variable:

$A\bar{B}C + \bar{A}\bar{B}\bar{C} = \bar{B}(AC + \bar{A}\bar{C})$

As another example, consider the expression *ABC + ABD.* Here the two terms have the variables *A* and *B* in common, and so *A• B* can be factored out of both terms. That is,

ABC + ABD = AB(C + D)

Theorems (9) to (13) are easy to remember and use since they are identical to those of ordinary algebra. Theorems (14) and (15), on the other hand, do not have any counterparts in ordinary algebra. Each can be proved by trying all possible cases for *x* and *y.* This is illustrated for theorem (14) as follows:

Case 1. For *x*= 0, y= 0,

x+ xy= x
0+0 •0=0
0=0

Case 2. For *x*= 0, y= 1, *x+xy=x*
0+0 • 1=0

0+0=0
0=0

Case 3. For *x*= l, y= 0, *x+xy=x*
1+1•0=1

1+0=1
1 = 1

Case 4. For x= 1, y=1, *x+xy=x*
1+1•1=1

1+1=1
1 = 1

Theorem, (14) can also be proved by factoring and using theorems (6) and (2) as follows:

x+xy=x (1+y)
= x 1 [using theorem (6)]
= x [using theorem (2)]

Example

Simplify the expression y = $A\overline{B}D + A\overline{B}\overline{D}$.

Example

Simplify $z = (\overline{A} + B)(A + B)$.

Example

Simplify $x = ACD + \overline{A}BCD$.

Check Your Progress 2

Note: i) Use the space below for your answer.

Exercise: Simplify the Boolean expression F = C (B + C) (A + B + C).

……

……

……

3.6 De Morgan's Theorems

Two of the most important theorems of Boolean algebra were contributed by a great mathematician named DeMorgan. DeMorgan's theorems are extremely useful in simplifying expressions in which a product or sum of variables is inverted.

De Morgan's first theorems: In words, the complement of a logical sum equals the logical product of the complements. In the term of circuits, a NOR gate equals a bubbled AND gate.

De Morgan's second theorems: In words, the complement of a logical product equals the logical sum equals the complements. In the term of circuits, a NAND gate is equivalent to a bubbled OR gate.

The two theorems are:

(1) $(\overline{x + y}) = \overline{x}\ \overline{y}$

(2) $(\overline{x \bullet y}) = \overline{x} + \overline{y}$

Theorem (1) says that when the OR sum of two variables is inverted, this is the same as inverting each variable individually and then ANDing these inverted variables. Theorem (2) says that when the AND product of two variables is inverted, this is the same as inverting each variable individually and then ORing them. Each of DeMorgan's theorems can be readily proven by checking for all possible combinations of *x* and *y*.

Although these theorems have been stated in terms of single variables *x* and *y*, they are equally valid for situations where *x* and/or *y* are expressions that contain more than one variable. For example, let's apply them to the expression

$(\overline{A\overline{B} + C})$ as shown below:

$(\overline{A\overline{B} + C}) = (\overline{A\overline{B}}) \bullet \overline{C}$

Note that here we treated $A\overline{B}$ as *x* and *C* as *y*. The result can be further simplified since we have a product $A\overline{B}$ that is inverted. Using theorem (2), the expression becomes

$\overline{A\overline{B}} \bullet \overline{C} = (\overline{A} + \overline{\overline{B}}) \bullet \overline{C}$

Notice that we can replace $\overline{\overline{B}}$ by *B*, so that we finally have

$(\overline{A} + B) \bullet \overline{C} = \overline{A}\overline{C} + B\overline{C}$

This final result contains only inverter signs that invert a single variable.

When using De Morgan's theorems to reduce an expression, we may break an inverter sign at any point in the expression and change the operator at that point in the expression to its opposite (+ is changed to •, and vice versa). This procedure is continued until the expression is reduced to one in which only single variables are inverted. Two or more examples are given below.

1. $z = \overline{A + \bar{B}.C}$

$= \bar{A} \bullet ()$

$= \bar{A} \bullet (\bar{\bar{B}} + \bar{C})$

$= \bar{A} \bullet (B + \bar{C})$

$= \bar{A}\bar{B} + \bar{A}\bar{C} + \bar{D}\bar{E} + \bar{D}F$

2. $\omega = (\overline{(A + BC) \bullet \overline{D + EF}})$

$= (\overline{A + BC}) + (\overline{D + EF})$

$= (\bar{A} \bullet \overline{BC}) + (\bar{D} \bullet \bar{E}\bar{F})$

$= [\bar{A} \bullet (\bar{B} + \bar{C})] + [\bar{D} \bullet (\bar{E} + \bar{F})]$

DeMorgan's theorems are easily extended to more than two variables. For example, it can be proved that

$\overline{x + y + z} = \bar{x} \bullet \bar{y} \bullet \bar{z}$

$\bar{x} \bullet \bar{y} \bullet \bar{z} = \overline{x + y + z}$

and so on for more variables. Again, realize that any one of these variables can be an expression rather than a single variable.

3.6.1 Implications of DeMorgan's Theorems

Let us examine these theorems (1) and (2) from the standpoint of logic circuits. First, consider theorem (1),

$\overline{x + y} = \bar{x} \bullet \bar{y}$

The left-hand side of the equation can be viewed as the output of a NOR gate whose inputs are *x* and *y*. The right-hand side of the equation, on the other hand, is the result of first inverting both *x* and *y* and then putting them through an AND gate. These two representations are equivalent and are illustrated in Figure 4 (a).

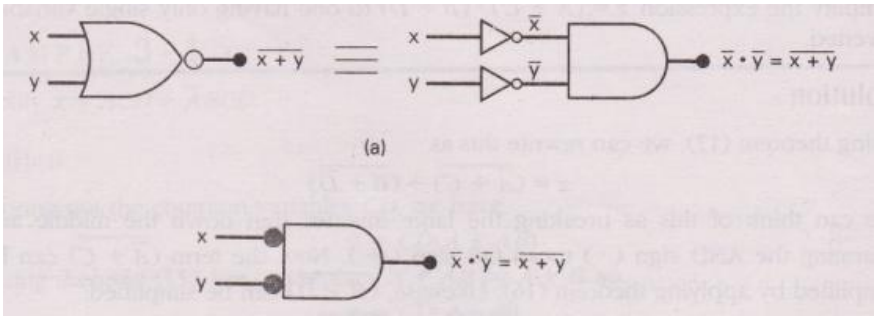


Figure.4 (a): Equivalent circuits implied by theorem (1) (b) alternative symbol for the NOR function.

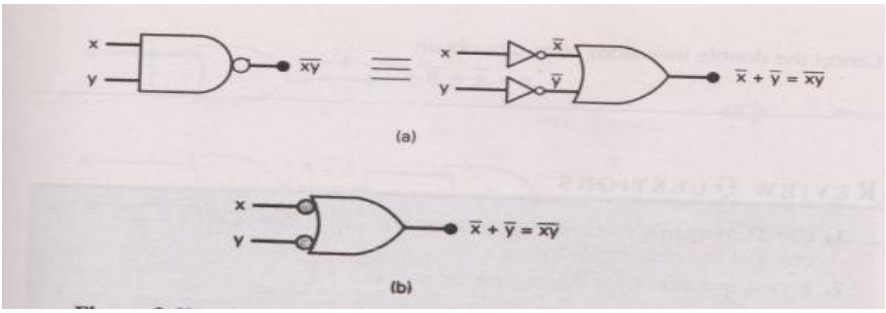


Figure 5 (a): Equivalent circuits implied by theorem (2); (b) alternative symbol for the NAND function.

That this means is that an AND gate with INVERTERs on each of its inputs is equivalent to a NOR gate. In fact, both representations are used to represent the NOR function. When the AND gate with inverted inputs is used to represent the NOR function, it is usually drawn as shown in Figure 3.8(b), where the small circles on the inputs represent the inversion operation.

Now consider theorem (2),

$x\,y = x + y$

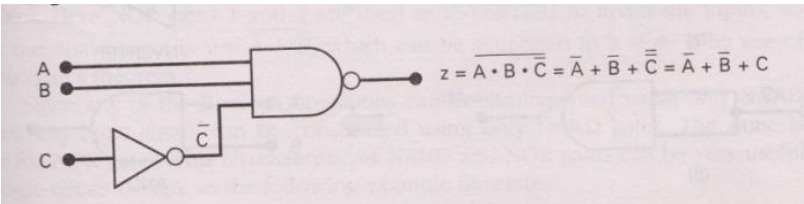
The left side of the equation can be implemented by a NAND gate with inputs *x* and *y*. The right side can be implemented by first inverting inputs *x* and *y* and then putting them through an OR gate. These two equivalent representations are show in Figure 6 (a). The OR gate with INVERTERS on each of its inputs is equivalent to the NAND gate. In fact, both representations are used to represent the NAND function. When the OR gate with inverted inputs is used to represent the NAND function.

Example

Simplify the expression $z = (\bar{A} + C) \bullet (B + \bar{D})$ to one having only single variables inverted.

Example

Determine the output expression for the circuit and simplify it using De-Morgan's theorems.



Check Your Progress 3

Note: i) Use the space below for your answer.

Exercise: Evaluate $x = \bar{A}.B + C(\overline{A.D})$ using the convention A = True and B = False.

……

……

……

3.7 Proving the Equations of Boolean Expressions by Truth Table

We can transform one Boolean expression into an equivalent expression by applying the postulates the theorems of Boolean algebra. This is important if you want to convert a given expression to a canonical form (a standardized form) or if you want to minimize the number of literals (primed or unprimed variables) or terms in an expression. Minimizing terms and expressions can be important because electrical circuits often consist of individual components that implement each term or literal for a given expression. Minimizing the expression allows the designer to use fewer electrical components and, therefore, can reduce the cost of the system. Unfortunately, there are no fixed rules you can apply to optimize a given expression. Much like constructing mathematical proofs, an individual's ability to easily do these transformations is usually a function of experience. Nevertheless, a few examples can show the possibilities:

ab + ab' + a'b = a(b+b') + a'b By P4
= a1 + a'b By P5
= a + a'b By Th4
= a + a'b + 0 By Th3
= a + a'b + aa' By P5
= a + b(a + a') By P4
= a + b1 By P5
= a + b By Th4
(a'b + a'b' + b')' = (a'(b+b') + b')' By P4
= (a' + b')' By P5
= ((ab)')' By Th8
= ab By definition of not
b(a+c) + ab' + bc' + c = ba + bc + ab' + bc' + c By P4
= a(b+b') + b(c + c') + c By P4
= a1 + b1 + c By P5
= a + b + c By Th4

Although these examples all use algebraic transformations to simplify a Boolean expression, we can also use algebraic operations for other purposes. For example, the next section describes a canonical form for Boolean expressions. Canonical forms are rarely optimal.

3.8 Principle of Duality

The duality theorem says that start with Boolean relation; you can drive another Boolean relation by

1. Changing OR operation i.e. + sing to an AND sign operation i.e. (dot) and vice versa.
2. Complement any 0 or 1 appearing in the expression i.e. replacing constant 0 and 1 by 1and o respectively.

Any statement or theorem which has this property is called dual and this quality of duality which characterizes switching algebra is known as the "Principle of duality".

For example (1), we know that

A + 0 = 0 and its dual relation is

A . 1 = A Which is true

(2) The distributed law state that

A.(B + C) = A.B + A.C

By changing each OR operation (+) and AND operation (.) we get the dual relation, which is

A + B.C = (A + B).(A + C)

The proof can be oriented by the perfect induction as shown in table 3.3.

Table 1: Algebraic Induction for Duality

A	B	C	A.B	A.C	B.C	A+B	A+C	B+C	A.B+A.C	A(B+C)	(A+B)(A+C)	A+BC
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	1	1	0	0	0	0
0	1	0	0	0	0	1	0	1	0	0	0	0
0	1	1	0	0	1	1	1	1	0	0	1	1
1	0	0	0	0	0	1	1	0	0	0	1	1
1	0	1	0	1	0	1	1	1	1	1	1	1
1	1	0	1	0	0	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1

In the above table, columns 8 and 9 are same and column 10 and 11 are same, hence the above both statements are proved.

Note: (1) The duality theorem is useful to produce a new Boolean relation.

(2) The dual of SOP from is POS from and vice versa.

3.8.1 Self Dual Function

If a Boolean function is equal to its dual function, then the given function is called Self Dual Function.

Example

$F_1 = ab + bc + ca$

Dual of $F_1 = F_1^D$ (a+b) (b+c) (c+a)

= (ac+b) (c+a)

= ac + bc +ac +ab Since $F_1 = F_1^D$, F_1 is a dual

Example

$F_2 = \Sigma m\ (0, 3, 5, 6)$

Dual of $F_2 = F_2^D = m_0^D\ m_3^D\ m_5^D\ m_6^D$

$= M_{7\text{-}0}\ M_{7\text{-}3}\ M_{7\text{-}5}\ M_{7\text{-}6}$

$= M_7\ M_4\ M_2\ M_1$

$= m_0 + m_3 + m_5 + m_6$
$= \Sigma m\ (0, 3, 5, 6) = F_2$
Hence F_2 is a self dual function.

Self Assessment Questions

1. The NAND gate output will be low if the two inputs are
(a) 00 (b) 01
(c) 10 (d) 11

2. The simplification of the Boolean expression $\overline{(\bar{A}B\bar{C})} + \overline{(A\bar{B}C)}$ is
(a) 0 (b) 1
(c) A (d) BC

3. The Boolean expression $\bar{A}.B + A.\bar{B} + A.B$ is equivalent to
(a) A + B (b) $\bar{A}$.B
(c) $\overline{A+B}$ (d) A.B

4. When simplified with Boolean Algebra (x + y)(x + z) simplifies to
(a) x (b) x + x(y + z)
(c) x(1 + yz) (d) x + yz

5. The gates required to build a half adder are
(a) EX-OR gate and NOR gate (b) EX-OR gate and OR gate
(c) EX-OR gate and AND gate (d) Four NAND gates.

6. DeMorgan's first theorem shows the equivalence of
(a) OR gate and Exclusive OR gate.
(b) NOR gate and Bubbled AND gate.
(c) NOR gate and NAND gate.
(d) NAND gate and NOT gate

3.9 Basic of Logic Gates

3.9.1 Symbols for Logic Gates

We initially examine the three simplest gates. These correspond to the logical operators AND, OR, and NOT. We have discussed the functional behavior of each of these Boolean operators. Figure 7 depicts the graphical representation of the gate that corresponds to each operator.

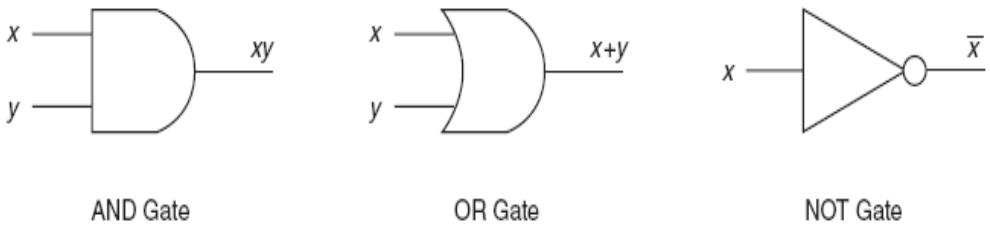


Figure 7: The Three Basic Gates

x	y	x XOR y
0	0	0
0	1	1
1	0	1
1	1	0

(a)



(b)

Figure 8: (a) The Truth Table for XOR (b) The Logic Symbol for XOR

3.9.2 Universal Gates

Two other common gates are NAND and NOR, which produce complementary output to AND and OR, respectively. Each gate has two different logic symbols that can be used for gate representation. (It is left as an exercise to prove that the symbols are logically equivalent. Hint: Use DeMorgan's Law.) Figures 9 and 10 depict the logic diagrams for NAND and NOR along with the truth tables to explain the functional behavior of each gate.

x	y	x NAND y
0	0	1
0	1	1
1	0	1
1	1	0

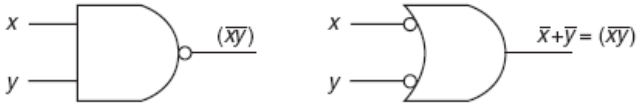


Figure 9: The Truth Table and Logic Symbols for NAND

x	y	x NOR y
0	0	1
0	1	0
1	0	0
1	1	0

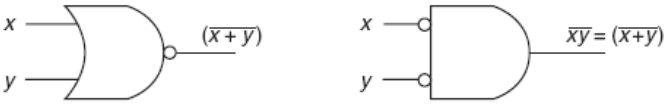


Figure 10: The Truth Table and Logic Symbols for NOR

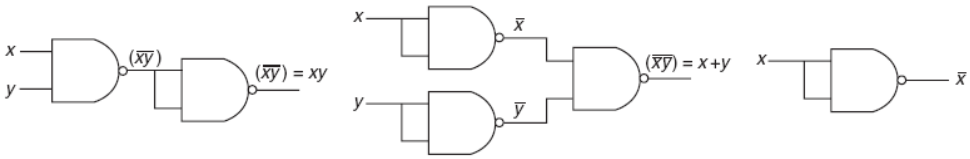


Figure 11: Three Circuits Constructed Using Only NAND Gates

3.9.3 Multiple Input Gates

In our examples thus far, all gates have accepted only two inputs. Gates are not limited to two input values, however. There are many variations in the number and types of inputs and outputs allowed for various gates. For example, we can represent the expression x + y + z using one OR gate with three inputs, as in Figure 12. Figure 13 represents the expression $x\overline{y}z$.

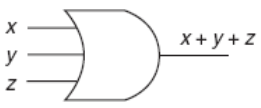


Figure 12: A Three-Input OR Gate Representing x + y + z

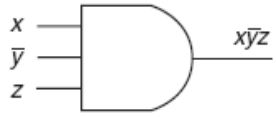


Figure 13: A Three-Input AND Gate Representing x$\bar{y}$z

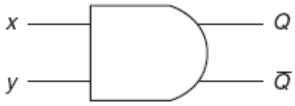


Figure 14 AND Gate with Two Inputs and Two Outputs.

3.10 Use of Logic Gates in Circuits

Upon opening a computer and looking inside, one would realize that there is a lot to know about all of the digital components that make up the system. Every computer is built using collections of gates that are all connected by way of wires acting as signal gateways. These collections of gates are often quite standard, resulting in a set of building blocks that can be used to build the entire computer system. Surprisingly, these building blocks are all constructed using the basic AND, OR, and NOT operations. In the next few sections, we discuss digital circuits, their relationship to Boolean algebra, the standard building blocks, and examples of the two different categories, combinational logic and sequential logic, into which these building blocks can be placed.

3.10.1 Digital Circuits and Their Relationship to Boolean Algebra

We have seen that a simple Boolean operation (such as AND or OR) can be represented by a simple logic gate. More complex Boolean expressions can be represented as combinations of AND, OR, and NOT gates, resulting in a logic diagram that describes the entire expression. This logic diagram represents the physical implementation of the given expression, or the actual digital circuit. Consider the function F(x,y,z) = x + $\bar{y}$z (which we looked at earlier). Figure 15 represents a logic diagram that implements this function.

We can build logic diagrams (which in turn lead to digital circuits) for any Boolean expression.

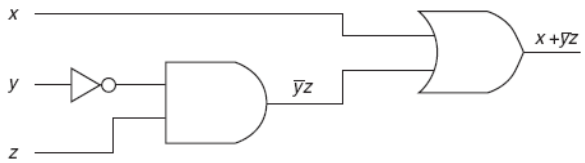


Figure 16: A Logic Diagram for F(x,y,z) = x + $\bar{y}$z.

3.10.2 Integrated Circuits

Computers are composed of various digital components, connected by wires. Like a good program, the actual hardware of a computer uses collections of gates to create larger modules, which, in turn, are used to implement various functions. The number of gates required to create these “building blocks” depends on the technology being used. Because the circuit technology is beyond the scope of this text, the reader is referred to the reading list at the end of this chapter for more information on circuit technology.

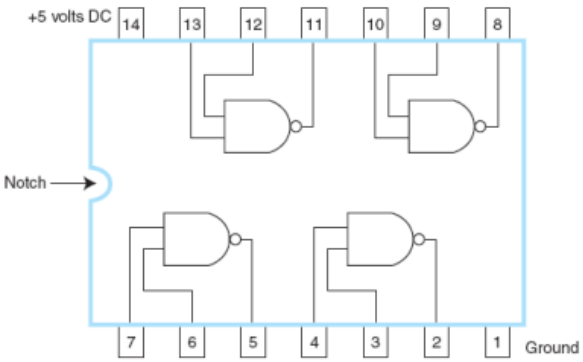


Figure 17: A Simple SSI Integrated Circuit.

3.11 Karnaugh Maps (K-Map)

One of the challenges that we encounter simplifying sums of minterms (or products of maxterms) is that it is often difficult to see which terms are logically adjacent (i.e. which terms can be simplified using the logic adjacency theorem). In the early 1950's, Maurice Karnaugh realized that this problem can be solved by writing the minterms or maxterms so that logically adjacent terms are also physically adjacent. For example, the minterms of the expression $A\bar{B} + \bar{A}\bar{B} + AB$ can be inserted in a table or map that places adjacent minterms next to each other (See Figure 18)

	$\bar{A}$	A
$\bar{B}$	$\bar{A}\bar{B}$	$A\bar{B}$
B		AB

Figure 18. Logical and Physical Adjacency.

From Figure 18 it is evident that $A\bar{B}$ is logically adjacent to both $\bar{A}\bar{B}$ and A.B. This suggests an approach to simplifying the expression.

$$A\bar{B} + \bar{A}\bar{B} + AB = \bar{A}\bar{B} + (A\bar{B} + A\bar{B}) + AB = (\bar{A}\bar{B} + A\bar{B}) + (A\bar{B} + AB) = \bar{B} + A \qquad (7.1)$$

What if there are three variables? It is easy if we use three dimensions, then each minterm can be placed in a box as shown in Figure 19.

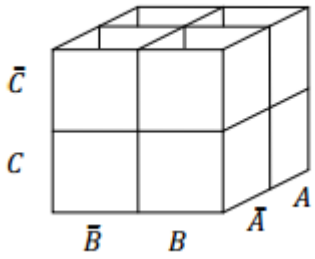


Figure 19: Logical and Physical adjacency in 3-D.

Unfortunately, this approach is hard to do on paper and does not generalize to four or more variables. Instead, we flatten Figure 19 by folding the back half of the cube to the front, giving us the map in Figure 20.

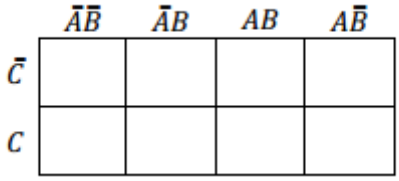


Figure 20: A Three Variable Map.

Note that the □ □ column and the □ column in Figure 20 are not physically adjacent, but were adjacent in Figure 19. We must always remember that the first and last columns of a map are logically adjacent.

Exercise: Use the map from Figure 20 to simplify the expression.

3.11.1 Reading the Karnaugh Map

The simplified logic expression we derive from a Karnaugh map is the sum of one or more products, or terms. Each term corresponds to one group on the map, so the first step to reading a Karnaugh map is to derive a term for each group. This can be done by applying the logic adjacency theorem to all the minterms in the group, but there is a shortcut. Simply write down the variable(s) that are common to all the minterms in the group. To illustrate this procedure, consider Figure 21, which shows the optimal grouping for the Karnaugh map in Figure 20.

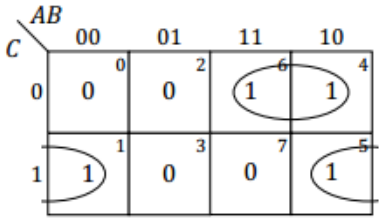
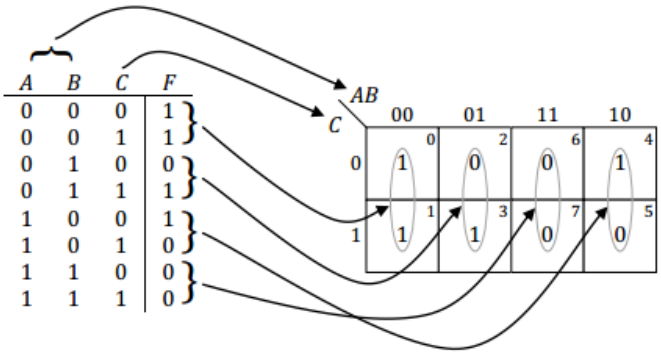


Figure: 21

In the top group of Figure 21, each minterm contains A and $\bar{C}$ (the right two columns have A=1, and the top row has C=0), so the term associated with that group is $A\bar{C}$. In the second group, each minterm contains $\bar{B}$ and C, so the term associated with that group is $\bar{B}C$. The sum of these terms gives $A\bar{C} + \bar{B}C$, , which matches the result in Example 18.

3.11.2 Filling the Karnaugh Map

Filling a Karnaugh map is straightforward if the behavior of the circuit is defined by a truth table. Recall that if the output on a row of a truth table is 1, the minterm associated with that row is included in the logic expression, but if a minterm is included in the logic expression, its cell in the Karnaugh map must also contain 1. The upshot is that if the truth table is in order, filling the Karnaugh map is a matter of copying each output into the Karnaugh map in minterm order. (See Figure 22.)



Note also from Figure 22 that the variable names on the Karnaugh map must match the inputs of the truth table, with the leftmost input first.

Check Your Progress 4

Case Study-Press Wiring

An electrical layout is needed for a hydraulic press. The press uses a 24Vdc double actuated solenoid valve to advance and retract the press. This device has a single common and two input wires. Putting 24Vdc on one wire will cause the press to advance, putting 24Vdc on the second wire will cause it to retract. The press is driven by a large hydraulic pump that requires 220Vac rated at 20A, this should be running as long as the press is on. The press is outfitted with three push buttons, one is a NC stop button, the other is a NO manual retract button, and the third is a NO start automatic cycle button. There are limit switches at the top and bottom of the press travels that must also be connected.

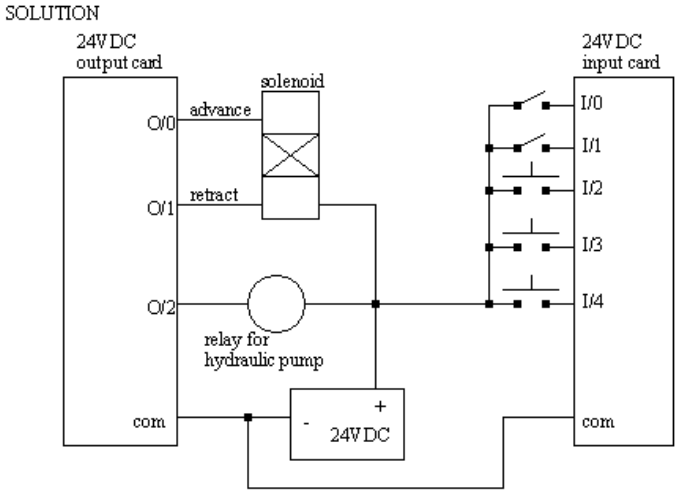


Figure: 23

The input and output cards were both selected to be 24Vdc so that they may share a single 24Vdc power supply. In this case the solenoid valve was wired directly to the output card, while the hydraulic pump was connected indirectly using a relay (only the coil is shown for simplicity). This decision was primarily made because the hydraulic pump requires more current than any PLC can handle, but a relay would be relatively easy to purchase and install for that load. All of the input switches are connected to the same supply and to the inputs.

Questions

1. Explain the need of hydraulic press.
2. What are limit switches at the top and bottom of the press?

3.12 Summary

- Karnaugh map is a graphical device used to simplify a logic equation or to convert a truth table to its corresponding logic circuit in a simple, orderly process.
- K map has been filled with 0s and 1s, the sum-of-products expression for the output X can be obtained by ORing together those squares that contain a 1.
- Looping an octet of 1s eliminates the three variables that appear in both complemented and un-complemented form.
- Minimal cost solution is a valid logic design with the minimum number of gates with the minimum number of inputs.

- Do not-care condition input-output condition that never occurs during normal operation. Since the condition never occurs, you can use an X on the Karnaugh map. This X can be a 0 or a 1, whichever you prefer.
- The operation of ternary rejection in Boolean algebra is the operation ( ) given by (abc) = a'b' + b'c' + c'a'.

3.13 Keywords

AND gate: The *AND gate* is so named because, if 0 is called "false" and 1 is called "true," the gate acts in the same way as the logical "and" operator.

Boolean algebra: It is used to help analyze a logic circuit and express its operation mathematically and it has its own unique identities based on the bivalent states of Boolean variables.

Boolean quantities: It has led to the simple rules of addition and multiplication, and has excluded both subtraction and division as valid arithmetic operations.

Boolean theorem: It is useful in, simplifying a logic expression that is, in reducing the number of terms in the expression.

DeMorgan's theorems: These are extremely useful in simplifying expressions in which a product or sum of variables is inverted

Distributive law: It states that an expression can be expanded by multiplying term by term just the same as in ordinary algebra.

***Identity*:** It is a statement true for all possible values of its variable or variables.

Inverter: A logical *inverter* sometimes called a *NOT gate* to differentiate it from other types of electronic inverter devices, has only one input. It reverses the logic state.

Logic gate: A logic gate is an elementary building block of a digital circuit. Most logic gates have two inputs and one output.

3.14 Review Questions

1. Describe the uses of logic gates in circuits.
2. Explain the principle of duality.
3. What do understand by K-map?
4. Simplify the following Boolean equation and implement it in ladder logic.

$X = A + BA + B\overline{C} + \overline{D + C}$

5. Simplify the following Boolean equation and write a ladder logic program to implement it.

$X = (A\overline{B}C + \overline{A}BC + A\overline{B}\,\overline{C} + AB\overline{C} + ABC)$

6. Given a system that is described with the following equation,

$X = A + (B \cdot (\overline{A} + C) + C) + A \cdot B \cdot (\overline{D} + \overline{E})$

A. Simplify the equation using Boolean algebra.

B. Implement the original and then the simplified equation with a digital circuit.

C. Implement the original and then the simplified equation in ladder logic.

7. Simplify the following and implement the original and simplified equations with gates and ladder logic.

$A + (\overline{B} + \overline{C} + \overline{D}) \cdot (B + \overline{C}) + A \cdot B \cdot (\overline{C} + \overline{D})$

8. Simplify the following boolean equations.

$\overline{(A + B)} \cdot \overline{(A + \overline{B})}$

$ABCD + \overline{A}BCD + ABC\overline{D} + AB\overline{C}\,\overline{D}$

9. Simplify the following Boolean equations,

$((A \cdot \overline{B}) + \overline{(\overline{B} + A)}) \cdot C + (\overline{B} \cdot C + B \cdot C)$

10. Given the Boolean expression a) draw a digital circuit and b) a ladder diagram (do not simplify), c) simplify the expression.

$$X = A \cdot \overline{B} \cdot C + \overline{(C + B)}$$

Answers of self assessment questions

1 (d) 2 (b) 3 (a) 4 (d) 5 (c) 6 (b)

3.15 Further Studies

1. https://www.cs.duke.edu/courses/cps104/spring11/lects/09-logic-design.pdf
2. Computer Fundamentals by P. K. Sinha

4

Number System

STRUCTURE

4.0 Objectives

4.1 Introduction

4.2 Digital and Analog Operations

4.3 Binary Data

4.4 Binary Number System

4.5 Decimal Number System

4.6 Octal Number System

4.7 Hexadecimal Number System

4.8 Fractional Conversion

4.9 Coding System

4.10 Summary

4.11 Keywords

4.12 Review Questions

4.13 Further Studies

4.0 Objectives

After studying this chapter, you will be able to:

- Explain the digital and analog operations
- Understand the binary data
- Explain the Number system
- Define the conversion of numbers
- Discuss the coding system
- Describe the error-detecting codes

4.1 Introduction

We are familiar with the decimal number system in which digits are 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9. The computer uses binary digits for its operation. In the binary system there are only two digits 0 and 1. The programmer feeds instruction and data in alphabets and decimal digits. But for the operation of the computer these are converted to binary bits. This chapter deals with the conversion of binary numbers to

decimal numbers and vice versa. It also deals with hexadecimal and octal system. Computer circuitry is usually designed to process hexadecimal or octal number.
Number Systems are two types:
1. Non Positional Number System
2. Positional Number System

4.2 Digital and Analog Operations

There are a variety of other methods for digitizing analog signals-too many to discuss in detail. Nevertheless, we shall take the time to examine two more techniques and the reasons for their importance. Probably the most important single reason for investigating other methods of conversion is to determine ways to reduce the conversion time. Recall that the simultaneous converter has a very fast conversion time. The counter converter is simple logically but has a relatively long conversion time. The continuous converter has a very fast conversion time once it is locked on the signal but loses this advantage when multiplexing inputs.

4.2.1 Successive Approximation

If multiplexing is required, the successive-approximation converter is most useful. The block diagram for this type of converter is shown in Figure 4.2a. The converter operates by successively dividing the voltage ranges in half. The counter is first reset to all 0s, and the MSB is then set. The MSB is then left in or taken out (by resetting the MSB flip-flop) depending on the output of the comparator. Then the second MSB is set in, and a comparison

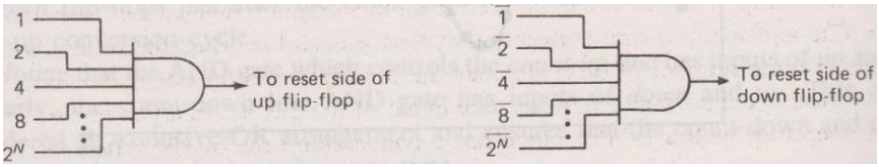


Figure 4.1: Count-limiting gates for the converter.

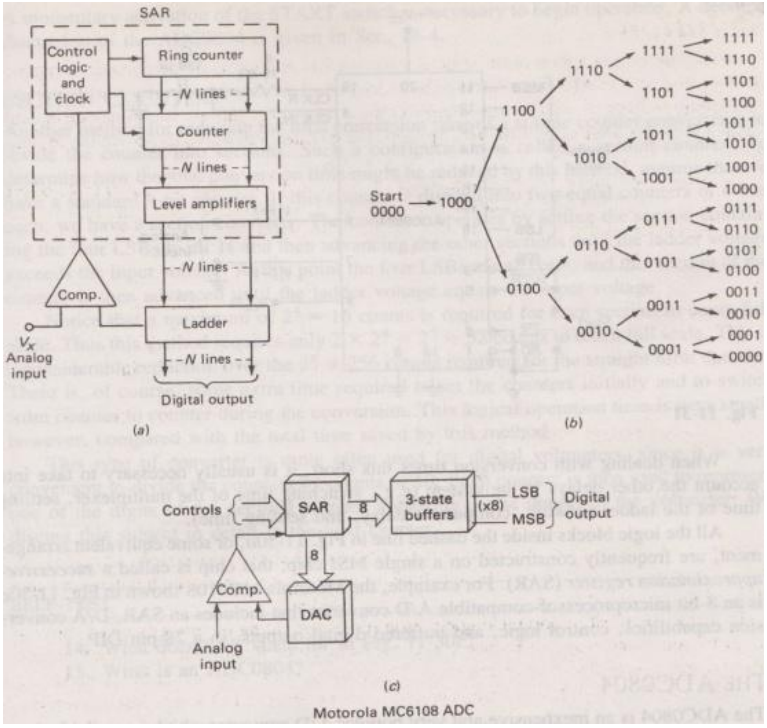


Figure 4.2: Successive approximation converter.

is made to determine whether to reset the second MSB flip-flop. The process is repeated down to the LSB,

and at this time the desired number is in the counter. Since the conversion involves operating on one flip-flop at a time, beginning with the MSB, a ring counter may be used for flip-flop selection.
The successive-approximation method thus is the process of approximating the analog log voltage by trying 1 bit at a time beginning with the MSB. The operation is shown in diagram form in Figure 4.2b. It can be seen from this diagram that each conversion takes the same time and requires one conversion cycle for each bit. Thus the total conversion time is equal to the number of bits, n, times the time required for one conversion cycle. One conversion cycle normally requires one cycle of the clock. As an example, a 10-bit converter operating with a 1-MHz clock has a conversion time of $10 * 10^{-6} = l0^{-5} = 10$ μs.

Did You Know?
Binary valued quantities terms were taken by Eric S. Raymond's "Jargon Lexicon," an indexed collection of terms, common and obscure, germane to the world of computer programming.

4.3 Binary Data

Binary data is a type of data that is represented or displayed in the binary numeral system. Binary data is the only category of data that can be directly understood and executed by a computer. It is numerically represented by a combination of zeros and ones.

4.4 Binary Number System

Given that digital logic and memory devices are based on two electrical states (on and off), it is natural to use a number system, called the binary number system, which contains only two symbols, namely 0 and 1.

4.5 Decimal Number System

The decimal system is composed of 10 numerals or symbols. These 10 symbols are 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9; using these symbols as digitals of a number, we can express any quantity. The decimal system, also called the base- 10 system because it has 10 digits, has evolved naturally as a result of the fact that man has 10 fingers.

Did You Know?
Blaise Pascal (French) invented the first adding machine in 1642. Twenty years later, an Englishman, Sir Samuel Moreland, developed a more compact device that could multiply, add, and subtract.

4.6 Octal Number System

Just as the decimal system with its ten digits is a base-ten system, the octal number system with its 8 digits, '0', '1', '2', '3', '4', '5', '6' and '7', is a base-eight system. Table 4.1 shows the weighting for the octal number system up to 3 decimal places before and 2 decimal places after the octal point (.).

Table 4**.1:** Octal weights

Weights	8^2	8^1	8^0	.	8^{-1}	8^{-2}

Did You Know?
About 1672, Gottfried Wilhelm von Leibniz (German) perfected a machine that could perform all the basic operations (add, subtract, multiply, divide), as well as extract the square root. Modern electronic digital computers still use von Liebniz's principles.

4.7 Hexadecimal Number System

The hexadecimal number system is used as an intermediary system in computers, such as are presentation of memory addresses or a representation of colours. The hexadecimal number system is also known as the base-16 number system, because each position in the number represents an incremental number with a base of 16 (see Table 4.3). For example, the first position (the furthest right) is represented as 16 the second position (one from furthest right) is represented as 16, and so forth. To determine what the actual number is in "decimal" representation, take the number that appears in the position, and multiply it by 16^x, where x is the power representation. For example, if a number appears in the furthest right position, take the number in the furthest right position and multiply it by 16 If there are multiple positions in the number (ex: 17AF), add all the results together.

Since the number system is represented in "sixteen's", there are only 10 numbers and 5 letters that can be a value in each position of the base-16 number. Below are the numbers that each position can hold:

Table 4.3: Comparing number hexadecimal to decimal values

Hexadecimal Representation	"Decimal Value"
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
A	10
B	11
C	12
D	13
E	14
F	15

Caution

Be careful to use ten different symbols like: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9 for hexadecimal number.

Self Assessment Questions

1. The decimal system is composed of …… numberals or symbols.

(a) 8 (b) 10 (c) 11 (d) 15

2. Binary system there are only two symbols or possible digit values, and ……..

(a) 1 and 2 (b) 8 and 10 (c) 0 and 1 (d) 8 and 2

3. The hexadecimal number system is also known as the base-16 number system, because each position in the number represents an incremental number with a base of 16.

(a) True (b) False

4. The octal, or base ……., number system is a common system used with computers.

(a) 0 (b) 10 (c) 11 (d) 8

5. The base of the decimal number is …….
(a) 0 (b) 10 (c) 11 (d) 8

6. In Hexadecimal number system A stands for:
(a) 12 (b) 10 (c) 11 (d) 13

7. In Hexadecimal number system F stands for:
(a) 14 (b) 10 (c) 15 (d) 13

8. When ……. is transmitted from one location to another there is always the possibility that an error may occur.
(a) data (b) number (c) hexadecimal (d) None of these.

4.8 Fractional Conversion

Numbers expressed in decimal number system are much more meaningful to us, than are numbers expressed in any other number system. This is because we have been using decimal numbers in our day-to-day life, right from childhood; however, we can represent any number in one number system in any other number system. Because the input and final output values are to be in decimal, computer professionals are often required to convert number in other systems to decimal and vice-versa. Many methods can be used to convert numbers from one base to another. A method of converting from another base to decimal, and a method of converting from decimal to another base are described here:

4.8.1 Converting From another Base to Decimal

The following steps are used to convert a number in any other base to a base 10 (decimal) number:
Step 1: Determine the column (positional) value of each digit (this depends on the position of the digit and the base of the number system).
Step 2: Multiply the obtained column values (in Step 1) by the digits in the corresponding columns.
Step 3: Sum up the products calculated in Step 2. The total is the equivalent value in decimal.

Example
$11001_2=?_{10}$
Solution:
Step1: Determine column Values

Column Number	Column (From Right) $2^0=1$
1	$2^1=2$
2	$2^2=4$
3	$2^3=8$
4	$2^4=16$

Step 2: multiply the column values by the corresponding column digits.

16	8	4	2	1
*1	*1	*0	*0	*1
16	8	0	0	1

Step 3: Sum up the products
16+8+0+0+1=25

Hence, $11001_2=25_{10}$

Example

$4706_8=?_{10}$

Solution:

Step 1: Determine column values

Column Number (From Right)	Column
1	$8^0=1$
2	$8^1=8$
3	$8^2=64$
4	$8^3=512$

Step 2: Multiply the column values by the corressponding column digits

512	64	8	1
* 4	* 7	* 0	* 6
2048	448	0	6

Step 3: Sum up the products

2048+448+0+6=2502

Hence, $4706_8=2502_{10}$

Example

$1AC_{16}=?_{10}$

Solution: $1AC_{16}=1*16^2+A*161+C*16^0$

$=1*256+10*16+12*1$

$=256+160+12$

$=428_{10}$

Example

$4052_7=?_{10}$

Solution: $4052_7 = 4*7^3+0*7^2+5*7^1+2*7^0$

$=4*343+0*49+5*7+2*1$

$= 1372+0+35+2$

1409_{10}

Example

$4052_6=?_{10}$

Solution: $4052_6=4*6^3+0*6^2+5*6^1+2*6^0$

$=4*216+0*36+5*6+2*1$

$=864+0+30+2$

$=896_{10}$

Example

$1AC_{13}=?_{10}$

Solution: $lAC_{13}=1*13^2+A*13^1+C*13^0$

$=1*169+10*13+12*1$

$=311_{10}$

4.8.2 Converting from Decimal to another Base (Division-Remainder Technique)

The following steps are used to convert a base 10 (decimal) number to a number in another base

Step 1: Divide the decimal number by the value of the new base.

Step 2: Record the remainder from Step 1 as the rightmost digit (least significant digit) of the new base number.

Step 3: Divide the quotient of the previous division by the new base.

Step 4: Record the remainder from Step 3 as the next digit (to the left) of the new base number. Repeat Steps 3 and 4, recording remainders from right to left, until the quotient becomes zero in Step 3.

Note that the last remainder, thus obtained, will be the most significant digit of the new base number.

Example

25_{10}

Solution:

Steps 1: 25/2 = 12 and remainder 1

Steps 2: 12/2 = 6 and remainder 0

Steps 3: 6/2 = 3 and remainder 0

Steps 4: 3/2 = 1 and remainder 1

Steps 5: 1/2 = 0 and remainder 1

The remainders are now arranged in the reverse order, making the first remainder the least significant digit (LSD) and the last remainder the most significant digit (MSD).

Hence, $25_{10}= 11001_2$

4.8.3 Converting from a Base Other Than 10 to another Base Other Than 10

The following steps are used to convert a number in a base other than 10, to number base other than 10:

Step 1: Convert the original number to a base (decimal) number.

Step 2: Convert the decimal number obtained in step 1 to the new base number.

Example

$545_6=?_4$

Solution: ***Step 1:*** Convert from base 6 to base 10

$545=5*6^2+4*6^1+5*6^0$

$= 5*36+4*6+5*1$

$=180+24+5$

$=209_{10}$

Step 2: Convert 209_{10} to base 4

4	209	Remainder
	52	1
	13	0
	3	1
	0	3

$209_{10}=3101_4$

Therefore, $545_6=209_{10}=3101_4$

Hence, $546_6=3101_4$

Above example illustrates the method of converting a binary number to an octal number.

4.8.4 Shortcut Method for Binary to Octal Conversion

The following steps are used in this method:

Step1: Divide the binary digits into groups of three (starting from the right)

Step 2: Convert each group of three binary digit to one octal digit. Since there are only 8 digits (0 to 7) in octal number system, 3 bits (2^3=8) are sufficient to represent any octal number in binary to decimal conversion method in this step.

Example

101110_2=?$_8$

Solution: ***Step 1:*** Divide the binary digits into groups of 3, starting the right (LSD).

101 110

Step 2: Convert each group into one digit of octal (use binary- to- decimal conversion method).

$101_2=1*2^2+0*2^1+1*2^0$ $110_2=1*2^2+1*2^1+0*2^0$

$=4+0+1$ $4+2+0$

$=5_8$ 6_8

Hence, $101110_2=56_8$

4.8.5 Shortcut Method for Octal to Binary Conversion

The following steps are used in this method:

Step 1: Convert each octal digit to a 3 digit binary number (the octal digits may be treated as decimal numbers for this conversion).

Step 2: Combine all the resulting binary groups (of 3 digits each) into a single binary number.

Example

562_8=?$_2$

Solution: ***Step 1:*** Convert each octal digit to 3 Binary digits

$5_8=101_2$

$6_8 = 110_2$

2_8-010_2

Step 2: Combine the binary groups.

$562_8 = \underset{5}{\underline{101}} \quad \underset{6}{\underline{110}} \quad \underset{2}{\underline{010}}$

Hence, $562_8 = 101110010_2$

4.8.6 Shortcut Method for Binary to Hexadecimal Conversion

The following steps are used in this method:

Step 1*:* Divide the binary digits into groups of four (starting from the right).

Step 2*:* Convert each group of four binary digits to one hexadecimal digit. Remember that hexadecimal digits 0 to 9 are equal to decimal digits 0 to 9 and hexadecimal digits A to F are equal to decimal values 10 to 15. Hence, for this step, we use binary to decimal conversion procedure and represent decimal values 10 to 15 as hexadecimal A to F.

Example:

11010011_2=?$_{16}$

Solution: ***Step 1:*** Divide the binary digit into groups of 4, starting from the right (LSD)

1101 0011

Step 2: Convert each group of 4 binary digits to 1 hexadecimal digit.

$1101_2=1*2^3+1*2^2+0*2^1+1*2^0$ $0011_2=0*2^3+0*2^2+1*2^1+1*2^0$

$=8+4+0+1$ $=0+0+2+1$

$=13_{10}$ $=3_{16}$

$=D_{16}$

Hence, $11010011_2=D3_{16}$

Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Convert the number $1001011010011_2 = ?_{16}$

……

……

……

4.8.7 Shortcut Method for Hexadecimal to Binary Conversion

The following steps are used in this method:

Step 1: Convert decimal equivalent of each hexadecimal digit to 4 binary digits.

Step 2: Combine all resulting binary groups (4 digits each) into a single binary number.

Example

$2AB_{16} = ?_2$

Solution:

Step 1: Convert decimal equivalent each hexadecimal digit to 4 binary digits.

$2_{16} = 2_{10} = 0010_2$

$A_{16} = 10_{10} = 1010_2$

$B_{16} = 11_{10} = 1011_2$

Step 2: Combine the binary groups.

$$2AB_{16} = \frac{0010}{2} \quad \frac{0010}{A} \quad \frac{1011}{B}$$

Hence, $2AB_{16} = 001010101011_2$

Table 4.4 summarizes the relationship among decimal, hexadecimal, binary, and octal number systems. Note that the maximum value for a single digit of octal (7) is equal to the maximum value of three digits of binary. The value range of one digit of octal duplicates the value range of three digits of binary. If we substitute octal digits for binary digits, the substitution is on a one-to-three basis. Hence, computers that print octal numbers instead binary, while taking memory dump, save one-third of printing space and time.

Similarly, note that the maximum value of one digit in hexadecimal is equal to the maximum value of four digits in binary. Hence, the value range of one digit of hexadecimal is equivalent to the value range of four digits of binary. Therefore, hexadecimal shortcut notation is a one-to-four reduction in space and time required for memory dump.

Table 4.4: Relationship among Decimal, Hexadecimal, Binary, and Octal number systems

Decimal	Hexadecimal	Binary	Octal
0	0	0	0
1	1	1	1
2	2	10	2
3	3	11	3
4	4	100	4
5	5	111	5
6	6	110	6
7	7	111	7
8	8	1000	10
9	9	1001	11
10	A	1010	12
11	B	1011	13
12	C	1100	14
13	D	1101	15
14	E	1110	16
15	F	1111	17
16	10	10000	20

Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Convert the value $2AFCB_{16}=?_{2}$

……..……

……..……

……

4.9 Coding System

To represent numeric, alphabetic, and special characters in a computer's internal storage and on magnetic media, we must use some sort of coding system. In computers, the code is made up of fixed size groups of binary positions. Each binary position in a group is assigned a specific value; for example 8, 4, 2, or 1. In this way, every character can be represented by a combination of bits that is different from any other combination.

In this section you will learn how the selected coding systems are used to represent data. The coding systems included are Extended Binary Coded Decimal Interchange Code (EBCDIC), and American Standard Code for Information Interchange (ASCII).

4.9.1 Extended Binary Coded Decimal Interchange Code (EBCDIC)

Using an 8-bit code, it is possible to represent 256 different characters or bit combinations. This provides a unique code for each decimal value 0 through 9 (for a total of 10), each uppercase and lowercase letter (for a total of 52), and for a variety of special characters. In addition to four numeric bits, four zone bit positions are used in 8-bit code as illustrated in figure 4.7. Each group of the eight bits makes up one alphabetic, numeric, or special character and is called a byte.

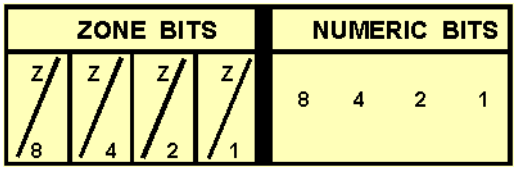
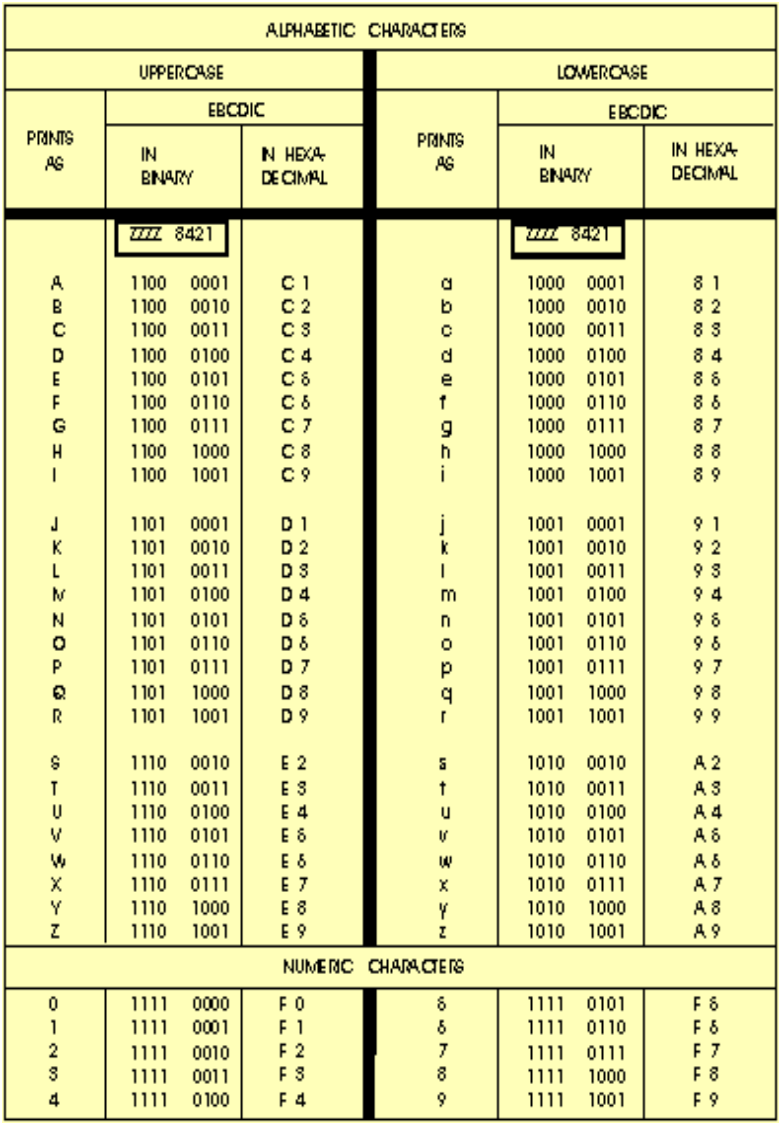


Figure 4.7: Format for EBCDIC and ASCII codes.

When you look at figure 4.7, you will notice that the four rightmost bits in EBCDIC are assigned values of 8, 4, 2, and 1. The next four bits to the left are called the zone bits. The EBCDIC coding chart for uppercase and lowercase alphabetic characters and for the numeric digits 0 through 9 is shown in figure 4.8, with their hexadecimal equivalents. Hexadecimal is a number system used with some computer systems. It has a base of 16 (0-9 and A-F). A represents 10; B represents 11; C represents 12; D represents 13; E represents 14; and F represents 15. In EBCDIC, the bit pattern 1100 is the zone combination used for the alphabetic characters A through I, 1101 is used for the characters J through R, and 1110 is the zone combination used for characters S through Z. The bit pattern 1111 is the zone combination used when representing decimal digits. For example, the code 11000001 is equivalent to the letter A; the code 11110001 is equivalent to the decimal digit 1. Other zone combinations are used when forming special characters. Not all of the 256 combinations of 8-bit code have been assigned characters. Figure 4.8 illustrates how the characters DP-3 are represented using EBCDIC.



ALPHABETIC CHARACTERS					
UPPERCASE			LOWERCASE		
PRINTS AS	EBCDIC IN BINARY (ZZZZ 8421)	EBCDIC IN HEXA-DECIMAL	PRINTS AS	EBCDIC IN BINARY (ZZZZ 8421)	EBCDIC IN HEXA-DECIMAL
A	1100 0001	C 1	a	1000 0001	8 1
B	1100 0010	C 2	b	1000 0010	8 2
C	1100 0011	C 3	c	1000 0011	8 3
D	1100 0100	C 4	d	1000 0100	8 4
E	1100 0101	C 5	e	1000 0101	8 5
F	1100 0110	C 6	f	1000 0110	8 6
G	1100 0111	C 7	g	1000 0111	8 7
H	1100 1000	C 8	h	1000 1000	8 8
I	1100 1001	C 9	i	1000 1001	8 9
J	1101 0001	D 1	j	1001 0001	9 1
K	1101 0010	D 2	k	1001 0010	9 2
L	1101 0011	D 3	l	1001 0011	9 3
M	1101 0100	D 4	m	1001 0100	9 4
N	1101 0101	D 5	n	1001 0101	9 5
O	1101 0110	D 6	o	1001 0110	9 6
P	1101 0111	D 7	p	1001 0111	9 7
Q	1101 1000	D 8	q	1001 1000	9 8
R	1101 1001	D 9	r	1001 1001	9 9
S	1110 0010	E 2	s	1010 0010	A 2
T	1110 0011	E 3	t	1010 0011	A 3
U	1110 0100	E 4	u	1010 0100	A 4
V	1110 0101	E 5	v	1010 0101	A 5
W	1110 0110	E 6	w	1010 0110	A 6
X	1110 0111	E 7	x	1010 0111	A 7
Y	1110 1000	E 8	y	1010 1000	A 8
Z	1110 1001	E 9	z	1010 1001	A 9
NUMERIC CHARACTERS					
0	1111 0000	F 0	5	1111 0101	F 5
1	1111 0001	F 1	6	1111 0110	F 6
2	1111 0010	F 2	7	1111 0111	F 7
3	1111 0011	F 3	8	1111 1000	F 8
4	1111 0100	F 4	9	1111 1001	F 9

Figure 4.8: Eight-bit EBCDIC coding chart (including hexadecimal equivalents).

Since one numeric character can be represented and stored using only four bits (8-4-2-1), using an 8-bit code allows the representation of two numeric characters (decimal digits) as illustrated in figure 4.9. Representing two numeric characters in one byte (eight bits) is referred to as packing or packed data. By packing data (numeric characters only) in this way, it allows us to conserve the amount of storage space required, and at the same time, increases processing speed.

DECIMAL VALUE	92	73
EBCDIC	10010010	01110011
BIT PLACE VALUES	84218421	8421
8421	B Y T E 1	B Y T E 2

Figure 4.9: Packed data.

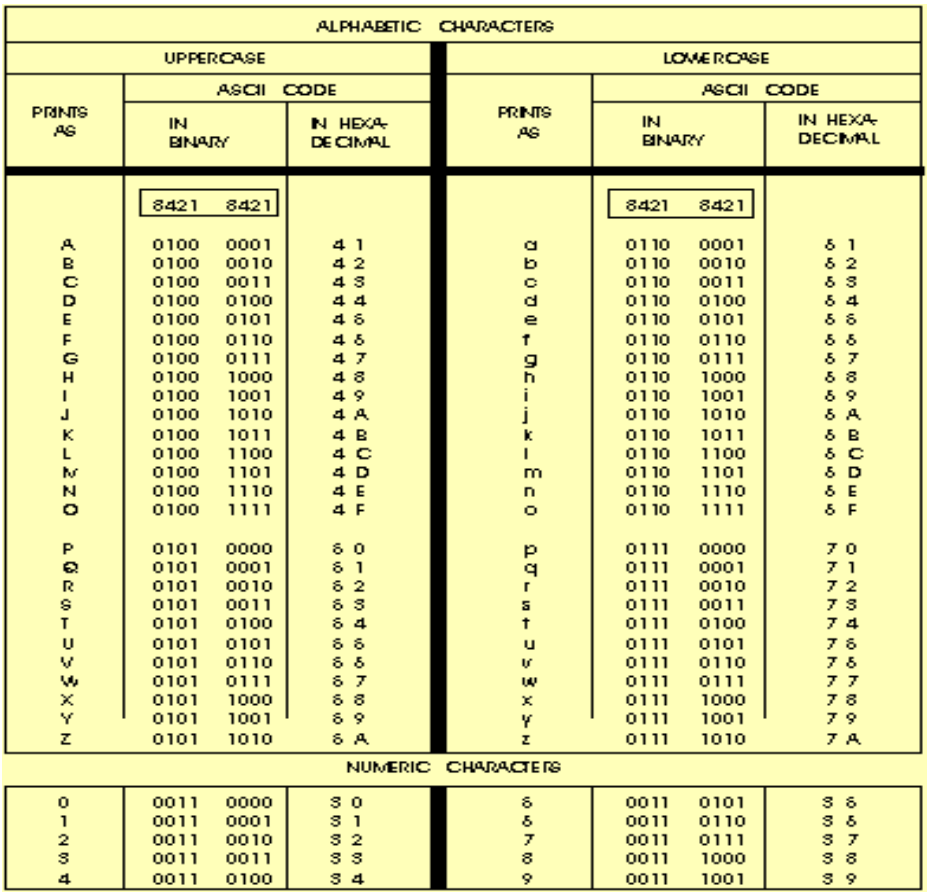
4.9.2 American Standard Code for Information Interchange (ASCII)

Another 8-bit code, known as the American Standard Code for Information Interchange (ASCII) (pronounced ASS-KEY), was originally designed as a 7-bit code. Several computer manufacturers cooperated to develop this code for transmitting and processing data. The purpose was to standardize a binary code to give the computer user the capability of using several machines to process data regardless of the manufacturer - IBM, HONEYWELL, UNIVAC, BURROUGHS, and so on. However, since most computers are designed to handle (store and manipulate) 8-bit code, an 8-bit version of ASCII was

developed. ASCII is commonly used in the transmission of data through data communications and is used almost exclusively to represent data internally in microcomputers. The concepts and advantages of ASCII are identical to those of EBCDIC. The important difference between the two coding systems lies in the 8-bit combinations assigned to represent the various alphabetic, numeric, and special characters. When using ASCII 8-bit code, you will notice the selection of bit patterns used in the positions differs from those used in EBCDIC. For example, let's look at the characters DP3 in both EBCDIC and ASCII to see how they compare.

Character	D	P	3
EBCDIC	1100 0100	1101 0111	1111 0011
ASCII	0100 0100	0101 0000	0011 0011

In ASCII, rather than breaking letters into three groups, uppercase letters are assigned codes beginning with hexadecimal value 41 and continuing sequentially through hexadecimal value 5A. Similarly, lowercase letters are assigned hexadecimal values of 61 through 7A. The decimal values 1 through 9 are assigned the zone code 0011 in ASCII rather that 1111 as in EBCDIC. Figure 4.10 is the ASCII coding chart showing uppercase and lowercase alphabetic characters and numeric digits 0 through 9.



ALPHABETIC CHARACTERS						
UPPERCASE				LOWERCASE		
PRINTS AS	ASCII CODE IN BINARY (8421 8421)		IN HEXA-DECIMAL	PRINTS AS	ASCII CODE IN BINARY (8421 8421)		IN HEXA-DECIMAL
A	0100	0001	4 1	a	0110	0001	6 1
B	0100	0010	4 2	b	0110	0010	6 2
C	0100	0011	4 3	c	0110	0011	6 3
D	0100	0100	4 4	d	0110	0100	6 4
E	0100	0101	4 5	e	0110	0101	6 5
F	0100	0110	4 6	f	0110	0110	6 6
G	0100	0111	4 7	g	0110	0111	6 7
H	0100	1000	4 8	h	0110	1000	6 8
I	0100	1001	4 9	i	0110	1001	6 9
J	0100	1010	4 A	j	0110	1010	6 A
K	0100	1011	4 B	k	0110	1011	6 B
L	0100	1100	4 C	l	0110	1100	6 C
M	0100	1101	4 D	m	0110	1101	6 D
N	0100	1110	4 E	n	0110	1110	6 E
O	0100	1111	4 F	o	0110	1111	6 F
P	0101	0000	5 0	p	0111	0000	7 0
Q	0101	0001	5 1	q	0111	0001	7 1
R	0101	0010	5 2	r	0111	0010	7 2
S	0101	0011	5 3	s	0111	0011	7 3
T	0101	0100	5 4	t	0111	0100	7 4
U	0101	0101	5 5	u	0111	0101	7 5
V	0101	0110	5 6	v	0111	0110	7 6
W	0101	0111	5 7	w	0111	0111	7 7
X	0101	1000	5 8	x	0111	1000	7 8
Y	0101	1001	5 9	y	0111	1001	7 9
Z	0101	1010	5 A	z	0111	1010	7 A
NUMERIC CHARACTERS							
0	0011	0000	3 0	5	0011	0101	3 5
1	0011	0001	3 1	6	0011	0110	3 6
2	0011	0010	3 2	7	0011	0111	3 7
3	0011	0011	3 3	8	0011	1000	3 8
4	0011	0100	3 4	9	0011	1001	3 9

Figure 4.10: Eight-bit ASCII coding chart (including hexadecimal equivalents).

At this point you should understand how coding systems are used to represent data in both EBCDIC and ASCII. Regardless of what coding system is used, each character will have an additional bit called a check bit or parity bit.

Exercise: Check Your Progress 3

Case Study-The Indian Numeral System

Although the Chinese were also using a decimal based counting system, the Chinese lacked a formal notational system that had the abstraction and elegance of the Indian notational system, and it was the Indian notational system that reached the Western world through the Arabs and has now been accepted as

universal. Several factors contributed to this development whose significance is perhaps best stated by French mathematician, Laplace: *"The ingenious method of expressing every possible number using a set of ten symbols (each symbol having a place value and an absolute value) emerged in India. The idea seems so simple nowadays that its significance and profound importance is no longer appreciated. It is simplicity lies in the way it facilitated calculation and placed arithmetic foremost amongst useful inventions."*

Emergence of Calculus

In the course of developing a precise mapping of the lunar eclipse, *Aryabhatta* was obliged to introduce the concept of infinitesimals - i.e. *tatkalika gati* to designate the infinitesimal, or near instantaneous motion of the moon, and express it in the form of a basic differential equation. *Aryabhatta's* equations were elaborated on by *Manjula* (10th C) and *Bhaskaracharya* (12th C) who derived the differential of the sine function. Later mathematicians used their intuitive understanding of integration in deriving the areas of curved surfaces and the volumes enclosed by them.

Applied Mathematics, Solutions to Practical Problems

Developments also took place in applied mathematics such as in creation of trigonometric tables and measurement units. *Yativrsabha's* work *Tiloyapannatti* (6th C) gives various units for measuring distances and time and also describes the system of infinite time measures.

In the 9th C, *Mahaviracharya* (Mysore) wrote *Ganit Saar Sangraha* where he described the currently used method of calculating the Least Common Multiple (LCM) of given numbers. He also derived formulae to calculate the area of an ellipse and a quadrilateral inscribed within a circle (something that had also been looked at by *Brahmagupta*) The solution of indeterminate equations also drew considerable interest in the 9th century, and several mathematicians contributed approximations and solutions to different types of indeterminate equations. In the late 9th C, *Sridhara* (probably Bengal) provided mathematical formulae for a variety of practical problems involving ratios, barter, simple interest, mixtures, purchase and sale, rates of travel, wages, and filling of cisterns. Some of these examples involved fairly complicated solutions and his *Patiganita* is considered an advanced mathematical work. Sections of the book were also devoted to arithmetic and geometric progressions, including progressions with fractional numbers or terms, and formulas for the sum of certain finite series are provided. Mathematical investigation continued into the 10th C. *Vijayanandi* (of Benares, whose *Karanatilaka* was translated by *Al-Beruni* into Arabic) and *Sripati* of Maharashtra are amongst the prominent mathematicians of the century.

The leading light of 12th C Indian mathematics was *Bhaskaracharya* who came from a long-line of mathematicians and was head of the astronomical observatory at Ujjain. He left several important mathematical texts including the *Lilavati* and *Bijaganita* and the *Siddhanta Shiromani*, an astronomical text. He was the first to recognize that certain types of quadratic equations could have two solutions. His Chakrawaat method of solving indeterminate solutions preceded European solutions by several centuries, and in his *Siddhanta Shiromani* he postulated that the earth had a gravitational force, and broached the fields of infinitesimal calculation and integration. In the second part of this treatise, there are several partss relating to the study of the sphere and it is properties and applications to geography, planetary mean motion, eccentric epicyclical model of the planets, first visibilities of the planets, the seasons, the lunar crescent etc. He also discussed astronomical instruments and spherical trigonometry. Of particular interest are his trigonometric equations: $\sin(a + b) = \sin a \cos b + \cos a \sin b$; $\sin(a - b) = \sin a \cos b - \cos a \sin b$;

Questions

1. What are the differences between the Indian and Chinese mathematics?
2. Why the invention of calculus was needed? How it affected the mathematics?

4.10 Summary

- A number system is a way of representing a number. Every number system has a base (the number of digits available).
- A number system does not change the value of the number, but only the manner in which it is represented.

- The decimal system is a positional-value system in which the value of a digit depends on its position.
- The decimal point separates the positive powers of 10 from the negative powers.
- The binary system is positional-value system, wherein each binary digit has its own value or weight expressed as a power of 2.
- The hexadecimal number system is known as the base-16 number system, because each position in the number represents an incremental number with a base of 16.

4.11 Keywords

Binary system: It is a positional-value system, where in each binary digit has its own value or weight expressed as a power of 2.

Digital systems: Digital systems process digital signals which can take only a limited number of values (discrete steps) usually just two values are used: the positive supply voltage (+Vs) and zero volts (0V).

Hexadecimal number system: It is known as the base-16 number system, because each position in the number represents an incremental number with a base of 16.

Number system: It is a basic counting various items. On hearing the word number all of us immediately think of the familiar decimal number system with its 10digit: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.

Octal: The octal, or base 8, number system is a common system used with computers. Because of its relationship with the binary system, it is useful in programming some types of computers.

4.12 Review Questions

1. Discuss the number system.
2. Explain the decimal number system and which decimal is equivalent of 1101011_2.
3. What is the largest number that can be represented using 8 bits?
4. Explain the method of conversion from octal number to binary number system.
5. What is the next binary after 10111_2 in the counting sequence?
6. Explain the method of conversion from hexadecimal to binary number system.
7. What is the hexadecimal representation for the binary number 100100101011?
8. What is the largest decimal value that can be represented using 12 bits?
9. Convert the following binary number system to their equivalent decimal value.

a) 11001_2=……$_{10}$

b) 1001.1001_2=……..$_{10}$

c) 10011011001.10110_2=…….$_{10}$

10. Convert from one number system to another

a) 5766_8=? $_{10}$

b) $21AFC_{16}$=? $_{10}$

c) 3552642_7=?$_{10}$

d) 3214052_6=?$_{10}$

e) $1DACC_{13}$=? $_{10}$

f) 32545_6=?$_4$

Answers for Self Assessment Questions

1. (b) 2. (c) 3. (a) 4. (d) 6. (b) 7. (b) 8. (c) 9. (a)

4.13 Further Studies

1) The Number System (Dover Books on Mathematics) by H. A. Thurston
2) Computer Fundamentals by P. K. Sinha

5

Data Representation and Binary Arithmetic

STRUCTURE

5.0 Objectives

5.1 Introduction

5.2 Data Organization

5.3 Data Representation

5.4 Binary Arithmetic

5.5 Character Representation

5.6 Checking the Result of Binary Arithmetic

5.7 Summary

5.8 Keywords

5.9 Review Questions

5.10 Further Studies

5.0 Objectives

After studying this chapter, you will be able to:

- Discuss about the data organization
- Explain the data representation
- Describe the binary arithmetic
- Explain the character representation
- Checking the result of binary arithmetic

5.1 Introduction

This chapter discusses several important concepts including the binary and hexadecimal numbering systems, binary data organization (bits, nibbles, bytes, words, and double words), signed and unsigned numbering systems, arithmetic, logical, shift, and rotate operations on binary values, bit fields and packed data, and the ASCII character set. This is basic material and the remainder of this text depends upon your understanding of these concepts.

5.2 Data Organization

In pure mathematics a value may take an arbitrary number of bits. Computers, on the other hand, generally work with some specific number of bits. Common collections are single bits, groups of four bits (called

nibbles), groups of eight bits (called bytes), groups of 16 bits (called words), and more. The sizes are not arbitrary. There is a good reason for these particular values. This section will describe the bit groups commonly used on the Intel 80x86chips.

5.2.1 Bits

The smallest "unit" of data on a binary computer is a single bit. Since a single bit is capable of representing only two different values (typically zero or one) you may get the impression that there are a very small number of items you can represent with a single bit. There are an infinite number of items you can represent with a single bit. You can represent any two distinct items. Examples include zero or one, true or false, on or off, male or female, and right or wrong. However, you are not limited to representing binary data types (that is, those objects which have only two distinct values). You could use a single bit to represent the numbers 723 and 1,245 or perhaps 6,254 and 5. You could also use a single bit to represent the colours red and blue. You could even represent two unrelated objects with a single bit. For example, you could represent the colour red and the number 3,256 with a single bit. You can represent any two different values with a single bit. However, you can represent only two different values with a single bit.

5.2.2 Nibbles

A nibble is a collection of four bits. It would not be a particularly interesting data structure except for two items: BCD (binary coded decimal) numbers and hexadecimal numbers. It takes four bits to represent a single BCD or hexadecimal digit. With a nibble, we can represent up to 16 distinct values. In the case of hexadecimal numbers, the values 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F are represented with four bits. BCD uses ten different digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) and requires four bits. In fact, any sixteen distinct values can be represented with a nibble, but hexadecimal and BCD digits are the primary items we can represent with a single nibble.

5.2.3 Bytes

Without question, the most important data structure used by the 80x86microprocessor is the byte. A byte consists of eight bits and is the smallest addressable datum (data item) on the 80x86 microprocessor. Main memory and I/O addresses on the 80x86 are all byte addresses. This means that the smallest item that can be individually accessed by an 80x 86 programs is an eight-bit value. To access anything smaller requires that you read the byte containing the data and mask out the unwanted bits. The bits in a byte are normally numbered from zero to seven using the convention shown in Figure 5.1:

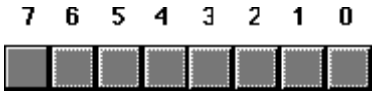


Figure 5.1: Numbered convention of byte.

Bit 0 is the low order bit or least significant bit; bit 7 is the high order bit or most significant bit of the byte. We will refer to all other bits by their number.
Note that a byte also contains exactly two nibbles. (see Figure 5.2)

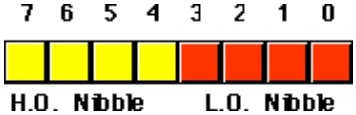


Figure 5.2: Nibbles in a byte.

5.2.4 Words

A word is a group of 16 bits. We will number the bits in a word starting from zero on up to fifteen. The bit numbering appears as shown in Figure 5.3.

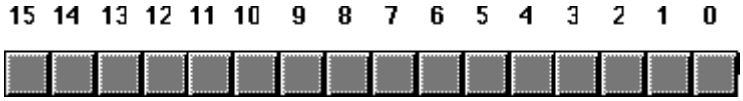


Figure 5.3: Bit numbering of a word.

Like the byte, bit 0 is the low order bit and bit 15 is the high order bit. When referencing the other bits in a word use their bit position number.

Notice that a word contains exactly two bytes. Bits 0 through 7 forms the low order byte, bits 8 through 15 forms the high order byte (see Figure 5.4):

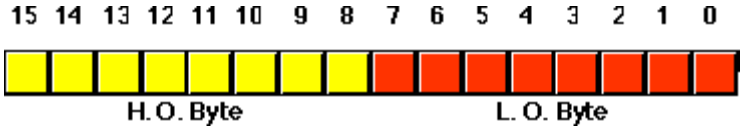


Figure 5.4: Numbering order of a word.

Naturally, a word may be further broken down into four nibbles as shown in Figure 5.5:

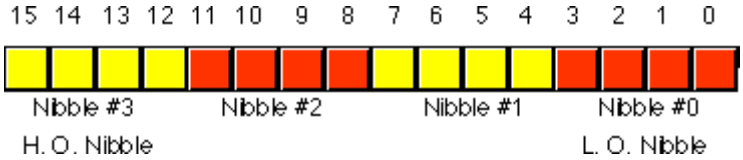


Figure 5.5: Nibbles in a word.

5.3 Data Representation

All data on digital computers is represented as a sequence of 0s and 1s. This includes numeric data, text, executable files, images, audio, and video. The ASCII standard associates a seven bit binary number with each of 128 distinct characters. The MP3 file format rigidly specifies how to encode each raw audio file as a sequence of 0s and 1s. All data are numbers, and all numbers are data.

In this section we describe how to represent integers in binary, decimal, and hexadecimal and how to convert between different representations. We also describe how to represent negative integers.

5.3.1 Number Systems

There are many ways to represent integers: the number of days in the month of October can be represented as 31 in decimal, 11111 in binary, 1F in hexadecimal, or XXXI in Roman Numerals. It is important to remember than an integer is an integer, no matter whether it is represented in decimal or with Roman Numerals.

- *Decimal numbers.* We are most familiar with performing arithmetic with the decimal (base 10) number system. This number system has been widely adopted, in large part because we have 10 fingers. However, other number systems still persist in modern society.
- *Sexagecimal numbers.* The Sumerians uses a sexagecimal (base 60) number system. We speculate that 60 were chosen since it is divisible by many integers: 1, 2, 3, 4, 5, 6, 10, 12, 15, 20, and 30. Most clocks are based on the sexagecimal system. The Babylonians inherited sexagecimal numbers from the Sumerians. They divided a circle into 360 degrees since they believed the Sun rotated around the Earth in about 360 days. Ptolemy tabulated trigonometric tables using base 360, and, even today, we still often use degrees instead of radians when doing geometry.
- *Binary numbers.* Computers are based on the binary (base 2) number system because each wire can be in one of two states (on or off).
- *Hexadecimal numbers.* Writing numbers in binary is tedious since this representation uses between 3 and 4 times as many digits as the decimal representation. The hexadecimal (base 16) number system is often used as shorthand for binary. Base 16 is useful because 16 is a power of 2, and numbers have roughly as many digits as in the corresponding decimal representation.

Did You Know?
The Indian scholar Pingala (circa 5th–2nd centuries BC) developed mathematical concepts for describing prosody, and in so doing presented the first known description of a binary numeral system.

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: Find the two's complement of 10111000.

……

……

……

5.4 Binary Arithmetic

In this section, you will see how the four basic arithmetic operations are performed inside a computer by using binary numbers. Actually, binary arithmetic is simpler to learn because binary number system deals with only two digits 0 and 1. Since all binary numbers are made up of only 0s and 1s, when arithmetic operations are performed on these numbers the results are also in 0s and 1s.

5.4.1 Addition

Binary addition is performed in the same manner as decimal addition. However, since binary number system has only two digits, the addition table for binary arithmetic is very simple and consists of only four entries. The complete table for binary addition is as follows:
0+0=0
0+1=1
1+0=1
1+1=0 plus a carry of 1 to next higher column
Carry-overs are performed in the same manner as in decimal arithmetic. Since 1 is the largest digit in binary number system, any sum greater than 1 requires a digit to be carried over. For instance, 10 plus 10 binary requires addition of two 1s in the second position. Since 1+ 1= 0 plus a carry-over of 1, the sum of 10 + 10 is 100 in binary.
By repeated use of the above rules, any two binary numbers can be added together by adding two bits at a time. The examples are illustrating the exact procedure.

Example: Add binary numbers 101 and 10in both decimal and binary forms.

Solution:	**Binary**	**Decimal**
	101	5
	+10	+2
	111	7

Example

Add binary numbers 10011 and 1001 in both decimal and binary form

Solution

Binary	Decimal
carry 11	carry 1
10011	19
+1001	+9
11100	28

In this example, carry are generated for first and second columns

5.4.2 Subtraction

The principles of decimal subtraction can as well be applied to subtraction of numbers in other number systems. It consists of two steps that are repeated for each column of the numbers. The first step is to determine if it is necessary to borrow. If the subtrahend (the lower digit) is larger than the minuend (the upper digit), it is necessary to borrow from the column to the left. It is important to note here that the value borrowed depends upon the base of the number system and is always the decimal equivalent of the base. Hence, in decimal 10 is borrowed, in binary 2 is borrowed, in octal 8 is borrowed, and in hexadecimal 16 is borrowed. The second step is simply to subtract the lower value from the upper value. The complete table for binary subtraction is as follows:

```
0 - 0 = 0
0 - 1 = 1 with a borrow from the next column
1 - 0 = 1
1 - 1 = 0
```

Observe that the only case in which it is necessary to borrow is when 1 is subtracted from 0. The examples given here illustrate the exact procedure.

Example:

Subtract 01110_2 from 1010_2 .

Solution:

```
  12
  0202
   10101
  -01110
  -------
   00111
  -------
```

In the first column (from right to left), 0 is subtracted from 1. No borrow is required in this case and the result is 1. In the second column, we have to subtract 1 from 0. A borrow is necessary to perform this subtraction. Hence, a 1 is borrowed from the third column that becomes 2 (binary 10) in the second column because the base is 2. The third column now becomes 0. Now in the second column, we subtract 1 from 2 giving 1. Since the third column is 0 due to earner borrow, we have to subtract 1 from 0 for which borrow is required. The fourth column contains a 0 and hence, has nothing to borrow. Therefore, we have to borrow from the fifth column. Borrowing 1 from the fifth column gives 2 in the fourth column and the fifth column becomes 0. Now the fourth column has something to borrow. When 1 of the 2 in the fourth column borrowed, it becomes 2 in the third column and 1 remains in the fourth column. Now in the third column, we subtract 1 from 2, giving 1. Subtraction of the fourth column is now 1 from 1, giving 0 and in the fifth column, subtraction is 0 from 0, giving 0. Hence, the result of subtraction is 00111_2. The result may be verified by subtracting 14_{10} (= 01110_2) from 21_{10} (=10101_2), which gives 7_{10} (= 00111_2).

5.4.3 Multiplication

Multiplication in binary number system also follows the same general rules as multiplication in decimal number system. However, learning binary multiplication is a trivial task because the table for binary multiplication is very short, with only four entries, instead of 100 entries necessary for decimal multiplication. The complete table for binary multiplication is as follows:

```
0 x 0 = 0
0 x 1 = 0
1 x 0 = 0
1 x 1 = 1
```

The example illustrates the method of binary multiplication. It is only necessary to copy the multiplicand, if the digit in the multiplier is 1 and to copy all 0s, if the digit in the multiplier is 0. The ease with which each step of the operation is performed is apparent.

Example

Multiply the binary numbers 1010 and 1001

Solution

```
    1010    Multiplicand
   x1001    Multiplier
  ------
    1010    Partial Product
   0000     Partial Product
  0000      Partial Product
 1010       Partial Product
 -------
 1011010    Final Product
 -------
```

5.4.4 Division

Once again, division in binary number system is very simple. As in decimal number system (or in any other number system), division by zero is meaningless. A computer deals with this problem by raising an error condition called 'Division by zero' error. Hence the complete table for binary division is as follows:

$0 \div 0 =$ Divide by zero error
$0 \div 1 = 0$
$1 \div 0 =$ Divide by zero error
$1 \div 1 = 1$

Binary division is performed in a manner similar to decimal division. The rules for binary division are:

1. Start from the left of the dividend.
2. Perform a series of subtractions, in which the divisor is subtracted from the dividend.
3. If subtraction is possible, put a 1 in the quotient and subtract the divisor from the corresponding digits of dividend.
4. If subtraction in not possible (divisor greater than reminder), record a 0 in the quotient.
5. Bring down the digit to add to the reminder digits. Proceed as before in manner similar to long division.

The example illustrates this method.

Example

Divide 100001_2 by 110_2

Solution

```
          0101  (Quotient)
      ------
110 ) 100001  (Dividend)
        110     1 <-- Divisor greater than 100, so put 0 in quotient
      ------
        1000    2 <-- Add digit from dividend to group used above
         110    3 <-- Subtraction possible, so put 1 in quotient
      ------
          100   4 <-- Remainder from subtraction plus digit from dividend
         110    5 <-- Divisor greater, so put 0 in quotient
      ------
          1001  6 <-- Add digit from dividend to group
           110  7 <-- Subtraction possible, so put 1 in quotient
      ------
           11     Remainder
      ------
```

Verify the result by dividing 33_{10} (100001_2) by 6_{10} (110_2), which gives a quotient of 5_{10} (101_2) and a remainder of 3_{10}(11_2).

5.5 Character Representation

Character data is not just alphabetic characters, but also numeric characters, punctuation, spaces, etc. Most keys on the central part of the keyboard (except shift, caps lock) are characters.

As we know with signed and unsigned integers, characters need to represent. In particular, they need to be represented in binary. After all, computers store and manipulate 0's and 1's (and even those 0's and 1's are just abstractions---the implementation is typically voltages).

Unsigned binary and two's complement are used to represent unsigned and signed integer respectively, because they have nice mathematical properties, in particular, you can add and subtract as you would expect.

However, there are not such properties for character data, so assigning binary codes for characters is somewhat arbitrary. The most common character representation is ASCII, which attends for American Standard Code for Information Interchange.

There are two reasons to use ASCII. First, we need some way to represent characters as binary numbers (or, equivalently, as bit string patterns). There is not much choice about this since computers represent everything in binary.

If you have noticed a common theme, it is that we need representation schemes for everything. However, most importantly, we need representations for numbers and characters. Once you have that (and perhaps pointers), you can build up everything you need.

The other reason we use ASCII is because of the letter "S" in ASCII, which stands for "standard". Standards are good because they allow for common formats that everyone can agree on.

Unfortunately, there is also the letter "A", which stands for American. ASCII is clearly biased for the English language character set. Other languages may have their own character set, even though English dominates most of the computing world (at least, programming and software).

5.5.1 ASCII Code (Decimal)

Table 5.4 shows the ASCII codes for the decimal numbers.

Table 5.4: ASCII codes for decimal number system

0 nul	16 dle	32 sp	48 0	64 @	80 P	96 `	112 p
1 soh	17 dc1	33 !	49 1	65 A	81 Q	97 a	113 q
2 stx	18 dc2	34 "	50 2	66 B	82 R	98 b	114 r
3 etx	19 dc3	35 #	51 3	67 C	83 S	99 c	115 s
4 eot	20 dc4	36 $	52 4	68 D	84 T	100 d	116 t
5 enq	21 nak	37 %	53 5	69 E	85 U	101 e	117 u
6 ack	22 syn	38 &	54 6	70 F	86 V	102 f	118 v
7 bel	23 etb	39 '	55 7	71 G	87 W	103 g	119 w
8 bs	24 can	40 (	56 8	72 H	88 X	104 h	120 x
9 ht	25 em	41)	57 9	73 I	89 Y	105 i	121 y
10 nl	26 sub	42 *	58 :	74 J	90 Z	106 j	122 z
11 vt	27 esc	43 +	59 ;	75 K	91 [	107 k	123 {
12 np	28 fs	44 ,	60 <	76 L	92 \	108 l	124 |
13 cr	29 gs	45 -	61 =	77 M	93]	109 m	125 }
14 so	30 rs	46 .	62 >	78 N	94 ^	110 n	126 ~
15 si	31 us	47 /	63 ?	79 O	95 _	111 o	127 del

5.5.2 ASCII Code (Hex)

Table 5.5 shows the ASCII codes for the hexadecimal number system.

This can be found by typing man ascii.

Table 5.5: ASCII codes for the hexadecimal number system

00 nul	10 dle	20 sp	30 0	40 @	50 P	60 `	70 p
01 soh	11 dc1	21 !	31 1	41 A	51 Q	61 a	71 q
02 stx	12 dc2	22 “	32 2	42 B	52 R	62 b	72 r
03 etx	13 dc3	23 #	33 3	43 C	53 S	63 c	73 s
04 eot	14 dc4	24 $	34 4	44 D	54 T	64 d	74 t
05 enq	15 nak	25 %	35 5	45 E	55 U	65 e	75 u
06 ack	16 syn	26 &	36 6	46 F	56 V	66 f	76 v
07 bel	17 etb	27 '	37 7	47 G	57 W	67 g	77 w
08 bs	18 can	28 (	38 8	48 H	58 X	68 h	78 x
09 ht	19 em	29)	39 9	49 I	59 Y	69 I	79 y
0a nl	1a sub	2a *	3a :	4a J	5a Z	6a j	7a z
0b vt	1b esc	2b +	3b ;	4b K	5b [	6b k	7b {
0c np	1c fs	2c ,	3c <	4c L	5c \	6c l	7c |
0d cr	1d gs	2d -	3d =	4d M	5d]	6d m	7d }
0e so	1e rs	2e .	3e >	4e N	5e ^	6e n	7e ~
0f si	1f us	2f /	3f ?	4f O	5f _	6f o	7f del

The difference in the ASCII code between an uppercase letter and its corresponding lowercase letter is 20_{16}. This makes it easy to convert lower to uppercase (and back) in hex (or binary).

char as a one byte int

It turns out that C supports two char types: char (which is usually considered “signed”) and unsigned char, which is unsigned.

Did You Know?

Extended Binary Coded Decimal Interchange Code (EBCDIC), the 8-bit character encoding was used mainly on IBM mainframe and IBM midrange computer operating systems in late 1950s and early1960s.

Caution

Transcoding could result in character data loss when encodings are incompatible.

Self Assessment Questions

1. The smallest “unit” of data on a binary computer is called.....

(a) nibble (b) bit (c) byte (d) word

2. A ..... is a collection of four bits.

(a) nibble (b) bit (c) byte (d) word

3. A .....consists of eight bits and is the smallest addressable data item.

(a) nibble (b) bit (c) byte (d) word

4. A word is a group of 16 bits.

(a) True (b) False

5. When arithmetic operations are performed on binary numbers the results are in 0s and 1s.

(a) True (b) False

6. Character data are just the alphabetic characters.

(a) True (b) False

5.6 Checking the Result of Binary Arithmetic

After representing the binary numbers or performing operations on them we need to check for their correctness. For example we used five binary bits to represent the magnitude of the number, and the left-most (sixth) bit as the negative-weight, or sign, bit. With five bits to represent magnitude, we have a representation range of 2^5, or thirty-two integer steps from 0 to maximum. This means that we can represent a number as high as $+31_{10}$ (011111_2), or as low as -32_{10} (100000_2). If we set up an addition problem with two binary numbers, the sixth bit used for sign, and the result either exceeds $+31_{10}$ or is less than -32_{10}, and our answer will be incorrect. Let us try adding 17_{10} and 19_{10} to see how this overflow condition works for excessive positive numbers:

. $17_{10} = 10001_2$ $19_{10} = 10011_2$
.
. 1 11 <--- Carry bits (Showing sign bits)
010001
. + 010011
. --------
. 100100

The answer (100100_2), interpreted with the sixth bit as the -32_{10} place, is actually equal to -28_{10}, not $+36_{10}$ as we should get with $+17_{10}$ and $+19_{10}$ added together. Obviously, this is not correct. What went wrong? The answer lies in the restrictions of the six-bit number field within which we are working, since the magnitude of the true and proper sum (36_{10}) exceeds the allowable limit for our designated bit field, we have an overflow error. Simply put, six places does not give enough bits to represent the correct sum, so whatever figure we obtain using the strategy of discarding the left-most "carry" bit will be incorrect.

A similar error will occur if we add two negative numbers together to produce a sum that is too low for our six-bit binary field. Let us try adding -17_{10} and -19_{10} together to see how this works (or does not work, as the case may be):

. $-17_{10} = 101111_2$ $-19_{10} = 101101_2$
.
. 1 1111 <--- Carry bits (Showing sign bits)
101111
. +101101
. --------
. 1|011100
. Discard extra bit

Final Answer: $011100_2 = +28_{10}$

The (incorrect) answer is a positive twenty-eight. The fact that the real sum of negative seventeen and negative nineteen was too low to be properly represented with a five bit magnitude field and a sixth sign bit is the root cause of this difficulty.

Let us try these two problems again, except this time using the seventh bit for a sign bit, and allowing the use of 6 bits for representing the magnitude:

. $17_{10} + 19_{10}$ $(-17_{10}) + (-19_{10})$
.
. 1 11 11 1111
. 0010001 1101111
. + 0010011 +1101101
. ---------- ----------
. 0100100_2 $1|1011100_2$
. Discard extra bit

Answers: $0100100_2 = +36_{10}$

. $1011100_2 = -36_{10}$

By using bit fields sufficiently large to handle the magnitude of the sums, we arrive at the correct answers.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Convert the number 36_{10} and 6_{10} in binary number and perform the division operation on them. Verify the answer also.

……..……

……..……

……

Exercise: Check Your Progress 3

Case Study-History of the Binary System

The Binary System of numeration is the simplest of all positional number systems. The base - or the radix - of the binary system is 2, which means that only two digits - 0 and 1 - may appear in a binary representation of any number. The binary system is of great help in the Nim-like games: Plainim, Nimble, Turning Turtles, Scoring, Northcott's game, etc. More importantly, the binary system underlies modern technology of electronic digital computers. Computer memory comprises small elements that may only be in two states - off/on - that are associated with digits 0 and 1. Such an element is said to represent one bit - binary digit.

The first electronic computer - ENIAC which stood for Electronic Numerical Integrator and Calculator - was built in 1946 at the University of Pennsylvania, but the invention of the binary system dates almost 3 centuries back. Gottfried Wilhelm Leibniz (1646-1716), the co-inventor of Calculus, published his invention in 1701 in the paper *Essay d'une nouvelle science des nombres* that was submitted to the Paris Academy to mark his election to the Academy. However the actual discovery occurred more than 20 years earlier.

According to the *Oxford Encyclopedic Dictionary*, an entry BINARY ARITHMETIC first appeared in English in 1796 in *A Mathematical and Philosophical Dictionary*.

Binary numbers are written with only two symbols - 0 and 1. For example, a = 1101. Since symbols 0 and 1 are also a part of the decimal system and in fact of a positional system with any base, there is an ambiguity as to what 1101 actually stands for. To avoid confusion, the base is often written explicitly, like in a = (1101)2 or b = (1101)10. In the decimal system, 1101 is interpreted as 1 thousand 1 hundred 1, which is just a sum of powers of 10 with coefficients that are the digits of the number. More accurately,

(1101)10 = 1·103 + 1·102 + 0·10 + 1

To represent numbers, the decimal system uses the powers of 10, whereas the binary system uses in a similar manner the powers of 2.

(1101)2 = 1·23 + 1·22 + 0·2 + 1

The numbers are different. In fact,

(1101)2 = 8 + 4 + 1 = 13 (= (13)10.)

There are several problems with using more than one number system at the same time. Should we read (1101)2 as 1 thousand 1 hundred 1 in binary? Or, after some mental calculations, just 13 without mentioning the base? The latter possibility is overtaxing and unreasonable: why to use a system other than the decimal in writing while depending on the decimal in speech? The former is inappropriate altogether for etymological reasons. We might say thousand to indicate a 1 in the fourth position from the right regardless of the base of the system in use, but this would conflict with the etymology of the word thousand, and the same is true of the word hundred. Both are related to the base 10 and no other.

Questions

1. What is the definition of binary number according to Oxford Encyclopaedic Dictionary?

2. What do you understand by positional number systems?

5.7 Summary

- There are 128 defined codes in the ASCII character set. IBM uses the remaining 128 possible values for extended character codes including European characters, graphic symbols, Greek letters, and math symbols.
- Unsigned binary and two's complement are used to represent unsigned and signed integer respectively
- Character data is at least as important as numeric data. Like numeric data, character data is represented using 0's and 1's.
- The most commonly used character representation is ASCII. Unicode is gaining popularity, and should eventually become the standard character set in programming languages.
- No bit overflow error occurs when two numbers of opposite signs are added together.
- Bit overflow occurs when the magnitude of a number exceeds the range allowed by the size of the bit field.

5.8 Keywords

Bits: The smallest "unit" of data on a binary computer is called a single bit. A single bit is capable of representing only two different values, zero or one.

Byte: It consists of eight bits and is the smallest addressable datum (data item) on the 80x86 microprocessor.

Data: It is information that has been translated into a form that is more convenient to move or process.

Nibble: It is a collection of four bits. It would not be a particularly interesting data structure except for two items: BCD (binary coded decimal) numbers and hexadecimal numbers.

Word: A word is a group of 16 bits. It represents integer values in the range 0...65,535 or -32,768...32,767.

5.9 Review Questions

1. What is the data organization? Differentiate between nibble and a word?
2. What is the data representation? How computer represent the different type of data?
3. Explain the number system and number conversion.
4. Define the negative integers? How do we negate a number?
5. What do you mean by the binary arithmetic?
6. Add the two numbers (110111_2) and (1011_2).
7. Convert the numbers 32_{10} and 3_{10} in binary and multiply them.
8. What is the character representation? How does it perform?
9. How do we check the result of binary arithmetic?
10. What do you mean by the term bit overflow?

Answers for Self Assessment Questions

1. (b) 2. (b) 3. (c) 4. (a) 5. (a) 6. (b)

5.10 Further Studies

1. http://en.wikibooks.org/wiki/A-level_Computing/AQA/Problem_Solving,_Programming,_Data_Representation_and_Practical_Exercise/Fundamentals_of_Data_Representation/Binary_arithmetic
2. Computer Fundamentals by P. K. Sinha

6

Input Devices

STRUCTURE

6.0 Objectives

6.1 Introduction

6.2 Input Device

6.3 Typing Input Devices

6.4 Pointing Input Devices

6.5 Scanning Input Devices

6.6 Audio Visual Input Devices

6.7 Summary

6.8 Keywords

6.9 Review Questions

6.10 Further Studies

6.0 Objectives

After studying this chapter, you will be able to:

- Discuss the typing and input device
- Explain the pointing input devices
- Discuss the scanning input devices
- Explain the audio visual input devices

6.1 Introduction

Input information and programs are entered into the computer through input devices such as the keyboard, disks, or through other computers via network connections or modems connected to the Internet. The input device also retrieves information off disks.

6.2 Input Device

Input devices are used to enter data and instructions into computer memory from external world. Examples include: Keyboard, Mouse, Track ball, Track Pad, Joy stick, Touch Screen, Light pen, Touch Screen, Scanner, Digital Camera, Microphone, Graphics tablet. They perform three main functions.

1. Receive data from user.

2. Convert data into binary form [The language of 0 and 1 also called the machine language. Machine language is the only language, a computer can understand directly.]

3. Provide converted data to computer memory for processing.

6.3 Typing Input Devices

6.3.1 Keyboard

Keyboard is the most familiar input device. It is most widely used input device to enter data and instructions into the computer. Keyboard has a set of keys like a typewriter. It has alphabetic keys for A,B,C…Z or a,b,c,…z. It has numeric keys like 0, 1, 2, 3,…9. It has function keys F1,F2,…F12 used to perform specific tasks. It has other keys used for editing like Delete, Backspace, Home, End, etc.



6.3.2 Touch Pad

A touch pad is a device for pointing (controlling input positioning) on a computer display screen. It is an alternative to the mouse. Originally incorporated in laptop computers, touch pads are also being made for use with desktop computers. A touch pad works by sensing the user's finger movement and downward pressure. The touch pad has since become the leading cursor controlling device in laptops. Many laptops use a trackball. IBM ThinkPad laptops use a "pointing stick" (called a Track Point) that is set into the keyboard.



Did You Know?

The first touch pad was invented by George E. Gerpheide in 1988 and the Apple computer was the first to license and use the touch pad in its power book laptops in 1994.

6.4 Pointing Input Devices

A Pointing input device is an input device which is used to control the movement of the pointer to select items on a display screen.

Examples of pointing devices Include: Mouse, Track ball, Joystick, Pointing Stick, Graphic Tablet, Touch pad or Track pad, Touch Screen and Light pen etc.

6.4.1 Mouse

Mouse is an input device used to control motion of pointer on screen. A mouse has two or three buttons called Left, Right and Middle button. Buttons are used to perform different functions.



Did You Know?

The mouse was invented in 1966 by Douglas Engelbart.

6.4.2 Track Ball

Track Ball is an input device like a mouse. It is used in Lap top computers to control motion of pointer on screen. It is a pointing device like upside down mouse. It has a ball on its upper side. This ball is moved by fingers or thumb and the pointer moves accordingly on screen.



6.4.3 Joystick

Joystick is an input device used to play games on computer.



6.4.4 Pointing Stick

A pointing stick is a pressure sensitive small nub (similar to pencil eraser) used like a joystick. It is usually found on laptops fixed between the 'G', 'H', and 'B'keys. It operates by sensing the pushing force applied by the user. The pointer on the screen moves, when user pushes pointing stick with finger. Normally, it is used with laptop computers.



6.4.5 Graphic Tablet

A graphics tablet consists of a special pen called stylus and a flat pad. The image is created on the monitor screen as the user draws it on the pad with the help of stylus (special pen).



6.4.6 Touch Screen

In ATM and in latest smart phones, touch screen is used to receive input from the user. The user enters data by the touch of his finger on different menu options or icons present on touch screen.



6.4.7 Light Pen

Light pen is an input device consisting of a special pen that is connected to a computer's monitor. The user points at the screen with the pen and selects items or chooses commands.



Self Assessment Questions

1. Input devices are used to …….and instructions into computer memory from external world.
(a). delete data (b). enter data (c). format data (d). None of these.

2. ……..data from user.
(a). Receive (b). Leave (c). Store (c). None of these.

3. Provide converted data to …….for processing.
(a).output (b). Computer memory (c).input (d). None of these.

4. A touch pad works by sensing the user's finger……..and downward pressure.
(a). movement (b). display (c).device (d). None of these.

5. A …….is an input device which is used to control the movement of the pointer to select items on a display screen.
(a).Pointing input device (b). keyboard device
(c).scanning device (d). None of these

6. Keyboard is an input device.
(a).True (b). False

7. A mouse has ……buttons.
(a).two (b).three (c). one (d).None of these

6.5 Scanning Input Devices

Data scanning devices are input devices, which are used for direct entry into the computer system from source documents. Data Scanning Devices are of many types:

6.5.1 Image Scanner

- An Image scanner is an input device.
- Which translates paper documents into an electronic format?
- Which can be stored in a computer?
- The Input document may be text, pictures and even hand written material.
- The copy of a document stored in a computer and can be displayed or printed whenever desired.

6.5.2 Flatbed Scanner

- A Flatbed Scanner is like a copy machine, which consists of box having glass plate on the top.

- The document to be scanned is placed upside down on the glass plate the light is below the glass plate and moves horizontally from left to right when activated.
- After scanning one line the light beam moves up a little and scans a next line.
- This process is repeated for all the lines.

6.5.3 Hand-held scanner

- A hand-held scanner to scan a document, the scanner is slowly dragged from one end of the document to its other end with its lights on.
- The scanner has to be dragged very carefully over the document. Otherwise the conversion of the document into its equivalent bitmap will not be correct due to this reason hand-held scanners are used only in cases where high accuracy is not needed.
- They also much cheaper as compared to flatbed scanners.
- When image scanners are used for inputting text documents, they have the following limitations Input document is stored as an image, instead of text.
- The computer cannot interpret the stored document as numbers characters and special symbols.
- The storage required for storing the document.

Caution

Be aware while scanning any document, the scanner cover must be covered properly as its rays may harm your eyes.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Search about the history of the scanners.

……..……

……..……

……

6.6 Audio Visual Input Devices

A device that captures digital photographs. Most digital cameras do not directly input data into a computer - they store photographs on memory cards. The photographs can later be transferred to a computer. A modern digital camera can capture 10 Megapixels or more per photograph - that's 10,000,000 colored dots (pixels) in every photo!



6.6.1 Video Camera

A device that captures moving images, or video. Like a digital camera, most video cameras do not directly input data into a computer – the captured movies are stored on video-tape or memory cards and later transferred to a computer. However, there are some situations where video cameras do feed video data directly into a computer: television production and video-conferencing. In these situations the video data is required in real-time.



6.6.2 Web Cam

This is a very basic video camera used to feed live video into a computer. The video data from a web cam is low quality compared to a full video camera. However it is good enough for web chats (e.g. using a messenger application such as MSN Messenger or Skype).

Usually a web cam is clipped to the top of a monitor, but many laptops now have web cams built into the edge of the screen.



Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Write few lines on CCTV camera.

……..……

……..……

……

6.6.3 Microphone

An input device that converts sound into a signal that can be fed into a computer. The signal from a microphone is usually analogue so, before it can be processed by a computer, it must be converted into digital data. An Analogue-to-Digital Convertor (ADC) is used for this (usually built into the computer's sound card). Many headphones now come with microphones to allow them to be used with chat and phone applications.



.

Caution

The signal from a microphone is analog and it must be converted into digital data before processed by a computer otherwise it will not respond properly.

Exercise: Check Your Progress 3

Case Study-A Brief History of the Personal Computer Trackball

Popular PC writings have it that a trackball is nothing more than "an upside down" mouse. This is, however, quite inaccurate revisionist history and, if anything, a mouse is an upside down trackball since development of the trackball pre-dates early development of the mouse by about 11 years. The history of both pointing devices intertwines with the development of the modern graphical user interfaces (GUI) now standard on virtually all computers.

Early Beginnings

The trackball was actually invented circa 1949 by Tom Cranston and Fred Long staff, two Canadian engineers who were working to develop a better way to coordinate target data displayed on a then state of the art radar system CRT display they were developing. Called “DATAR”, the original trackball used a 4" Canadian duck pin bowling ball and weighed several pounds. The DATAR project was eventually cancelled but the idea of using the trackball to position a cursor and select items on a computer display screen using X-Y axis orthogonal outputs survived and trackballs ultimately ended up being incorporated into radar tracking and fire control systems by a number of defense contractors worldwide. Thus the trackball was already a fairly mature military/industrial special purpose input device in the late 50's/early 60's when Doug Engle art began his groundbreaking work at SRI (Stanford Research Institute) on human-computer user interfaces and his mouse. Also, far from being an upside down trackball, the mouse upon which Engle art was finally granted his original patent in 1970 the patent was issued long after his original 1963 development work was carved from a wood block and used orthogonal direct drive wheels for the X and Y axis, not the now familiar caged ball or optical sensors (the “ball mouse” was invented by Bill English, another SRI alumnus, in 1972). Arguments over the superiority of orthogonal direct drive mice compared to ball mice and optical mice remains a persistent mouse heresy to this day.

Industrial and Military Controller

During the 1960's and 1970's trackballs continued to evolve as rugged, high cost, high precision input devices for industrial process control and military radar fire control systems. Using balls from 2"-4" in diameter, these industrial trackballs from companies like Litton, Honeywell and Measurement Systems were OEM devices designed to be mounted directly into work station desk surfaces or set on the work surface in rudimentary box enclosures.

The 4" trackball in the arcade game, “Missile Command”, introduced by Atari in 1980 and the 2-1/4" trackball in the 1981 arcade game, “Centipede”, also from Atari, are typical of the open frame panel mounted trackballs of the time. Mice of the day were similarly regarded as exotic and specialized input devices requiring custom hardware and software for integration into an application system; the notion of using a mouse or a trackball as a standalone peripheral device that just plugged into a host computer via a standardized hardware interface and using standardized software was non-existent. However, the input device world was about to change.

Graphical User Interfaces Arrive

Both trackballs and mice were fairly obscure specialty input devices in the late 1970's and early 1980's; the text based systems of the time (including early personal computers) all needed keyboards but they had no need for pointing devices. But, behind the scenes, a quiet revolution in user interfaces was brewing at the Xerox Palo Alto Research Center (PARC). There at PARC the Alto I, Alto II and Star computers introduced the “graphical user interface” (GUI), a new user paradigm that combined Engle art’s research on human computer interaction, English's improved ball mouse, and graphical as opposed to text based display hardware to form a software environment which required both a keyboard and a pointing device for effective use. While these Xerox systems and their early GUIs are historical footnotes of technical interest but little commercial success, their influence on Microsoft and Apple, the emerging twin giants of personal computing changed the face of computing forever and, input device wise, mice were destined to become the big winners!

Apple took its first run at a GUI based computer in 1983 with the Lisa, a desktop workstation with an integrated mouse and GUI application software suite. That same year Microsoft introduced its first mouse (actually bundled with the Microsoft Word processor). While neither the Lisa nor the first Microsoft mouse achieved great commercial success, both companies persevered and Apple's introduction of the Macintosh in 1984 and the introduction of Windows by Microsoft in 1985 catapulted pointing devices from obscure special purpose input device to computer mass market necessity because it was simply not possible to effectively use a GUI based computer system without a pointing device. Thus, there was instantly created a market for a pointing device to be shipped with each new GUI equipped computer system plus an aftermarket for millions more pointing devices for existing systems being upgraded to use the new GUI software. And, since both Apple and Microsoft followed Xerox's lead by using the mouse as their pointing

device of choice, mice really hit the big time. Had the roles been reversed with the trackball as the pointing device of choice for use with these early GUI systems, the odds are good that today the trackball would be the familiar mass market product and the mouse the obscure option.

Mice Roll Out

The driving impulse of the widespread switch from text to GUI based user interfaces led to an explosion in mouse manufacturing. Apple was now shipping a mouse with every Macintosh computer and names now familiar in the PC mouse business like Microsoft, Logitech and mouse Systems began producing ever higher quantities of retail aftermarket mice in a variety of designs to capitalize on the growing acceptance of Microsoft Windows. As with any new technology, there were endless debates about how many buttons were optimum, what was the best encoder technology, who had the mouse with the best ergonomic shape, optimum cord length etc., many of which assumed religious issue intensity. In addition, there were significant technology issues involved in developing the software protocols, drivers and hardware interfaces required to seamlessly integrate the pointing device into the universe of software applications as they made the painful transition from text based to graphical user interface. Like any manufactured product subject to the immutable laws of high volume manufacturing, mice were refined and manufacturing costs reduced in a steeply declining classic Boston Consulting Group experience curve. A mouse for the early xerox workstations circa 1975 cost around about $400.00. By 1986 a Microsoft or Logitech PC mouse for Windows 1.0 had a retail street price of around $89. By the time Windows 3.1 launched in 1992 these "name brand" mice were down under $50. By the launch of Windows 95 in 1995 generic mice were selling for under $10.00. After the production of hundreds of millions of units plastic OEM mice can be now be had for under $1.00 each.

Trackballs Struggle to Keep Up With the Rodents

Contrary to popular opinion, mice were never regarded as the best possible input device for use with graphical user interfaces. The ground breaking ergonomics research in the original "Handbook of Human Computer Interaction", first published in 1986 by Elsevier Press, rated six graphical input devices (mouse, trackball, joystick, graphics tablet, light pen, and touch screen) according to their effectiveness as tested in a variety of user input pointing tasks with graphical user interfaces. In overall ratings the graphics tablet ranked first and the trackball ranked a close second; mice finished third. (Joysticks, lightens and touch screens were judged to be specialized input devices effective for specific classes of pointing applications but not good selections for use as general purpose computer system pointing devices.)

Technical merit is great but in consumer markets, marketing muscle counts for more. (Remember the videotape format war between Bemata and VHS?) Unfortunately, trackballs did not have anybody with anything like the marketing muscle of Microsoft or Apple to drive them into the market place. While every computer soon arrived including a mouse as standard, there was no such OEM volume/cost driver for trackballs. As a result, early PC and Macintosh trackballs were all aftermarket products manufactured in much lower volume and as a result much more expensive than mice - a situation which persists to this day. Early PC trackballs, particularly in the Windows markets, were also handicapped by the lack of early standards for mouse drivers, display attributes and hardware interfaces which often resulted in frustrating software compatibility issues. By the late 80's there were a only a handful of trackballs like the Micro Speed Fast TRAP, the Kensington Turbo Mouse and the Honeywell Lynx available for use on PCs and Macintoshes and their shipment volume was miniscule compared to the volume of mice being shipped.

Questions

1. Explain the role of trackball in mouse.
2. How useful trackball in industrial and military controller.

6.7 Summary

- The work of a computer is characterized by an input-process-output model in which a program receives input from an input device.

- Users employ a variety of input devices to interact with the computer, but most user interfaces today are based upon a keyboard and a mouse pointing input device.
- A keyboard consists of a number of switches and a keyboard controller. The keyboard controller is built into the keyboard itself.
- Keyboard scan codes are sent to the computer via a serial port.
- Digital camera records and stores photographic images in digital form that is fed to a computer for viewing and printing.

6.8 Keywords

***Data scanning devices*:** it is input devices used for direct data entry into a computer system from source documents

***Graphics tablet*:** it consists of a special pen called stylus and a flat pad. The image is created on the monitor screen as the user draws it on the pad with the help of stylus (special pen).

***Input device*:** it is an electromechanical device that accepts data from outside world and translates them into form of a computer can interpret.

***Keyboard devices*:** the most commonly used input devices today. They allow data entry into a computer system by pressing asset of keys (labeled buttons) neatly mounted on a keyboard connected to the computer system.

***Pointing stick*:** It is a pressure sensitive small nub (similar to pencil eraser) used like a joystick.

6.9 Review Questions

1. What is an input device? Name some commonly used input devices.
2. What is a mouse? Explain how it is used to notify the system of a particular user choice out of a given set of choices on a monitor's screen.
3. What are keyboard devices?
4. What are the difference between keyboards and touch pad?
5. What is a joystick? How it is different from a trackball? Write some uses of a joystick.
6. What are data scanning devices?
7. What are the difference between pointing stick and graphic tablet?
8. What is a light pen?
9. What are the difference between flatbed scanner and hand-held scanner?
10. What are audio visual input devices?

Answers for Self Assessment Questions

1. (b) 2.(a) 3.(b) 4.(a) 5.(a) 6. (a)

6.10 Further Studies

1. http://en.wikibooks.org/wiki/A-level_Computing/AQA/Computer_Components,_The_Stored_Program_Concept_and_the_Internet/Hardware_Devices/Input_devices
2. Computer Fundamentals by P. K. Sinha

7

Output Devices

STRUCTURE

7.0 Objectives

7.1 Introduction

7.2 Output Devices

7.3 Soft Copy Verses Hard Copy Output

7.4 Monitor

7.5 Printers

7.6 Electrostatic Technique

7.7 Special Purpose Output Equipments

7.8 Summary

7.9 Keywords

7.10 Review Questions

7.11 Further Studies

7.0 Objectives

After studying this chapter, you will be able to:

- Discuss the output devices
- Differentiate the soft and hard copy output
- Explain the monitor
- Discuss the electrostatic technique
- Explain the special purpose output equipments

7.1 Introduction

An Output Device is a piece of hardware that is used receiving information from a computer. Some devices are both an input and output device and can transfer information in one of two directions depending on the current situation. A disk drive is an example of an input/output device. Some devices can be used only for output - for example, a printer and a monitor. This is the most commonly used output device. It displays what you have typed or otherwise entered in to the computer on the screen in front of you, monitors can be monochrome (black and white, black and green or black and amber), or colour. Colour monitors come in various types. Each type has a different number of colours to use and a different quality of picture (resolution). The higher the resolution, the better the quality of the pictures on the screen. Resolution is measured in pixels. The screen is divided into a grid. Each square on the grid is a pixel.

7.2 Output Devices

An output device is an electromechanical device that accepts data from a computer and translates them into a form suitable for use by outside world (users). Several output devices are available today. They can be broadly classified into following categories:

1 Monitors
2 Printers
3 plotters
4. Screen image projector
5. Voice response systems

7.3 Soft Copy Verses Hard Copy Output

Output devices generate computer output that can be broadly classified into following two types:

*Soft-copy output***:** A soft-copy output is not produced on a paper or some material that can be touched and carried for being shown to others. They are temporary in nature.

Example: Content displayed on a terminal screen, or words spoken out by a voice response system are S011-copy output.

Hard-copy output: A hard-copy output is produced on a paper or some material that can be touched and carried for being shown to others. They are permanent in nature and can be kept in paper files or can be looked later when the person is not using a computer.

Example: Output produced on paper by printers or plotters are hard-copy output.

7.4 Monitor

Monitors are the most popular output devices used today for producing soft-copy output. They display the generated output on a television like screen. A monitor is associated usually with a keyboard and together they form a video display terminal (VDT). A VDT (often referred to as just terminal) is the most popular input/output (I/O) device used with today's computers. It serves as both an input and output device. The keyboard is used for input to a computer and the monitor is used to display the output from the computer. The name "terminal" comes from the fact that a terminal is at the terminus or end of a communication path.

Two basic types of monitors used today are cathode-ray-tube (CRT) and LCD (Liquid Crystal Display) flat-panel. CRT monitors work much like a television screen and are used with non-portable computer systems. On the other hand, LCD flat-panel monitors are thinner, lighter and are used commonly with portable computer systems like notebook computers. With gradual reduction in price of LCD flat panel monitors, they are used increasingly with non-portable desktop computer systems also. They are also preferred because they occupy less table space.

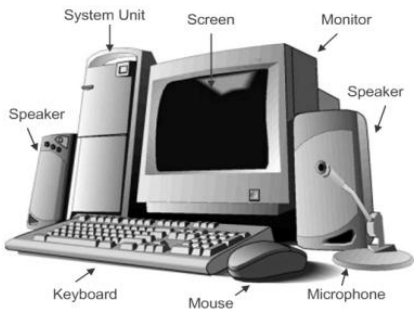


Figure 7.1: A video display terminal consists of a monitor and a keyboard.

Did You Know?

The first standalone LCD displays appeared in the mid 1990s.

7.4.1 LCD Monitor

LCD stands for *Liquid Crystal Display*, referring to the technology behind these popular flat panel monitors. An LCD monitor is distinguishable from a traditional CRT monitor as the latter has a bulky footprint with a depth of several inches and a weight 13 - 23 kilograms or more, while LCDs are commonly 1 - 3 inches (2.5 - 7.5 cm) thick and weigh less than 4.5 k. LCD displays were used on laptop computers before the technology improved enough to make the jump to desktop monitors. A LCD monitor consists of five layers: a backlight, a sheet of polarized glass, a "mask" of colored pixels, a layer of liquid crystal solution responsive to a wired grid of x, y coordinates, and a second polarized sheet of glass. By manipulating the orientations of crystals through precise electrical charges of varying degrees and voltages, the crystals act like tiny shutters, opening or closing in response to the stimulus, thereby allowing degrees of light that have passed through specific colored pixels to illuminate the screen, creating a picture.



Figure 7.2: LCD monitor.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Select a monitor and connect to the CPU and power supply.

……..……

……..……

……

Did You Know?

Liquid crystal was discovered by the Austrian botanist Fredreich Rheinizer in 1888

7.5 Printers

Printers are the most popular output devices used today for producing hard-copy output.

Different types of printers are described below:

7.5.1 Dot-Matrix Printers

Dot-matrix printers are character printers that print one character at a time. They form characters and all kinds of images as patterns of dots. Figure 7.3 shows how various types of characters can be formed as patterns of dots. A Dot matrix printer has a print head that moves horizontally (left to right and right to left) across the paper. Print contains an array of pins that can be activated independent of each other to extend and strike against an inked -n to form patterns of dots on the paper. To print a character, the printer activates the appropriate set of pins as the print head moves horizontally. For faster printing, many dot-matrix printers print both ways - while the printer head moves from left to right and while it moves from right to left, on return. Such method is called bidirectional printing.



Figure 7.3: A dot-matrix printer.

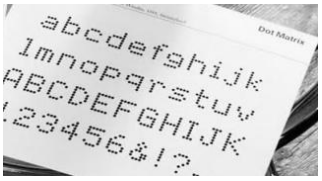


Figure 7.4: Formation of characters as patterns of dots.

7.5.2 Inkjet Printers

Inkjet printers are character printers that form characters and all kinds of images by spraying small drops of ink on paper. The print head of an inkjet printer contains up to 64 tiny nozzles that can be heated up selectively in a few microseconds by an integrated circuit resistor. When the resistor heats up, the ink near it vaporizes and is ejected through the nozzle making a dot on the paper placed in front of the print head. To print a character, the printer selectively heats the appropriate set of nozzles as the print head moves horizontally. Figure 7.5 shows an inkjet printer.



Figure 7.5: An inkjet printer

Inkjet printers are slower than dot-matrix printers are with printing speeds ranging from 40 to 300 characters per second. Typically, an inkjet printer is more expensive than a dot-matrix printer. They are preferred if speed of printing is not an important factor.

7.5.3 Drum Printers

Drum printers are line printers that print an entire line at a time. Recall that dot matrix and inkjet printers print one character at a time. Figure 7.6 shows the print mechanism of a drum printer. It consists of a solid cylindrical drum with characters embossed (raised characters) on its surface in the form of circular bands. Each band consists of all the printing characters supported by the printer in its character set. Total number of bands is equal to the maximum number of characters (print positions) that can be printed on a line. Hence, a drum printer with 132 characters per line and supporting a character set of 96 characters will have altogether 12,672 (132x96) characters embossed on its surface. In addition to the drum, the printer has a set of hammers (one opposite each band) mounted in front of the drum in a manner that an inked ribbon and paper can be placed between the hammers and the drum. The total number of hammers is equal to the total number of bands on the drum, which is also equal to the total print positions per line. The drum rotates at a high speed. A character is printed at a desired print position by activating the appropriate hammer when the character embossed on the band at the print position passes below it. Hence, the drum would have to complete one full revolution to print each line of output. This means that not all characters on a line are printed at exactly the same time, but the time required to print an entire line is so fast that it appears as if one line is printed at a time. The drum of a drum printer is expensive and cannot be changed often. Hence, drum printers can print a pre-defined set of characters only in a pre-defined style embossed on the drum. Drum printers, therefore, do not have ability to print any shape of characters, different sizes of print, and graphics such as charts and graphs. Drum printers are impact printers because they print by hammering on a paper and inked ribbon against the -3cters embossed on the drum. Hence, they can be used to produce multiple copies by using carbon paper or its equivalent. Due to impact printing, drum printers are noisy in operation and often use a cover to reduce the noise level. Printing speeds of drum printers are in the range of 300 to 2000 lines per minute.

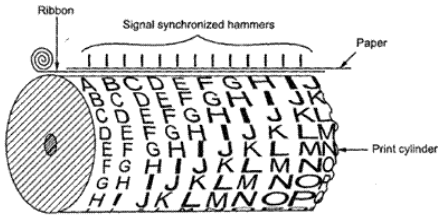


Figure 7.6: Drum printer

7.5.4 Chain/Band Printers

Chain/band printers are line printers that print one line at a time. Figure 7.7 shows the print mechanism of a chain/band printer. It consists of a metallic chain/band on which all characters of the character set supported by the printer are embossed. A standard character set may have 48, 64, or 96 characters. In order to enhance printing, peed, the characters in the character set are embossed several times on the chain/band. For example, the .chain/band of a 64 character set printer may have four sets of 64 characters each embossed on it. In this case, the .chain/band will have altogether 256 (64x4) characters embossed on it. The printer has a set of hammers mounted in front of the chain/band in a manner that an inked ribbon and paper placed between the hammers and chain/band. The total number of hammers is equal to the total number of print positions. Therefore, a printer supporting 132 print positions will have 132 hammers. The chain/band rotates at a high speed. A character is printed at a desired print position by activating the appropriate hammer when the character embossed on the chain/band passes below it. Since the character set is repeated several times on the chain/band, it is not necessary to wait for the chain/band to make a complete revolution to position the desired character in the correct print position.

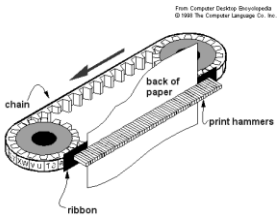


Figure 7.7: Printing mechanism of a chain/band printer.

Chain/band printers are impact printers because they print by hammering on a paper and inked ribbon against the characters embossed on the chain/band. Hence, they can be used to produce multiple copies by using carbon paper or its equivalent. Due to impact printing, chain/band printers are noisy in operation and often use a cover to reduce the noise level. Printing speeds of chain/band printers are in the range of 400 to 3000 lines per minute.

7.5.5 Laser Printers

Laser printers are page printers that print one page at a time. Main components of a laser printer are a laser beam source, a multi-sided mirror, a photoconductive drum, and toner (tiny particles of oppositely charged ink). To print a page of output, the laser beam is focused on the electro statically charged drum by the spinning multi-sided mirror. The mirror focuses the laser beam on the surface of the drum in a manner to create the patterns of characters/images to be printed on the page. Since the drum is photoconductive, a difference in electric charge is created on those parts of the drum surface that are exposed to the laser beam.



Figure 7.8: A laser printer.

The paper with heat and pressure to generate the printed output. The drum is then rotated and cleaned with a rubber blade to remove the toner sticking to its surface to prepare the drum for next page printing. Figure 7.8 shows a laser printer. Laser printers produce very high quality output because they form characters by very tiny ink particles. The most -common laser printers have resolution of 600 dpi (dots per inch), whereas some high-end laser printers have resolution of 1200 dpi. Due to their high resolution, these printers give excellent graphics art quality. Since laser printers produce printed output as patterns generated by the laser beam, they can print any shape of characters that a program can describe. They, therefore, can print many special characters, different sizes of print; also have the ability to print graphics such as charts and graphs. Laser printers are non-impact printers. Hence, they are very quiet in operation. Being of non-impact type, they cannot be used to produce multiple copies of a document in a single printing. Although most laser printers can print in black color only because they have a single toner-drum arrangement. A colour laser printer like color inkjet printer four color toners, each with a drum of its own. The usage of drums and toners are same as earlier, but drum is charged in such a way that together they produce the desired color output. Laser printers are faster in printing speed than other printers discussed earlier. Low speed laser printers can print 4 pages per minute. High-speed laser printers are also available. Owing to their better print quality and -printing speed, laser printers are more expensive than other types of printers.

Caution

In laser printer, the toner composed of oppositely charged ink particles, sticks to the drum and then fused permanently on it.

Self Assessment Questions

1. RAM chips plugged into special sockets on the motherboard are known as

(a). SIMMs (b). SINNs

(c). SIMNs (d). None of these

2. Megabytes (MB) is equal to 1,048,576 (2^{20})

(a).True (b) False

3. The act of retrieving data from a storage location is called……

(a). memory write operation (b). memory read operation

(c) memory operation (d) None of these

4. ……. storage system consists of a rotating disk coated with a thin metal.

(a). Memory disc (b).Optical disk

(c). Hard disk (d). All of these

5. LED indicator stands for……

(a).Light-emitting diode (b).Level-emitting diode

(c).Low-emitting diode (d).None of these

6. A …….. is a pointing device similar to a roller-ball mouse.

(a).Electronic pen (b).Trackball

(c) mouse (d) None of these

7. A color inkjet printer comes with two ink cartridges-black and ……..

(a).Tricolor (b).Blue, green, white

(c) Orange (d). None of these

7.6 Electrostatic Technique

An electrostatic technique converts text information into spoken sentences. To produce speech, these devices combine basic sound units called phonemes. From a given text information, sequence of words are combined into phonemes, amplified, and output through a speaker attached to the system. Electrostatic technique are still in their infancy because currently they can produce only limited unique sounds with only limited vocal inflections and phrasing. However, they are very useful in a wide range of applications.

1. For reading out text information to blind persons. For example, a recently published book may be scanned using a scanner, converted into text using OCR software, and read out to blind persons using a speech synthesizer.
2. For allowing those persons who cannot speak, to communicate effectively. For example, a person with this type of disability simply types the information and the electrostatic technique converts it into spoken words.
3. For translation systems that convert an entered text into spoken words in a selected language. For example, a foreigner coming to India may enter a text he/she wants to communicate to an Indian, and the electrostatic technique converts it into spoken words of the selected Indian language.

Did You Know?

The laser printer was invented at Xerox in 1969 by researcher Gary Starkweather, who had an improved printer working by 1971 and incorporated into a fully functional networked printer system by about a year late.

7.7 Special Purpose Output Equipments

7.7.1 Computer Output Microfilm (COM)

Computer output microfilm is a system that converts stored data directly from electronic media. It is a 16 mm roll film type which is blipped and the frames have fixed sizes and equidistant from each other.

7.7.2 Document Printers (Multi Functional Devices)

A product or device that has multiple functions. An example of this might be a printer that also makes copies, faxes, and scans. Another example is a CD or DVD that might contain multiple applications on the same disk; this may be a Mac and PC version of the same software or media meant to be played on more than one platform. Also called multi function product (MFP), all-in-one. A typical MFP may act as a combination of some or all of the following devices:

- Printer
- Scanner
- Photocopier
- Fax
- E-mail

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Write down notes on how printer evolution takes place.

……..……

……..……

……

7.7.3 Plotter

We learnt earlier that dot matrix, inkjet, and laser printers are capable of producing graphics output. However, engineering design applications like architectural plan of a building, design of mechanical

components of an air craft or a car, etc., often require high-quality, perfectly-proportioned graphic output on large sheets. The various: types of printers discussed above are not suitable for meeting this output requirement of such applications. A special type of output device, called plotters, is used for this purpose. Plotters are an ideal output device for architects, engineers, city planners, and others who need to routinely generate high-precision, hard-copy, graphic output of widely varying sizes. Two commonly used types of plotters are drum plotter and flatbed plotter.

Drum Plotter

In a drum plotter, the paper on which the design is to be made is placed over a drum that can rotate in both clockwise and anti-clockwise directions to produce vertical motion. The mechanism also consists of one or more penholder mounted perpendicular to the drum's surface. The pen(s) clamped in the holder(s) can move left to or right to left to produce horizontal motion. A graph-plotting program controls the movements of the drum and pen (s) that is, under computer control, the drum and pen(s) move simultaneously to draw designs and graphs; sheet placed on the drum. The plotter can also annotate the designs and graphs so drawn by using the pen to draw characters of various sizes. Since each pen is program selectable, pens having ink of different colors can be mounted in different holders to produce multi-colored designs. Figure 7.9 shows a drum plotter.

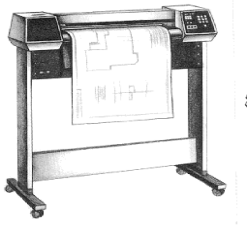


Figure 7.9: A drum plotter.

Flatbed Plotter

A flatbed plotter plots a design or graph on a sheet of paper spread and fixed over a rectangular flatbed table. In this type of plotter, normally the paper does not move and the pen holding mechanism provides all types of motions necessary to draw complex designs and graphs. That is under computer control, the pen(s) in the required manner to draw design and graphs on the sheet placed on the flatbed table. The plotter can also annotate the designs and graphs so drawn by using the pen to draw characters of various sizes. Provision is also there to mount more than one pen in the pen(s) holding mechanism.

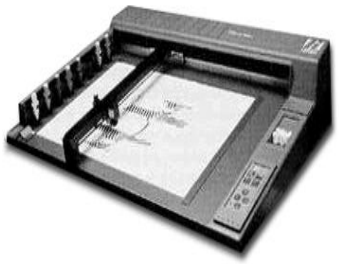


Figure 7.10: A flatbed plotter.

Exercise: Check Your Progress 3

Case study-Germany's Heidelberg uses OKI's C9600 for color proofing

For several years now, Germany's Heidelberger Druckmaschinen AG (Heidelberg) has been using OKI Printing Solutions' C9600 Series printer as a color proofing machine. The C9600（MICROLINE 9600 PS)was first used in tandem with Heidelberg's Color Proof Pro Station in 2006 at the German company's headquarters. Heidelberg is the world's leading solutions provider for print media. It has a strong reputation for driving the efficiency and profitability of its customers through the delivery of advanced

printing techniques and solutions. As well as optimizing prepress, press and post press procedures, Heidelberg offers a workflow system and associated applications which can manage the complete process, encompassing everything from production flow to administrative tasks. The priest workflow, system specifically designed for print shops, is capable of delivering efficiencies across the entire process from preliminary preparation to actual production workflow.

Facing up to the Challenges

Historically, Heidelberg had used large-scale inkjet printers for color proofing. This approach presented the company with many challenges. Typically, it was not possible to use paper for the proofing process that was the same quality as that used in the final print as the printed color would be too thick on the page. Usually, the preparation of bulk samples was difficult and unwieldy since they were generally printed on such large paper sizes that the edges would need to be trimmed and folded so that the finished image could be clearly seen once the binding process had taken place. This was especially important when the proof was needed as a reference at the post press stage. As result, there was increasing demand for page printers that would be convenient for simple proofing. The issue was that these kinds of solutions are typically unsuitable for color proofing.

Finding a Solution

The ability to use the C9600 Series printer, incorporating Heidelberg's RIP, the world's most advanced color-conversion processor, has changed this situation completely. The solution directly controls MICROLINE's colors when processing proofs. It is also equipped with lithography technology. As a result, it can be used for simple proofing and can also be deployed at the actual printing site. In addition, print shops can now review work in the same environment throughout the process since many designers are also OKI users. The solution's processing speed is attractive, too. OKI's printers have a reputation for fast processing anyway. Now it seems that the engine itself is faster still. In addition, it is very easy to bind-print with OKI's page printers, enabling rapid delivery to customers. And spotting imposition mistakes is easier than ever. In Japan, OKI's printers have been widely used as standardized machines at graphic designers' and prepress sites. In fact, they are known as the first PostScript printers in Japan. And now, following a further refinement in technology and performance. they are helping to support seamless and smooth digital workflows.

In the future, the ability of print shops, production firms and project owners and to install and use the same brand of printer will help to enable remote color proofing, ultimately resulting in even greater efficiency levels.

Questions

1. What was the main problem with **OKI's C9600 printers?**
2. **How the OKI's C9600 printer's problem was solved?**

7.8 Summary

- The plotters annotate the designs and graphs so drawn by using the pen to draw characters of various sizes.
- Monitors are the most popular output device used today for producing soft-copy output.
- The drum and pen(s) move simultaneously to draw designs and graphs; sheet placed on the drum.
- Plotters are ideal output device for architects, engineers, city planners, and others who need to routinely. Generate high-precision, hard-copy, graphic output of widely varying sizes.
- Digitizers are used commonly in the area of computer aided design (CAD) by architects and engineers to design cars, buildings, medical devices, robots, mechanical parts, etc.

7.9 Keywords

Flush memory: Storage technology recall that flashes is non-volatile, electrically erasable programmable read only memory (EEPROM) chips.

Hot-spot of graphics: The graphics cursor, irrespective of its size and shape, has a pixel-size point that is considered the point of reference to decide where the cursor is positioned on the screen. This point is called hot-spot of the graphics cursor.

LCD: LCD stands for Liquid Crystal Display, referring to the technology behind these popular flat panel monitors.

Plotter: A special type of output device, called plotters, is purpose. plotters are an ideal output device for architects, engineers, city planners, and others who need to routinely generate high-precision, hard-copy, graphic output of widely varying sizes.

Terminal: A monitor is associated usually with a keyboard and together they form a video display terminal (VDT). A VDT (often referred to as just terminal) is the most popular input/output (I/O) device used with today's computers.

7.10 Review Questions

1. What are peripheral devices?
2. Why I/O devices are necessary for a computer system?
3. Differentiate between a flatbed and drum plotter.
4. What is a printer? What are the commonly used types of printers?
5. What is differentiating between soft-copy and hard-copy outputs?
6. What are the inkjet printers?
7. What is the electrostatic technique?
8. What are the special purpose output equipments?
9. What are the main limitations of primary storage of a computer system?
10. What are point-and-draw devices? Name some commonly used point-and-draw devices.

Answers for Self Assessment Questions

1. (a) 2. (a) 3. (b) 4. (b) 5. (a) 6. (b) 7. (a).

7.11 Further Studies

1. http://en.wikibooks.org/wiki/A-level_Computing/AQA/Computer_Components,_The_Stored_Program_Concept_and_the_Internet/Hardware_Devices/Output_devices
2. Computer Fundamentals by P. K. Sinha

8

Central Processing Unit

STRUCTURE

8.0 Objectives

8.1 Introduction

8.2 Central Processing Unit

8.3 Arithmetic and Logic Unit (ALU)

8.4 Control Unit

8.5 Registers

8.6 Instruction Set

8.7 Processor Speed

8.8 Summary

8.9 Keywords

8.10 Review Questions

8.11 Further Studies

8.0 Objectives

After studying this chapter, you will be able to:

- Understand the central processing unit
- Discuss the concept of arithmetic and logic unit
- Define and declare control unit
- Explain the registers
- Understand the instruction set
- Define and declare processor speed

8.1 Introduction

Central processing unit (CPU) is an older term for processor and microprocessor, the central unit in a computer containing the logic circuitry that performs the instructions of a computer's programs. It otherwise known as a processor is an electronic circuit that can execute computer programs. Both the miniaturization and standardization of CPUs have increased their presence far beyond the limited application of dedicated computing machines. Modern microprocessors appear in everything from automobiles to mobile phones. The clock rate is one of the main characteristics of the CPU when performance is concerned. Clock rate is the fundamental rate in cycles per second (measured in hertz, kilohertz, megahertz or gigahertz) for the frequency of the clock in any synchronous circuit. A single clock

cycle (typically shorter than a nanosecond in modern non-embedded microprocessors) toggles between a logical zero and a logical one state.
Engineers are working hard to push the boundaries of the current architectures and are constantly searching for new ways to design CPUs that tick a little quicker or use slightly less energy per clock. This produces new cooler CPUs that can run at higher clock rates.
Scientists also continue to search for new designs that allow CPUs to run at the same or at a lower clock rate as older CPUs, but which get more instructions completed per clock cycle.
The clock rate of a processor is only useful for providing comparisons between computer chips in the same processor family and generation. Clock rates can be very misleading since the amount of work different computer chips can do in one cycle varies. Clock rates should not be used when comparing different computers or different processor families. Rather, some kind of software benchmarks should be used. Smartphone are equipped with more advanced embedded chipsets that can do many different tasks depending on their programming.
The performance of the CPU that is at the core of the chipset is vital for the daily user experience and the general computing performance of the Smartphone. People tend to use the clock rate of the main CPU to compare the performance of competing end products. But as we already pointed out, the clock rate of a processor is only useful for providing performance comparisons between computer chips in the same processor family and generation. For all other purposes, it's best to use software benchmarks for determining comparative performance.

8.2 Central Processing Unit

The CPU can be thought of as the "brains" of the device. It is responsible for fetching the correct instruction for execution, decoding that instruction, and then executing that instruction.
The CPU sometimes works in conjunction with the ALU to complete the execution of the instruction (in arithmetic and logical operations).
The CPU controls the program memory address bus, the data memory address bus, and accesses to the stack. The main unit inside the computer is the CPU. This unit is responsible for all events inside the computer. It controls all internal and external devices, performs arithmetic and logic operations. The operations a microprocessor performs are called the instruction set of this processor. The instruction set is "hard wired" in the CPU and determines the machine language for the CPU. The more complicated the instruction set is, the slower the CPU works. Processors differ from one another by the instruction set. If the same program can run on two different computer brands they are said to be compatible. Programs written for IBM compatible computers will not run on Apple computers because these two architectures are not compatible.
There is an exception to this rule. Apple Macintosh with a program Soft PC loaded can run programs written for IBM PC. Programs like Soft PC make one CPU "pretend" to be another. These programs are called software emulators. Although software emulators allow the CPU to run incompatible programs they severely slow down the performance.
The CPU is composed of several units

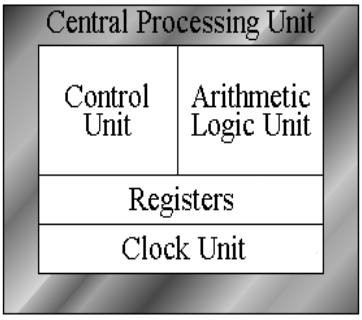


Figure 8.1 *A simplified diagram of the*

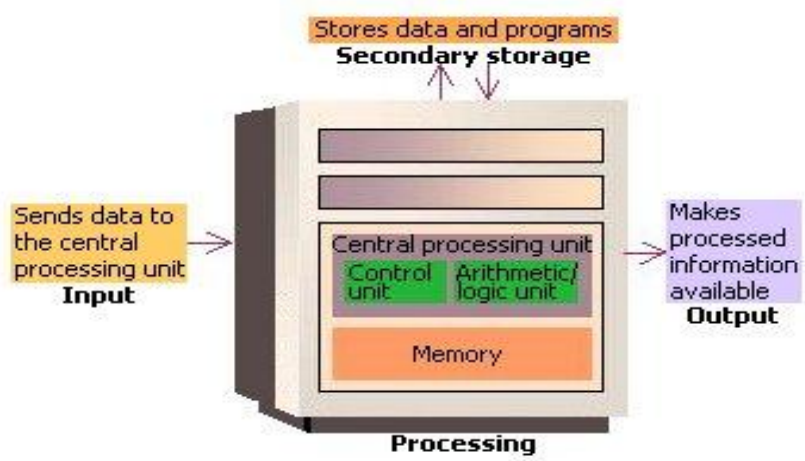


Figure 8.2: The Central Processing Unit

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: Prepare the block diagram of CPU.

……..……

……..……

……

8.3 Arithmetic and Logic Unit (ALU)

An arithmetic-logic unit (ALU) is the part of a computer processor (CPU) that carries out arithmetic and logic operations on the operands in computer instruction words. In some processors, the ALU is divided into two units, an arithmetic unit (AU) and a logic unit (LU). Some processors contain more than one AU for example, one for fixed-point operations and another for floating-point operations. (In personal computers floating point operations are sometimes done by a floating point unit on a separate chip called a numeric coprocessor.)

Typically, the ALU has direct input and output access to the processor controller, main memory (random access memory or RAM in a personal computer), and input/output devices. Inputs and outputs flow along an electronic path that is called a bus. The input consists of an instruction word (sometimes called a machine instruction word) that contains an operation code (sometimes called an "op code"), one or more operands, and sometimes a format code. The operation code tells the ALU what operation to perform and the operands are used in the operation. (For example, two operands might be added together or compared logically.) The format may be combined with the op code and tells, for example, whether this is a fixed-point or a floating-point instruction. The output consists of a result that is placed in a storage register and settings that indicate whether the operation was performed successfully. (If it is not, some sort of status will be stored in a permanent place that is sometimes called the machine status word.)

In general, the ALU includes storage places for input operands, operands that are being added, the accumulated result (stored in an accumulator), and shifted results. The flow of bits and the operations performed on them in the subunits of the ALU is controlled by gated circuits. The gates in these circuits are controlled by a sequence logic unit that uses a particular algorithm or sequence for each operation code. In the arithmetic unit, multiplication and division are done by a series of adding or subtracting and shifting operations. There are several ways to represent negative numbers. In the logic unit, one of 16 possible logic operations can be performed such as comparing two operands and identifying where bits do not match.

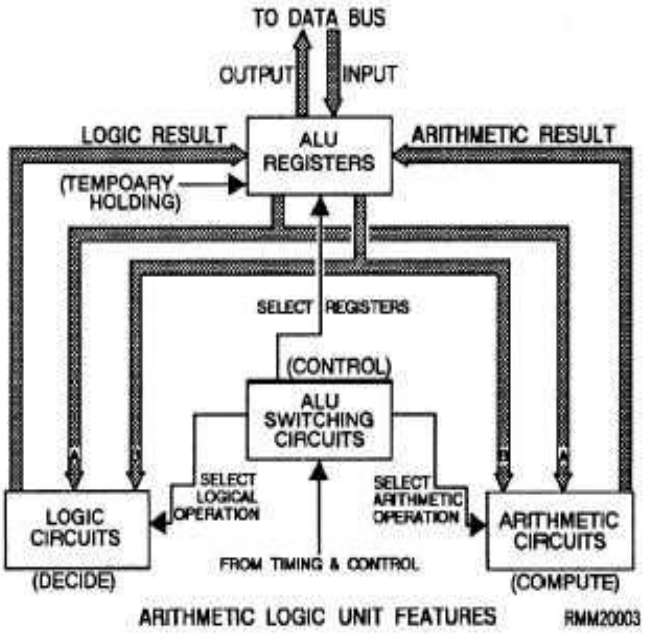


Figure 8.3 Block diagram of an ALU

The design of the ALU is obviously a critical part of the processor and new approaches to speeding up instruction handling are continually being developed.

The ALU is the part where actual computations take place. It consists of circuits which perform arithmetic operations (e.g. addition, subtraction, multiplication, division) over data received from memory and capable to compare numbers.

Did You Know?

The first single chip CPU was the Intel 4004 invented in Nov 1971, a 4-bit processor meant for a calculator. It processed data in 4 bits, but its instructions were 8 bits long. Program and data were separate.

8.4 Control Unit

It is a typical component of the CPU that implements the microprocessor instruction set. It extracts instructions from memory and decodes and executes them, and sends the necessary signals to the ALU to perform the operation needed. Control Units are either *hardwired* (instruction register is hardwired to rest of the microprocessor) or *micro-programmed*.

The control unit directs and controls the activities of the internal and external devices. It interprets the instructions fetched into the computer, determines what data, if any, are needed, where it is stored, where to store the results of the operation, and sends the control signals to the devices involved in the execution of the instructions.

8.5 Registers

Registers is special, high-speed storage area within the CPU. For example, if two numbers are to be multiplied, both numbers must be in registers, and the result is also placed in a register. (The register can contain the address of a memory location where data is stored rather than the actual data itself.)

The number of registers that a CPU has and the size of each (number of bits) help determine the power and speed of a CPU. For example a 32-bit CPU is one in which each register is 32 bits wide. Therefore, each CPU instruction can manipulate 32 bits of data.

Usually, the movement of data in and out of registers is completely transparent to users, and even to programmers. Only assembly language programs can manipulate registers. In high-level languages, the compiler is responsible for translating high-level operations into low-level operations that access registers.
While performing these operations the ALU takes data from the temporary storage area inside the CPU named registers. Registers are a group of cells used for memory addressing, data manipulation and processing. Some of the registers are general purpose and some are reserved for certain functions. It is a high-speed memory which holds only data for immediate processing and results of this processing. If these results are not needed for the next instruction, they are sent back to the main memory and registers are occupied by the new data used in the next instruction.

Caution
Before processing data, all data must be represented in a register.

Self Assessment Questions

1. ALU stands for
(a) Arithmetic Logic Unit (b) Array Logic Unit
(c) Application Logic Unit (d) None of above

2. VGA stands for
(a) Volatile Graphics Array (b) Visual Graphics Array
(c) Video Graphics Array (d) Video Graphics Adapter

3. CPUs with a small instruction set, fixed-length instructions, and reduced references to memory to retrieve operands are said to employ RISC (Reduced Instruction Set Computer) architecture.
(a). True (b) False

4. Clock speed is measured in
(a) Megahertz (MHz) (b) Megahertz (MHz) or gigahertz (GHz).
(c) Gigahertz (GHz). (d) None of these.

5. The number of registers that a CPU has and the size of each (number of bits) help determine the power and speed of a CPU.
(a). True (b) False

8.6 Instruction Set

An instruction set is a collection of machine code instructions which enable a processor to carry out all its tasks. A high level program is compiled into a number of machine instructions one high level instruction (for example, a print command issued by a programming software like, Visual Basic) must be changed into a series of low-level machine instructions (or machine language) which perform that single operation. The total number of instructions in an instruction set can be up to 200. Processors use one of two instruction set types.
Most personal computers use a CISC (Complex Instruction Set Computer) architecture, in which the CPU supports as many as two hundred instructions. An alternative architecture, used by many workstations and also by some personal computers, is RISC (Reduced Instruction Set Computer), which supports fewer instructions. Reduced Instruction Set Computer (RISC) is a type of instruction set for a microprocessor that recognizes a relatively limited number of instructions. Until the mid-1980s, the tendency among computer manufacturers was to build increasingly complex CPUs that had ever-larger sets of instructions.
However, a number of computer manufacturers decided to reverse this trend by building crus capable of executing only a very limited set of instructions. One advantage of RISC is that it can execute their

instructions very fast because the instructions are so simple. Another, perhaps more important, advantage is that RISC chips require fewer transistors, which makes them cheaper to design and produce. To some extent, CISC and RISC implementations are becoming more and more alike. Many of today's RISC chips support as many instructions as yesterday's CISC chips. And today's CISC chips use many techniques formerly associated with RISC chips.

Superscalar describes a microprocessor design that makes it possible for more than one instruction at a time to be executed during a single clock cycle. In a superscalar design, the processor or the instruction compiler is able to determine whether an instruction can be carried out independently of other sequential instructions, or whether it has a dependency on another instruction and must be executed in sequence with it. The processor then uses multiple execution units to simultaneously carry out two or more independent instructions at a time. Superscalar design is sometimes called "second generation RISC."

8.7 Processor Speed

The CU and ALU perform operations at incredible speed. These operations are usually synchronized by a built-in electronic clock (known as system clock) that emits millions of regularly spaced electric pulses per second (known as clock cycles). Instructions are fetched, decoded, and executed at proper intervals with intervals timed by a specific number of clock cycles. One cycle is the time it takes to perform one operation such as moving a byte of data from one memory location to another. Normally, several clock cycles are required to fetch, decode, and execute a single program instruction. Shorter the clock cycle, faster the processor. Hence, the speed at which an instruction is executed is related directly to a computer's built-in clock speed, which is the number of pulses produced per second. This clock speed is measured in megahertz (MHz) or gigahertz (GHz), where mega means 106, giga means 109, and hertz means cycles per second. Over the years, clock speeds of processors have increased steadily. Most of today's popular personal computers have clock speeds in the 500 MHz (0.5 GHz) to 4000 MHz (4.0 GHz) range. However, the effort to build processors with even faster ratings continues. With all other parameters being equal (although they never are), a CPU operating at 500 MHz can process data five times as fast as one operating at 100 MHz.

8.7.1 Types of Processors

CISC Processors

One of the earlier goals of CPU designers was to provide more and more instructions in the instruction set of a CPU to ensure that the CPU supports more functions directly. This makes it easier to translate high-level language programs to machine language and ensures that the machine language programs run more effectively. Of course, every additional instruction in the instruction set of a CPU requires the necessary hardware circuitry to handle that instruction, adding more complexity to the CPU's hardware circuitry. Another goal of CPU designers was to optimize the usage of expensive memory. To achieve this, the designers tried to pack more instructions in memory by introducing the concept of variable-length instructions such as half-word, one-and-half-word, etc. For example, an operand in an immediate instruction needs fewer bits and can be designed as a half-word instruction. Additionally, CPUs were designed to support a variety of addressing modes (discussed later in this chapter during the discussion of memory). CPUs with large instruction set, variable-length instructions, and a variety of addressing modes are said to employ CISC (Complex Instruction Set Computer) architecture. Since CISC processors possess so many processing features, they make the job of machine language programmers easier. However, they are complex and expensive to produce. Most personal computers of today use CISC processors.

RISC Processors

In early 1980s, some CPU designers realized that many instructions supported by a CISC-based CPU are rarely used. Hence, an idea evolved that the design complexity of a CPU can be reduced greatly by implementing only bare minimum basic set of instructions and some of the more frequently used instructions in the hardware circuitry of the CPU. Other complex instructions need not be supported in the instruction set of the CPU because they can always be implemented in software by using the basic set of

instructions. While working on simpler CPU design, the designers also came up with the idea of making all the instructions of uniform length so that the decoding and execution of all instructions becomes simple and fast. Furthermore, to speed up computation and to reduce the complexity of handling a number of addressing modes they decided to design all the instructions in such a way that they retrieve operands stored in registers in CPU rather than from memory. These design ideas resulted in producing faster and less expensive processors. CPUs with a small instruction set, fixed-length instructions, and reduced references to memory to retrieve operands are said to employ RISC (Reduced Instruction Set Computer) architecture. Since RISC processors have a small instruction set, they place extra demand on programmers who must consider how to implement complex computations by combining simple Instructions. However, RISC processors are faster for most applications, less complex, and less expensive to produce CISC processors because of simpler design.

EPIC Processors

The Explicitly Parallel Instruction Computing (EPIC) technology breaks through the sequential nature of conventional processor architectures by allowing the software to communicate explicitly to the processor when operations can be done in parallel. For this, it uses tighter coupling between the compiler and the processor. It enables the compiler to extract maximum parallelism in the original code and explicitly describe it to the processor. Processors based on EPIC architecture are simpler and more powerful than traditional CISC or RISC processors. These processors are mainly targeted to next-generation, 64-bit, high-end server and workstation market (not for personal computer market).

Did You Know?

The microprocessors have begun in 1971 when Intel Company has let out the first microprocessor i4004.

Exercise: Check Your Progress 2

Case Study-Motorola 68000

The 68000 grew out of the MACSS (Motorola Advanced Computer System on Silicon) project, begun in 1976 to develop an entirely new architecture without backward compatibility. It would be a higher-power sibling complementing the existing 8-bit 6800 line rather than a compatible successor. In the end, the 68000 did retain a bus protocol compatibility mode for existing 6800 peripheral devices, and a version with an 8-bit data bus was produced. However, the designers mainly focused on the future, or forward compatibility, which gave the M68K platform a head start against later 32-bit instruction set architectures. For instance, the CPU registers are 32 bits wide, though few self-contained structures in the processor itself operate on 32 bits at a time. The MACSS team drew heavily on the influence of minicomputer processor design, such as the PDP-11 and VAX systems, which were similarly microcoded.

In the mid 1970s, the 8-bit microprocessor manufacturers raced to introduce the 16-bit generation. National Semiconductor had been first with its IMP-16 and PACE processors in 1973-1975, but these had issues with speed. The Intel 8086 in 1977 quickly gained popularity. The decision to leapfrog the competition and introduce a hybrid 16/32-bit design was necessary, and Motorola turned it into a coherent mission. Arriving late to the 16-bit arena afforded the new processor more transistors (roughly 40000 active versus 20000 active in the 8086), 32-bit macroinstructions, and acclaimed general ease of use.

The original MC68000 was fabricated using an HMOS process with a 3.5-micron feature size. Formally introduced in September 1979, Initial samples were released in February 1980, with production chips available over the counter in November. Initial speed grades were 4, 6, and 8 MHz. 10 MHz chips became available during 198, and 12.5 MHz chips by June 1982. The 16.67 MHz "12F" version of the MC68000, the fastest version of the original HMOS chip, was not produced until the late 1980s. Tom Gunter, retired Corporate Vice President at Motorola, is known as the "Father of the 68000."

The 68000 had many high-end design wins early on. It became the dominant CPU for Unix based workstations including Sun workstations and Apollo/Domain workstations, found its way into heralded computers such as the Amiga, Atari ST, Apple Lisa and Macintosh, and was used in the first generation of desktoplaser printers including the original Apple Inc. LaserWriter and the HP LaserJet. In 1982, the

68000 received an update to its ISA allowing it to support virtual memory and to conform to the Popek and Goldberg virtualization requirements. The updated chip was called the 68010. A further extended version which exposed 31 bits of the address bus was also produced, in small quantities, as the 68012.

To support lower-cost systems and control applications with smaller memory sizes, Motorola introduced the 8-bit compatible MC68008, also in 1982. This was a 68000 with an 8-bit data bus and a smaller (20 bit) address bus. After 1982, Motorola devoted more attention to the 68020 and 88000 projects.

Second-sourcing

Several other companies were second-source manufacturers of the HMOS 68000. These included Hitachi (HD68000), who shrank the feature size to 2.7-microns for their 12.5 MHz version, Mostek (MK68000), Rockwell (R68000), Signetics (SCN68000), Thomson/SGS-Thomson (originally EF68000 and later TS68000), and Toshiba (TMP68000). Toshiba was also a second-source maker of the CMOS 68HC000 (TMP68HC000).

CMOS versions

The 68HC000, the first CMOS version of the 68000, was designed by Hitachi and jointly introduced in 1985. Motorola's version was called the MC68HC000, while Hitachi's was the HD68HC000. The 68HC000 was eventually offered at speeds of 8-20 MHz. Except for using CMOS circuitry, it behaved identically to the HMOS MC68000, but the change to CMOS greatly reduced its power consumption. The original HMOS MC68000 consumed around 1.35 watts at an ambient temperature of 25 °C, regardless of clock speed, while the MC68HC000 consumed only 0.13 watts at 8 MHz and 0.38 watts at 20 MHz. (Unlike CMOS circuits, HMOS still draws power when idle, so power consumption varies little with clock rate.) Apple selected the 68HC000 for use in the Macintosh Portable.

Motorola replaced the MC68008 with the MC68HC001 in 1990. This chip resembled the 68HC000 in most respects, but its data bus could operate in either 16-bit or 8-bit mode, depending on the value of an input pin at reset. Thus, like the 68008, it could be used in systems with cheaper 8-bit memories.

The later evolution of the 68000 focused on more modern embedded control applications and on-chip peripherals. The 68EC000 chip and SCM68000 core expanded the address bus to 32 bits, removed the M6800 peripheral bus, and excluded the MOVE from SR instruction from user mode programs. In 1996, Motorola updated the standalone core with fully static circuitry drawing only 2 µW in low-power mode, calling it the MC68SEC000.

Motorola ceased production of the HMOS MC68000 and MC68008 in 1996, but its spin-off company, Freescale Semiconductor, is still producing the MC68HC000, MC68HC001, MC68EC000, and MC68SEC000, as well as the MC68302 and MC68306 microcontrollers and later versions of the DragonBallfamily. The 68000's architectural descendants, the 680x0, CPU32, and Coldfire families, are also still in production.

As a microcontroller core

After being succeeded by “true” 32-bit microprocessors, the 68000 was used as the core of many microcontrollers. In 1989, Motorola introduced the MC68302 communications processor.

Questions

1. Explain the brief history of Motorola 68000 CPU.
2. Discuss the CMOS versions of Motorola 68000.

8.8 Summary

- Registers are a group of cells used for memory addressing, data manipulation and processing.
- The CPU thought of as the “brains” of the device.
- The control unit directs and controls the activities of the internal and external devices
- Control unit is a typical component of the CPU that implements the microprocessor instruction set.
- The three commonly known processor architectures are CISC (Complex Instruction Set Computer), RISC (Reduced Instruction Set Computer), and EPIC (Explicitly Parallel Instruction Computing).

8.9 Keywords

Arithmetic and logic unit*:* The two basic components of a CPU are the control unit the arithmetic logic unit.

Control Unit**:** A control unit in general is a central (or sometimes distributed but clearly distinguishable) part of the machinery that controls its operation, provided that a piece of machinery is complex and organized enough to contain any such unit.

CPU*:* It is the brain of a computer system. All major calculations and comparisons performed by a computer are carried out inside its CPU. CPU is also responsible for activating and controlling the operations of other unit of the computer system. Hence, no other single component of a computer determines its overall performance as much as its CPU.

Registers: It is Special, high-speed storage area within the CPU.

System clock and clock cycles: The CU and ALU perform operations at incredible speed. These operations are usually synchronized by a built-in electronic clock (known as system clock) that emits millions of regularly spaced electric pulses per second (known as clock cycles).

8.10 Review Questions

1. What is the CPU?
2. What is an arithmetic logic unit?
3. What are the types of processor speed? Explain it.
4. What is instruction set of computer?
5. Differentiate between the CISC and RISC processors?
6. What are the registers?
7. What are the EPIC Processors?
8. Discuss the Control Unit.
9. What problem Apple computers?
10. Define the CISC.

Answers for Self Assessment Questions

1. (a) 2. (c). 3. (a) 4. (b) 5. (a)

8.11 Further Studies

1. http://my.safaribooksonline.com/book/information-technology-and-software-development/9788131733097/the-computer-system-hardware/ch02lev1sec2
2. Computer Fundamentals by P. K. Sinha

9

Storage Devices

STRUCTURE

9.0 Objectives

9.1 Introduction

9.2 Storage and its Needs

9.3 Brain versus Memory

9.4 Storage Evaluation Units

9.5 Data Access Methods

9.6 Memory Card

9.7 Driving Naming Conventions in a PC

9.8 Summary

9.9 Keywords

9.10 Review Questions

9.11 Further Studies

9.0 Objectives

After studying this chapter, you will be able to:

- Discuss the basic concept of storage and its needs
- Explain the Brain versus Memory
- Understand about the Storage Evaluation Units

9.1 Introduction

Computer storage devices are used to store huge amounts of data and information permanently. If you want any of your data kept safe and lastingly, then your choice should be these devices. Usually these kinds of devices are called secondary storage or permanent storage.

When we choose storage devices, we need to understand their characteristics. The three main characteristics of storage media are access method, capacity and portability. Access method refers to how data is accessed from storage devices. Sequential and direct are the two kinds of methods used to access data from secondary devices.

Storage devices are the building blocks of storage in disk subsystems as well as being used as standalone products in server systems. The disk drive technology as the device that is used tar more than any other.

9.2 Storage and its Needs

With hard drives now storing tons of data and SSDs becoming more and more common, your computer's storage can now range from 64GB to 3TB. That is a pretty big range. How much space do you actually need?

For those considering purchasing a new computer for work, whether it is a laptop or desktop, and storage needs remain a key concern. However, with the ability to extend a computer's storage capacity with external hard-drives, opting for a computer with the greatest amount of storage is no longer a necessity, but rather beneficial.

Not every person has unlimited amounts of resources to buy a computer with boundless amounts of storage. Although purchasing a device with minimal storage isn't the proper strategy either, identifying the specific needs for the device is essential.

People using their computer to simply access documents and the internet may not want to delve into their wallet to buy a top-of-the-line device, while workers using their computers to store documents, spreadsheets, presentations and any other file may look to more expensive devices with greater storage.

9.3 Brain versus Memory

Hey we all use computers everyday and store enormous amount of data for further processing. Not restricting only to that, we also have memory storage facilities in mobile phones, play stations (other gaming devices) and televisions too. This started from calculators and the most recent flash memories. Recently with the rise of external hard disk the memory capacity storage has become almost infinite. But these devices store data mostly in "Bits-Bytes" format.

So after the evolution of these electronic data storage devices, there has been a question which ruins every human brain. There are also debates at various levels on comparison between Human Brain and Computer memory.

9.3.1 Computers and other Memory Storage Devices

Computers have limited memory capacity, even though we have lots of additional external aids for memory storage like external hard disks, flash memories, memory cards etc. These devices are just electronic devices which store data in a particular format. These data s gets distributed around various "sectors" on a compact disk or the memory slot. These are stored in a language which are only "0"s & "1"s. They can be arranged, edited, formatted, and retrieved when ever required. There is a greater flexibility in using such memories. They take few seconds to retrieve from their memory space. Let us say for e.g.: we have a computer of memory 160 GB. So we can store data s up to say 140 GB easily. After that there is always a limitation that it can no more accept data s. The effect of such huge memory storage results in slowing down the processor. So the speeds of retrieval or any operation performed are slow. The data's as such stored in the memory space of the computers cannot integrate by itself until any programmer is coded to perform that particular operation. The data's stored are just like cargoes stored onboard a ship. They cannot perform or be of any use until a human integrates them. These memory storages are used at hazardous areas like nuclear reactors to monitor and store records. These are also used in "Black Boxes" of an aircraft to store data's which remains undistracted even after plane crash. These types of memories are modified to store data's and compute statistics for even decades together. With the usage of "Internet", we can definitely say the world has shrunk so much that we get every in-depth details of anything which happens in any part of the world in micro-seconds (search engines).

9.3.2 Human Brain the Power House of Everything

Human Brain weighs more than electronic memory storage devices. It is bigger too. It may be slower in performing some tedious calculations than computers. It may not perform all automated jobs that a computer performs. With all the above said functions performed by memories of computer, Definitely Human Brain cannot. But the beauty of Human Brain is it can analyze, integrate, decide, learn, teach etc. And many such things which computer memories cannot. The computer memory can hold various details, but cannot understand. For e.g. it can get the details of supreme Sun in few micro-seconds, but it cannot understand or see the Sun. It can calculate and store distance between stars and planets, but it does not know what they are..! It can save data s and store complicated records, but Human Brain creates a purpose for those data. The improvement in every inch of the memory storage devices are done with Human

Brains. The technology may develop multi-folds, but all these developments are for the Humans, by the Humans.
The Human Brain can process emotions too. It controls the most complex mechanism, the most complex machine in the world... "The Human Being"...! The brain has various sections performing various tasks. Emotional centre takes care of the emotions, the memory centre stores data s which can be processed with any kind of matter, it has nervous centre which controls every movement and reflexes of Human Being.
When we talk about the memory storage capacity of Human Brain, it is definitely infinite...!
One should keep in mind that whatever we see is what we learn. Whatever we experience is what we learn and this learning is end-less. We learn everyday unconsciously and with consciousness too.

9.4 Storage Evaluation Units

9.4.1 Primary Storage

Primary storage, also known as main storage or memory, is the main area in a computer in which data is stored for quick access by the computer's processor. On today's smaller computers, especially personal computers and workstations, the term random access memory (RAM) - or just memory is used instead of primary or main storage, and the hard disk, diskette, CD, and DVD collectively describe secondary storage or auxiliary storage.

The terms main storage and auxiliary storage originated in the days of the mainframe computer to distinguish the more immediately accessible data storage from storage that required input/output operations. An earlier term for main storage was core in the days when the main data storage contained ferrite cores.

Primary storage is sometimes used to mean storage for data that is in active use in contrast to storage that is used for backup purposes. In this usage, primary storage is mainly the secondary storage referred to in meaning 1. (It should be noted that, although these two meanings conflict, the appropriate meaning is usually apparent from the context.)

9.4.2 Secondary Storage

Secondary storage devices, as indicated by the name, save data after it has been saved by the primary storage device, usually referred to as RAM (Random Access Memory). From the moment you start typing a letter in Microsoft Word, for example, and until you click on "Save," your entire work is stored in RAM. However, once you power off your machine, that work is completely erased, and the only copy remaining is on the secondary storage device where you saved it, such as internal or external hard disk drive, optical drives for CDs or DVDs, or USB flash drive.
As a result, additional memory, called auxiliary memory or secondary storage, is used with most computer system. Secondary storage is non-volatile and has lower cost per bit stored but it generally has an operating speed far slower than that of primary storage. It is used primarily to store large volume of data on permanent basis that can be partially transferred to primary storage, whenever required for processing.
Over the years, several devices and media have been used as secondary storage in computer systems. Many of them such as punched paper tape and punched cards have become obsolete now. They are magnetic tape, magnetic disk, optical disk, and memory storage devices. You will learn about the terminologies, principles of operation, uses, and trade-offs of different types of secondary storage devices.

Hard Disk Operations

The basic physical construction of a hard disk drive consists of spinning disks with heads that move over the disks and store data in tracks and sectors. The heads read and write data in concentric rings called tracks, which are divided into segments called sectors, which typically store 512 bytes each (see Figure 1).

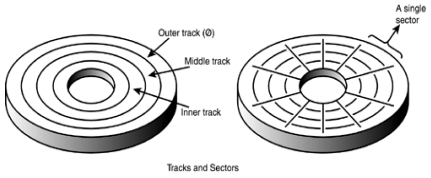


Figure 1: The tracks and sectors on a disk.

Hard disk drives usually have multiple disks, called platters, that are stacked on top of each other and spin in unison, each with two sides on which the drive stores data. Most drives have two or three platters, resulting in four or six sides, but some PC hard disks have up to 12 platters and 24 sides with 24 heads to read them (Seagate Barracuda 180). The identically aligned tracks on each side of every platter together make up a cylinder. A hard disk drive usually has one head per platter side, with all the heads mounted on a common carrier device or rack. The heads move radially across the disk in unison; they can't move independently because they are mounted on the same carrier or rack, called an actuator.

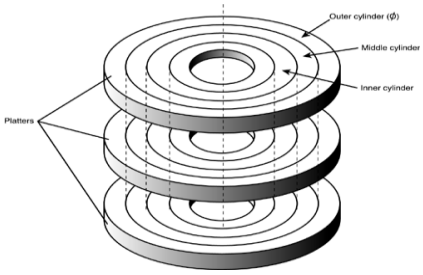


Figure 2: Hard disk cylinders.

Originally, most hard disks spun at 3,600rpm—approximately 10 times faster than a floppy disk drive. For many years, 3,600rpm was pretty much a constant among hard drives. Now, however, most drives spin even faster. Although speeds can vary, modern drives typically spin the platters at either 4,200rpm; 5,400rpm; 7,200rpm; 10,000rpm; or 15,000rpm. Most standard-issue drives found in PCs today spin at 5,400rpm, with high performance models spinning at 7,200rpm. Some of the small 2 1/2" notebook drives run at only 4,200rpm to conserve power, and the 10,000rpm or 15,000rpm drives are usually found only in very high-performance workstations or servers, where their higher prices, heat generation, and noise can be more easily dealt with. High rotational speeds combined with a fast head-positioning mechanism and more sectors per track are what make one hard disk faster overall than another.

The heads in most hard disk drives do not (and should not!) touch the platters during normal operation. However, on most drives, the heads do rest on the platters when the drive is powered off. In most drives, when the drive is powered off, the heads move to the innermost cylinder, where they land on the platter surface. This is referred to as contact start stop (CSS) design. When the drive is powered on, the heads slide on the platter surface as they spin up, until a very thin cushion of air builds up between the heads and platter surface, causing the heads to lift off and remain suspended a short distance above or below the platter. If the air cushion is disturbed by a particle of dust or a shock, the head can come into contact with the platter while it is spinning at full speed. When contact with the spinning platters is forceful enough to do damage, the event is called a head crash. The result of a head crash can be anything from a few lost bytes of data to a completely ruined drive. Most drives have special lubricants on the platters and hardened surfaces that can withstand the daily “takeoffs and landings” as well as more severe abuse.

Some newer drives do not use CSS design and instead use a load/unload mechanism that does not allow the heads to contact the platters, even when the drive is powered off. First used in the 2 1/2" form factor notebook or laptop drives where resistance to mechanical shock is more important, traditional load/unload mechanisms use a ramp positioned just off the outer part of the platter surface, whereas some newer designs position the ramp near the spindle. When the drive is powered off or in a power saving mode, the

heads ride up on the ramp. When powered on, the platters are allowed to come up to full speed before the heads are released down the ramp, allowing the airflow (air bearing) to prevent any head/platter contact.
Because the platter assemblies are sealed and nonremovable, the track densities on the disk can be very high. Hard drives today have up to 96,000 or more tracks per inch (TPI) recorded on the media (Hitachi Travelstar 80GN). Head disk assemblies (HDAs), which contain the platters, are assembled and sealed in clean rooms under absolutely sanitary conditions. Because few companies repair HDAs, repair or replacement of the parts inside a sealed HDA can be expensive. Every hard disk ever made eventually fails. The only questions are when the failure will occur and whether your data is backed up.

Floppy Disk Drives

Most modern personal computers do not have floppy drives anymore. The small, 3 1/2" disks they read only hold 1.44 MB of data so they have become somewhat obsolete when compared to the data storing capabilities of USB flash drives (up to 256 GB) and CD drives (700 MB).
Floppies are non-volatile storage. The data remains when the computer power is turned off. They can be read or written to over and over again.
The standard types of floppy disk drives and disks that have been used in PCs since the beginning. It explores the various types of drives and disks, how they function, and how to properly install and service them. The high-capacity floppy drives such as the SuperDisk (LS-120 and LS-240) and flash memory devices such as USB keychain are covered separately the floppy is still sometimes used as a system installation and configuration device, especially when troubleshooting. In older systems that don't support the El Torito CD-ROM boot specification, the floppy drive is the only way to load an operating system from scratch or to run bootable diagnostics. Newer systems that support El Torito (bootable CDs) do not require floppy drives because they can boot operating systems and diagnostics directly from a CD.



Figure 3: Floppy Disk Drives.

Winchester Disk

An early removable disk drive from IBM that put the heads and platters (disks) in a sealed unit for greater speed. Before the Winchester architecture, removable disks were like removable disks today, in which the read/write heads remain in the drive and make contact with the platter after the cartridge is inserted.
Introduced in 1973 as the model 3340, the drive had one permanent and one removable spindle, each holding 30MB. The “30-30” storage capacities led to the Winchester nickname after the Winchester 30-30 rifle. The term later referred to all fixed hard disks because the heads and platters are always encased in the same, sealed unit. See also Winchester.

From Computer Desktop Encyclopedia
Reproduced with permission.
© 1996 International Business Machines Corporation
Unauthorized use not permitted.

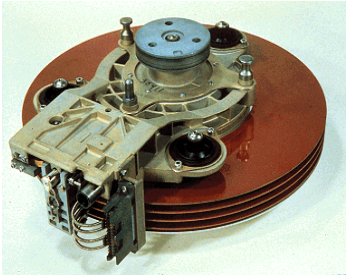


Figure 4: The Winchester Disk.

The BM's Winchester disk was a removable cartridge, but the heads and platters were built in a sealed unit and were not separable.
A type of disk storage device characterized by no removable or sealed disk packs; extremely narrow tracks; a lubricated surface that allows the head to rest on the surface during start and stop operations; and servomechanisms which utilize a magnetic pattern, recorded on the medium itself, to position the head.

Optical Disk
An optical disk is mounted on an optical disk drive for reading/writing of information on it. An optical disk drive contains all the mechanical, electrical, and electronic components for holding an optical disk and for reading/writing of information on it. That is, it contains the tray on which the disk is kept, read/write laser beams assembly, and motor to rotate the disk Figure 5 Shows an optical disk drive.



Figure 5 Optical Disk Drive

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: Consider a disk size is 32MB, and a block size is 1KB, calculate the size of FAT16.
……..……

……..……

……

Did You Know?
The optical disc was invented in 1958.

Zip Drive: A Zip drive is a small, portable disk drive used primarily for backing up and archiving personal computer files. The trademarked Zip drive was developed and is sold by Lomega Corporation. Zip drives and disks come in two sizes. The 100 megabyte size actually holds 100,431,872 bytes of data or the equivalent of 70 floppy diskettes. There is also a 250 megabyte drive and disk. The Lomega Zip drive comes with a software utility that lets you copy the entire contents of your hard drive to one or more Zip disks.

Flash Drives: flash drive is a compact device of the size of a pen, comes in various shapes and stylish designs (such as pen shape, wallet shape etc.), and may have different added features (such as with a camera, with a built-in N1P3lWMA/FM Radio play back for music on the go, etc.). It enables easy transport of data from one computer to another. It is a plug-and-play device that simply plugs into a USB (Universal Serial Bus) port of a computer. The computer detects it automatically as removable drive. No one can read, write, copy, delete, and move data from the computer's hard disk drive to the flash drive or from the flash drive to the hard disk drive. One can even run applications, view videos, or play MP3 files from it directly. Once done, it can be simply plugged out of the USB port of the computer and kept into the pocket for being carried anywhere. A flash drive does not require any battery cable, or software, and is compatible with most PCs, desktop, and laptop computers with USB 2.0 port. All these features make it ideal external data storage for mobile people to carry or transfer data from one computer to another. As the

name implies, it is based on *flush memory* storage technology Recall that flash is non-volatile, Electrically Erasable Programmable Read Only Memory (EEPROM) chip. It is a highly -Aid-state storage having data retention capability of more than 10 years.



Figure 6: A flash drive (pen drive).

Available storage capacities are 8MB, 16MB, 64MB, 128MB, 256MB, 512MB, 1GB, 2GB, 4GB, and 8GB. A: of 16MB capacity has 5600 times more storage capacity than a IAMB floppy disk.
Figure 9.7 shows a flash drive. It has a main body and usually a port connector cover. The cover is removed or port connector is pushed out when the drive is to be plugged into the USB port of' a computer. The main body usually has a write protect tab, a read/write LED (Light Emitting Diode) indicator, and a strap hole. Some manufacturers also provide software to be used with the drive.

Blu Ray Disk: Blu-ray Disc (BD) is a next-generation optical disc format meant for storage of high-definition video and high-density data. The Blu-ray standard was jointly developed by a group of consumer electronics and PC companies called the Blu-ray Disc Association (BDA). As compared to the HD DVD format, its main competitor, Blu-ray has more information capacity per layer, 25 instead of 15 gigabytes, but may initially be more expensive to produce.

Did You Know?
In 1953 IBM recognized the immediate application for what it termed a “Random Access File” having high capacity and rapid random access at a relatively low cost.

9.5 Data Access Methods

How data files are stored in secondary storage varies with the types of media and devices you are using. Data files may be stored on or in sequential-access storage, direct-access storage, or random-access storage.

9.5.1 Sequential Access Storage

Punched cards, paper tape, and magnetic tape are examples of sequential-access storage media. When operating in a sequential environment, a particular record can be read only by first reading all the records that come before it in the file. When you store a file on tape, the 125th record cannot be read until the 124 records in front of it are read. The records are read in sequence. You cannot read just any record at random. This is also true when reading punched cards or paper tape.

9.5.2 Direct Access Storage

Direct-access storage allows you to access the 125th record without first having to read the 124 records in front of it. Magnetic disks and drums are examples of direct-access storage media. Data can be obtained quickly from anywhere on the media. However, the amount of time it takes to access a record is dependent to some extent on the mechanical process involved. It is usually necessary to scan some (but not all) of the preceding data.

9.5.3 Random Access Storage

Random-access storage media refers to magnetic core, semiconductor, thin film, and bubble storage. Here, a given item of data can be selected from anywhere in storage without having to scan any preceding items. And, the access time is independent of the storage location.

In computing, an access method is a program or a hardware mechanism that moves data between the computer and an outlying device such as a hard disk (or other form of storage) or a display terminal. The term is sometimes used to refer to the mechanics of placing or locating specific data at a particular place on a storage medium and then writing the data or reading it. It is also used to describe the way that data is located within a larger unit of data such as a data set or file. An access method is also an application program interface (API) that a programmer uses to create or access data sets or to read from or write to a display terminal or other output device. Examples are the Virtual Sequential Access Method (VSAM) and the Virtual Telecommunication Access Method (VTAM).

Exercise: Check Your Progress 2
Note: i) Use the space below for your answer.
Ex1: Recover files on storage device.

……..……

……..……

……

9.6 Memory Card

These disks operate like the common hard disk. But have less storage space, consumes less power and data can be accessed quickly. Its ease of usage makes acceptable, however it is susceptible to virus attack and failure.
You should not hold your data permanently on these types of disks; rather you should use them to hold copy files on which you are going to work now. Do not use these disks as a temporary storage device.

9.7 Driving Naming Conventions in a PC

The following fundamental rules enable applications to create and process valid names for files and directories, regardless of the file system:

- Use a period to separate the base file name from the extension in the name of a directory or file.
- Use a backslash (\) to separate the components of a path. The backslash divides the file name from the path to it, and one directory name from another directory name in a path. You cannot use a backslash in the name for the actual file or directory because it is a reserved character that separates the names into components.
- Use a backslash as required as part of volume names, for example, the "C:\" in "C:\path\file" or the "\\server\share" in "\\server\share\path\file" for Universal Naming Convention (UNC) names. For more information about UNC names, see the Maximum Path Length Limitation section.
- Do not assume case sensitivity. For example, consider the names OSCAR, Oscar, and oscar to be the same, even though some file systems (such as a POSIX-compliant file system) may consider them as different. Note that NTFS supports POSIX semantics for case sensitivity but this is not the default behavior.
- Volume designators (drive letters) are similarly case-insensitive. For example, "D:\" and "d:\" refer to the same volume.
- Use any character in the current code page for a name, including Unicode characters and characters in the extended character set (128–255), except for the following:
- The following reserved characters:
- < (less than)
- (greater than)
  : (colon)

" (double quote)
/ (forward slash)
\ (backslash)
| (vertical bar or pipe)
? (question mark)

- (asterisk)
- Integer value zero, sometimes referred to as the ASCII NUL character.
- Characters whose integer representations are in the range from 1 through 31, except for alternate data streams where these characters are allowed. For more information about file streams, see File Streams.
- Any other character that the target file system does not allow.
- Use a period as a directory component in a path to represent the current directory, for example ".\temp.txt". For more information, see Paths.
- Use two consecutive periods (..) as a directory component in a path to represent the parent of the current directory, for example "..\temp.txt". For more information, see Paths.
- Do not use the following reserved device names for the name of a file:

The CON, PRN, AUX, NUL, COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9, LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, and LPT9. Also avoid these names followed immediately by an extension; for example, NUL.txt is not recommended. For more information, see Namespaces.

- Do not end a file or directory name with a space or a period. Although the underlying file system may support such names, the Windows shell and user interface does not. However, it is acceptable to specify a period as the first character of a name. For example, ".temp".

Self Assessment Questions

1. Which is the most appropriate device to back up data held on a server each night?

(a) Magnetic tape (b) Zip disk
(c) Flash memory stick (d) Hard disk

2. Which would be the most appropriate device for storing a 2 hour film?

(a) CD-ROM (b) Zip disk
(c) DVD (d) Hard disk

3. Which would be the most appropriate device for transferring files between school and home?

(a) Zip disk (b) Hard disk
(c) CD-ROM (d) Flash memory stick

4. Which would be the most appropriate to store the computers BIOS instructions?

(a) Flash memory stick (b) RAM
(c) ROM (d) Hard disk

5. Which of these would be the most likely storage device for a music album?

(a) CD-ROM (b) Hard disk
(c) Flash memory (d) RAM

9.8 Summary

- Secondary storage of a computer system is non volatile and has low cost per bit stored, but it generally has an operating speed far slower than that of primary storage.
- A computer's main memory built of volatile RAM chips.

- Memory card disks operate like the common hard disk. But have less storage space, consumes less power and data can be accessed quickly.
- Computer memory is organized into a hierarchy. At the highest level are the processor registers.
- Any storage unit of a computer system is characterized and evaluated based on following properties storage capacity, access time cost per bit of storage, volatile, and random access.
- A primary storage or main memory of a computer system is made up of several small storage areas called location or cell. Each of these location store a fixed number of bits, called word length the memory.
- Memory storage devices use flash memory technology for secondary storage devices. Two popular memory storage devices are flash device (pen drive) and memory card.

9.9 Keywords

Blu-ray Disc BD is a next-generation optical disc format meant for storage of high-definition video and high-density data.

Digital Video (or Versatile) Disk DVD was designed primarily to store and distribute movies. However, it is fast becoming mainstream optical disk as prices are reducing and need for large capacity storage is increasing.

Flash Drives It enables easy transport of data from one computer to another

Hard Disk Operations The basic physical construction of a hard disk drive consists of spinning disks with heads that move over the disks and store data in tracks and sectors. The heads read and write data in concentric rings called tracks, which are divided into segments called sectors, which typically store 512 bytes each.

Primary storage It is a computer system has limited capacity and is volatile. Hence, additional memory, called auxiliary memory or secondary storage is used with most computer systems.

Video Compact Disc' VCD stands for 'Video Compact Disc' and basically it is a CD that contains moving pictures and sound.

9.10 Review Questions

1. What is memory cell?
2. Which of the following is not a writable DVD standard?
3. What does it mean to reimage a hard disk?
4. What are the differences between serial access memory and random access memory?
5. What is a secondary storage? How it differs from a primary storage?
6. What is optical disk? How are data recorded/read from an optical disk?
7. List typical uses of optical disks.
8. What is the difference between the main memory RAM and the cache memory in computers?
9. What is a secondary storage? How it differs from a primary storage?
10. What are the physical memory devices of computers?

Answers for Self Assessment Questions

1. (a) 2. (C) 3. (d) 4. (c) 5. (a)

9.11 Further Studies

1. http://my.safaribooksonline.com/book/hardware/upgrading-and-repairing/059600866x/external-and-removable-storage-devices/repairpc-chp-9
2. Computer Fundamentals by P. K. Sinha

10

Basics of Software

STRUCTURE

10.0 Objectives

10.1 Introduction

10.2 Software

10.3 Types of Software

10.4 Open Source Software

10.5 Integrated Development Environment (IDE)

10.6 Needs of Software

10.7 Summary

10.8 Keywords

10.9 Review Questions

10.10 Further Studies

10.0 Objectives

After studying this chapter, you will able to:

- Define software
- Explain the type of software
- Describe the open source software
- Explain the integrated development environment
- Understand need of software

10.1 Introduction

Software is a general term used to describe a collection of computer programs, procedures and documentation that perform some tasks on an operating system. Software is the way to perform different tasks electronically. Software is a set of rules to perform a specific task. The software is the information that the computer uses to get the job done. Software needs to be accessed before it can be used. There are many terms used for the process of accessing software including running, executing, starting up, opening, and others. Computer programs allow users to complete tasks. A program can also be referred to as an application and the two words are used interchangeably.

10.2 Software

A computer cannot do anything on its own. It must be instructed to do a job desired by us. Hence, it is necessary to specify a sequence of instructions a computer must perform to solve a problem. Such a

sequence of instructions written in a language understood by a computer is called a computer program. A program controls a computer's processing activity, and the computer performs precisely what the program wants it to do. When a computer is running a program to perform a task, we say, it is running or executing that program. Hardware you can touch, software you can not.
The term software refers to a set of computer programs, procedures, and associated documents (flowcharts, manuals, etc.) describing the programs, and how they are to be used.

Software package is a group of programs that solve a specific problem or perform a specific type of job. For sample, a word-processing package may contain programs for text editing, text formatting, drawing graphics, spelling checking, etc. Hence, a multipurpose computer system, like a personal computer in your home, has several software packages, one each for every type of job it can perform. Software is a collection of instructions that enables a user to interact with the computer or have the computer perform specific tasks for them. Without any software the computer would be useless.

10.2.1 Relationship between Hardware and Software

For a computer to produce useful output its hardware and software must work together. Nothing useful can be done with the hardware on its own, and software cannot be utilized without supporting hardware.

Can a computer run without software?

In most situations, yes, a computer can run without software being installed. However, if an operating system or interpreter is not found on the computer it would either generate an error or sit at a black screen. Installing additional programs onto the computer after the operating system has been installed will only give those computer additional abilities. For example, a word processor is not required, but will allow you to create documents and letters.

How to install a software program

How to install a software program can depend on the operating system being used and the program being installed. Because of all the different possibilities, we have created the above for installing programs in each of the major operating systems.
Computer hardware is any physical device, something that you are able to touch and software is a collection of instructions and code installed into the computer and cannot be touched. For example, the computer monitor you are using to read this text on and the mouse you are using to navigate this web page is computer hardware. The Internet browser that allowed you to visit this page and the operating system that the browser is running on is software.

Can a computer run without hardware?

Depends on the hardware. Almost all computer setups will require at least a disk drive (e.g. hard drive), display, keyboard, memory, motherboard, processor, power supply, and video card in order to function properly. If any of these devices was missing or had problems, an error would be encountered, or the computer would not start. Adding other hardware such as a disc drive (e.g. CD-ROM or DVD), modem, mouse, network card, printer, sound card, speakers, etc. are not required, but will give the computer additional capabilities.
To take an analogy, a cassette player and its cassettes purchased from the market are hardware. However, the songs recorded on the cassettes are its software. To listen to a song, that song has to be recorded on one of the cassettes first, this is then mounted on the cassette player and played. Similarly, to get a job done by a computer, the corresponding software has to be loaded in the hardware first and then executed.
Following important points regarding the relationship between hardware and software are brought out by this analogy:
1. Both hardware and software are necessary for a computer to do useful job. Both are complementary to each other.
2. Same hardware can be loaded with different software to make a computer perform different types of jobs just as different songs can be played using the same cassette player.
3. Except for upgrades (like increasing main memory and hard disk capacities, or adding speakers, modems, etc.), hardware is normally a one-time expense, whereas software is a continuing expense. Like

we buy new cassettes for newly released songs or for songs whose cassettes, we do not have, we buy, new software to be run on the same hardware as and when need arises, or funds become available.

10.2.2 Software Development Steps

All software needs to be developed by someone. Developing software and putting it to use is a complex process involving following steps:

1. Analyzing the problem at hand, and planning the program(s) to solve the problem.
2. Coding the program(s).
3. Testing, debugging, and documenting the program(s).
4. Implementing the program(s).
5. Evaluating and maintaining the program(s).

Did You Know?

The first theory about software was proposed by Alan Turing in his 1935 essay Computable numbers with an application to the Entscheidungs problem (Decision problem).

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Write the steps in software development.

……...……

……...……

……

10.3 Types of Software

Although the range of software available today is vast and varied, most software can be divided into two major categories:

10.3.1 System Software

System software is a set of one or more programs designed to control the operation and extend the processing capability of a computer system. In general, a computer's system software performs one or more of the following functions:

1. Supports development of other application software.
2. Supports execution of other application software.
3. Monitors effective use of various hardware resources such as CPU, memory, peripherals, etc.
4. Communicates with and controls operation of peripheral devices such as printer, disk, tape, etc.

Some commonly known types of system software are:

Operating Systems Operating system software takes care of effective and efficient utilization of all hardware and software components of a computer system.

Programming Language Translators Programming language translators transform the instructions prepared by programmers in a programming language into a form that can be interpreted and executed by a computer system.

Communications Software In a network environment (where multiple computers are interconnected together by communications network) communications software enables transfer of data and programs from one computer system to another.

Utility Programs Utility programs (also known as *utilities)* are a set of programs that help users in system maintenance tasks, and in performing tasks of routine nature. Some tasks commonly performed by utility programs include formatting of hard disks or floppy disks, taking backup of files stored on hard disk on to a tape or floppy disk, sorting of the records stored in a file based on some key field(s), etc.

Did You Know?
The term "software" was first used in print by John W. Tukey in 1958. Colloquially, the term is often used to mean application software.

10.3.2 Application Software

Application software is a set of one or more programs designed to solve a specific problem, or do a specific task. For example, payroll processing software, examination results processing software, railway/airline reservation software, computer games software are all application software. Similarly, a program written by a scientist to solve a research problem is also application software. The programs included in an application software package are called *application programs.* The programmers who prepare application software are referred to as *application programmers.*

Some commonly known application software is:

Word-Processing Software: Word-processing software enables us to make use of a computer for creating, editing, viewing, formatting, storing, retrieving, and brinting documents (written material, such as letters, reports, books, etc.).

Spreadsheet Software: Spreadsheet software is a numeric-data-analysis tool that allows us to create a kind of computerized ledger. A manual ledger is a book having rows and columns that accountants use for keeping a record of financial transactions and for preparing financial statements.

Database software: *A database* is a collection of related data stored and treated as a unit for information retrieval purposes. *A database software* is a set of programs that enable us to create a database, maintain it (add, delete, and update its records), organize its data in desired fashion (for example, sort its records alphabetically name-wise), and selectively retrieve useful information from it. For example, queries such as get the telephone number of the person named Kashyap Rana from the address database, or get the names of all currently enrolled students whose birthdays fall today from the student database can be handled easily.

Graphics Software: Graphics software enables us to use a computer system for creating, editing, viewing, storing, retrieving, and printing of designs, drawings, pictures, graphs, etc.

Personal Assistance Software: Personal assistance software allows us to use personal computers for storage and retrieval of our personal information, as well as planning and management of our schedules, contacts, finances, and inventory of important items.

Education Software: Education software allows a computer to be used as a teaching and learning tool. A few examples of such software include those used for teaching mathematics, grammar, language, or any other subject.

Entertainment Software: Entertainment software allows a computer to be used as an entertainment tool. Computer video games belong to this category of software.

10.3.3 Programming Software

Programming software usually provides tools to assist a programmer in writing computer programs, and software using different programming languages in a more convenient way. The tools include:

Compilers

A compiler is a computer program (or set of programs) that transforms source code written in a programming language (the source language) into another computer language (the target language, often having a binary form known as object code). The most common reason for wanting to transform source code is to create an executable program.

Debuggers

A debugger or debugging tool is a computer program that is used to test and debug other programs (the "target" program). The code to be examined might alternatively be running on an instruction set simulator (ISS), a technique that allows great power in its ability to halt when specific conditions are encountered but which will typically be somewhat slower than executing the code directly on the appropriate (or the same) processor. Some debuggers offer two modes of operation - full or partial simulation, to limit this impact.

Interpreters

An interpreter normally means a computer program that executes, i.e. performs, instructions written in a programming language. An interpreter may be a program that either

1. Executes the source code directly
2. Translates source code into some efficient intermediate representation (code) and immediately executes this
3. Explicitly executes stored precompiled code made by a compiler which is part of the interpreter system

Linkers

A linker or link editor is a program that takes one or more objects generated by a compiler and combines them into a single executable program.

Text editors

A text editor is a type of program used for editing plain text files. Text editors are often provided with operating systems or software development packages, and can be used to change configuration files and programming language source code.

Did You Know?

The history of computer software is most often traced back to the first software bug in 1946.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: List down the names of application software and system software.

……..……

……..……

……

10.4 Open Source Software

Open source software (OSS) is computer software that has its underlying 'source-code' made available under a license. This can allow developers and users to adapt and improve it. Policy on the use of OSS in government was updated in 2004. This briefing explains how OSS works, outlines current and prospective uses and examines recent policy developments. It discusses its advantages and disadvantages and examines factors affecting uptake.

Computer software can be broadly split into two development models *Proprietary* or 'closed' software, owned by a company or individual. Copies of the 'binary' are made public; the 'source-code' is not usually made public.

Open source software (OSS), where the source-code is released with the binary. Users and developers can be licensed to use and modify the code, and to distribute any improvements they make.

In practice, software companies often develop both types of software. OSS is developed by an on-going, iterative process where people share the ideas expressed in the source-code. The aim is that a large community of developers and users can contribute to the development of the code, check it for errors and bugs, and make the improved version available to others. Project management software is used to allow developers to keep track of the various versions.

Both OSS and proprietary approaches allow companies to make a profit. Companies developing proprietary software make money by developing software and then selling licenses to use the software, for example Microsoft receives a payment for every copy of Windows sold with a personal computer. OSS companies make their money by providing services, such as advising clients on the version that best suits their needs, installing and customizing software and development and maintenance.

The software itself may be made available at no cost. There are two main types of OSS licenses:
Berkeley Software Distribution (BSD) License: this permits a licensee to 'close' a version (by withholding the most recent modifications to the source-code) and sell it as a proprietary product;
The GNU General Public License (GNU, GPL, or GPL) under this license, licensees may not 'close' versions. The licensee may modify copy and redistribute any derivative version, under the same GPL license. The licensee can either charge a fee for this service or work free of charge.

Desirable software attributes

There is widespread debate over the relative merits of proprietary software and OSS. However, it is difficult to make general comparisons; most analysts say comparisons should be made only on a case-by-case basis. It is generally agreed that whether software is open source or proprietary, the following attributes are of key importance:

- Reliability: defined as how long a system can stay in operation without user intervention;
- Quality: commonly defined as the number of errors in a fixed number of lines of code;
- Security: how resilient the software is to unauthorized actions (e.g. viruses);
- Flexibility: how easily the software can be customized to meet specific needs and run on different types of device;
- Project management: how well organized the development process is; open standards: documents created with one type of software being readable in another. This avoids 'lock- in' to a particular document format.
- Switching costs: the cost of moving from one system to another;
- Total cost of ownership (TCO): the full costs incurred over the lifetime of the software;
- User-friendliness: how easy the software is to use.

10.4.1 Use of Open Source Software

The private sector

There is increasing awareness and uptake of OSS within the private sector, with OSS and proprietary software becoming increasingly interwoven. Major corporations such as IBM believe it enables them to make use of a worldwide community of developers to improve their products and services. Some industry commentators suggest that OSS will lead to a more competitive software industry. Currently over 67% of web-servers run open source software called Apache. The majority of websites and email systems run on OSS. Worldwide, around 30% of infrastructural computers run GNU/Linux, an open source operating system. However, use of OSS on the desktop is more limited: over 96% of desktop computers still use Microsoft Windows. OSS has inspired new portable device projects, such as the 'Simputer'. This is a small, inexpensive, handheld computer, intended to bring computing power to India and other emerging economies.

Open source software in government

Governments' interest in OSS is increasing, due to their reliance on sophisticated software. The UK Office of Government Commerce released a series of case studies
in October 2004 outlining how OSS has been used in the public sector. However, UK parliamentary responses to questions on the use of OSS in government show that uptake is still limited. The Office of the Deputy Prime Minister is funding the 'Open Source Academy' project. This is intended to overcome barriers to uptake of OSS in local government such as lack of information, skills, confidence and lack of suitable products.
Policy on use of OSS within government is outlined in the updated e-Government Unit's policy document released in October 2004. Key points are:

- Re-affirmation of the UK Government's commitment to 'procurement neutrality': OSS solutions should be considered alongside proprietary ones in IT procurements;
- Contracts will be awarded on a case-by-case basis, based on value for money. The UK Government

will seek to avoid 'lock-in' to proprietary IT products.

10.4.2 Other Usage

Advocates of OSS argue that, in principle, the OSS model allows software to be developed for minority markets, that is, product development can be need-driven rather than market-driven. In practice, it is not clear there is such a clear distinction between the two models: for example both GNU/Linux and Windows now have versions in a number of minority languages.

10.4.3 Open Source Culture

The principle of open source can be applied to a variety of other applications as well as software development. Some commentators believe that several sectors of government and industry could benefit from the open source approach. The ideas behind it are spreading into pharmaceutical drug production; music; book and journal publishing; television broadcasting and many other cultural areas. The BBC is planning to make some material available in a 'Creative Archive' for viewing, copying and reuse but with some rights reserved, such as commercial exploitation.

Did You Know?

The Open Source Definition was written by Bruce Perens as "The Debian Free Software Guidelines", and refined it using the comments of the Debian developers in a month-long e-mail conference in June, 1997. He removed the Debian-specific references from the document to create the "Open Source Definition."

Self Assessment Questions

1 Software is a ……to perform a specific task.

(a) Set of rules (b) Set of low

(c) Set of method (d) None of these

2. Software refers to a set of computer programs, procedures, and associated documents describing the programs, and how they are to be used.

(a) True (b) False

3. All computer setups will not require at least a disk drive, display, keyboard, memory, motherboard, processor, power supply, and video card in order to function properly.

(a) True (b) False

4. The programs included in a system software package are called…….

(a) Reporting tools (b) Application programs

(c) System programs (d) None of these

5. **.....**are a set of programs that help users in system maintenance tasks, and in performing tasks of routine nature.

(a) Utility programs (b) Operating system

(c) System programs (d) None of these

10.5 Integrated Development Environment (IDE)

An integrated development environment (IDE) (also known as integrated design environment, integrated debugging environment or interactive development environment) is a software application that provides comprehensive facilities to computer programmers for software development. An integrated development environment (IDE) is a programming environment that has been packaged as an application program, typically consisting of a code editor, a compiler, a debugger, and a graphical user interface (GUI) builder.

The IDE may be a standalone application or may be included as part of one or more existing and compatible applications. The basic programming language, for example, can be used within Microsoft Office applications, which makes it possible to write a WordBasic program within the Microsoft Word application. IDEs provide a user-friendly framework for many modern programming languages, such as Visual Basic, Java, and PowerBuilder.

The IDE is a standard electronic interface used between a computer motherboard's data paths or bus and the computer's disk storage devices. The IDE interface is based on the IBM PC Industry Standard Architecture (ISA) 16-bit bus standard, but it is also used in computers that use other bus standards. Most computers sold today use an enhanced version of IDE called Enhanced Integrated Drive Electronics (EIDE). In today's computers, the IDE controller is often built into the motherboard. IDE was adopted as a standard by American National Standards Institute (ANSI) in November, 1990.

IDEs for developing HTML applications are among the most commonly used. For example, many people designing Web sites today use an IDE (such as Home Site, Dream Weaver, or FrontPage) for Web site development that automates many of the tasks involved. An IDE normally consists of:

- A source code editor
- A compiler and/or an interpreter build automation tools
- A debugger

Did you know?

Dartmouth BASIC was the first language to be created with an IDE (and was also the first to be designed for use while sitting in front of a console or terminal). Its IDE (part of the Dartmouth Time Sharing System) was command-based, and therefore did not look much like the menu-driven, graphical IDEs prevalent today.

Caution

Be cautious before using any software, it should be compatible to hardware otherwise it may not run properly.

10.6 Needs of Software

The problem with computers is that most of us do not speak their language. We need a translator, something that can understand our needs and put the computer to work for us. The translator is called software and it makes computers useful.

Look at it this way. Like a typewriter, a computer without software is just a lifeless machine. By adding software, the computer becomes more alive, easy to use, and built for you. Most computers have two basic kinds: The operating system and software programs.

If we want to use a computer, we have to use an operating system. From saving files, to using a mouse or fixing problems, the operating system covers the basics. Operating systems come with all new computers and do a lot of same things. You have seen those called Windows, Mac and Linux.

But the operating system is only part of what we need. To make them personalized and more useful, we can add software programs. For example, if you need to edit a photo you can add a software program that is built for that purpose. If you need to design a house, you can add a software program that lets you see the house from all sides. By adding and removing software programs, you can make the computer fit with exactly what you want to do. And adding most software programs is easy. You can get them from a computer store or download them from the Internet. No computer nerds required.

What is a software program? What’s really happening when you open one? Think about it this way. Computers are really good at following instructions. And a software program is essentially a set of instructions that tells the computer exactly what to do.

When you open a program, the computer goes to work, completing the instructions until the program is ready for you to use. The ability to add and remove software programs means that everyone's computer can be different and unique to them.

Now we have talked about operating systems that take care of the basics, and software programs that make computers personalized. It's this combination that makes computers so useful. But it's not limited to computers on your desk.

Consider your cell phone. Just like a computer without software, it's a lifeless machine that does not speak our language. Thankfully, cell phones have software that brings them to life.

The same thing is true for many cameras, music players, and even our cars. Every day we rely on software to bring machines to life and make them personalized and useful.

The next time you use a computer or cell phone, think about software's role in translating your needs into instructions that put the machine to work for you.

Exercise: Check Your Progress 3:

Case Study Client CRM

Company Profile

Client is an IT organization that intends to launch to major software products in Europe and USA. The main USP of the package will be:

- *Say Goodbye to licensing*
- *Going beyond the platform*

The client currently has several lines of software products including an Operating System. They have been shipping products that range from a full fledged server operating system to business applications such as CRM and Accounts. Above to the quick summary of products and offerings from Client:

- Linux Server, which includes productivity and collaboration tools.
- Linux Desktop which include productivity tools like office suites, multimedia tools.
- CRM, a contact relationship management application.
- Accounts, a business accounting application

Business situation

The major goals of the client as an IT organization are to deliver subscription free applications to its customers and secondly, to remove any kind of platform boundaries, so that the customers may choose their choice of platform to work on.

As a result their requirement is of a comprehensive CRM application that can run on any Operating System, especially targeting an OS that is free from licensing hassles.. These situations may be summed up follows:

- Low cost solution.
- Solution should work on all major platforms including Windows, Linux and Solaris.

Technical situation

The CRM application requested by the client posed some really interesting challenges for our software designers. The major challenge was how to design an application that could run in any operating platform. The first thought was to create a web based application but that cannot be shipped and managed as easily as a desktop application. A CRM application requires extensive functionality from the native computer environment such as use of ports, printers, internet access etc. Write an application that should run in Windows, Linux, Mac and still be able to perform all required functions flawlessly was quite a challenge. Other major challenges were as follows:

- The technical aspects such as file system, shortcut keys, user interface looks etc. are quite varied across different operating systems.
- User Interface should not vary much across different platforms in order to create a brand value.
- Cross Platform desktop application means a cross platform database which again was a big challenge for us.
- File operations should work across different platforms.

- One of the key requirements of a CRM applications is integration with the productivity applications such as Office suites, E-Mail clients etc. This was again a big challenge as each OS carries it is own set of productivity tools.
- These are just a few of the key major technical issues which our design teams had to tackle with great efficiency and in a cost effective manner.

Solution

Having a look at the technical challenges, our designers proposed the following solution to Client:

- Development and runtime platform was proposed to be Java in order to avoid any kind of platform variances. Java in itself is a full fledged operating platform which provides application and native operating system, in order to allow them to execute in a platform independent manner.
- User Interface was proposed to be built in Java Swings using one of the available Plastic Look and Feel, in order to overcome any kind of user interface differences across the platforms.
- Database employed for this application was selected as HSQLDB. Being a cross platform database, it has an ability to operate in both server as well as embedded mode. Also being open source and free, it was best choice to use at the back end.
- As far as productivity tools are concerned, we proposed a solution of choice to be OpenOffice.org. Again OpenOffice.org is an open source project and has its own Java wrappers in order to make it programmable. It is also available for major operating systems including Windows, Linux and Mac. We proposed a tight integration with OpenOffice.org suite in order to follow Going beyond the platform objective.
- As far as e-mail is concerned we could have easily integrate the CRM application with Outlook but that would result in Windows based solution. Therefore we opted out on creating our own E-Mail engine in order to provide more value to the system.
- We also proposed on creating add ons for the applications on later date in case a customization is required where integration with native productivity tools is require.

Benefits

The solutions proposed by our designers and thinkers and the implementation of these solutions by our developers completed the project successfully. Following are the positives of these development efforts:

- Platform/Database Independent Solution
- On time delivery of the solution.
- Objectives were met successfully.
- Future plans to start with Web Based and PDA version of the CRM application.

Third Party Tools and Software

We used the following software and tools to develop the software in order to meet all the requirements of the project:

- *Development IDE:* Net beans
- *Development platform:* Java
- *Database Layer:* Hibernate
- *Back end:* HSQLDB
- *GUI:* Swings component suites

Other utilities and libraries:

- OpenOffice.org developers API were used to create Client OpenOffice.org bridge.
- Calendar API were used to create the Calendar module of the application.
- Mail API were used to create E-Mail module
- Jasper Reports was used to create reporting engine of the application.

Question

1. Differentaite between technical and business situation.

2. Discuss third party tools and software.

10.7 Summary

- Software is a general term used to describe a collection of computer programs, procedures and documentation that perform some tasks on an operating system.
- Computer hardware is any physical device, something that you are able to touch and software is a collection of instructions and code installed into the computer and cannot be touched.
- System software makes the operation of a computer system more effective and efficient. It helps the hardware components work together, and provides support for the development and execution of application software (programs).
- Personal assistance software allows us to use personal computers for storage and retrieval of our personal information, as well as planning and management of our schedules, contacts, finances, and inventory of important items.
- Open source software (OSS) is computer software that has its underlying 'source-code' made available under a license.
- An integrated development environment (IDE) is a programming environment that has been packaged as an application program, typically consisting of a code editor, a compiler, a debugger, and a graphical user interface (GUI) builder.

10.8 Keywords

Application software: It is a set of one or more programs designed to solve a specific problem, or do a specific task.

Compiler: It is a computer program (or set of programs) that transforms source code written in a programming language (the source language) into another computer language.

Database software: *A database* is a collection of related data stored and treated as a unit for information retrieval purposes.

Integrated development environment (IDE): It is a software application that provides comprehensive facilities to computer programmers for software development.

***Operating Systems*:** Operating system software takes care of effective and efficient utilization of all hardware and software components of a computer system.

Software: It is a collection of instructions that enables a user to interact with the computer or have the computer perform specific tasks for them. Without any software the computer would be useless.

System software: It is a set of one or more programs designed to control the operation and extend the processing capability of a computer system.

10.9 Review Questions

1. Define software.
2. Differentiate the relationship between hardware and software.
3. Explain the type of software.
4. Differentiate between system and application software.
5. Differentiate between system and programming software.
6. Write the short notes on:

(a) Education Software

(b) Graphics software

(c) Database software.

7. Write the short notes on:

(a) Compilers

(b) Debuggers
(c) Interpreters.
8. .Explain the open source software.
9. What are the uses of open source software?
10. Describe the integrated development environment.

Answers for Self Assessment Questions

1. (a) 2. (a) 3. (b) 4. (c) 5. (a)

10.10 Further Studies

1. Basics of Software Engineering Experimentation by Natalia Juristo, Ana M. Moreno
2. Computer Fundamentals by P. K. Sinha

11

Operating System

STRUCTURE

11.0 Objectives

11.1 Introduction

11.2 Why Use Operating Systems

11.3 Meaning of Operating System

11.4 Functions of Operating System

11.5 Types of Reboot

11.6 The Booting Process

11.7 Types of Operating System

11.8 Summary

11.9 Keywords

11.10 Review Questions

11.11 Further Studies

11.0 Objectives

After studying this chapter, you will be able to:

- Explain the use of operating system
- Discuss the function and types of operating system
- Explain the types of reboot
- Explain briefly the booting process

11.1 Introduction

Modern general-purpose computers, including personal computers and mainframes, have an operating system to run other programs, such as application software. Examples of operating systems for personal computers include Microsoft Windows, Mac OS (and Darwin), UNIX, and LINUX. The lowest level of any operating system is its kernel. This is the first layer of software loaded into memory when a system boots or starts up. The kernel provides access to various common core services to all other system and application programs. These services include, but are not limited to: disk access, memory management, task scheduling, and access to other hardware devices.

As well as the kernel, an operating system is often distributed with tools for programs to display and manage a graphical user interface (although Windows and the Macintosh have these tools built into the operating system), as well as utility programs for tasks such as managing files and configuring the operating system. They are also often distributed with application software that does not relate directly to the operating system's core function, various camps advocate microkernels, monolithic kernels, and so on.

Operating systems are used on most, but not all, computer systems. The simplest computers, including the smallest embedded systems and many of the first computers did not have operating systems. Instead, they relied on the application programs to manage the minimal hardware themselves, perhaps with the aid of libraries developed for the purpose. Commercially-supplied operating systems are present on virtually all modern devices described as computers, from personal computers to mainframes, as well as mobile computers such as PDAs and mobile phones.

11.2 Why Use Operating Systems

An operating system (OS) is a software program that manages the hardware and software resources of a computer. The OS performs basic tasks, such as controlling and allocating memory, prioritizing the processing of instructions, controlling input and output devices, facilitating networking, and managing files. The first computers did not have operating systems. However, software tool for managing the system and simplifying the use of hardware appeared very quickly afterwards and gradually expanded in scope. By the early 1960s, commercial computer vendors were supplying quite extensive tools for streamlining the development, scheduling, and execution of jobs on batch processing systems. Examples were produced by UNIVAC and Control Data Corporation, amongst others.

Through the 1960s, several major concepts were developed, driving the development of operating systems. The development of the IBM System/360 produced a family of mainframe computers available in widely differing capacities and price points, for which a single operating system OS/360 was planned (rather than developing ad-hoc programs for every individual model). This concept of a single OS spanning an entire product line was crucial for the success of System/360 and, in fact, IBM's current mainframe operating systems are distant descendants of this original system; applications written for the OS/360 can still be run on modern machines. OS/360 also contained another important advance: the development of the hard disk permanent storage device (which IBM called DASD). Another key development was the concept of time-sharing: the idea of sharing the resources of expensive computers amongst multiple computer users interacting in real time with the system. Time sharing allowed all of the users to have the illusion of having exclusive access to the machine; the Multics timesharing system was the most famous of a number of new operating systems developed to take advantage of the concept.

Commercially-popular minicomputer operating system was VMS. One notable early operating system was CP/M, which was supported on many early microcomputers and was largely cloned in creating MS-DOS, which became wildly popular as the operating system chosen for the IBM PC (IBM's version of it was called IBM-DOS or PC-DOS), its successors making Microsoft one of the world's most profitable companies. The major alternative throughout the 1980s in the microcomputer market was Mac OS, tied intimately to the Apple Macintosh computer.

By the 1990s, the microcomputer had evolved to the point where, as well as extensive GUI facilities, the robustness and flexibility of operating systems of larger computers became increasingly desirable. Microsoft's response to this change was the development of Windows NT, which served as the basis for Microsoft's entire operating system line starting in 1999. Apple rebuilt their operating system on top of a UNIX core as Mac OS X, released in 2001. Hobbyist-developed reimplementation's of UNIX, assembled with the tools from the GNU project, also became popular; versions based on the Linux kernel are by far the most popular, with the BSD derived UNIX holding a small portion of the server market. The growing complexity of embedded devices has a growing trend to use embedded operating systems on them.

Did you know?

Multics was an inspiration to a number of operating systems developed in the 1970s, notably UNIX.

11.2.1 Present Scenario

Command line interface (or CLI) operating systems can operate using only the keyboard for input. Modern OS's use a mouse for input with a graphical user interface (GUI) sometimes implemented as a shell. The appropriate OS may depend on the hardware architecture, specifically the CPU, with only Linux and BSD

running on almost any CPU. Windows NT has been ported to other CPUs, most notably the Alpha, but not many. Mainframe computers and embedded systems use a variety of different operating systems, many with no direct connection to Windows or UNIX, but typically more similar to Unix than Windows.

- Personal computers
- IBM PC compatible - Microsoft Windows and smaller Unix-variants (like Linux and BSD)
- Apple Macintosh - Mac OS X, Windows, Linux and BSD
- Mainframes - A number of unique OS's, sometimes Linux and other UNIX variants.
- Embedded systems - a variety of dedicated OS's, and limited versions of Linux or other OS's.

11.2.2 Unix-like

The UNIX-like family is a diverse group of operating systems, with several major subcategories including System V, (Berkeley Software Distribution) BSD, and Linux. The name "UNIX" is a trademark of the open group which licenses it for use to any operating system that has been shown to conform to the definitions that they have cooperatively developed. The name is commonly used to refer to the large set of operating systems which resemble the original UNIX systems run on a wide variety of machine architectures. They are used heavily as server systems in business, as well as workstations in academic and engineering environments. Free software UNIX variants, such as Linux and BSD, are increasingly popular. They are used in the desktop market as well, for example Ubuntu, but mostly by hobbyists. Some UNIX variants like HP's HP-UX and IBM's AIX are designed to run only on that vendor's proprietary hardware. Others, such as Solaris, can run on both proprietary hardware and on commodity x86 PCs. Apple's Mac OS X, a microkernel BSD variant derived from next step, mach, and free BSD, has replaced Apple's earlier (non-UNIX) Mac OS. Over the past several years, free UNIX systems have supplanted proprietary ones in most instances. For instance, scientific modeling and computer animation were once the province of SGI's IRIX, present scenario they are dominated by Linux-based.

The team at bell labs who designed and developed UNIX went on to develop and inferno, which were designed for modern distributed environments. They had graphics built-in, unlike UNIX counterparts that added it to the design later did not become popular because, unlike many UNIX distributions.

Did you know?

The first microcomputers did not have the capacity or need for the elaborate operating system that had been developed for mainframes and minis; minimalistic operating systems were developed.

11.2.3 Microsoft Windows

The Microsoft Windows family of operating systems originated as a graphical layer on top of the older MS-DOS environment for the IBM PC. Modern versions are based on the newer Windows NT core that first took shape in OS/2 and borrowed from open VMS. Windows runs on 32-bit and 64-bit Intel and AMD computers, although earlier versions also ran on the DEC alpha, MIPS, and power PC architectures (some work was done to port it to the SPARC architecture). As of 2004, Windows held a near-monopoly of around 90% of the worldwide desktop market share, although this is thought to be dwindling due to the increase of interest focused on open source operating systems. It is also used on low-end and mid-range servers, supporting applications such as web servers and database servers. In recent years, Microsoft has spent significant marketing and R&D money to demonstrate that Windows is capable of running any enterprise application.

11.3 Meaning of Operating System

An operating system is an important part of almost every computer system. A computer system can be divided roughly into four components: the hardware, the operating system, the application programs, and the users (Figure 11.1).

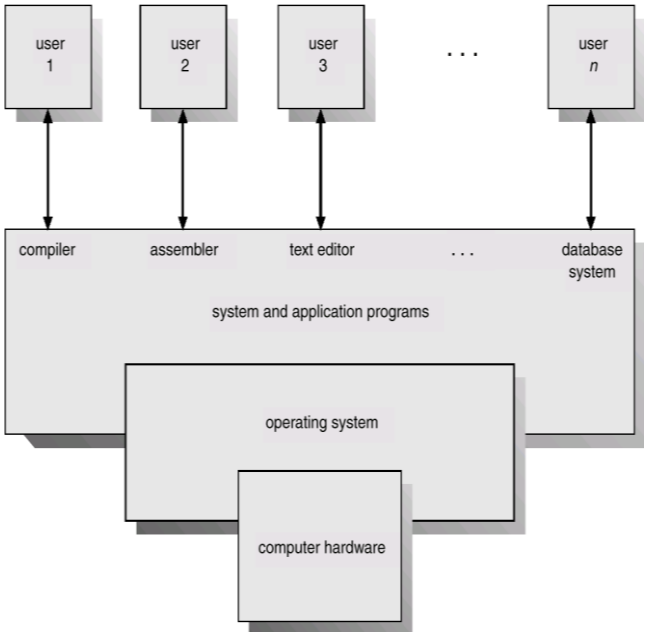


Figure 11.1: Abstract view of the components of a computer system.

The hardware-the central processing unit (CPU), the memory, and the input/output (110) devices-provides the basic computing resources. The application programs-such as word processors, spreadsheets, compilers, and web browsers-define the ways in which these resources are used to solve the computing problems of the users. The operating system controls and coordinates the use of the hardware among the various application programs for the various users. The components of a computer system are its hardware, software, and data.

The operating system provides the means for the proper use of these resources in the operation of the computer system operating systems can be explored from two viewpoints the user and the system.

Did you know?

Since the early 1990s the choice for personal computers has been largely limited to the Microsoft windows family and the UNIX-like family, of which Linux and Mac OS are becoming the major choices.

11.4 Functions of Operating System

Process Management: Process management module takes care of creation and deletion of processer scheduling of system resources to different processers requesting them and providing mechanism for synchronization and communication among processes.

Memory Management: memory management module takes care of allocation and de-allocation of memory space to programs in need of this resource.

File Management: File management module takes care of file-related activities such as organization, storage, retrieval, naming sharing, and protection of files.

Security: Security module protects the resources and information of a computer system against destruction and unauthorized access.

Command Interpretation: Command interpretation module takes care of interpreting user commands, and directing system resources to process the commands. With this mode of interaction with a system, users are not much concerned about hardware details of the system.

11.5 Types of Reboot

The M7225 can be rebooted in three different ways: cold boot, clean boot, and firmware update mode. There is no warm boot on the M7225 since it uses the more modern persistent storage model; when

performing a cold boot, all data is retained. When comparing windows CE or windows mobile 5.0 devices, the M7225's cold boot is the same as an older devices warm boot except the operating system is reloaded from flash memory rather than RAM. The reason behind the naming differences is because the reboot terminology actually describes the devices initial state before the reboot. A cold boot means that the device was completely dead and without any power at all before the reboot was initiated (i.e. a battery is inserted). A warm boot describes that the unit was up and running with a good power source before the reboot was started.

Cold Boot	• Factory settings restored. • All user data lost. • Operating system is reloaded from flash memory
Warm Boot	• User data retained. • Operating system is reloaded from RAM

Older Windows CE 5.0/Windows Mobile 5.0 Terminology

Clean Boot	• Factory settings restored. • Operating system is reloaded from Flash memory. • All user data is lost.
Cold Boot	• User data retained. • Operating system is reloaded from Flash memory.

M7225 with Windows CE 6.0 Terminology

11.5.1 When to Cold Boot and When to Clean Boot

The M7225 is designed for "always-on" operation. Under normal operating conditions, pressing the button will simply suspend and resume the device from ultra-low-power sleep mode. There are only some special cases when reboots are needed such as storing the unit for long periods of time, shipping the M7225, rectifying an application lock-up, updating the system firmware, and restoring factory settings and applications. When the unit has had an application fault or has entered a locked-up state, the unit should be cold booted. When requiring a factory reset the unit should be clean booted. Refer to the firmware update guide when updating the unit's internal operating system software.

Performing a cold boot

To cold boot the unit, simply press the reset button located above the key with a stylus on pistol grip models. On non-pistol grip models, remove the main battery, press the reset button, and reinsert the battery.

Performing a clean boot

To clean boot the unit, hold down the button while pressing the reset button located above the key with a stylus on pistol grip models. On non-pistol grip models, remove the main battery, press the reset button, and while holding the button reinsert the battery.

Did you know?

When storing or shipping the M7225 while booting, the unit should be fully powered down, otherwise it will harm your system.

Self Assessment Questions

1. Which of the command is used to remove files?

(a).erase (b).rm

(c).delete (d).dm

2. The commonly used UNIX commands like date, Is, cat, etc. are stored in

(a)./dev directory (b)./bin and /usr/bin directories
(c)./unix directory (d)./tmp directory

3. When a computer is first turned on or restarted, a special type of absolute loader called ____ is executed
(a).Compile and Go loader (b).Boot loader
(c).Bootstrap loader (d).Relating loader

4. Which of the following Operating systems is better for implementing a Client-Server network
(a).MS DOS (b).Windows 95
(c) Windows 98 (d) Windows 2000

5. The operating system manage
(a).Memory (b) Processes
(c).Disks and I/O devices (d) all of the above

11.6 The Booting Process

Most computers are also capable of booting over a computer network. In this scenario, the operating system is stored on the disk of a server, and certain parts of it are transferred to the client using a simple protocol such as the trivial file transfer protocol (TFTP) after these parts have been transferred, the operating system then takes over control of the booting process.

11.6.1 Booting From Different Operating System

Hard Disk and Partitions

Partitioning is a process of dividing the hard disk into several chunks, and uses any one of the partition to install OS or use two or more partitions to install multiple OSs. But you can always have one partition, and use up the entire hard disk space to install a single OS, but this will become data management nightmare for users of large hard disks. You can have only four partitions, and these four partitions are called (Primary Partitions) again, if we have a large hard disk, we cannot have only four primary partitions,.(Extended Partition. This extended partition is not a usable partition by itself, but it is like a "container" and it is used to hold logical drives that are this extended partition can be subdivided into multiple logical partitions. (Active Partition**)** active partition is that partition which is flagged as bootable or which contains OS, this is generally a primary partition.

Partition boot sector (PBR): This is the logical first sector that is sector at the start of a primary partition. This is also 512 byte area, which contains some programs to initialize or run OS files all primary partitions have it is own PBRs.

Extended boot sector (EBR): This is the logical first sector that is the sector at the start of the extended partition. This EBR contains a partition which lists the available logical partitions inside extended partition. That is it contains the starting addresses of each logical partitions.

Logical extended boot sector (LEBR): This is the logical first sector residing at the start of each logical partition. This is similar to PBR for primary partitions.

Single OS Boot Process

Whenever PC is turned ON, BIOS takes the control, and it performs a lot of operations. It checks the hardware, ports etc and finally it loads the MBR program into memory (RAM). Now, MBR takes control of the booting process. Functions of MBR, when there is only one OS are installed in the system are as given below:-

- The boot process starts by executing code in the first sector of the disk, MBR.
- The MBR looks over the partition table to find the Active Partition.
- Control is passed to that partition's boot record (PBR) to continue booting.
- The PBR locates the system-specific boot files

- Then these boot files continue the process of loading and initializing the rest of the OS.

Multiple OS Boot Process

Whenever there are multiple OSs, be it multiple windows or windows with Linux, then boot process will be slightly different. Actually, there can be two different types of Boot Process in multiple OS environment, Microsoft way and non-Microsoft way. Windows multiboot can be represented as below.

- When BIOS hands over control to Microsoft MBR, this program looks into Partition Table for Active Partition.
- Then it hands over the control to the PBR of Active Partition. In this case, the Active Partition is where Win98 was installed.
- But Win98 PBR has been altered by WinXP, and it no longer contains Win98 boot program (like io.sys or msdos.sys). But it contains ntldr.peculiarity is that one OS's boot program is in another OS's Partition.
- ntldr looks into boot.ini file and finds out the Microsoft OSs installed in the System and displays the option menu.
- When user slects Win98, the file bootsect.dos (which is in same partition) is executed, and if WinXP is selected, ntoskrnl is executed (which is in another partition).
- The good thing about Microsoft way is that, it's very easy to configure

This boot process also has two limitations

- There can be only one real mode DOS based OSs like Win95/Win98 along with NT based OSs. If you want both Win95, Win98 with any NT based OS, then it is simply not possible.
- Microsoft MBR looks for active status in primary partitions only and not in logical partitions. This means, Microsoft OSs should be installed in primary partitions only if it should be bootable (For this reason itself, WinXP boot file ntldr is placed in primary partition of Win98 instead of it is own Logical Partition).

11.7 Types of Operating System

There are generally four types, categorized based on the types of computers they control and the sort of applications they support. The categories are real-time operating system, single user single task, and single user multitasking and multi-user.

11.7.1 Real-Time Operating System (RTOS)

Real-time operating systems are used to control machinery, scientific instruments and industrial systems such as embedded systems (programmable thermostats, household appliance controllers), industrial robots, spacecraft, industrial control (manufacturing, production, power generation, fabrication, and refining), and scientific research equipment.

A real time operating system typically has very little user-interface capability, and no end-user utilities, since the system will be a "sealed box" when delivered for use. A very important part of an RTOS is managing the resources of the computer so that a particular operation executes in precisely the same amount of time, every time it occurs. In a complex machine, having a part move more quickly just because system resources are available may be just as catastrophic as having it not move at all because the system is busy.

A RTOS facilitates the creation of a real-time system, but does not guarantee the final result will be real-time; this requires correct development of the software. A RTOS does not necessarily have high throughput; rather, an RTOS provides facilities which, if used properly, guarantee deadlines can be met generally (soft real-time) or deterministically (hard real-time).

A RTOS will typically use specialized scheduling algorithms in order to provide the real-time developer with the tools necessary to produce deterministic behavior in the final system. A RTOS is valued more for

how quickly and/or predictably it can respond to a particular event than for the given amount of work it can perform over time. Key factors in an RTOS are therefore a minimal interrupt latency (the time between the generation of an interrupt by a device and the servicing of the device which generated the interrupt) and a minimal thread switching latency (the time needed by the operating system to switch the CPU to another thread).

11.7.2 Single User, Single Task

The name implies, this operating system is designed to manage the computer so that one user can effectively do one thing at a time. The palm OS for palm handheld computers is a good example of a modern single-user, single-task operating system.

11.7.3 Single User, Multitasking

This is the type of operating system most people use on their desktop and laptop computers today. Microsoft's windows and Apple's Mac OS platforms are both examples of operating systems that will let a single user have several programs in operation at the same time. For example, it's entirely possible for a windows user to be writing a note in a word processor while downloading a file from the Internet while printing the text of an e-mail message.

11.7.4 Multi-user

Multi-user defines operating system or application software that allows concurrent access by multiple users of a computer. A multi-user operating system allows many different users to take advantage of the computer's resources simultaneously. The operating system must make sure that the requirements of the various users are balanced, and that each of the programs they are using has sufficient and separate resources so that a problem with one user does not affect the entire community of users. UNIX, VMS and mainframe operating systems, such as *MVS*, are examples of multi-user operating systems. Time-sharing systems are multi-user systems. Most batch processing systems for mainframe computers may also be considered "multi-user", to avoid leaving the CPU idle while it waits for I/O operations to complete.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Prepare a list of system requirements before installing an operating system.

……..……

……..……

……

11.8 Summary

- An operating system (OS) is a software program that manages the hardware and software.
- A better method of implementing multitasking is for an operating system to employ preemptive.
- Multiprogramming is interleaved execution of two or more different and independent programs by a computer.
- It provides strict system security.
- Operating system is an integrated set of programs that controls the resources (CPU, memory I/O devices, etc.)

11.9 Keywords

Asymmetric multi processing: Asymmetric hardware systems commonly dedicated individual processors to specific tasks.

Multitasking: An operating system that utilizes multitasking is one that allows more than one program to run simultaneously.

Operating System: An operating system (OS) is a software program that manages the hardware and software resources of a computer.

Real time operating system (RTOS): Real-time operating systems are used to control machinery, scientific instruments and industrial systems such as embedded systems.

Symmetric multi processing: SMP involves a multiprocessor computer architecture where two or more identical processors can connect to a single shared main memory.

UNIX-like: UNIX-like family is a diverse group of operating systems, with several major subcategories including System V, BSD, and Linux.

11.10 Review Questions

1. What does an operating system do?
2. What are the three main purposes of an operating system?
3. What is the main advantage of multiprogramming?
4. What are the main differences between operating systems for mainframe computers and PCs?
5. What are the types of reboot? Explain.
6. How do clustered systems differ from multiprocessor systems?
7. How can differentiate between the multiprocessor and multitasking?
8. What is the real-time operating system? Explain with examples.
9. Define the single-processor systems and multiprocessor systems?
10. Define the essential properties of the following types of operating systems:
    a. Batch
    b. Interactive
    c. Time sharing
    d. Real time
    e. Network

Answers for Self Assessment:

1. (b) 2. (b) 3. (c) 4. (d) 5. (d)

11.11 Further Studies

1. OPERATING SYSTEMS by WILLIAM STALLINGS
2. Computer Fundamentals by P. K. Sinha

12

Disk Operating System (DOS)

STRUCTURE

12.0 Objectives

12.1 Introduction

12.2 What is DOS?

12.3 Function of DOS?

12.4 Versions of DOS

12.5 The DOS Commands

12.6 Non-executable and Executable files

12.7 Summary

12.8 Keywords

12.9 Review questions

12.10 Further Studies

12.0 Objectives

After studying this chapter, you will be able to:

- Discuss the versions of DOS
- Explain the DOS
- Explain the DOS system files
- Discuss about the DOS commands

12.1 Introduction

The DOS Stands for "Disk Operating System." DOS was the first operating system used by IBM-compatible computers. It was originally available in two versions that were essentially the same, but marketed under two different names. "PC-DOS" was the version developed by IBM and sold to the first IBM-compatible manufacturers. "MS-DOS" was the version that Microsoft bought the rights to, and was bundled with the first versions of Windows.

DOS uses a command line, or text-based interface, that allows the user to type commands. By typing simple instructions such as pwd (print working directory) and cd (change directory), the user can browse the files on the hard drive, open files, and run programs. While the commands are simple to type, the user must know the basic commands in order to use DOS effectively (similar to Unix). This made the operating system difficult for novices to use, which is why Microsoft later bundled the graphic-based Windows operating system with DOS.

The first versions of Windows (through Windows 95) actually ran on top of the DOS operating system. This is why so many DOS-related files (such as .INI, .DLL, and .COM files) are still used by Windows. However, the Windows operating system was rewritten for Windows NT (New Technology), which

enabled Windows to run on its own, without using DOS. Later versions of Windows, such as Windows 2000, XP, and Vista, also do not require DOS.
DOS is still included with Windows, but is run from the Windows operating system instead of the other way around. The DOS command prompt can be opened in Windows by selecting "Run..." from the Start Menu and typing cmd.

12.2 What is DOS?

The operating system is used for operating the system or the computer. It is a set of computer programs and also known as DOS. The main functions of DOS are to manage disk files, allocate system resources according to the requirement. The DOS provides features essential to control hardware devices such as keyboard, screen, disk devices, printers, modems and programs.
Basically, DOS is the medium through which the user and external devices attached to the system communicate with the system. The DOS translate the command issued by the user in the format that is understandable by the computer and instruct computer to work accordingly. It also translates the result and any error message in the format for the user to understand.
At the heart of DOS there are three files. One is always called command.com. The other two are by tradition hidden and have different names depending on the DOS version. In PC-DOS as well as in DR-DOS, they are called ibmbio.com and ibmdos.com. In MS-DOS. They are called io.sys and msdos.sys.
If you place a floppy into your computer and type sys a: at the command prompt, these three files will be copied to the floppy, and the floppy will be made bootable. You can start a computer from this floppy now.

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: Install your server and configure the service processor.

……..……

……..……

……

12.2.1 Physical Structure of Disk

The files in the computer come from various sources. Some files come with DOS, while other comes with publications such as a word processor. These files contain codes and other information that is necessary to make the computer application operational. Before long, there will be hundreds or even thousands of files in the computer, which can make it difficult to locate specific files.
The names of all the files created in a disk are stored in its directory. Directory is just like a file folder, which contain all the logically related files. The DOS files are organized in a hierarchical or an inverted tree like structure. The general analogy is with a file cabinet containing a number of drawers, which in turn may contain folders. The content of these folders is the needed information.
The file cabinet here is the ROOT DIRECTORY, the drawer is INDIVIDUAL DIRECTORY, the folders are SUBDIRECTORY and the information in these folders may in turn be classified as FILES.
Otherwise, the large number of files that get created for various purposes in a disk can make the directory huge and difficult to view and manage. Therefore, DOS enables the user to organize the files in a disk into directories and sub directories in a hierarchical structure. Directories can contain other directories. A directory within another directory is called a sub directory.
Of course, there may be sub directories of sub directories, but a file name is the furthest you may descend down the (inverted) tree of directories and files. Thus, a file name corresponds to a tree leaf, a sub-directory to a branch, the directory to the trunk, and the root directory to the root of the tree, hence the name ROOT DIRECTORY.

Sample of Directory Structure

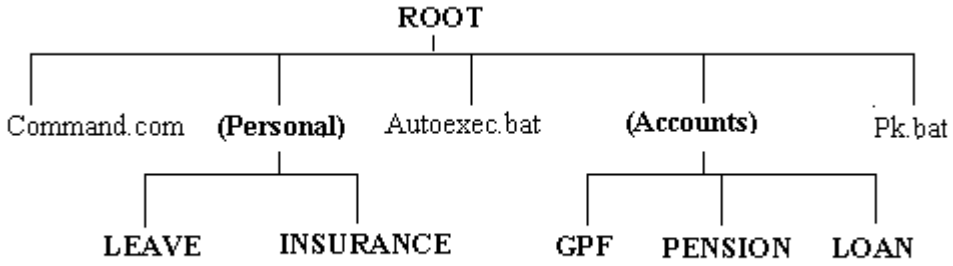


Figure 12.1: The directory/sub-directory is represented in bold letters.

12.2.2 Drive Name

A path is the route that leads from the root directory of a drive to the file you want to use.
For example, to access the NOS.LET file in the LETTER subdirectory of NOS directory, DOS must go from the ROOT (\) directory through the NOS directory to the LETTER directory, as shown in the Figure 12.2:

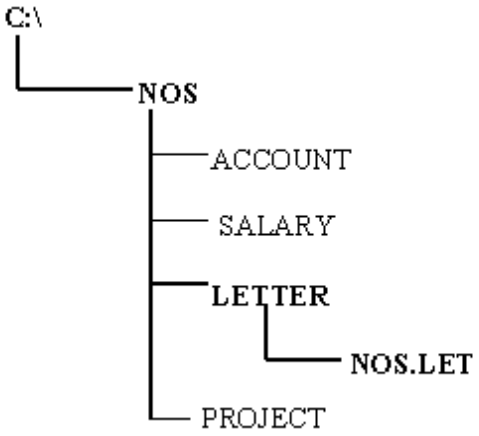


Figure 12.2: directory.

To specify the same path at the command prompt, you would type it as shown in the illustration:

Caution

Without using an emulator, DOS application run under following versions of Windows like, Windows XP x64, Windows Vista x64 and Windows 7 x64 editions, as these do not contain NTVDM as well as 16-bit DOS applications cannot run directly because COMMAND.COM is missing.

12.3 Function of DOS?

An operating system is a group of computer programs that coordinates all the activities among computer hardware devices. It is the first program loaded into the computer by a boot program and remains in memory at all times.

Functions of an operating system

The basic functions of an operating system are:

Booting the computer

Performs basic computer tasks eg managing the various peripheral devices eg mouse, keyboard
Provides a user interface, e.g. command line, graphical user interface (GUI)

Handles system resources such as computer's memory and sharing of the central processing unit (CPU) time by various applications or peripheral devices
Provides file management which refers to the way that the operating system manipulates, stores, retrieves and saves data.
Booting the computer
The process of starting or restarting the computer is known as booting. A cold boot is when you turn on a computer that has been turned off completely. A warm boot is the process of using the operating system to restart the computer.

Performs basic computer tasks
The operating system performs basic computer tasks, such as managing the various peripheral devices such as the mouse, keyboard and printers. For example, most operating systems now are plug and play which means a device such as a printer will automatically be detected and configured without any user intervention.

Provides a user interface
A user interacts with software through the user interface. The two main types of user interfaces are: command line and a graphical user interface (GUI). With a command line interface, the user interacts with the operating system by typing commands to perform specific tasks. An example of a command line interface is DOS (disk operating system). With a graphical user interface, the user interacts with the operating system by using a mouse to access windows, icons, and menus. An example of a graphical user interface is Windows Vista or Windows 7.
The operating system is responsible for providing a consistent application program interface (API) which is important as it allows a software developer to write an application on one computer and know that it will run on another computer of the same type even if the amount of memory or amount of storage is different on the two machines.

Handles system resources
The operating system also handles system resources such as the computer's memory and sharing of the central processing unit (CPU) time by various applications or peripheral devices. Programs and input methods are constantly competing for the attention of the CPU and demand memory, storage and input/output bandwidth. The operating system ensures that each application gets the necessary resources it needs in order to maximise the functionality of the overall system.

Provides file management
The operating system also handles the organisation and tracking of files and directories (folders) saved or retrieved from a computer disk. The file management system allows the user to perform such tasks as creating files and directories, renaming files, coping and moving files, and deleting files. The operating system keeps track of where files are located on the hard drive through the type of file system. The type two main types of file system are File Allocation table (FAT) or New Technology File system (NTFS).

Types of file system
- File Allocation table (FAT)
- New Technology file system (NTFS)

File Allocation table (FAT) uses the file allocation table which records, which clusters are used and unused and where files are located within the clusters.

NTFS is a file system introduced by Microsoft and it has a number of advantages over the previous file system, named FAT32 (File Allocation Table).

One major advantage of NTFS is that it includes features to improve reliablity. For example, the new technology file system includes fault tolerance, which automatically repairs hard drive errors without displaying error messages. It also keeps detailed transaction logs, which tracks hard drive errors. This can help prevent hard disk failures and makes it possible to recover files if the hard drive does fail.
NTFS also allows permissions (such as read, write, and execute) to be set for individual directories and files.

12.4 Versions of DOS

The origin of MS-DOS can be traced back to 1980 when Seattle Computer Producers developed a microcomputer operating system for in house use and was called QDOS and then renamed 86-DOS in late 1980 after modification.
Microsoft bought the rights to 86-DOS which had a contract with IBM to produce an operating system for IBM's new PC.
The 86-DOS was modified and called PC-DOS 1.0. When PC compatible machines were produced they used a similar version of PC-DOS called MS-DOS. Version 1.0 of DOS was released in 1981 giving single sided disc drive capability. Version 1.1 of DOS was released in 1982 giving double sided disc drive capability (single sided on V1.0) and output to a serial printer. Version 2.0 released in 1983 gave hard disc support, sophisticated hierarchical file directory, installable device Drivers and file handles. Version 3.0 released in 1984 gave improved hard disc and network support. Version 3.3 released in 1987 continued this trend. Version 4.0 released in 1988 provided the DOSSHELL, expanded memory driver, larger than 32Mb hard disc partitions. Version 5.0 released in 1991 designed as an upgrade which enabled device drivers to be placed in upper memory leaving more conventional memory available to programs.

Did You Know?
Several competing products, during its life, were released for the x86 platform, and MS-DOS itself would go through eight versions, until development ceased in 2000. Ultimately it was the key product in Microsoft's growth from a programing languages company to a diverse software development firm, providing the company with essential revenue and marketing resources. It was also the underlying basic operating system on which early versions of Windows ran as a GUI.

Exercise: Check Your Progress 2
Note: i) Use the space below for your answer.
Ex1: Prepare flowchart for the steps of installing the system specific drivers in your PC.
……..……

……..……

……

12.5 The DOS Commands

There are two types of commands:
12.5.1 Internal commands
These are those commands for which the code for executing them is stored in COMMAND.COM and they execute without the requirement of any other file. When aninternal command is issue by the user, the transient portion knows what to do and executes that command.
The list of internal commands is as follows:
1. ***Date:*** It displays the current system date and prompts you to change the date, if necessary.
Syntax: C:\>date <mm-dd-yy>

2. *Time:* It sets the computer's internal clock if commandis issued with parameters otherwise displays the systemtime.
Syntax: C:\>time <hours: minutes: seconds: hundredths>
3. *Clear Screen:* It clears the screen. The cleared screenshows only the command prompt and the blinking cursor.
Syntax: C:\>cls
4. *Ver:* Ver command displays the MS-DOS version no.
Syntax: C:\>ver
5. *Vol:* It displays the current volume label and the serialnumber of the specified drive. If drive specifier is omittedVol takes the current drive by default.
Syntax: C:\vol <d:>
6. ***Break:*** It enables the DOS to check for Ctrl + C and Ctrl – break as the signal to terminate the current program lessfrequently. If you run Break without any parameters, thecurrent state of break checking is displayed.
Syntax: C:\>break <on/off>
7. *MKDIR (or MD):* It creates a multilevel directory.
Syntax: C:\>MKDIR (or MD) <drive:> <path>
8. *CHDIR (or CD):* It changes the current directory or displays the name of the current directory.
Syntax: C:\>CHDIR (or CD) <drive:> < path>
9. *Dir:* It displays the list of the files and sub-directories thatare in the directory specified.
Syntax: C:\> dir <drive:> <path> <filename>
10. *Copy:* It copy one or more files to the location specified and can also be used to combine.
Syntax: C:\>copy <source> <target>
11. *EDLIN* :It is the line-oriented text editor.
Syntax: C:\>EDLIN <filename>
12. *Del:* It deletes the files specified.
Syntax : C:\>del <drive:> <path> <filename>
13. *Prompt:* It chanes the appearance of the command prompt or displays the current prompt.
Syntax: C:\>prompt <text>
14. *RMDIR (or RD):* It removes the directory.
Syntax: C:\>RMDIR (or RD) <drive:> <path>
15. *Rename (or REN):* It changes the name of the file or files specified and can also rename all files matching thespecified filename.
Syntax: C:\>rename (or REN) <drive:> <path><original filename> <changed filename>
16. *Type:* It displays the contents of a text file or view thecontents of any text file without modifying it.
Syntax: C:\>type <drive:> <path> <filename>
17. *Path:* It is used to earch for the executable files in thedirectories specified.
Syntax: C:\> path <drive:> <path> <……>
18. *Verify:* It verifies that the files are written correctly tothe disk.
Syntax: C:\>verify

12.5.2 External commands

These are the independent executable files that can be used as separate utilities also. Information about these files is also stored in the transient portion. When an external command is issued, the transient portion picks up the relevant file of that command and executes it.
Following are the list of external commands:
1. **Attrib***:* It lets you display or alter the attributes assigned to a file or directory.
Syntax: C:\>attrib <+A/-A><+H/-H><+R/-R><+S/-S><d:><path> <filename>
+A: add archive attributes
-A: remove archive attribute
+H: add hidden attributes

-H: remove hidden attribute
+R: add read-only attributes
-R: remove read-only attribute
+S: add system attributes
-S: remove system attribute
2. **Chkdsk*:*** It checks a disk's file allocation table entries for errors and find whether the files are o.k. or not.
Syntax: C:\>chkdsk <d:> <path> <filename>
3. **Comp*:*** It compares two files byte-by-byte and reports the differences.
Syntax: C:\>comp <d:>[<path + filename>]<d:>[<path+ filename>]
4. **Format*:*** It creates a new directory and file allocation table for the disk.
Syntax: C:\>format <drive name>
5. **Scandisk*:*** It checks a drive for errors and corrects any problems that it finds.
Syntax: C:\> scandisk <drive name>
6. **Tree*:*** The structure of the specified drive from the specified directory down, listing all the sub-directories it encounters is displayed graphically.
Syntax: C:\>tree <drive name>
7. **Deltree*:*** It deletes a directory and all the files and sub-directories contained within it.
Syntax: C:\>deltree <drive><path>
8. **XCopy*:*** It is used to copy all the files in a directory, including the files in the sub-directories of that directory.
Syntax: C:\>xcopy <source> <destination>
9. **Print*:*** It prints a text file.
Syntax: C:\>print <filename>
10. **FDisk*:*** It is the tool, DOS provides for setting up and managing hard disk partitions.
Syntax: C:\>fdisk
11. **Label*:*** It lets you create, modify or delete the label assigned to a disk.
Syntax: C:\>label <drive name>
12. **Doskey*:*** It edits command lines, recalls command lines and creates macros.
Syntax: C:\>doskey
13. **More*:*** It displays one screen of output at a time and in end of each screen displays the message.
Syntax: C:\>more <drive:> <path> <filename>
14. **Backup *:*** It lets the user to take the backup of hard disk files to floppies.
Syntax: C:\>backup <source> <destination>
15. **Restore*:*** It restores the files that were backed up on a floppy.
Syntax: C:\>restore <drive1> <drive2> <path> <filename>
16. **Mem*:*** It displays amount of used and free memory in the system.
Syntax: C:\>mem
17. **FC*:*** It compares two files or sets of files and displays the difference between them.
Syntax: C:\>fc <drive><filename1> <drive><filename2>

12.6 Non-executable and Executable files

Data files - (non-executable)
These are the files that contain the data you produce. They are produced when you are using an application program, such as a word-processor, spreadsheet or database and save the data to disk.
When you type in a file name of a data file, and press the Return (or Enter) key, it will not execute (or run). You will see the message 'Bad command or file name' Data files can only be used within an application.
Executable files
These are program files. The file names end in either COM or EXE. When you type in a file name that ends in COM or EXE, and press the Return (or Enter) key it will execute (or run).

In some cases when you do this you may see a message 'This program requires Microsoft Windows'. This is because that particular application program is written to use Microsoft Windows, and will not run under DOS without Microsoft Windows. You must load Microsoft Windows first, and then run a Windows program, from within Microsoft Windows.

Self Assessment Questions

1. The startup routine runs, when machine boots up is known as…….
(a) POST (b) BOOT up (c) Operating Routine (d) I/O operation.

2. Operating system is also known as……
(a) Database (b) System software (c) Hardware (d) Printer.

3. The maximum length allowed for primary name of a computer file under DOS is ……
(a) 8 (b) 12 (c) 3 (d) None of these.

4. Which of the following could be a valid DOS file specification?
(a) NOSFILE.POST (b) NOSFILE.P.OST
(c) NOSFILE.DOC (d) NOST.FILEDOC.

5. How many characters form a secondary name for a file?
(a) 8 (b) 12 (c) 3 (d) None of these.

6. What is the name given to something that the computer will automatically use unless you tell it otherwise?
(a) a specification (b) a wildcard (c) a default (d) a rule.

12.7 Summary

- The operating system is used for operating the system or the computer. It is a set of computer programs and also known as DOS.
- Basically, DOS is the medium through which the user and external devices attached to the system communicate with the system.
- In DOS, programs are started by typing their name into the command line. Directory is just like a file folder, which contain all the logically related files. Data is stored in individual 512-byte sectors on the hard disk. The hard disk is instead broken into larger pieces called *clusters*, or alternatively, *allocation units*.
- The modern DOS operating system is distributed on 3-5 high density floppy disks.

12.8 Keywords

Backup: It lets the user to take the backup of hard disk files to floppies.
File Allocation table (FAT) uses the file allocation table which records, which clusters are used and unused and where files are located within the clusters.
NTFS is a file system introduced by Microsoft and it has a number of advantages over the previous file system, named FAT32 (File Allocation Table).
Hard Disk Drive**:** A hard disk drive is a device for storing and retrieving digital information, primarily computer data.
Path: It is used to each for the executable files in the directories specified.
Prompt: It changes the appearance of the command prompt or displays the current prompt.

12.9 Review questions

1. What is the history of disk operating system?
2. How many versions of disk operating system?
3. What is the physical structure of disk? Explain the disk name.
4. Discuss about the FAT file system.
5. What is the common DOS Windows file?
6. What are the rules for DOS file and directory name creation?
7. Discuss about the Long File Names (LFNS).
8. Describe the steps in the DOS boot process.
9. How many types of files are in the core DOS operating system?
10. What are the DOS Commands? Explain briefly.

Answers for Self Assessment Questions

1 (b) 2 (b) 3 (a) 4 (b) 5 (b) 6 (c)

12.10 Further Studies

1. http://en.wikipedia.org/wiki/Disk_operating_systemS
2. Computer Fundamentals by P. K. Sinha

13

Programming Languages

STRUCTURE

13.0 Objectives

13.1 Introduction

13.2 Data, Information and Knowledge

13.3 Characteristics of Information

13.4 Comparison between Human Language and Computer Language

13.5 Program and Programming Language

13.6 Programming Development Cycle Algorithm

13.7 Program Flowcharts

13.8 Pseudocode

13.9 Programming Approaches

13.10 Programming Paradigms

13.11 Types of Programming Language

13.12 Third/Fourth Generation Language

13.13 Summary

13.14 Keywords

13.15 Review Questions

13.16 Further Studies

13.0 Objectives

After studying this chapter, you will be able to:

- Explain the data information and knowledge.
- Discuss the characteristics of information.
- Discuss the comparison between human language and computer language.
- Define the program and programming language
- Explain the programming development cycle algorithm.
- Discuss the program flowcharts.
- Define the pseudocode.
- Explain the approaches and programming paradigms.
- Explain the types of programming language.

- Discuss about the third/fourth generation language.

13.1 Introduction

A program is a set of instructions that tell the computer to do various things; sometimes the instruction it has to perform depends on what happened when it performed a previous instruction. This section gives an overview of the two main ways in which you can give these instructions or "commands" as they are usually called. One way uses an interpreter, the other a compiler. As human languages are too difficult for a computer to understand in an unambiguous way, commands are usually written in one or other languages specially designed for the purpose.

13.2 Data, Information and Knowledge

13.2.1 Data

Data is the raw material of information. It is the product of research and discovery. A single piece of data has no meaning unless the context is understood. Data needs to be transformed to information.
Data represents unorganized and unprocessed facts.

- Usually data is static in nature.
- It can represent a set of discrete facts about events.
- Data is a prerequisite to information.
- An organization sometimes has to decide on the nature and volume of data that is required for creating the necessary information.

13.2.2 Information

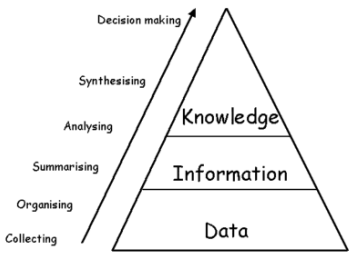
Information is a flow of messages. The patterns and relationship in the data is pointed out and discussed. The data is made informative and must be put into a context and linked like data.

- Information can be considered as an aggregation of data (processed data) which makes decision making easier.
- Information has usually got some meaning and purpose.

13.2.3 Knowledge

Knowledge is a multifaceted concept with multilayered meaning. The history of philosophy since the classical Greek period can be regarded as never ending search for the meaning of knowledge

- By knowledge we mean human understanding of a subject matter that has been acquired through proper study and experience.
- Knowledge is usually based on learning, thinking, and proper understanding of the problem area.
- Knowledge is not information and information is not data.
- Knowledge is derived from information in the same way information is derived from data.
- We can view it as an understanding of information based on its perceived importance or relevance to a problem area.
- It can be considered as the integration of human perceptive processes that helps them to draw meaningful conclusions.



13.3 Characteristics of Information

Good information is that which is used and which creates value. Experience and research shows that good information has numerous qualities. Good information is relevant for its purpose, sufficiently accurate for its purpose, completes enough for the problem, reliable and targeted to the right person. It is also communicated in time for its purpose, contains the right level of detail and is communicated by an appropriate channel, i.e. one that is understandable to the user.
Further details of these characteristics related to organizational information for decision-making follows.

13.3.1 Availability/Accessibility

Information should be easy to obtain or access. Information kept in a book of some kind is only available and easy to access if you have the book to hand. A good example of availability is a telephone directory, as every home has one for its local area. It is probably the first place you look for a local number. But nobody keeps the whole country's telephone books so for numbers further afield you probably phone a directory enquiry number. For business premises, say for a hotel in London, you would probably use the Internet.
Businesses used to keep customer details on a card-index system at the customer's branch. If the customer visited a different branch a telephone call would be needed to check details. Now, with centralized computer systems, businesses like banks and building societies can access any customer's data from any branch.

13.3.2 Accuracy

Information needs to be accurate enough for the use to which it is going to be put. To obtain information that is 100% accurate is usually unrealistic as it is likely to be too expensive to produce on time. The degree of accuracy depends upon the circumstances. At operational levels information may need to be accurate to the nearest penny on a supermarket till receipt, for example. Accuracy is important. As an example, if government statistics based on the last census wrongly show an increase in births within an area, plans may be made to build schools and construction companies may invest in new housing developments. In these cases any investment may not be recouped.

13.3.3 Reliability or Objectivity

Reliability deals with the truth of information or the objectivity with which it is presented. You can only really use information confidently if you are sure of its reliability and objectivity.
When researching for an essay in any subject, we might make straight for the library to find a suitable book. We are reasonably confident that the information found in a book, especially one that the library has purchased, is reliable and (in the case of factual information) objective. The book has been written and the author's name is usually printed for all to see. The publisher should have employed an editor and an expert in the field to edit the book and question any factual doubts they may have. In short, much time and energy goes into publishing a book and for that reason we can be reasonably confident that the information is reliable and objective.
Compare that to finding information on the Internet where anybody can write unedited and unverified material and 'publish' it on the web. Unless you know who the author is, or a reputable university or government agency backs up the research, then you cannot be sure that the information is reliable. Some Internet websites are like vanity publishing, where anyone can write a book and pay certain (vanity) publishers to publish it.

13.3.4 Relevance/Appropriateness

Information should be relevant to the purpose for which it is required. It must be suitable. What is relevant for one manager may not be relevant for another. The user will become frustrated if information contains data irrelevant to the task in hand.

For example, a market research company may give information on users' perceptions of the quality of a product. This is not relevant for the manager who wants to know opinions on relative prices of the product and its rivals. The information gained would not be relevant to the purpose.

13.3.5 Completeness

Information should contain all the details required by the user. Otherwise, it may not be useful as the basis for making a decision. For example, if an organization is supplied with information regarding the costs of supplying a fleet of cars for the sales force, and servicing and maintenance costs are not included, then a costing based on the information supplied will be considerably underestimated.

Ideally all the information needed for a particular decision should be available. However, this rarely happens; good information is often incomplete. To meet all the needs of the situation, you often have to collect it from a variety of sources.

13.3.6 Level of Detail/Conciseness

Information should be in a form that is short enough to allow for its examination and use. There should be no extraneous information. For example, it is very common practice to summaries financial data and presents this information, both in the form of figures and by using a chart or graph. We would say that the graph is more concise than the tables of figures as there is little or no extraneous information in the graph or chart. Clearly there is a trade-off between level of detail and conciseness.

13.3.7 Presentation

The presentation of information is important to the user. Information can be more easily assimilated if it is aesthetically pleasing. For example, a marketing report that includes graphs of statistics will be more concise as well as more aesthetically pleasing to the users within the organization. Many organizations use presentation software and show summary information via a data projector. These presentations have usually been well thought out to be visually attractive and to convey the correct amount of detail.

13.3.8 Timing

Information must be on time for the purpose for which it is required. Information received too late will be irrelevant. For example, if you receive a brochure from a theatre and notice there was a concert by your favorite band yesterday, then the information is too late to be of use.

13.3.9 Value of Information

The relative importance of information for decision-making can increase or decrease its value to an organization. For example, an organization requires information on a competitor's performance that is critical to their own decision on whether to invest in new machinery for their factory. The value of this information would be high. Always keep in mind that information should be available on time, within cost constraints and be legally obtained.

13.3.10 Cost of Information

Information should be available within set cost levels that may vary dependent on situation. If costs are too high to obtain information an organization may decide to seek slightly less comprehensive information elsewhere. For example, an organization wants to commission a market survey on a new product. The survey could cost more than the forecast initial profit from the product. In that situation, the organization would probably decide that a less costly source of information should be used, even if it may give inferior information.

13.3.11 Difference between Value and Cost

Many students in the past few years have confused the definitions of value and cost. Information gained or used by an organization may have a great deal of value even if it may not have cost a lot. An example would be bookshops, which have used technology for many years now, with microfiche giving way to

computers in the mid to late 1990s. Microfiche was quite expensive and what the bookshops received was essentially a list of books in print. By searching their microfiche by publisher they could tell you if a particular book was in print. Eventually this information became available on CD-ROM. Obviously this information has value to the bookshops in that they can tell you whether or not you can get the book. The cost of subscribing to microfiche was fairly high; subscribing to the CD-ROM version only slightly less so. Much more valuable is a stock system which can tell you instantly whether or not the book is in stock, linked to an on-line system which can tell you if the book exists, where it is available from, the cost and delivery time. This information has far more value than the other two systems, but probably actually costs quite a bit less. It is always up-to-date and stock levels are accurate.

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: Define Data, Information and Knowledge
……..……

……..……

……

13.4 Comparison between Human Language and Computer Language

When some people hear the word 'language' in computer programming language, they instantly think of naturally flowing conversation as occurs in human language. While computer programming language does flow in its own way, this flow is starkly different from the flow of human language. Here are a few major differences between both types of language as well as one similarity just to mix things up a bit.

While humans use language to communicate with each other for a variety of different reasons, computer programming exists for one purpose to tell computers how to perform a task. Granted, there are many different kinds of computer programming languages but they all exist to perform a specific function. While human language exists for communication, it can be used in countless different ways, making it extremely unpredictable and, in a way, unreliable.

Another difference between human language and computer programming language is computer language must stick to a specific script to be of any use in performing its designated task. Human language on the other hand doesn't have to follow a specific script as evidenced in naturally flowing conversations. While official human languages exist, these languages are adapted with many short hand versions such as slang. It computer programming language were used in such a way, it would essentially be useless. One similarity between human language and computer programming language is there are many different types of each. Just as many countries have their own languages, so to exist different programming languages. Just a few examples of the thousands of computer programming languages that currently exist include C and Java. Each programming language was created to perform a specific task that previous ones didn't perform.

Again although human language and computer programming languages are very different in many ways, they both exist to create communication. With humans, language exists so people can communicate easily with each other. With computers, language exists so computers can communicate in order to perform specific tasks.

13.5 Program and Programming Language

13.5.1 Program

A set of coded instructions that a computer can understand to solve a problem or produce a desired result. Two basic types of computer programs are

(1) An operating system, which provides the most fundamental instructions a computer uses in its operations, and
(2) An application program, which runs on the operating system and does a specific job such as word processing.
Programs are written either in one of high-level programming languages (such as BASIC, C, and Java) which are easier but execute relatively slowly, or in one of low-level languages (assembly language or machine language) which are very complex but execute very fast.

13.5.2 Programming Language

Computer program is a set of instructions that guide a computer to execute a particular task. It is like a recipe for a cook in making a particular dish. The recipe contains a list of ingredients called the data or variables, and a list of steps that guide the computer what to do with the data. So programming is the technique of making a computer to perform something you want to do.
Programming or coding is a language that is used by operating systems to perform the task. We know computer understands binary languages with digits 1s and 0s. These binary languages are difficult to understand by human; so we generally use an intermediate language instead of binary language. Again the program uses high-level language that is interpreted into bytes that the computer understands. So a programmer writes a source code and uses a tool or interpreter that allows the computer to read, translate and execute the programs to perform a function.

13.6 Programming Development Cycle Algorithm

In mathematics and computer science, an algorithm is an effective method expressed as a finite list of well-defined instructions for calculating a function. Algorithms are used for calculation, data processing, and automated reasoning. In simple words an algorithm is a step-by-step procedure for calculations. An algorithm is procedure consisting of a finite set of unambiguous rules (instructions) which specify a finite sequence of operations that provides the solution to a problem, or to a specific class of problems for any allowable set of input quantities (if there are inputs). In other word, an algorithm is a step-by-step procedure to solve a given problem.
They are essential to the way computers process data. Many computer programs contain algorithms that detail the specific instructions a computer should perform (in a specific order) to carry out a specified task, such as calculating employees' paychecks or printing students' report cards.
Typically, when an algorithm is associated with processing information, data is read from an input source, written to an output device, and/or stored for further processing. Stored data is regarded as part of the internal state of the entity performing the algorithm. In practice, the state is stored in one or more data structures. For some such computational process, the algorithm must be rigorously defined: specified in the way it applies in all possible circumstances that could arise. That is, any conditional steps must be systematically dealt with, case-by-case; the criteria for each case must be clear (and computable). Instructions are usually assumed to be listed explicitly, and are described as starting "from the top" and going "down to the bottom", an idea that is described more formally by *flow of control*.

13.6.1 Classification of Algorithms

- ***Recursion** or **iteration***: A recursive algorithm is one that invokes (makes reference to) itself repeatedly until a certain condition matches, which is a method common to functional programming. Iterative algorithms use repetitive constructs like loops and sometimes additional data structures like stacks to solve the given problems. Some problems are naturally suited for one implementation or the other. For example, towers of Hanoi is a well understood in recursive implementation. Every recursive version has an equivalent (but possibly more or less complex) iterative version, and vice versa.
- ***Logical***: An algorithm may be viewed as controlled logical deduction. This notion may be expressed as: **Algorithm = logic + control**. The logic component expresses the axioms that may be used in the computation and the control component determines the way in which deduction is applied to the

axioms. This is the basis for the logic programming paradigm. In pure logic programming languages the control component is fixed and algorithms are specified by supplying only the logic component. The appeal of this approach is the elegant semantics: a change in the axioms has a well-defined change in the algorithm.

- ***Serial** or **parallel** or **distributed***: Algorithms are usually discussed with the assumption that computers execute one instruction of an algorithm at a time. Those computers are sometimes called serial computers. An algorithm designed for such an environment is called a serial algorithm, as opposed to parallel algorithms or distributed algorithms. Parallel algorithms take advantage of computer architectures where several processors can work on a problem at the same time, whereas distributed algorithms utilize multiple machines connected with a network. Parallel or distributed algorithms divide the problem into more symmetrical or asymmetrical subproblems and collect the results back together. The resource consumption in such algorithms is not only processor cycles on each processor but also the communication overhead between the processors. Sorting algorithms can be parallelized efficiently, but their communication overhead is expensive. Iterative algorithms are generally parallelizable. Some problems have no parallel algorithms, and are called inherently serial problems.
- ***Deterministic** or **non-deterministic***: Deterministic algorithms solve the problem with exact decision at every step of the algorithm whereas non-deterministic algorithms solve problems via guessing although typical guesses are made more accurate through the use of heuristics.
- ***Exact** or **approximate***: While many algorithms reach an exact solution, approximation algorithms seek an approximation that is close to the true solution. Approximation may use either a deterministic or a random strategy. Such algorithms have practical value for many hard problems.
- ***Quantum algorithm***: Quantum algorithm run on a realistic model of quantum computation. The term is usually used for those algorithms which seem inherently quantum, or use some essential feature of quantum computation such as quantum superposition or quantum entanglement.

13.6.2. Cycle-based algorithm

It is a pointer algorithm that uses only two pointers, which move through the sequence at different speeds. The algorithm thus maintains two pointers into the given sequence. At each step of the algorithm, it increases by one; moving one pointer one step forward and the second pointer two steps forward in the sequence, and then compare the sequence values at these two pointers.

Did You Know?

Floyd's cycle-finding algorithm, also called the "tortoise and the hare" algorithm, is a pointer algorithm that uses only two pointers, which move through the sequence at different speeds. The algorithm is named for Robert W. Floyd, who invented it in the late 1960sv.

13.7 Program Flowcharts

A flowchart is a type of diagram that represents an algorithm or process, showing the steps as boxes of various kinds, and their order by connecting these with arrows. This diagrammatic representation can give a step-by-step solution to a given problem. Process operations are represented in these boxes, and arrows connecting them represent flow of control. Data flows are not typically represented in a flowchart, in contrast with data flow diagrams; rather, they are implied by the sequencing of operations. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields.

Symbols

A typical flowchart have the following kinds of symbols:

Start and end symbols: Represented as circles, ovals or rounded rectangles, usually containing the word "Start" or "End", or another phrase signaling the start or end of a process, such as "submit enquiry" or "receive product".

Arrows: Showing "flow of control". An arrow coming from one symbol and ending at another symbol represents that control passes to the symbol the arrow points to.
Generic processing steps: Represented as rectangles. Examples: "Add 1 to X"; "replace identified part"; "save changes" or similar.
Subroutines: Represented as rectangles with double-struck vertical edges; these are used to show complex processing steps which may be detailed in a separate flowchart. Example: PROCESS-FILES. One subroutine may have multiple distinct entry points or exit flows; if so, these are shown as labeled 'wells' in the rectangle, and control arrows connect to these 'wells'.
Input/Output: Represented as a parallelogram. Examples: Get X from the user; display X.
Prepare conditional: Represented as a hexagon. Shows operations which have no effect other than preparing a value for a subsequent conditional or decision step.
Conditional or decision: Represented as a diamond (rhombus) showing where a decision is necessary, commonly a Yes/No question or True/False test. The conditional symbol is peculiar in that it has two arrows coming out of it, usually from the bottom point and right point, one corresponding to Yes or True, and one corresponding to No or False. (The arrows should always be labeled.) More than two arrows can be used, but this is normally a clear indicator that a complex decision is being taken, in which case it may need to be broken-down further or replaced with the "pre-defined process" symbol.
Junction symbol: Generally represented with a black blob, showing where multiple control flows converge in a single exit flow. A junction symbol will have more than one arrow coming into it, but only one going out. In simple cases, one may simply have an arrow point to another arrow instead. These are useful to represent an iterative process (what in Computer Science is called a loop). A loop may, for example, consist of a connector where control first enters, processing steps, a conditional with one arrow exiting the loop, and one going back to the connector. For additional clarity, wherever two lines accidentally cross in the drawing, one of them may be drawn with a small semicircle over the other, showing that no junction is intended.
Labeled connectors: Represented by an identifying label inside a circle. Labeled connectors are used in complex or multi-sheet diagrams to substitute for arrows. For each label, the "outflow" connector must always be unique, but there may be any number of "inflow" connectors. In this case, a junction in control flow is implied.
Concurrency symbol: Represented by a double transverse line with any number of entry and exit arrows. These symbols are used whenever two or more control flows must operate simultaneously. The exit flows are activated concurrently when all of the entry flows have reached the concurrency symbol. A concurrency symbol with a single entry flow is a *fork*; one with a single exit flow is a *join*.
Data-flow extensions: A number of symbols have been standardized for data flow diagrams to represent data flow, rather than control flow. These symbols may also be used in control flow charts (e.g. to substitute for the parallelogram symbol).

- A *Document* represented as a rectangle with a wavy base;
- A *Manual input* represented by quadrilateral, with the top irregularly sloping up from left to right. An example would be to signify data-entry from a form;
- A *Manual operation* represented by a trapezoid with the longest parallel side at the top, to represent an operation or adjustment to process that can only be made manually.
- A *Data File* represented by a cylinder.

Types of flowchart

Flowcharts can be modelled from the perspective of different user groups and that there are four general types:

- *Document flowcharts*, showing controls over a document-flow through a system
- *Data flowcharts*, showing controls over a data-flow in a system
- *System flowcharts* showing controls at a physical or resource level
- *Program flowchart*, showing the controls in a program within a system

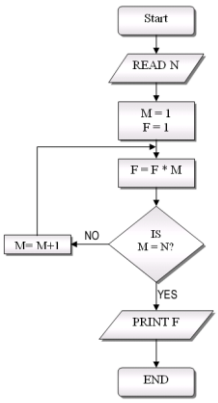


Figure.13.1 A flowchart for computing the factorial of N (10!) where N! = (1*2*3*4*5*6*7*8*9*10)

Caution

An algorithm is a precise list of precise steps, the order of computation will always be critical to the functioning of the algorithm.

13.8 Pseudocode

In computer science and numerical computation, pseudocode is a compact and informal high-level description of the operating principle of a computer program or other algorithm. It uses the structural conventions of a programming language, but is intended for human reading rather than machine reading. Pseudocode typically omits details that are not essential for human understanding of the algorithm, such as variable declarations, system-specific code and some subroutines. The programming language is augmented with natural language descriptions details, where convenient, or with compact mathematical notation. The purpose of using pseudocode is that it is easier for people to understand than conventional programming language code, and that it is an efficient and environment-independent description of the key principles of an algorithm. It is commonly used in textbooks and scientific publications that are documenting various algorithms, and also in planning of computer program development, for sketching out the structure of the program before the actual coding takes place.

Exercise: Check Your Progress

Note: i) Use the space below for your answer.

Ex1: Identify Pseudocode.

……..……

……..……

……

13.9 Programming Approaches

13.9.1 Non-Structured vs. Structured Approaches

Structured programming is a standardization technique used for software development. This approach works by having all programmers use the same structured design techniques. Structured programming was invented to address the shortcomings of non-structured programming, which frequently employed GO TO branch points to transfer from one part of the program to another part. Using GO TO codes, one could transfer backward, forward, or anywhere else within the program. The problem is that the connections between parts of the program by using GO TO commands can become quite haphazard.

The haphazard and sometimes convoluted pattern of linkages between parts of the program has been called spaghetti code. This type of programming is difficult to understand and debug. Non-structured

programming of this nature is now viewed as an ineffective programming strategy. To develop good software, developers have to carefully think out and design the programs. In the earliest days of computing, programmers wrote software according to their own whims, with the result that programs were often confusing and difficult to work with. Software today is expected to follow recognized design principles. The prevailing design standards are structured programming and structured design.

13.10 Programming Paradigms

There are several kinds of programming paradigm:

- Imperative
- Logical
- Functional
- Object-Oriented

It can be shown that anything solvable using one of these paradigms can be solved using the others; however, certain types of problems lend themselves more naturally to specific paradigms.

13.10.1 Imperative

The imperative programming paradigm assumes that the computer can maintain through environments of variables any changes in a computation process. Computations are performed through a guided sequence of steps, in which these variables are referred to or changed. The order of the steps is crucial, because a given step will have different consequences depending on the current values of variables when the step is executed.

Imperative Languages

Popular programming languages are imperative more often than they are any other paradigm studies in this course. There are two reasons for such popularity:

The imperative paradigm most closely resembles the actual machine itself, so the programme is much closer to the machine;

Because of such closeness, the imperative paradigm was the only one efficient enough for widespread use until recently.

Advantages

- Efficient
- Close to the machine
- Popular
- Familiar

Disadvantages

- The semantics of a program can be complex to understand or prove, because of referential transparency does not hold(due to side effects)
- Side effects also make debugging harder.
- Abstraction is more limited than with some paradigms;
- Order is crucial, which doesn't always suit itself to problems.

13.10.2 Logical

The Logical Paradigm takes a declarative approach to problem-solving. Various logical assertions about a situation are made, establishing all known facts. Then queries are made. The role of the computer becomes maintaining data and logical deduction.

Logical Paradigm Programming

A logical program is divided into three sections:

- A series of definitions/declarations that define the problem domain
- Statements of relevant facts

- Statement of goals in the form of a query

Any deducible solution to a query is returned. The definitions and declarations are constructed entirely from relations. i.e. X is a member of Y or X is in the internal between a and b etc.

Advantages

The advantages of logic oriented programming are beheld:

- The system solves the problem, so the programming steps themselves are kept to a minimum;
- Proving the validity of a given program is simple.

13.10.3 Functional

The Functional Programming paradigm views all subprograms as functions in the mathematical sense-informally; they take in arguments and return a single solution. The solution returned is based entirely on the input, and the time at which a function is called has no relevance. The computational model is therefore one of function application and reduction.

Languages

Functional languages are created based on the functional paradigm. Such languages permit functional solutions to problems by permitting a programmer to treat functions as first-class objects (they can be treated as data, assumed to have the value of what they return; therefore, they can be passed to other functions as arguments or returned from functions).

Advantages

The following are desirable properties of a functional language:

- The high level of abstraction, especially when functions are used, suppresses many of the details of programming and thus removes the possibility of committing many classes of errors.
- The lack of dependence on assignment operations, allowing programs to be evaluated in many different orders. This evaluation order independence makes function-oriented languages good candidates for programming massively parallel computers.
- The absence of assignment operations makes the function-oriented programs much more amenable to mathematical proof and analysis than are imperative programs, because functional programs possess referential transparency.

Disadvantages

- Perhaps less efficiency.
- Problems involving many variables or a lot of sequential activity are sometimes easier to handle imperatively or with object-oriented programming.

13.10.4 Object-Oriented

Object Oriented Programming (OOP) is a paradigm in which real-world objects are each viewed as separate entities having their own state which is modified only by built in procedures, called methods. Because objects operate independently, they are encapsulated into modules which contain both local environments and methods. Communication with an object is done by message passing.

Objects are organized into classes, from which they inherit methods and equivalent variables. The object-oriented paradigm provides key benefits of reusable code and code extensibility.

Features and Benefits

A new class (called a derived class or subclass) may be derived from another class (called a base class or super class) by a mechanism called inheritance. The derived class inherits all the features of the base class: its structure and behaviour (response to messages). In addition, the derived class may contain additional state (instance variables), and may exhibit additional behaviour (new methods to resend to new messages). Significantly, the derived class can also override behaviour corresponding to some of the methods of the base class: there would be a different method to respond to the same message. Also, the inheritance mechanism is allowed even without access to the source code of the base class.

13.11 Types of Programming Language

There is no single standard for classifying programming languages. In fact, dozens of categories exist. One of the most fundamental ways programming languages are characterized is by programming paradigm. A programming paradigm provides the programmer's view of code execution. The most influential paradigms are examined in the next three sections, in approximate chronological order. Each of these paradigms represents a mature worldview, with enormous amounts of research and effort expended in their development. A given language is not limited to use of a single paradigm. Java, for example, supports elements of both procedural and object-oriented programming, and it can be used in a concurrent, event-driven way. Programming paradigms continue to grow and evolve, as new generations of hardware and software present new opportunities and challenges for software developers.

13.10.1 Procedural Programming Languages

Procedural programming specifies a list of operations that the program must complete to reach the desired state. This one of the simpler programming paradigms, where a program is represented much like a cookbook recipe. Each program has a starting state, a list of operations to complete, and an ending point. This approach is also known as imperative programming. Integral to the idea of procedural programming is the concept of a procedure call.

Procedures, also known as functions, subroutines, or methods, are small sections of code that perform a particular function. A procedure is effectively a list of computations to be carried out. Procedural programming can be compared to unstructured programming, where all of the code resides in a single large block. By splitting the programmatic tasks into small pieces, procedural programming allows a section of code to be re-used in the program without making multiple copies. It also makes it easier for programmers to understand and maintain program structure.

Two of the most popular procedural programming languages are FORTRAN and BASIC.

13.11.2 Structured Programming Languages

Structured programming is a special type of procedural programming. It provides additional tools to manage the problems that larger programs were creating. Structured programming requires that programmers break program structure into small pieces of code that are easily understood. It also frowns upon the use of global variables and instead uses variables local to each subroutine. One of the well known features of structural programming is that it does not allow the use of the GOTO statement. It is often associated with a "top-down" approach to design. The top-down approach begins with an initial overview of the system that contains minimal details about the different parts. Subsequent design iterations then add increasing detail to the components until the design is complete.

The most popular structured programming languages include C, Ada, and Pascal.

13.11.3 Object-Oriented Programming Languages

Object-oriented programming is one the newest and most powerful paradigms. In object-oriented programs, the designer specifies both the data structures and the types of operations that can be applied to those data structures. This pairing of a piece of data with the operations that can be performed on it is known as an object. A program thus becomes a collection of cooperating objects, rather than a list of instructions. Objects can store state information and interact with other objects, but generally each object has a distinct, limited role.

There are several key concepts in object-oriented programming (OOP). A class is a template or prototype from which objects are created, so it describes a collection of variables and methods (which is what functions are called in OOP). These methods can be accessible to all other classes (public methods) or can have restricted access (private methods). New classes can be derived from a parent class. These derived classes inherit the attributes and behavior of the parent (inheritance), but they can also be extended with new data structures and methods.

The list of available methods of an object represents all the possible interactions it can have with external objects, which means that it is a concise specification of what the object does. This makes OOP a flexible system, because an object can be modified or extended with no changes to its external interface. New classes can be added to a system that uses the interfaces of the existing classes.

Objects typically communicate with each other by message passing. A message can send data to an object or request that it invoke a method. The objects can both send and receive messages. Another key characteristic of OOP is encapsulation, which refers to how the implementation details of a particular class are hidden from all objects outside of the class. Programmers specify what information in an object can be shared with other objects. A final attribute of object oriented programming languages is polymorphism. Polymorphism means that objects of different types can receive the same message and respond in different ways. The different objects need to have only the same interface (that is, method definition). The calling object (the client) does not need to know exactly what type of object it is calling, only that is has a method of a specific name with defined arguments. Polymorphism is often applied to derived classes, which replace the methods of the parent class with different behaviors. Polymorphism and inheritance together make OOP flexible and easy to extend.

Object-oriented programming proponents claim several large advantages. They maintain that OOP emphasizes modular code that is simple to develop and maintain. OOP is popular in larger software projects, because objects or groups of objects can be divided among teams and developed in parallel. It encourages careful up-front design, which facilitates a disciplined development process. Object-oriented programming seems to provide a more manageable foundation for larger software projects.

The most popular object-oriented programming languages include Java, Visual Basic, C#, C++, and Python.

Self Assessment Questions

1. ……..is a synonym for third-generation programming language.

(a).High level language (b).low level language

(c).middle level language (d).None of these

2. First introduced in the late 1950s,……ALGOL, and COBOL

(a).Pascal (b).C++

(c). .Net (d) FORTRAN

3. The ……..tends to make the languages more programmer-friendly.

(a). Fourth generation (b). Third generation

(c). Fifth generation (d). Second generation

4. Most popular object-oriented programming languages include …….C#, C++, and Python

(a).Java (b).None of these

(c). Visual Basic (d). Both of these

5. ……..is often applied to derived classes

(a). Polymorphism (b).Abstraction

(d).Encapsulation (d). None of these

6. Polymorphism and …….together make OOP flexible and easy to extend

(a).Encapsulation (b). Inheritance

(c). Polymorphism (d). None of these

13.12 Third/Fourth Generation Language

The third generation tends to make the languages more programmer-friendly. This includes features like improved support for aggregate data types, and expressing concepts in a way that favors the programmer, not the computer.



A third generation language improves over a second generation language by having the computer take care of non-essential details, not the programmer. "High level language" is a synonym for third-generation programming language.

First introduced in the late 1950s, FORTRAN, ALGOL, and COBOL are early examples of this sort of language. Most popular languages today, such as C, C++, C#, Java, BASIC and Delphi, are also third-generation languages.

Most 3GLs support structured programming. A fourth-generation programming language (4GL) is a programming language or programming environment designed with a specific purpose in mind, such as the development of commercial business software. In the history of computer science, the 4GL followed the 3GL in an upward trend toward higher abstraction and statement power. The 4GL was followed by efforts to define and use a 5GL.

The natural-language, block-structured mode of the third-generation programming languages improved the process of software development. However, 3GL development methods can be slow and error-prone. It became clear that some applications could be developed more rapidly by adding a higher-level programming language and methodology which would generate the equivalent of very complicated 3GL instructions with fewer errors. In some senses, software engineering arose to handle 3GL development. 4GL and 5GL projects are more oriented toward problem solving and systems engineering. All 4GLs are designed to reduce programming effort, the time it takes to develop software, and the cost of software development.

A quantitative definition of 4GL has been set by Capers Jones, as part of his work on function point analysis. Jones defines the various generations of programming languages in terms of developer productivity, measured in function points per staff-month. A 4GL is defined as a language that supports 12–20 function points per staff month. This correlates with about 16–27 lines of code per function point implemented in a 4GL.

13.13 Summary

- Computer software is a set of programming instructions. Before starting coding, programmers must understand the user requirements and the flow of logic of the program.
- Assembly language is easier to use than machine language as a programmer can use symbols to sum up program instructions.
- Fourth-generation languages free programmers from worrying about the procedures to be followed to solve a problem
- The design is then broken down into modules to facilitate programming.

13.14 Keywords

Compiler: It supports the assembler instructions. If an exceptional speed of execution of a part of a code is required, and the user possesses the corresponding knowledge of the microcontroller architecture and assembler instructions, then the critical part of the program could be written in the assembler (user-optimized parts of the code).

Modularity: SDF definitions can be modular because they accept all context-free languages, including the ambiguous ones. This will help you compose embedded languages and deal with language dialects in a natural manner.

Object-oriented programs: The designer specifies both the data structures and the types of operations that can be applied to those data structures.

Programming languages**:** It usually has several kinds of identifiers. Consider Java for example, it has class names, variable names, package names, etc.

Structured programming**:** It requires that programmers break program structure into small pieces of code that are easily understood.

13.15 Review Questions

1. What are the structured programming languages?
2. What are procedural programming languages?
3. Write the examples of algorithms and flowcharts.
4. What are the general rules for flow chart?
5. What is the programming language?
6. What are the comparison between human language and computer language?
7. What is the difference between value and cost?
8. What is the difference between value of information and cost of information?
9. Write the characteristics of information.
10. What is the pseudocode?

Answers for Self Assessment Questions

1. (a) 2.(d) 3.(b) 4.(d) 5.(a) 6. (b)

13.16 Further Studies

1. http://cs.brown.edu/~sk/Publications/Books/ProgLangs/
2. Computer Fundamentals by P. K. Sinha

14

Virus

STRUCTURE

14.0 Objectives

14.1 Introduction

14.2 History of Virus

14.3 Mechanism of Virus

14.4 How a Virus Spreads

14.5 How is Virus Named?

14.6 A Few Prominent Viruses

14.7 Types of Computer Virus

14.8 Norton Antivirus

14.9 Execution of Norton Anti-Virus

14.10 Summary

14.11 Keywords

14.12 Review Questions

14.13 Further Studies

14.0 Objectives

After studying this chapter, you will be able to:

- Explain history of virus
- Discuss mechanism of virus
- Understand how a virus spreads
- Understand how virus is named
- Explain a few prominent viruses
- Discuss types of computer virus
- Understand Norton anti virus
- Understand execution of Norton anti virus

14.1 Introduction

The person might have a computer virus infection when the computer starts acting differently. For instance getting slow or when they turn the computer on, it says that all the data is erased or when they start writing a document, it looks different, some chapters might be missing or something else abnormal has happened.

The next thing usually the person whose computer might be infected with virus, panics. The person might think that all the work that has been done is missing. That could be true, but in most cases viruses have not done any harm jet, but when one start doing something and are not sure what you do, that might be harmful. When some people try to get rid of viruses they delete files or they might even format the whole hard disk.

14.2 History of Virus

Technically, the concept of a computer virus was first imagined in 1949, well before computers became commonplace. In that year, computer pioneer John von Neumann wrote a paper titled "Theory and Organization of Complicated Automata." In this, von Neumann postulated that a computer program could be self-replicating—and thus predicted today's self-replicating virus programs.

The theories of von Neumann came to life in the 1950s, at Bell Labs. Programmers there developed a game called "Core Wars," where two players would unleash software "organisms" into the mainframe computer, and watch as the competing programs would vie for control of the machine—just as viruses do today.

In the real world, computer viruses came to the fore in the early 1980s, coincident with the rise of the very first personal computers. These early viruses were typically spread by users sharing programs and documents on floppy disks; a shared floppy was the perfect medium for spreading virus files.

The first virus "in the wild," as they say, infected Apple II floppy disk in 1981. The virus went by the name of Elk Cloner, and did not do any real damage; all it did was display a short rhyme onscreen:

It will get on all your disks
It will infiltrate your chips
Yes it is Cloner!
It will stick to you like glue
It will modify ram too

At the time, Elk Cloner was not identified as a virus, because the phrase "computer virus" had yet to be coined. That happened in 1983, when programmer Len Adleman designed and demonstrated the first experimental virus on a VAX 11/750 computer. From Adleman's lab to the real world was but a short step.

In 1986, the Brain virus became the first documented file infector virus for MS-DOS computers. That same year, the first PC-based Trojan horse was released, disguised as the then-popular shareware program PC Write.

From there, things only went downhill; with the popularity of computer bulletin board services (BBSs) helping to spread viruses beyond what was previously physically possible. The BBSs were the online precursors to the Internet; users could use their low-speed modems to dial into public and private BBSs, both to exchange messages and to download files. As any morning quarterback could predict, there were viruses hiding among the standard utilities and applications that users downloaded, thus facilitating the spread of those viruses.

The year 1996 saw the first virus designed specifically for Windows 95 and the first macro viruses for Word and Excel files. That year also saw the first virus for the Linux operating system.

By 1999, viruses had become almost mainstream. The Melissa virus, released that year, was a combination macro virus and worm that spread itself by e-mailing contacts in a user's Outlook or Outlook Express Address Book. Melissa did untold amounts of damage to computers and company networks around the world, and was followed (in 2000) by the Love Letter worm (also known as the "Love Bug"), which shut down tens of thousands of corporate e-mail systems. Since then, viruses have continued to proliferate and mutate, with viruses being developed for personal digital assistants (PDAs), file-swapping networks, instant messaging systems, and more.

Did You Know?

Computer viruses hit the big time in 1992, when the Michelangelo virus hit. Michelangelo was one of the first viruses to spread worldwide, and garnered much media attention. Fortunately, its bark was worse than

its bite, and little actual damage occurred.

14.3 Mechanism of Virus

The term *virus* was applied to this type of software very early in its history. It is an apt metaphor, because a computer virus is, in many ways, similar to the biological viruses that attack human bodies.

For example a biological virus is not truly a living, independent entity; as biologists will tell you, a virus is nothing more than a fragment of DNA sheathed in a protective jacket. It reproduces by injecting its DNA into a host cell. The DNA then uses the host cell's normal mechanisms to reproduce itself.

A computer virus is like a biological virus in that it also is not an independent entity; it must piggyback on a host (another program or document) in order to propagate.

Many viruses are hidden in the code of legitimate software programs—programs that have been "infected," that is. These viruses are called *file infector viruses*, and when the host program is launched, the code for the virus is also executed, and the virus loads itself into your computer's memory. From there, the virus code searches for other programs on your system that it can infect; if it finds one, it adds its code to the new program, which, now infected, can be used to infect other computers.

This entire process is shown in Figure 14.1.

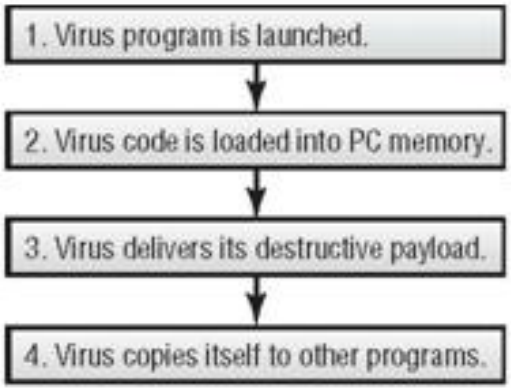


Figure: 14.1: How a virus infect your computer.

Did You Know?

In 1990, the first BBS (bulletin board services) was specifically created for virus writers.

14.4 How a Virus Spreads

14.4.1 Why Do People Write and Spread Viruses

It is difficult to know why people write them. Everyone has their own reasons. Some general reasons are to experiment how to write viruses or to test their programming talent. Some people just like to see how the virus spreads and gets famous around the World. The following is a list from news group postings alt.comp.virus and tries to explain why people write and spread viruses.

- They do not understand or prefer not to think about the consequences for other people
- They simply do not care
- They do not consider it to be their problem if someone else is inconvenienced
- They draw a false distinction between creating/publishing viruses and distributing them
- They consider it to be the responsibility of someone else to protect systems from their creations
- They get a buzz, acknowledged or otherwise, from vandalism
- They consider they're fighting authority
- They like 'matching wits' with antivirus vendors
- It is a way of getting attention, getting recognition from their peers and their names (or at least that of their virus) in the papers and the Wild List
- They are keeping the antivirus vendors in a job

14.4.2 How Viruses Act

Viruses' main mission is to spread out and then get active. Some viruses just spread out and never activate. Viruses when they spread out, they make copies of self and spreading is harmful.

14.4.3 How Viruses Spread Out

Virus's mission is to hop from program to other and this should happen as quickly as possible. Usually viruses join to the host program in some way. They even write over part of the host program.

A computer is infected with a boot sector virus if it is booted from an infected floppy disk. Boot sector infections cannot normally spread across a network. These viruses spread normally via floppy disks which may come from virtually any source:

- Unsolicited demonstration disks
- Brand-new software
- Disks used on your PC by salesmen or engineers
- Repaired hardware

A file virus infects other files, when the program to which it is attached is run, and so a file virus can spread across a network and often very quickly. They may be spread from the same sources as boot sector viruses, but also from sources such as Internet FTP sites and newsgroups. Trojan horses spread just like file viruses.

A multipartite virus infects boot sectors and files. Often, an infected file is used to infect the boot sector: thus, this is one case where a boot sector infection could spread across a network.

14.4.4 How Viruses Activate

We are always afraid that viruses do something harmful to files when they get active, but not all the viruses activate. Some viruses just spread out, but when viruses activate they do very different things. Might play a part of melody or play music in the background, show a picture or animated picture, show text, format hard disk or do changes to files.

As an example, in one unnamed company, over a long period of time, the files in a server were corrupted just a bit. So, backup copies were taken from the corrupted files. And after they noticed that something was wrong, it was too late to get back the data from the backups. That kind of event is the worst that can happen for the uses.

There is also talk that viruses have done something to hardware like hard disk or monitor. Viruses cannot do any harm to hardware but they can do harm to programs and for example to BIOS so that computer does not start after that. Usually you can start the computer from a boot diskette if the computer does not start otherwise.

14.5 How is Virus Named?

Antivirus vendors generally assign virus names consisting of a prefix, the name, and a suffix. Not all vendors follow this convention, however, and even those who do may sometimes use different designators. When attempting to find information about a particular virus, it can be helpful to understand how the names are formed.

14.5.1 The Prefix

The prefix (when used) identifies the type of virus or malware it is. W32 or Win32, for example, denote that it is a Windows 32-bit infector and thus impacts Windows 95, 98, 2000, 2003, XP, Me, NT 4.0. Those that impact only Windows 95/98 often have prefixes of W95. Other vendors apply prefixes that are more indicative of the type of threat, rather than the platform it infects. For example, a TROJ prefix implies the file is a Trojan Horse, an I-Worm prefix indicates it is an Internet/email worm, and OM signifies that it is a Microsoft Office macro virus.

W97M, WM, X2KM are other examples of macro virus prefixes that denote both the fact that it is a macro virus and provides clues as to what versions of Office (or products within Office) are impacted. For example, an X2KM prefix in a virus name indicates that it is a macro virus impacting the Office 2000 version of Excel.
The prefix is usually separated from the name by an underscore, a period, or a slash.

14.5.2 The Name

Following the prefix is the actual name of the malware. For example, W32/Bagle has a prefix of W32 and the worm itself is dubbed Bagle.

14.5.3 The Suffix

Many viruses belong to the same family but are slightly different. To differentiate between these variants, antivirus vendors assign an alphabetical suffix. The original virus (or worm, Trojan, etc.) generally does not have a suffix assigned until after further variants of the same threat are discovered. For example, W32/Bagle became W32/Bagle.A after the 'B' variant was discovered.
Subsequent variants are assigned descending letters of the alphabet, i.e. Bagle.A, Bagle.B, Bagle.C through to Bagle.Z. When the end of the alphabet has been reached, the count starts over. Thus, following Bagle.Z will be Bagle.AA, Bagle.AB, Bagle.AC, etc. The third pass through the alphabet would begin with Bagle.BA, Bagle.BB, Bagle.BC, etc. This will repeat as many times as necessary. As of October 2004, the prolific Gaobot variants had reached W32/Gaobot.BOW.
The suffix is generally separated from the virus name by either a period or a dash.

14.5.4 The Modifier

Some vendors also add a modifier after the suffix that further describes what type of malware it is. For example, @mm signifies a mass-mailing email worm and @dl is used by some to designate a downloader.
Using the above information, we can quickly see that W32/Bagle.BB@mm is a Bagle variant that is a mass-mailing email worm impacting Windows 32-bit systems.

14.5.5 One Virus, Many Names

It is one thing to understand how the name is constructed, but what if you are looking for information on the threat? It is important to remember that different vendors assign different names to the same virus. Thus when searching for information on a particular virus, it is imperative that both the vendor and the virus name be referenced.
For example, if using a search engine to find information on Bagle.AT, make sure you also include the name of the vendor that identified it as such. Otherwise, a generic search on Bagle.AT could lead you to information that did not pertain to the particular virus your antivirus software had identified. What Trend Micro calls WORM_BAGLE.AT is W32/Bagle-AU to Sophos, W32/Bagle.bb@mm to McAfee, Win32.Bagle.AQ to Computer Associates, and I-Worm.Bagle.at to Kaspersky. Antivirus vendor Symantec not only considers it a different variant, they also have assigned a different name to the worm family. Instead of Bagle, Symantec persists in calling the family Beagle, thus the Bagle.AT variant used in this example is W32.Beagle.AW@mm to Symantec.

Self Assessment Questions

1. Technically, the concept of a computer virus was first imagined in ……

(a) 1949 (b) 1981

(b) 1972 (d) None of these.

2. A computer virus is like a …….virus in that it also is not an independent entity.

(a) warm (b) fragment

(c) biological (d) All of these.

3. …… viruses are hidden in the code of legitimate software programs.
(a) File infector viruses (b) Boot sector virus
(c) Chat virus (d) None of these

4. Viruses that replicate themselves via e-mail or over a computer network cause the subsidiary problem of increasing the amount of……..
(a) Internet (b) data
(c) network traffic (d) Both (a) and (c)

5. ……mission is to hop from program to other and this should happen as quickly as possible.
(a) Antivirus (b) Virus's
(c) Program (d) None of these

14.6 A Few Prominent Viruses

With so many different types of viruses out there, what are the most widespread computer viruses today?
Unfortunately, that is a bit of a trick question. That is because most viruses have a defined and relatively short life cycle; they appear on the scene with a bang, doing considerable damage, but then—as protective methods are employed—just as quickly disappear from the radar scope.
Figure 14.2 illustrates the typical virus life cycle, from creation to eradication.

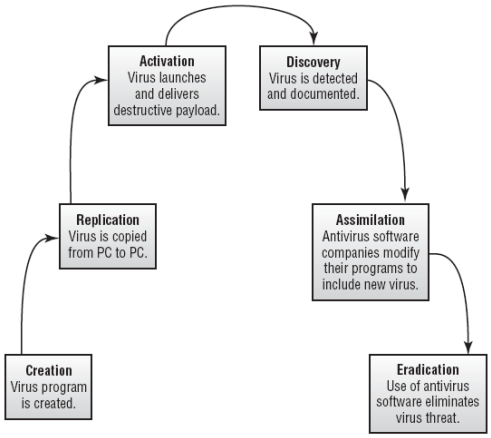


Figure 14.2: The life cycle of a computer virus.

Top 10 viruses in December 2011
Troj/Invo-Zip
W32/Netsky
Mal/EncPk-EI
Troj/Pushdo-Gen
Troj/Agent-HFU
Mal/Iframe-E
Troj/Mdrop-BTV

Troj/Mdrop-BUF
Troj/Agent-HFZ
Troj/Agent-HGT

Caution
Always scan email and instant messages for viruses before opening any attachments, as sometimes it may contain harmful viruses.

14.7 Types of Computer Virus

Technically, a computer virus is a piece of software that surreptitiously attaches itself to other programs and then does something unexpected. There are other types of programs—such as Trojan horses and worms—that do similar damage but do not embed themselves within other program code. These programs are not technically viruses, but they pose the same danger to computer systems everywhere. For that reason, all these programs—virus and non-virus, alike—are typically lumped together and referred to, in common parlance, as viruses. (Or, as some experts prefer, *malware*—for "malicious software."). The following different types of malicious programs, since the best defense against one are a defense against all. That is not to say that all malicious programs work the same way, or pack the same potential punch. They do not. So it helps to know a little bit about each type of virus, to help better protect against them.

14.7.1 File Infector Viruses

The most "traditional" form of computer virus is the file infector virus, which hides within the code of another program. The infected program can be a business application, a utility, or even a game—just as long as it is an executable program, typically with an EXE, COM, SYS, BAT, or PIF extension.
When an infected program is launched, the virus code copies itself into your computer's memory, typically before the program code is loaded. By loading itself into memory separately from the host program, the virus can continue to run in your system's memory, even after the host program is closed down.
Before the advent of the Internet and coincident creation of macro viruses, file infector viruses accounted for probably 85% of all virus infections. Today that number is much lower, because the other types of viruses are much easier to propagate.

14.7.2 Boot Sector Viruses

Boot sector viruses reside in the part of the disk that is read into memory and executed when your computer first boots up. (On a floppy disk, that is the *boot sector;* on a hard disk, the equivalent area is called the *Master Boot Record.*) Once loaded, the virus can then infect any other disk used by the computer; a disk-based boot sector virus can also infect a PC's hard disk.
Most boot sector viruses were spread by floppy disk, especially in the days before hard disks were common. Since removable disks are less widely used today, boot sector viruses have become much less prevalent than they were in the early 1990s.

14.7.3 Macro Viruses

Some computer viruses are created with the *macro* coding languages used with many of today's software applications. Macros are small programs that are created to do highly specific tasks within an application and are written in a pseudo-programming language designed to work with the application. The most common macro language, used in all Microsoft applications, is called Visual Basic for Applications (VBA). The VBA code can be added to a Word document to create custom menus and perform automatic

operations; unfortunately, VBA code can also be used to modify files and send unwanted e-mail messages, which is where the virus writers come in.

What makes macro viruses potentially more dangerous than file infector or boot sector viruses is that macros—and thus macro viruses—can be attached to document files. Older virus types had to be embedded in executable programs, which made them relatively easy to find and stop. But when any Word or Excel document you open could contain a macro virus, the world is suddenly a much more dangerous place.

14.7.4 Script Viruses

Script viruses are based on common scripting languages, which are macro-like pseudo-programming languages typically used on Web sites and in some computer applications. These viruses are written into JavaScript, ActiveX, and Java applets, which often run automatically when you visit a Web page or open a Word or Excel application. With the increasing use of the Web, these script viruses are becoming more common—and more deadly.

14.7.5 Trojan Horses

A *Trojan horse* is a program that claims to do one thing but then does something totally different. A typical Trojan horse has a filename that makes you think it is a harmless type of file; it looks innocuous enough to be safe to open. But when you run the file, it is actually a virus program that proceeds to inflict its damage on your system. It delivers its payload through deception, just like the fabled Trojan horse of yore.

Trojan horses are becoming more common, primarily through the spread of Internet-based e-mail. These e-mail Trojans spread as innocent-looking attachments to e-mail messages; when you click to open the attachment, you launch the virus.

14.7.6 Worms

A *worm* is a program that scans a company's network, or the Internet, for another computer that has a specific security hole. It copies itself to the new machine (through the security hole), and

Different Types of Viruses then starts replicating itself there. Worms replicate themselves very quickly; a network infected with a worm can be brought to its knees within a matter of hours.

Worms do not even have to be delivered via conventional programs; so-called "fileless" worms are recent additions to the virus scene. While in operation, these programs exist only in system memory, making them harder to identify than conventional file-hosted worms. These worms— such as the CodeRed and CodeBlue viruses—could cause considerable havoc in the future.

14.7.7 E-Mail Viruses

An *e-mail virus* is a program that is distributed as an attachment to an e-mail message. These viruses are typically separate programs (Trojan horses, mainly) that do their damage when they're manually executed by you, the user. These viruses masquerade as pictures, Word files, and other common attachments, but are really EXE, VBS, PIF, and other types of executable files in disguise. Many e-mail viruses hijack your e-mail program and send themselves out to all the contacts in your address book.

14.7.8 Chat and Instant Messaging Viruses

Many computer users like to chat online, either in public chat rooms or in private instant messaging (IM) conversations. Most chat and IM programs let you send files across to other users, and it is that capability that has contributed to the spread of so-called "instant" viruses.

Just as many users are in the habit of automatically opening all attachments to their incoming e-mail messages, many users are also accustomed to accepting any files sent to them when they are chatting. Unfortunately, a significant percentage of files sent via chat or IM are virus files, often Trojan horses masquerading as photographs or helpful utilities. Downloading and then opening one of these files begins the infection process.

14.7.9 Signature Detection

Most anti-virus programs work like the human immune system by scanning your computer for the signatures (patterns) of digital pathogens and infections. They refer to a dictionary of known malware, and if something in a file matches a pattern in the dictionary, the anti-virus software attempts to neutralize it. Like the human immune system, the dictionary approach requires updates, like flu shots, to provide protection against new strains of malware. Anti-virus can only protect against what it recognizes as harmful. Again, the problem is the bad guys are developing new malware so fast that anti-virus developers cannot keep up. Your computer is vulnerable during the delay between the time new malware is identified and the time a dictionary update is released by anti-virus vendors. This is why it is important that you keep your anti-virus product as up-to-date as possible.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Prepare a list of viruses that mainly affects the system files.

……..……

……..……

……

14.7.10 Behavior Detection

In this approach, instead of attempting to identify known malware, anti-virus software monitors the behavior of software installed on your computer. When a program acts suspiciously, such as trying to access a protected file or to modify another program, anti-virus spots the suspicious activity and alerts you to it. This approach provides protection against brand new types of malware that do not yet exist in any dictionary. The problem with this approach is that it can generate a large number of false warnings. You, the computer user, may be unsure about what to allow or not allow and over time become desensitized to all those warnings. You might be tempted to click Accept on every warning, leaving your computer wide open to attack and infection.

Caution

For surfing on Internet your computer must be virus protected.

14.8 Norton Antivirus

Norton antivirus fast and light all threat protection. It guards your PC, network, online activities, and your identity with innovative detection technologies optimized to combat today's aggressive, rapid-fire attacks. Norton antivirus makes online shopping, banking, and browsing safer and more convenient than ever. The improved Norton Protection System features multilayered security technologies. They work together to provide the comprehensive protection that detects and removes threats before they can harm your PC. The Norton Protection System stops online identity theft, viruses, hackers, bots, spyware, Trojan horses, and more without slowing you down or getting in your way.



Figure 14.3: Norton antivirus Window.

This version of Norton antivirus offers the following new features:

Improved Norton Protection System

Features the multilayered technologies that work together to proactively block attacks and detect and remove threats traditional virus scanning alone cannot stop.

Improved Vulnerability Protection

Guards vulnerabilities in your operating system, applications, browsers, and browser plug-ins to prevent threats from sneaking in.

Norton Insight Network

Uses Symantec's global security intelligence network to provide real-time protection from the newest threats.

Monthly Report

Tells you what Norton Internet Security has done in the past month so you can stay actively aware of how it is protecting you and your PC.

Norton Threat Insight

Reports tell you what actions threats have taken and where they came from so you can avoid future attacks.

Improved Norton Safe Web

Adds the additional protection when you search, shop, and surf the Web to stop Internet threats before they can infect your PC.

- Identifies unsafe and dangerous Web sites right in your search results.
- Analyzes downloaded files and applications and lets you know if they can be trusted to be safe before you install and run them.

Did You Know?

In May 1989, Symantec launched Symantec Antivirus for the Macintosh (SAM). SAM 2.0, released March 1990, incorporated technology allowing users to easily update SAM to intercept and eliminate new viruses, including many that did not exist at the time of the program's release.

14.9 Execution of Norton Anti-Virus

14.9.1 Norton Antivirus Installation and Setup

Installation of Norton was completely automated. There was not any messing around with various options, it is just a straight forward case of clicking install and watching the percentage count increase. Installation took around a minute in total.

Unfortunately, after installation, you hit a slight roadblock in the form of forced product activation. There was no way that you see to close the activation window, and it remained on my screen until gave it an email address to tag the serial on to. It is bad enough when you are bugged to activate a product after installation, but when, for whatever reason, you have no choice but to register, we get a little more pissed off. It seems Norton wanted to twist your arm on this one, and if it was not clear enough with the lack of an exit button on the window.

14.9.2 Norton Antivirus User Interface

Getting away from the somewhat rude introduction to Norton, at the top of the main window, you can clearly see the status of both your computer protection and network connection. One thing there is really like here is the ability to turn various features off for a certain amount of time simply by clicking a switch which turns the feature off. This is great if you are downloading a file that you trust, but often gets flagged as a virus, because you can temporarily flick your anti-virus off while you grab the file and do whatever you need to do with it.

There are other, really nice tweaks here in the UI (user interface), such as the performance option, which flips your window 180 degrees horizontally to reveal a window with all performance related analyses and tasks. Although it may not be super productive, it is nice to have a few pieces of eye candy, even if it is in your security software!

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Prepare a list of ten antivirus which currently available in market.

……..……

……..……

……

14.9.3 Security Performance of Norton Antivirus

Naturally, an anti-virus application review would not be complete without a mention of, you know, virus protection. After updating the software, it runs a full system scan, which took a little under 40 minutes, although this figure will differ depending on the size of your hard drive and the speed of your computer.

Exercise: Check Your Progress 3

Case Study-ZSecurity Virus

Melissa.V In the spring of 1999, Dan, a worker in an upstate NY research and development lab called his computer repair tech to make an appointment. He needed to bring his machine in because it had started acting funny a few days earlier. When pressed about the symptoms Dan mentioned his computer running extremely slow, especially when he first booted it up and tried to check his email. He also mentioned that he was having trouble with some of his Microsoft Office documents. Dan was on vacation for the week so he was able to get the repair tech to make a house call that same day.

After an initial inspection the tech decided he needed to take the machine back to his shop for a closer look. That closer look revealed the computer had been infected with the now famous "Melissa" virus, also known as W97M. This virus, originally created by New Jersey resident David L. Smith, was not intended to be malicious. However, it spread so rapidly it caused entire email systems to be overrun and shutdown. Malicious variants of Melissa were created soon after and it was the variant Melissa. V that Dan's computer had contracted.

Melissa's attack begins as an infected Microsoft Office file that takes advantage of the interoperability of Microsoft software. It copies itself to various files on the infected machine, then emails itself to entries found in address books on the machine with an attachment bearing the Microsoft .doc extension. Originally, Smith's attachment was passed off as a list of names and passwords to get access to pornographic websites. Once a machine became infected Melissa could send out any Office file as the attachment, so in just a few hours every .doc attachment was suspect.

In addition to reproducing and emailing itself, Melissa can also modify the infected Office documents in a variety of ways including data corruption, replacing the current data with something completely unrelated, damaging macros or adding its own, even harvesting data found in some documents. This is was Dan's experience with the Office documents on his machine. Another variant, Melissa. U went so far as to change the properties of Windows system files and then delete them, rendering the machine un-bootable as soon as it was shut down. Fortunately Dan was not struck with this variant.

Removal of the virus needed to be done manually since Dan's antivirus vendor had not yet released an automatic removal tool. The technician first needed to isolate Melissa's original source file, usually found still residing in the email folders. That source file had to be deleted along with any copies it made of itself and placed elsewhere on the machine, but unfortunately no source file was found initially. Next, all documents had to be scanned and cleaned where possible, deleted when cleaning was not possible.

Finally, the tech had to clean the system registry and the Microsoft Office preferences. Melissa modified a registry entry that was originally produced by the operating system. This modification told the virus whether or not it had mailed itself out previously. Oddly enough, the author programmed Melissa to run the email only once. As for the Office preferences, Melissa disabled macro tools, macro virus protection,

verification of template saving, and confirmation of document conversion. Disabling these options allowed the virus to modify documents without the knowledge of the user. All these features were turned back on as part of the removal process.
Once removal was complete and the computer returned, Dan and the tech needed to figure out where the infection came from in order to kill the source and prevent a second attack. The usual suspects were checked first; teenagers in the house who frequently exchanged files, unusual email attachments that had been opened, questionable websites that might have been visited. Yet all of these possibilities came up empty.

Dan mentioned he had been off work all week due to a mandatory facility furlough and was looking forward to returning in a couple of days. He had brought some work home with him the previous Friday so he would not be behind after the furlough, but the computer being down prevented him from doing much work. As it turned out, in his remarks Dan had revealed the source: the documents he had brought home from work. The floppy disk was checked and there it was; a file Dan had received in his email and which he brought home and opened on his computer.
The lab where Dan worked had been infected, but due to the week-long furlough it had not been able to do significant damage to the system. A call was placed to the lab's IT department who went in immediately and cleaned all the computers. When the doors opened the following Monday it was business as usual, thanks to a dedicated employee and a repair tech who knew what he was doing.
ZSecurity detects and cleans thousands of computer viruses, including Melissa and it variants. Make sure your program is updated and running at all times.

Questions
1. Write the brief conclusion of the case study.
2. How Melissa's attack begins as an infected Microsoft Office? Discuss.

14.10 Summary

- Viruses that replicate themselves via e-mail or over a computer network cause the subsidiary problem of increasing the amount of Internet and network traffic.
- Viruses are nasty little bits of computer code, designed to inflict as much damage as possible, and to spread to as many computers as possible—a particularly vicious combination.
- Antivirus vendors generally assign virus names consisting of a prefix, the name, and a suffix.
- The most "traditional" form of computer virus is the file infector virus, which hides within the code of another program.
- Boot sector viruses reside in the part of the disk that is read into memory and executed when your computer first boots up.
- Norton antivirus makes online shopping, banking, and browsing safer and more convenient than ever.

14.11 Keywords

Master Boot Record: The MBR is a type of boot sector popularized by the IBM Personal Computer.
Prefix: In the virus naming the prefix identifies the type of virus or malware.
Script Viruses**:** *Script viruses* are based on common scripting languages, which are macro-like pseudo-programming languages typically used on Web sites and in some computer applications.
Virus: The term "virus" is commonly but erroneously used to refer to other types of malware, including but not limited to adware and spyware programs that do not have the reproductive ability.
Wild virus: It is the first virus infected Apple II floppy disk in 1981.
Worms**:** A *worm* is a program that scans a company's network, or the Internet, for another computer that has a specific security hole.

14.12 Review Questions

1. Discuss the history of virus in brief.
2. Explain the file infector viruses.
3. What do you understand by mechanism of virus?
4. Write five reasons for spread the virus.
5. Explain the concept of virus naming with suitable example.
6. Differentiate between boot sector and macro virus.
7. What do you understand by chat and instant messaging viruses?
8. Explain in brief about antivirus.
9. Write five tips to safe computer from virus.
10. How viruses activate? Explain.

Answers for Self Assessment Questions

1. (a) 2. (c) 3. (b) 4. (d) 5. (b)

14.13 Further Studies

1. http://www.virology.net/big_virology/bvviruslist.html
2. Computer Fundamentals by P. K. Sinha

15

Communication and IT

STRUCTURE

15.0 Objectives

15.1 Introduction to Network

15.2 Communication process and Transmission Media

15.3 MODEM

15.4 Types of Modem

15.5 Summary

15.6 Keywords

15.7 Review Questions

15.8 Further Studies

15.0 Objectives

After studying this chapter, you will be able to:

- Explain Network
- Discuss MODEM
- Understand Types of Modem

15.1 Introduction to Network

A network comprises two or more computers that have been connected in order to enable them to communicate with each other, and share resources and files.

15.1.1 Computer Networks

A computer network is interconnection of various computer systems located at different places. In computer network two or more computers are linked together with a medium and data communication devices for the purpose of communication data and sharing resources. The computer that provides resources to other computers on a network is known as server. In the network the individual computers, which access shared network resources, are known as nodes.

Parts of a network

There are five basic components of a network: clients, servers, channels, interface devices and operating systems.

Servers: Sometimes called host computers, servers are powerful computers that store data or applications and connect to resources that are shared by the users of a network.

Clients: These computers are used by the users of the network to access the servers and shared resources (such as hard disks and printers). These days, it is typical for a client to be a personal computer that the users also use for their own non-network applications.

Channels: Called the network circuit, the channel is the pathway over which information travels between the different computers (clients and servers) that comprises the network.
Interface devices: These are hardware devices that connect clients and servers (and sometimes other networks) to the channel. Examples include modems and network interface cards.
Operating systems: The network operating system is the software of the network. It serves a similar purpose that the operating system serves in a stand-alone computer.

15.1.2 Understanding Network Channels

Network channels come in a variety of types, speeds and capabilities. For our purposes, there are four important dimensions of channels:
Transmission medium: This is the actual physical medium of the channel. Computer network channels use either wire line or wireless media.
Wire line media: Also called guided media and line-based media. In networks that use wire line media, the transmission of information takes place on a wire or cable. The three types of wire line media are twisted-pair wire, coaxial cable and fibre-optic cable. (Try and find examples of each of these media, and their relative speeds). While twisted-pair and coaxial cable are more commonly used today, fibre optic cables are becoming increasingly popular.
Wireless media also called radiated media. As the name indicates, in networks that use wireless media, there is no physical wire along which information travels; instead, information is transmitted through the air, from one transmission station to the next. Networking examples include radio, cellular, microwave and satellite. Broadcast TV and FM radio use wireless transmission as well (though the underlying engineering is a little different).
***Transmission rate or bandwidth*:** This property of a network channel describes how fast information can be transmitted over the channel. It is measured in bits per second people very commonly use the term bandwidth to mean transmission rate.

***Transmission directional capability*:** The direction in which information can be transmitted over a channel depends on whether the channel is simple, half-duplex or full-duplex.
Simplex: Information can be transmitted only in one direction.
Half-duplex: Information can be transmitted in both directions, but only in one direction at a time.
Full-duplex: Information can be transmitted in both directions simultaneously.
***Signal type*:** There are two signal types analog and digital. It is a little hard to understand the exact difference without discussing a lot of electrical engineering and physics, so we would not go there. What you need to take away is that:

- Analog signals are 'continuous' (they take on a wide range of values) and digital signals are 'discrete', and binary.
- Digital signals are more 'natural' for computer networks, since, as we know, computers represent all information in binary.
- The reason why we have to worry about analog signals is because the communications. Channels that predated computer networks (like telephone lines, cable TV lines and radio transmitters) were all designed to carry analog signals.

Did You Know?
Early networks of communicating computers included the military radar system Semi-Automatic Ground Environment (SAGE), started in the late 1950s.

15.2 Communication process and Transmission Media

The sharing of meaningful information between two or more people with the goal of the receiver understanding the sender's intended message. In business, the effectiveness of a company's internal and external communication process is often very important to its overall success.

Twisted-Pair Cable

Twisted-pair cable comes in two forms: unshielded and shielded.

Unshielded Twisted-Pair (UTP) Cable

Unshielded twisted-pair (UTP) cable is the most common type of telecommunication medium in use today. Although most familiar from its use in telephone systems, its frequency range is suitable for transmitting-both data and voice (see Figure 15.1). A twisted pair consists of two conductors (usually copper), each with its own colored plastic insulation. The plastic insulation is color-banded for identification (see Figure 15.2). Colors are used both to identify the specific conductors in a cable and to indicate which wires belong in pairs and how they relate to other pairs in a larger bundle.

A twisted pair consists of two conductors each surrounded by an insulating material.

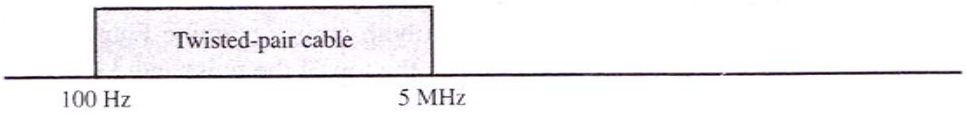


Figure 15.1: Frequency range for twisted-pair cable.

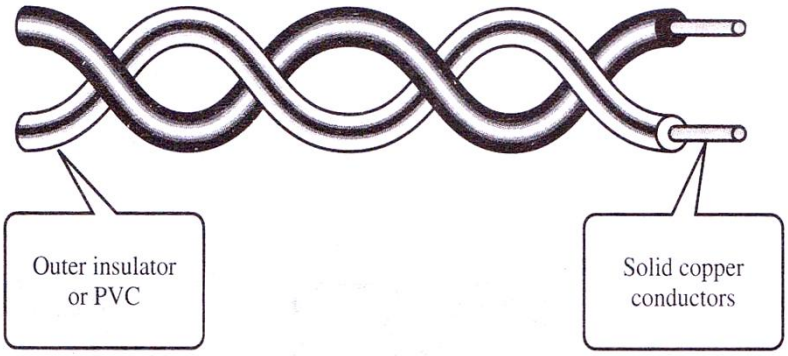


Figure 15.2: Twisted-pair cable.

In the past, two parallel flat wires were used for communication. However electromagnetic interference from devices such as a motor can create noise over those wires. If the two wires are parallel, the wire closest to the source of the noise gets more interference and ends up with a higher voltage level than the wire farther away, which results in an uneven load and a damaged signal (see Figure 15.3).

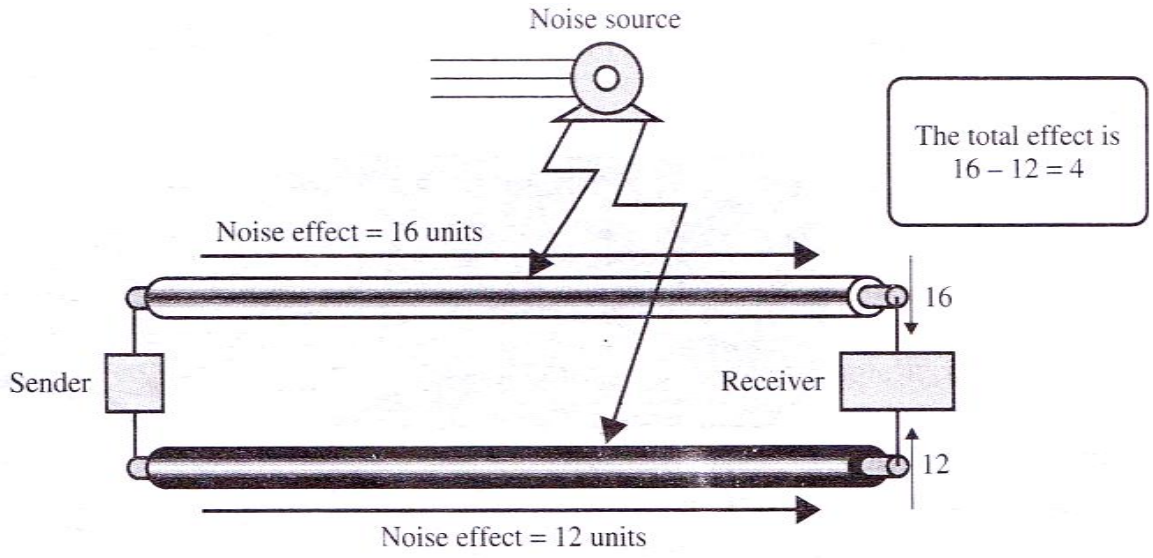


Figure 15.3: Effect of noise on parallel lines.

If, however, the two wires are twisted around each other at regular intervals (between 2 and 12 twists per foot), each wire is closer to the noise source for half time and farther away for the other half. With twisting, therefore, the cumulative effect of the interference is equal on both wires (see Figure 15.4). Each section of W has a "load" of 4 when it is on the top of the twist and 3 when it is on the bottom. The total effect of the noise at the receiver is therefore 0(14 - 14). Twisting does not always eliminate the impact of noise, but it does significantly reduce it.

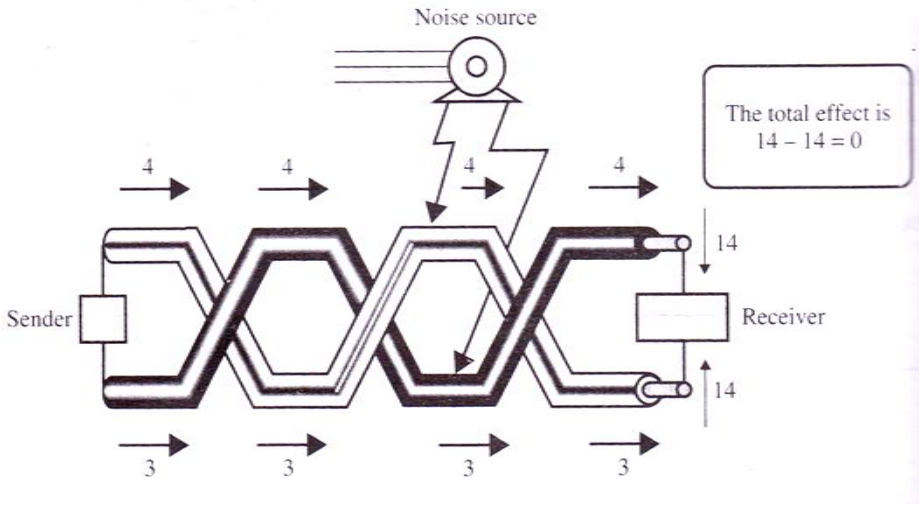


Figure 15.4: Effect of noise on twisted-pair lines.

Advantages of UTP are its cost and ease of use. UTP is cheap, flexible, and easy to install. Higher grades of UTP are used in many LAN technologies, including Ethel and Token Ring. Figure 15.5 shows a cable containing five unshielded twisted pairs.

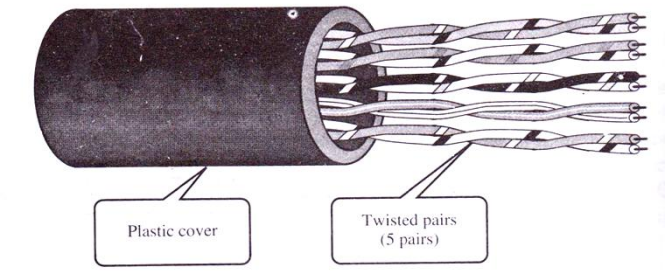


Figure 15.5: Cable with five unshielded twisted pairs of wires.

The Electronic Industries Association (EIA) has developed standards to grade UTP cables by quality. Categories are determined by cable quality, with 1 as the lowest and 5 as the highest. Each EIA category is suitable for certain uses and not for others:
Category 1: The basic twisted-pair cabling used in telephone systems. This level of quality is fine for voice but inadequate for all but low-speed data communication.
Category 2: The next higher grade, suitable for voice and for data transmission of up to 4 Mbps.
Category 3: Required to have at least three twists per foot and can be used for data transmission of up to 10 Mbps. It is now the standard Cable for most telephone systems.
Category 4: Must also have at least three twists per foot as well A other conditions to bring the possible transmission rate to 15 Mbps.
Category 5: Used for data transmission up to 100 Mbps.
UTP Connectors UTP is most commonly connected to network devices via a type of snap-in plug like that used with telephone jacks. Connectors are either male (the plug) or female (the receptacle). Male connectors snap into female connectors and have a repressible tab (called a key) that locks them in place. Each wire in a cable is attached to one conductor (or pin) in the

connector. The most frequently used of these plugs is an RJ45 connector with eight conductors, one for each wire of four twisted pairs (see Figure 15.6).

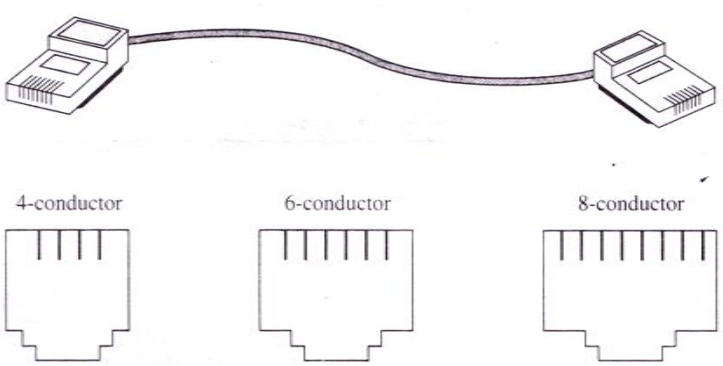


Figure 15.6: UTP connection.

Shielded Twisted-Pair (STP) Cable
Shielded twisted-pair (STP) cable has a metal foil or braided-mesh covering that encases each pair of insulated conductors (see Figure 15.7). The metal casing prevents the penetration of electromagnetic noise. It also can eliminate a phenomenon called crosstalk, which is the undesired effect of one circuit (or channel) on another circuit (or channel). It occurs when one line (acting as a kind of receiving antenna) picks up some of the signals traveling down another line (acting as a kind of sending antenna). This effect can be experienced during telephone conversations when one can hear other conversations in the background. Shielding each pair of a twisted-pair cable can eliminate most crosstalk.

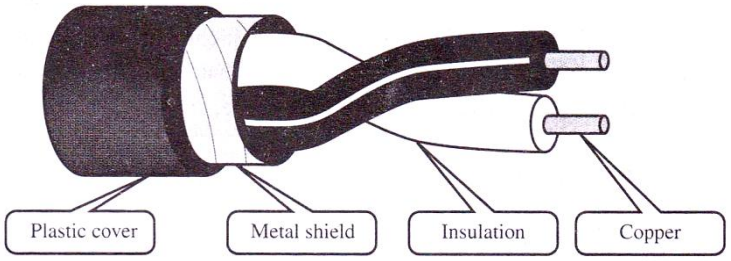


Figure 15.7: Shielded twisted-pair cable.

Materials and manufacturing requirements make STP more expensive than UTP but less susceptible to noise.

Did You Know?
The earliest telephones used telegraph lines, or open-wire single-wire earth return circuits. In the 1880s electric trams were installed in many cities, which induced noise into circuits.

Coaxial Cable
Coaxial cable (or coax) carries signals of higher frequency ranges than twisted-pair cable (see Figure 15.8), in part because the two media are constructed quite differently. Instead of having two wires, coax has a central core conductor of solid or stranded wire (usually copper) enclosed in an insulating sheath, which is, in turn, encased in an outer conductor of metal foil, braid, or a combination of the two (also usually copper). The outer metallic wrapping serves both as a shield against noise and as the second conductor which completes the circuit. This outer conductor is also enclosed in an insulating sheath, and the whole cable is protected by a plastic cover (see Figure 15.9).

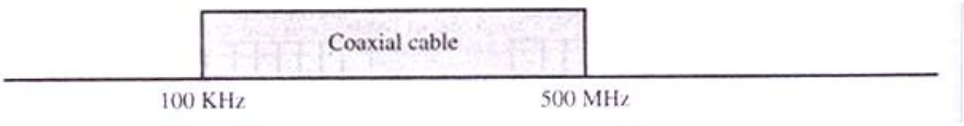


Figure 15.8: Frequency range of coaxial cable.

Coaxial Cable Standards

Different coaxial cable designs are categorized by their radio government (RG) ratings. Each RG number denotes a unique set of physical specifications, including the wire gauge of the inner conductor, the thickness and type of the inner insulator, the construction of the shield, and the size and type of the outer casing.

Each cable defined by RG ratings is adapted for a specialized function. The following are a few of the common ones:

RG-8: Used in thick Ethernet.
RG-9: Used in thick Ethernet.
RG-11: Used in thick Ethernet.
RG-58: Used in thin Ethernet.
RG-59: Used for TV.

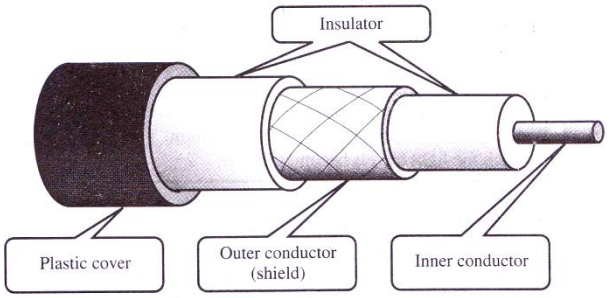


Figure 15.9: Coaxial cable.

Coaxial Cable Connectors

Over the years, a number of connectors have been designed for use with coaxial cable, usually by manufacturers seeking specific solutions to specific product requirements. A few of the most widely used connector designs have become standardized. The most common of these is called a barrel connector because of its shape. Of the barrel connectors, the most popular is the bayonet network connector (BNC), which pushes on and locks into place with a half turn. Other types of barrel connectors either screw together, or thus require more effort to install, or push on without locking, which is less secure. Generally, a cable terminates in a male connector that plugs or screws onto a corresponding female connector attached to the device. All coaxial connectors have a single pin protruding from the centre of the male connector that slides into a ferrule in the female connector. Coaxial connectors are familiar from cable TV and VCR hookups, which employ both threaded and slip-on styles.

Two other commonly used types of connectors are T -connectors and terminators. A T connector (used in thin Ethernet) allows a secondary cable or cables to branch off from a main line. A cable running from a computer, for example, can branch to connect several terminals. Terminators are required for bus topologies where one main cable acts as a backbone with branches to several devices but does not itself terminate in a device. If the main cable is left unterminated, any signal transmitted over the line echoes back and interferes with the original signal. A terminator absorbs the wave at the end and eliminates echo-back.

Optical Fiber

Up until this point, we have discussed conductive (metal) cables that transmit signals in the form of current. Optical fiber, on the other hand, is made of glass or plastic and transmits signals in the form of light. To understand optical fiber, we first need to explore several aspects of the nature of light.

The Nature of Light

Light is a form of electromagnetic energy. It travels at its fastest in a vacuum: 300,000 kilometers/second (approximately 186,000 miles/second). The speed of light depends on the density of the medium through which it is travelling (the higher the density, the slower the speed). Light, a form of electromagnetic energy, travels at 300,0001cilometers/second, or approximately 186,000 miles/second, in a vacuum. This speed decreases as the medium through which the light travels becomes denser.

Refraction

Light travels in a straight line as long as it is moving through a single uniform substance. If a ray of light travelling through one substance suddenly enters another (more or less dense) substance, its speed changes abruptly, causing the ray to change direction. This change is called refraction. A straw sticking out of a glass of water appears bent, or even broken, because the light by which we see it changes direction as it moves from the air to the water.

The direction in which a light ray is refracted depends on the change in density encountered. A beam of light moving from a less dense into a more dense medium is bent toward the vertical axis (examine Figure 15.10). The two angles made by the beam of light in relation to the vertical axis are called I, for incident, and R, for refracted. In Figure 15.10a, the beam travels from a less dense medium into a denser medium. In this case, angle R is smaller than angle I. In Figure 15.10b, however, the beam travels from a denser medium into a less dense medium. In this case, the value of I is smaller than the value of R. In other words, when light travels into a denser medium, the angle of incidence is greater than the angle of refraction; and when light travels into a less dense medium, the angle of incidence is less than the angle of refraction.

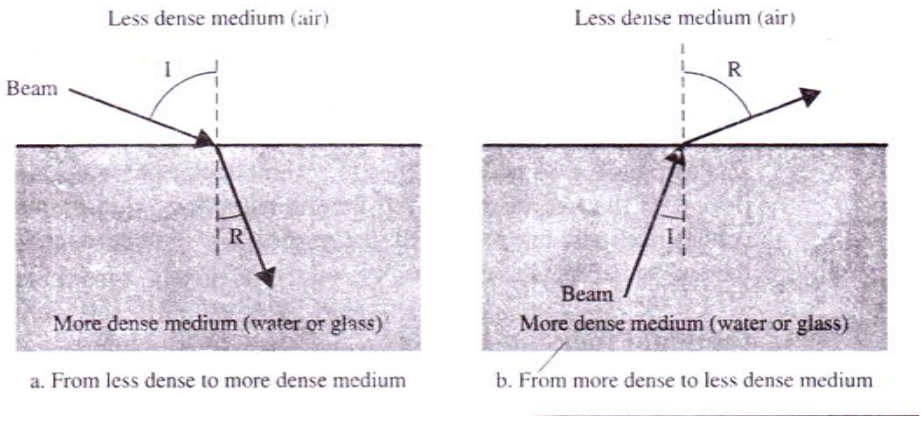


Figure 15.10: Refraction.

Fiber-optic technology takes advantage of the properties shown in Figure 15.10b to control the propagation of light through the fiber channel.

Critical Angle

Now examine Figure 15.11. Once again we have a beam of light moving from a denser into a less dense medium. In this example, however, we gradually increase the angle of incidence measured from the vertical. As the angle of incidence increases, so does the angle of refraction. It, too, moves away from the vertical and closer and closer to the horizontal.

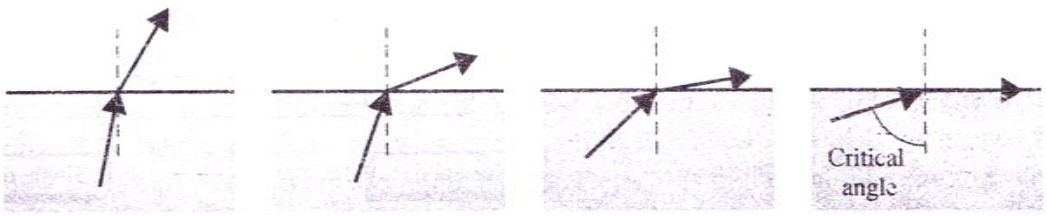


Figure 15.11 Critical angle.

At some point in this process, the change in the incident angle results in a refracted angle of 90 degrees, with the refracted beam now lying along the horizontal. The incident angle at this point is known as the critical angle.

Reflection

When the angle of incidence becomes greater than the critical angle, a new phenomenon occurs called reflection (or, more accurately, complete reflection, because some aspects of reflection always coexist with refraction). Light no longer passes into the less dense medium at all. In this case, the angle of incidence is always equal to the angle of reflection (see Figure 15.12).

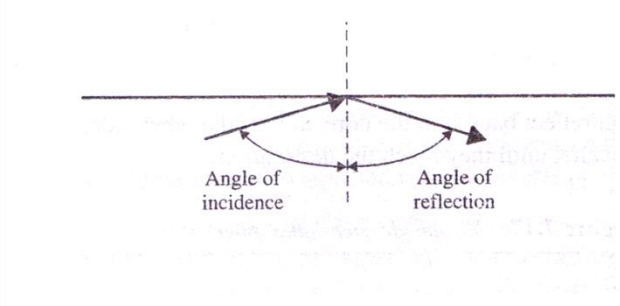


Figure 15.12: Reflection.

Optical fibers use reflection to guide light through a channel. A glass or plastic core is surrounded by a cladding of less, dense glass or plastic. The difference in density of the two materials must be such that a beam of light moving through the core is reflected off the cladding instead of being refracted into it. Information is encoded onto a beam of light as a series of on-off flashes that represent 1 and 0 bits.

Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Define Communication process and

……..……

……..……

……

Ex2: Explain Transmission Media

……..……

……..……

……

15.3 MODEM

The need to communicate between distant computers led to the use of the existing phone network for data transmission. Most phone lines were designed to transmit analog information - voices, while the computers and their devices work in digital form - pulses. So, in order to use an analog medium, a converter between the two systems is needed. This converter is the MODEM which performs MODulation and DEModulation of transmitted data. It accepts serial binary pulses from a device, modulates some property (amplitude, frequency, or phase) of an analog signal in order to send the signal in an analog medium, and performs the opposite process, enabling the analog information to arrive as digital pulses at the computer or device on the other side of connection.

Modems, in the beginning, were used mainly to communicate between DATA TERMINALS and a HOST COMPUTER. Later, the use of modems was extended to communicate between END COMPUTERS. This required more speed and the data rates increased from 300 bps in early days

to 28.8bps today. Today, transmission involves data compression techniques which increase the rates, error detection and error correction for more reliability.
In order to enable modems of various types and different manufacture to communicate, interface standards were developed by some standard organizations
Today's modems are used for different functions. They act as textual and voice mail systems, facsimiles, and are connected or integrated into cellular phones and in notebook computers enabling sending data from anywhere. The future might lead to new applications. Modem speeds are not expected to be increased much over today's 28.8 kbps. Further dramatic speed increases will require digital phone technology such as ISDN and fiber optic lines.
New applications might be implemented such as simultaneous voice and data. Videophones are an example of this.
Modems can be characterised by the following properties:

- Internal / External / PCMCIA modem
  - An internal modem is installed in one of the computer's expansion slot.
  - External modems are fully functioning external devices. The external modem is connected to a computer using a serial cable to one of the computer's serial ports, and draws power from an external power source.
- PCMCIA - Personal Computer Memory Card International Association. (Or People Can't Memorise Computer Industry Acronyms)
- Transmission speed
- Error detection and correction
- Compression

Self Assessment Questions

1……have been widely used in many areas of business since the mid-1960s.
(a). Database management systems (b). Networking
(c). Management (d). None of these

2.These systems must address a number of problems, such as……
(a). Security (b). Accuracy
(c). Both a and b (d). None of these

3……..the central connection point for network cables that connect to computers or other devices on a network.
(a). Network (b). Network hub
(c). Network adapter cards (d). None of these

4……expansion cards that provide the physical connection between each computer and the network.
(a). Network cards (b). Pen cards
(c). Network adapter cards (d). None of these

5. ……are more ‘natural’ for computer networks, since, as we know, computers represent all information in binary.
(a). Analog signals (b). Network signals
(c). Digital signals (d). None of these

6. ……..are ‘continuous’ (they take on a wide range of values) and digital signals are ‘discrete’, and binary.
(a). Analog signals (b). Digital signals
(c). Communication (d). None of these

7 **……**Information can be transmitted in both directions simultaneously.

(a). Half-duplex (b). Full-duplex

(c). Signals (d). None of these

15.4 Types of Modem

Internal and External Modems:

Internal modems

Internal modems are in the form of circuit boards, fitted in expansion slots of the motherboard. An internal modem is a device installed inside desktop or laptop computer, allowing the computer to communicate over a network with other connected computers. There are two types of internal modems: dial-up and WiFi ® (wireless). Dial-up operates over a telephone line and requires a network access phone number and logon credentials to make a connection. WIFI connect wirelessly and without credentials in certain cases.

External modems

External modems are enclosed in separate cases and not inside the system units. The modem is connected to the computer and telephone line by means of two separate cables.

Intelligent and Standard Modems:

Standard Modems

Most modems used today are called standard modems. These modems are usually operated by commands entered from a microcomputer keyboard. Users control the functions (dialling, etc.) of a modem through the keyboard. Modems may use different command languages to control their functions, but most modems (called "Hayes compatible") use a popular language developed by accompany "Hayes Microcomputer Products, Inc."

Intelligent Modems

Intelligent modems are also called advanced modems. These modems can accept new instructions and then respond to the commands while transmitting data and information. These can be done by microprocessor chips and internal read only memory (ROM) contained in the modem.

Wireless and Short-Haul Modems:

Short-Haul Modems

Short- haul modems are devices that transmit signals down the cable through anyCOM1 port. They sometimes are called modem eliminators, because they do not require an external power source. They are also called line drivers that can send data for a distance of more than one mile. This type of modem can be used within or across several buildings in a company or a university campus.

Wireless Modems

Wireless modems transmit the data signals through the air instead of by using a cable. They sometimes are called a radiofrequency modem. This type of modem is designed to work with cellular technology, and wireless local area networks. Wireless modems are not yet perfected, but the technology is rapidly improving.

15.5 Summary

- The modern form of communication like e-mail and Internet is possible only because of computer networking.
- Data Routing is the process of finding the most efficient route between source and destination before sending the data.
- In simplex mode the communication take place in one direction. The receiver receives the signal from the transmitting device.
- In half-duplex mode the communication channel is used in both directions, but only in one direction at a time. Thus a half-duplex line can alternately send and receive data.
- The computer that provides resources to other computers on a network is known as server.
- In the network the individual computers, which access shared network resources, are known as nodes.

15.6 Keywords

Communication Satellite: The problem of line-sight and repeaters are overcome by using satellites which are the most widely used data transmission media in modern days.

Data sequencing: A long message to be transmitted is broken into smaller packets of fixed size for error free data transmission.

Internet: The newest type of network to be used within an organisation is an internet or Internet Web. Such networks enable computers (or network) of any type to communicate easily.

Transmission: Communication of data achieved by the processing of signals.

Teleconferencing: It refers to electronic meetings that involve people who are at physically different sites. Telecommunication technology allows participants to interact with one another without travelling to the same location.

15.7 Review Questions

1. What is the model and communication task?
2. What are the needs of computer networks?
3. Differentiate between half-duplex and full-duplex.
4. What is the use of computer networks?
5. What is the application of computer network?
6. Differentiate between LAN and WAN.
7. How many types of server and network?
8. What are the software and hardware networking?
9. What are the network criteria?
10. What are the hub and network cables?

Answers for Self Assessment Questions

1. (a) 2. (c) 3. (b) 4. (c) 5. (c) 6. (a) 7. (b)

15.8 Further Studies

1. http://www.businessdictionary.com/definition/communication-process.html

16

Networks

STRUCTURE

16.0 Objectives

16.1 Introduction

16.2 Internet V/s Intranet

16.3 Network Topology

16.4 Network Devices

16.5 Types of Networks

16.6 Types of Connectivity

16.7 Summary

16.8 Keywords

16.9 Review Questions

16.10 Further Studies

16.0 Objectives

After studying this chapter, you will be able to:

- Explain Internet V/s Intranet
- Discuss Network Topology
- Understand Network Devices
- Understand how virus is named

16.1 Introduction

In the information age that we live in today, the speed at which information can travel inside a company would often indicate the productivity of that company. It is often necessary to create an environment where the flow of data is unimpeded and the intended recipient gets it instantaneously. Computers make this possible and there are multiple ways to implement such a network.

An Intranet is a computer network that is designed to work like the internet but in a much smaller scale and is restricted only to the employees of the company. It is possible to run FTP, HTTP, and mail servers in the intranet that is independent and inaccessible from the internet without proper authorization. This allows the employees to send progress reports to their manager even when they cannot meet in person. Workers could also work collaboratively on a certain project while keeping their paperwork properly synchronized. It is often necessary to have access to the internet from within your intranet, which is why intranets are placed behind a firewall. Some companies even deploy two firewalls and place some services inside the DMZ in order to raise their security further.

An intranet, although very helpful, wouldn't be very effective if it is totally removed from the internet. The internet is the massive network of computers from all around the world. It allows people to virtually any point in the world at a very minimal cost. Services like Email and VoIP has allowed many people to keep in touch despite geographical locations and time zones.

Being connected to the internet, a company can have their people in the field or those who are working at home to still be able to do what they would usually do when they are inside the office. They can connect to services inside the intranet and submit their work or contact their coworkers and superiors. They can even call online if their office supports IP-PABX systems.

The Intranet and the Internet are two domains that are very alike but are often segregated in order to maintain security. If properly configured and guarded, an Intranet that is connected to the Internet could raise your company's productivity by leaps and bound; not to mention cutting down the cost of traditional communications. It could also open the door to malicious people who can do major damage or even steal confidential company data if done haphazardly. It should be up to the management to make sure that all precautions are taken.

16.2 Internet V/s Intranet

Difference between Intranet & Internet:

- Users: Intranet users are your own employees who know a lot about the company, its organizational structure, and special terminology and circumstances while Internet site is used by customers who will know much less about your company and also care less about it.
- The tasks: The intranet is used for everyday work inside the company, including some quite complex applications; the Internet site is mainly used to find out information about the products or other things.
- Type of information: The intranet will have many draft reports, project progress reports, human resource information, and other detailed information, whereas the Internet site will have marketing information and customer support information or other information.
- Amount of information: Typically, an intranet has between ten and a hundred times as many pages as the same company's public website. The difference is due to the extensive amount of work-in-progress that is documented on the intranet and the fact that many projects and departments never publish anything publicly even though they have many internal documents.
- Bandwidth and cross-platform needs: Intranets often run between a hundred and a thousand times faster than most Internet users' Web access which is stuck at low-band or mid-band, so it is feasible to use rich graphics and even multimedia and other advanced content on intranet pages. Also, it is sometimes possible to control what computers and software versions are supported on an intranet, meaning that designs need to be less cross-platform compatible (again allowing for more advanced page content).

16.3 Network Topology

A network consists of multiple computers connected using some type of interface, each having one or more interface devices such as a Network Interface Card (NIC) and/or a serial device for PPP networking. Each computer is supported by network software that provides the server or client functionality. The hardware used to transmit data across the network is called the media. It may include copper cable, fiber optic, or wireless transmission. The standard cabling used for the purposes of this document is 10Base-T category 5 Ethernet cable. This is twisted copper cabling which appears at the surface to look similar to TV coaxial cable. It is terminated on each end by a connector that looks much like a phone connector. Its maximum segment length is 100 meters.

In a server based network, there are computers set up to be primary providers of services such as file service or mail service. The computers providing the service are are called servers and the computers that request and use the service are called client computers.

In a peer-to-peer network, various computers on the network can act both as clients and servers. For instance, many Microsoft Windows based computers will allow file and print sharing. These computers can act both as a client and a server and are also referred to as peers. Many networks are combination peer-to-peer and server based networks. The network operating system uses a network data protocol to communicate on the network to other computers. The network operating system supports the applications on that computer. A Network Operating System (NOS) includes Windows NT, Novell Netware, Linux, Unix and others.

The physical topology of a network refers to the configuration of cables, computers, and other peripherals. Physical topology should not be confused with logical topology which is the method used to pass information between workstations. Main Types of Network Topologies In networking, the term "topology" refers to the layout of connected devices on a network. This article introduces the standard topologies of computer networking.

One can think of a topology as a network's virtual shape or structure. This shape does not necessarily correspond to the actual physical layout of the devices on the network. For example, the computers on a home LAN may be arranged in a circle in a family room, but it would be highly unlikely to find an actual ring topology there.

16.3.1 Types Network Topologies

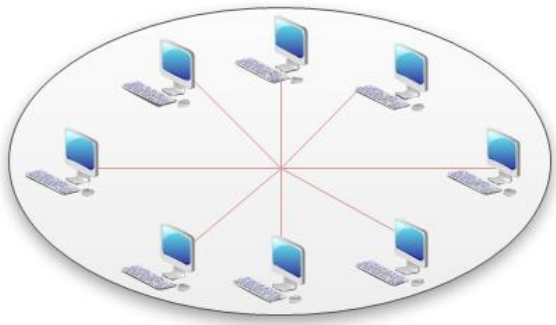
- Star Topology
- Ring Topology
- Bus Topology
- Tree Topology
- Mesh Topology
- Hybrid Topology

More complex networks can be built as hybrids of two or more of the above basic topologies.

Star Topology

Star Topology Many home networks use the star topology. A star network features a central connection point called a "hub" that may be a hub, switch or router. Devices typically connect to the hub with Unshielded Twisted Pair (UTP) Ethernet.

Compared to the bus topology, a star network generally requires more cable, but a failure in any star network cable will only take down one computer's network access and not the entire LAN. (If the hub fails, however, the entire network also fails.)



Star Topology

Advantages of a Star Topology :

a. Easy to install and wire.
b. No disruptions to the network then connecting or removing devices.
c. Easy to detect faults and to remove parts.

Disadvantages of a Star Topology:

a. Requires more cable length than a linear topology.
b. If the hub or concentrator fails, nodes attached are disabled.
c. More expensive than linear bus topologies because of the cost of the concentrators.

The protocols used with star configurations are usually Ethernet or LocalTalk. Token Ring uses a similar topology, called the star-wired ring.

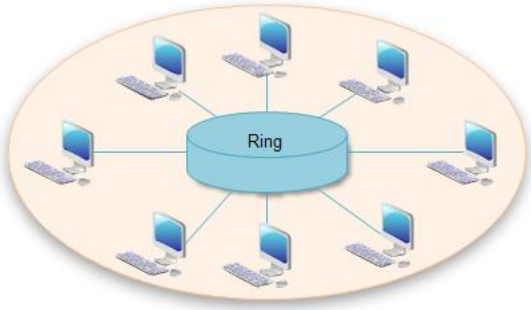
Star-Wired Ring

A star-wired ring topology may appear (externally) to be the same as a star topology. Internally, the MAU of a star-wired ring contains wiring that allows information to pass from one device to another in a circle or ring (See fig. 3). The Token Ring protocol uses a star-wired ring topology.

Ring Topology

Ring Topology In a ring network, every device has exactly two neighbors for communication purposes. All messages travel through a ring in the same direction (either "clockwise" or "counterclockwise"). A failure in any cable or device breaks the loop and can take down the entire network.

To implement a ring network, one typically uses FDDI, SONET, or Token Ring technology. Ring topologies are found in some office buildings or school campuses.

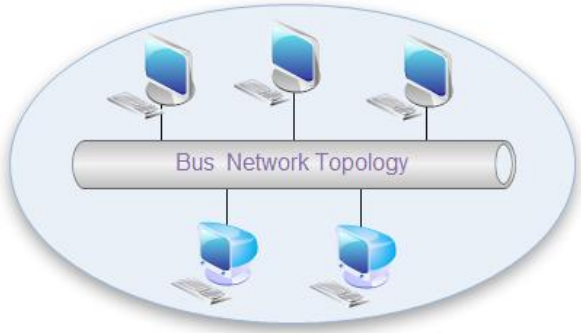


Ring Topology

Bus Topology

Bus Topology Bus networks (not to be confused with the system bus of a computer) use a common backbone to connect all devices. A single cable, the backbone functions as a shared communication medium that devices attach or tap into with an interface connector. A device wanting to communicate with another device on the network sends a broadcast message onto the wire that all other devices see, but only the intended recipient actually accepts and processes the message.

Ethernet bus topologies are relatively easy to install and don't require much cabling compared to the alternatives. 10Base-2 ("ThinNet") and 10Base-5 ("ThickNet") both were popular Ethernet cabling options many years ago for bus topologies. However, bus networks work best with a limited number of devices. If more than a few dozen computers are added to a network bus, performance problems will likely result. In addition, if the backbone cable fails, the entire network effectively becomes unusable.



Bus Topology

Advantages of a Linear Bus Topology:

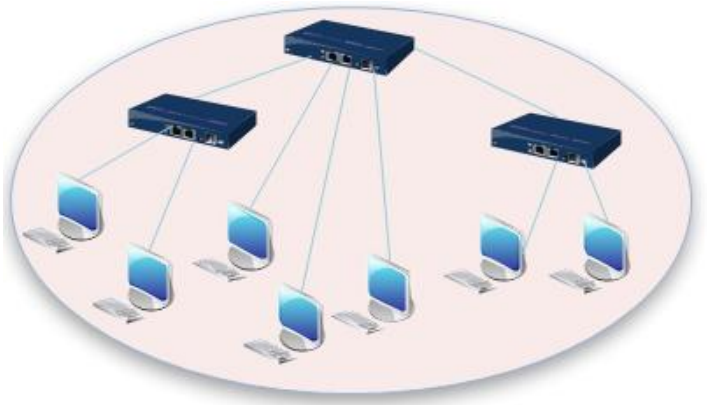
a. Easy to connect a computer or peripheral to a linear bus.
b. Requires less cable length than a star topology.

Disadvantages of a Linear Bus Topology:

a. Entire network shuts down if there is a break in the main cable.
b. Terminators are required at both ends of the backbone cable.
c. Difficult to identify the problem if the entire network shuts down.
d. Not meant to be used as a stand-alone solution in a large building.

Tree Topology

Tree Topology Tree topologies integrate multiple star topologies together onto a bus. In its simplest form, only hub devices connect directly to the tree bus, and each hub functions as the "root" of a tree of devices. This bus/star hybrid approach supports future expandability of the network much better than a bus (limited in the number of devices due to the broadcast traffic it generates) or a star (limited by the number of hub connection points) alone.



Tree Topology

Advantages of a Tree Topology:

a. Point-to-point wiring for individual segments.
b. Supported by several hardware and software venders.

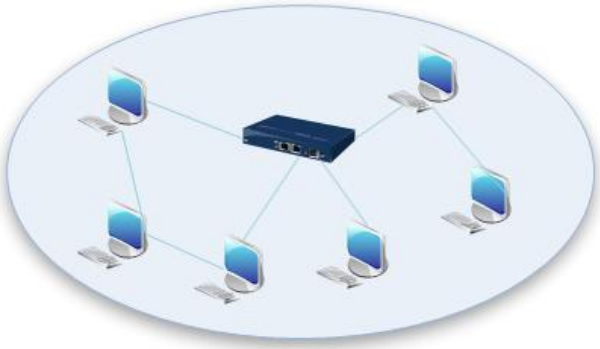
Disadvantages of a Tree Topology:

a. Overall length of each segment is limited by the type of cabling used.
b. If the backbone line breaks, the entire segment goes down.
c. More difficult to configure and wire than other topologies.

Mesh Topology

Mesh Topology Mesh topologies involve the concept of routes. Unlike each of the previous topologies, messages sent on a mesh network can take any of several possible paths from source to destination. (Recall that even in a ring, although two cable paths exist, messages can only travel in one direction.) Some WANs, most notably the Internet, employ mesh routing.

A mesh network in which every device connects to every other is called a full mesh. As shown in the illustration below, partial mesh networks also exist in which some devices connect only indirectly to others.



Mesh Topology

Hybrid Topology

A combination of any two or more network topologies. Note 1: Instances can occur where two basic network topologies, when connected together, can still retain the basic network character, and therefore not be a hybrid network. For example, a tree network connected to a tree network is still a tree network. Therefore, a hybrid network accrues only when two basic networks are connected and the resulting network topology fails to meet one of the basic topology definitions. For example, two star networks connected together exhibit hybrid network topologies. Note 2: A hybrid topology always accrues when two different basic network topologies are connected.

16.4 Network Devices

Network adapter cards: expansion cards that provide the physical connection between each computer and the network. The card installs into a slot on your computer, just like a sound card or modem card. Some newer computers have a network adapter already built into the system. Laptop computers often use a card that slides into a PC card slot.



Figure Network adapter card.

Network hub: the central connection point for network cables that connect to computers or other devices on a network. The hub has several network cable jacks or ports that you use to connect network cables to computers. The hub contains circuitry that enables each computer to communicate with any other computer connected to the hub (see Figure).



Figure : Network Hub.

Network cables: special, unshielded twisted-pair (UTP) cables used to connect each computer to the hub. The cable you need is Category 5 UTP cable with a square plastic RJ-45 connector on each end.

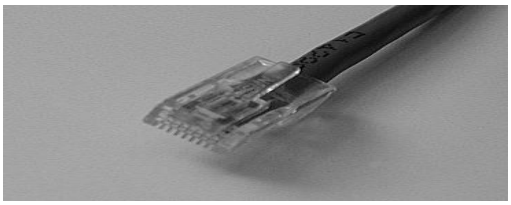


Figure : Network cable with RJ-45 connector.

All the networking hardware described here is known as Ethernet. Ethernet is the industry-wide standard for computer networks. Standard Ethernet networks transmit data at 10 million bits per second (Mbps). A newer Ethernet standard, called Fast Ethernet, transmits data at 100 Mbps. Computer networks often contain a mixture of 10 Mbps and 100 Mbps devices.

Wireless Networking Hardware

You may want to network computers where it is expensive or difficult to run network cables, for example, between two rooms or two buildings. However, recent advances in wireless networking technology make wireless networking practical and affordable. New wireless standards have facilitated the development of wireless products with good performance and the ability to integrate easily into a wired Ethernet network. The Ethernet standard for wireless networking is the IEEE 802.11b wireless standard. The 802.11b standard supports wireless connections at speeds up to 11 Mps, comparable to 10 Mbps wired Ethernet. Wireless industry leaders formed the Wireless Ethernet Compatibility Alliance (WECA) to certify cross-vendor compatibility with the 802.11b standard. These products display the WECA "Wireless Fidelity" (Wi-Fi) logo.

Figure : Wireless Fidelity Logo.

Suppose you want to network a few computers together in a small area where it would be expensive to have network cabling installed in an existing building. Or perhaps you just have a desktop computer and a notebook computer at home and you would like to be able to roam the house with the notebook computer and perhaps even browse the Web from the hammock in the back yard. Wireless Ethernet makes all this possible. You can install wireless adapters in each computer and form a wireless network Figure .



Figure : PC Card and USB Wireless Adapters.

Other Types of Networking Hardware

Other networking technologies are available that enable you to network a small number of computers together in a home or office. These technologies often use the telephone wiring or power lines to connect computers. Some use alternative wireless standards. While it is sometimes easier to install these networking systems, their performance and capabilities are limited. The performance is typically 10 times slower than the current Ethernet capabilities. While Fast Ethernet transmits data at 100 Mbps, these technologies typically transmit data between 1 Mbps and 10 Mbps. Also, the capabilities of these devices are often limited to the capabilities of the devices sold by one manufacturer. You usually cannot mix devices made by different manufacturers. The following table compares Ethernet to some of these different networking technologies.

Table: Different networking technologies.

Technology	**Medium**	**Speed**	**Notes**
Ethernet and Fast Ethernet	Category 5 networking cable	10 Mbps 100 Mbps	IEEE 802.3 standard for networks. Most computer networks use Ethernet, Fast Ethernet or a combination of the two.
Wireless Ethernet	Radio frequency (RF)	11 Mbps	IEEE 802.11b standard for wireless networking. Easily connects to a wired Ethernet network using an access point.
HomePNA (Home Phone line Networking Alliance)	Telephone wiring	1 Mbps or 10 Mbps	Uses existing home telephone wiring, which can also be used for simultaneous phone service. Easy to install, but is not compatible with Ethernet. Look for products that support 10 Mbps.
HomeRF (Home Radio Frequency Working Group)	Radio frequency (RF)	1.6 Mbps	A wireless networking technology designed to carry voice and data in the home. A new 2.0 version of HomeRF supports 10 Mbps. Not compatible with 802.11b.
Power line network	Electrical power lines	2 Mbps	New HomePlug Power line Alliance specification supports 14 Mbps. Few products available.
Bluetooth	Radio frequency (RF)	700 Kbps	A low-power, short-range wireless technology used for communication between digital devices, or between computers and peripherals.

Recommendations

If you are installing a new network, the best choice is standard Ethernet hardware. This is the same networking hardware used by thousands of businesses and corporations to connect millions of computers

together. Ethernet networking components are standardized, inexpensive, dependable, and easy to install and maintain. Ethernet hardware is widely available. You can find network hubs, adapters, and cables at most stores that specialize in computer sales. Because all manufacturers of Ethernet hardware adhere to the Ethernet standards, you can buy any component from any manufacturer and connect it to Ethernet components you already have. Wireless Ethernet is the best choice if you are installing a wireless network. To make sure the hardware is 802.11b compatible, look for the Wi-Fi logo on the product box. The Wi-Fi logo indicates the product is certified by the Wireless Ethernet Compatibility Alliance (WECA). Because these products are standardized, you can buy products from different manufacturers and use them together. Wireless Ethernet products have become widely available and continue to drop in price. If you use standard Ethernet and 802.11 wireless networking products, you can easily connect wireless and wired networks together using a wireless access point.

New Technologies

New Ethernet standards support even higher data rates for both wired and wireless networks.

Gigabit Ethernet: This new Ethernet standard transfers data at 1000 Mbps (1 Gbps) using standard Category 5 networking cables. If you install this cable today, you can migrate to the faster hardware should the need arise. Gigabit adapters, hubs and switches are available today, but Fast Ethernet is likely to provide adequate bandwidth for most networking applications on a small network. In most cases, the Ethernet hardware that you purchase today will be able to interoperate with newer Gigabit hardware.

802.11a: This new wireless standard supports speeds up to 54 Mbps. It uses technology similar to 802.11b, but operates at 5 GHz rather than at the 2.4 GHz band used for 802.11b. The higher frequency makes 802.11a less susceptible to interference from other devices such as cell phones, cordless phones, and microwave ovens. An 802.11a network can operate without interference in the same location as an 802.11b network, or near Bluetooth devices, which operate in the same frequency spectrum as 802.11b.

802.11g: This new wireless standard also supports speeds up to 54 Mbps. It is an extension of 802.11b and operates in the same RF spectrum as 802.11b. While 802.11g offers a clean upgrade path from 802.11b, 802.11a is less likely to be affected by interference. It is likely that one of these two competing technologies will become widely adopted. These new technologies are likely to be more expensive until their use becomes widespread. If you choose to use any of these new technologies, make sure your new hardware is compatible with any existing hardware you have. If you choose 802.11a or 802.11g, you may want to choose adapters that are compatible with 802.11b. Compatibility with 802.11b will let you connect to networks that do not support the newer technology.

16.5 Types of Networks

There are many different types of networks. However, from an end user's point of view there are two basic types:

16.5.1 Local Area Network (LAN)

The LAN is a computer network that spans a relatively small area. Most LANs are confined to a single building or group of buildings. However, one LAN can be connected to other LAN's over any distance via telephone lines and radio waves. A system of LANs connected in this way is called a wide-area network (WAN).

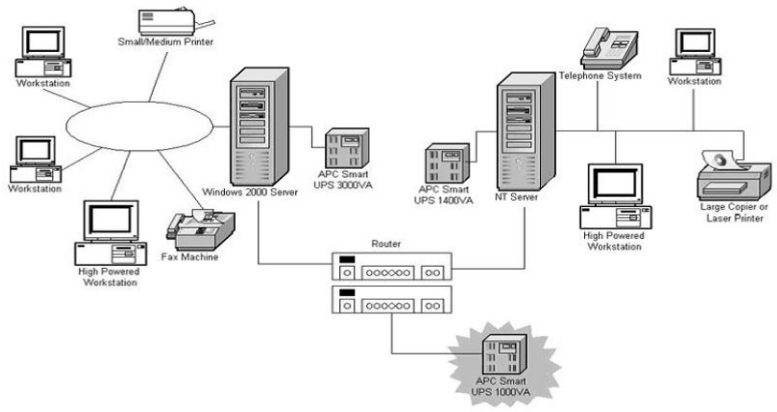


Figure Local Area Network.

Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Draw the basic flow diagram of communication in the network.

……..……

……..……

……

16.5.2 Wide-Area Networks (WANs)

The computers are farther apart and are connected by telephone lines or radio waves.

In addition to these types, the following characteristics are also used to categorize different types of networks.

Topology: The geometric arrangement of a computer system. Common topologies include bus, star, and ring.

Protocol

The protocol defines a common set of rules and signals that computers on the network use to communicate. One of the most popular protocols for LANs is called Ethernet. Another popular LAN protocol for PCs is the IBM token-ring network.

Architecture

Networks can be broadly classified as using either peer-to-peer or client/server architecture. Computers on a network are sometimes called nodes. Computers and devices that allocate resources for a network are called servers.

The types of networks can be further classified into two more divisions:

16.5.3 Peer-to-Peer Networking

This is a simple network configuration that requires some basic know-how to set up. Each of the interconnected machines shares dual capability and responsibility on the network. That is to say, that each machine serves a dual purpose or role, i.e. they are both clients and servers to some extent.

The server capability of the machines is very basic. The services provided by each, is no more than the ability to share resources like files, folders, disk drives and printers. They even have the ability to share Internet access.

However, the server functionality of these machines stops there. They cannot grant any of the benefits mentioned previously, since these are functions provided only by a dedicated server operating system.

Because all machines on the network have equal status, hence the term peers; there is no centralized control over shared resources. Sharing is endorsed or repealed by each machine's user. Passwords can be

assigned to each individual shared resource whether it is a file, folder, drive or peripheral, again done by the user.

Although this solution is workable on small networks, it introduces the possibility that users may have to know and remember the passwords assigned to every resource and then re-learn them if the user of a particular machine decides to change them! Due to this flexibility and individual discretion, institutionalised chaos is the norm for peer-to-peer networks.

Security can also be a major concern, because users may give passwords to other unauthorised users, allowing them to access areas of the network that the company does not permit. Furthermore, due to lack of centralisation, it is impossible for users to know and remember what data lives on what machine, and there are no restrictions to prevent them from over-writing the wrong files with older versions of the file. This of course cripples attempts to organise proper backups.

It may appear that peer-to-peer networks are hardly worthwhile. However, they offer some powerful incentives, particularly for smaller organisations. Networks of this type are the cheapest and easiest to install, requiring only Windows XP, a network card for each machine and some cabling. Once connected, users can start to share information immediately and get access to devices.

16.5.4 Client/Server Networks

Server based networks, or client/server networks as they are properly called, has a machine at the heart of its operations called the server. A server is a machine that provides services over a network by responding to client requests. Servers rarely have individuals operating it, and even then, it is usually to install, configure or manage its capabilities. The server's essential role on the network is to be continuously available to handle the many requests generated by its clients.

Server-based networks provide centralised control of the entire network environment. The computer systems used for this role are generally more powerful than end-user machines, incorporating faster CPUs, more memory, larger disk drives and other drive types installed, like a tape drive for backup purposes. These are required, because servers are dedicated to handling multiple simultaneous requests from their client communities.

Server based networks provide centralised verification of user accounts and passwords. Only valid account name and password combinations are allowed access to the network. Client/Server networks typically require a single login to the network itself, meaning that users need to remember long password lists to access various resources. Concentrations of resources on a single server mean that they are easier to find, as opposed to the peer-to-peer model, were resources were distributed throughout the network since they were attached to multiple machines. The server being a central data repository, means that not only is data more accessible to users, but it also makes life much easier in terms of performing backups, since the data is in a location know to the administrator.

Server-based networks are easier to scale. Peer-to-peer networks bog down seriously as they grow beyond ten users, and serious slow up with 20 users. On the other hand, client/server networks can handle a few users; up to a thousand users as such networks grow to keep pace with an organisations growth and expansion.

Unlike peer-to-peer networks, client/server networks do not come cheap. The server machine itself may cost several thousands of pounds, along with the software to make it run; another thousand pounds. Because of the complex nature of this kind of networking environment, a dedicated administrator is required to be on site at all times to be involved in the day to day running of the network. Hiring an individual of this nature adds considerably to the cost of client/server networks.

Lastly, because the networks operability is so dependent upon the server, this introduces a single point of failure, if the server goes down the network goes down. There are measures available, that can legislate for such failures, and however these techniques add even more cost to this solution.

Did You Know?

In 1965 Thomas Merrill and Lawrence G. Roberts created the first wide area network (WAN).

Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Draw the block diagram of peer to peer network.

……..……

……..……

……

16.6 Types of Connectivity

(A) Point-to-Point Connection
(B) Multipoint Connection.

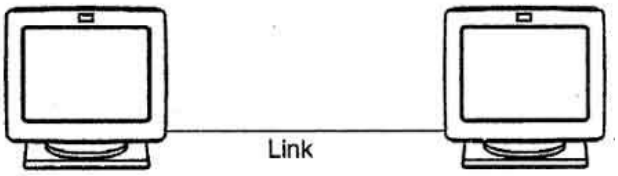
(A) Point-to-Point Connection

A point-to-point connection is a direct link between two devices such as a computer and a printer. It uses dedicated link between the devices. The entire capacity of the link is used for the transmission between those two devices. Most of today's point-to-point connections are associated with modems and PSTN (Public Switched Telephone Network) communications. In point to point networks, there exist many connections between individual pairs of machines.

To move from sources to destination, a packet (short message) may follow different routes. In networking, the Point-to-Point Protocol (PPP) is a data link protocol commonly used in establishing a direct connection between two networking nodes. It can provide connection authentication, transmission encryption, and compression PPP is used over many types of physical networks including serial cable, phone line, trunk line, cellular telephone, specialized radio links, and fiber optic links such as SONET. PPP is also used over Internet access connections (now marketed as "broadband").

Internet service providers (ISPs) have used PPP for customer dial-up access to the Internet, since IP packets cannot be transmitted over a modem line on their own, without some data link protocol. Two encapsulated forms of PPP, Point-to-Point Protocol over Ethernet (PPPoE) and Point-to-Point Protocol over ATM (PPPoA), are used most commonly by Internet Service Providers (ISPs) to establish a Digital Subscriber Line (DSL) Internet service connection with customers.

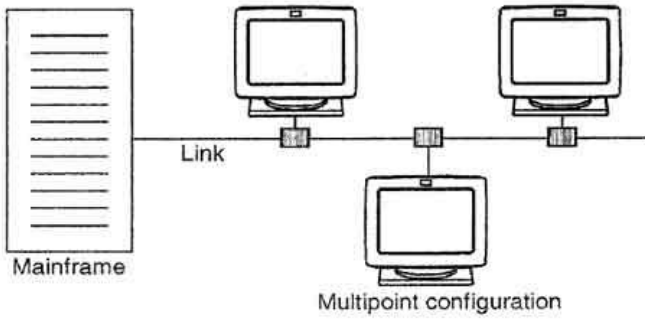
PPP is commonly used as a data link layer protocol for connection over synchronous and asynchronous circuits, where it has largely superseded the older Serial Line Internet Protocol (SLIP) and telephone company mandated standards (such as Link Access Protocol, Balanced (LAPB) in the X.25 protocol suite). PPP was designed to work with numerous network layer protocols, including Internet Protocol (IP), TRILL, Novell's Internetwork Packet Exchange (IPX), NBF and AppleTalk.



Point to Point Configuration

B) Multipoint Connection.

A multipoint connection is a link between three or more devices. It is also known as Multi-drop configuration. The networks havjng multipoint configuration are called Broadcast Networks. In broadcast network, a message or a packet sent by any machine is received by all other machines in a network. The packet contains address field that specifies the receiver. Upon receiving a packet, every machine checks the address field of the packet. If the transmitted packet is for that particular machine, it processes it; otherwise it just ignores the packet.



Multipoint configuration

Broadcast network provides the provision for broadcasting & multicasting. Broadcasting is the process in which a single packet is received and processed by all the machines in the network. It is made possible by using a special code in the address field of the packet. When a packet is sent to a subset of the machines i.e. only to few machines in the network it is known as multicasting. Historically, multipoint connections were used to attach central CPs to distributed dumb terminals. In today's LAN environments, multipoint connections link many network devices in various configurations.

Self Assessment Questions

1. Transmission media are usually categorized as……..

(a) fixed or unfixed (b) guided or unguided

(c) determinate or indeterminate (d). metallic or non-metallic

2. . Radio communication frequencies range from……

(a). 3 KHz to 300 KHz (b). 300 KHz to 3 GHz

(c). 3 KHz to 300 GHz (d). 3KHz to 3000 GHz

3. In fiber optics, the signal source is……waves.

(a) Light (b) radio

(c) Infrared (d) very low frequency

4. As frequency increase, the period …….

(a) Doubles (b) Increases

(c) Remains the same (d) Decreases

5. In cellular telephony, a service area is divided into small regions called……

(a) Cell (b) Cell offices

(c) MTSOs (d) Relay sites

6. In an optical fiber, the inner core is……..the cladding.

(a). More dense than (b) Less dense than

(c). The same density as (d). Another-name for

16.7 Summary

- Signals travel from transmitter to receiver via a path. This path, called the medium, guided or unguided.

- A guided medium is contained within physical boundaries, while an unguided, medium is boundless.
- Radio waves used to transmit data. These waves use unguided are usually propagated through the air.
- Fiber-optic cables are composed of a glass or plastic inner core surrounded by cladding, all encased in an outside jacket.
- Satellite communication uses a satellite in geosynchronous orbit to relay signals. A system of three correctly spaced satellites covers most of the earth.
- The Shannon capacity is a formula to determine the theoretical maximum data rate for a channel.

16.8 Keywords

Cellular telephony: Cellular telephony is moving fast toward integrating the existing system with satellite communication.

Guided media: It provides a conduit from one device to another; include twisted-pair cable, coaxial cable, and fiber-optic cable. A signal travelling along any of these media is directed and contained by the physical limits of the medium.

Optical fiber: Optical fiber is a glass or plastic cable that accepts and transports signals in the form of light.

Reflection: When the angle of incidence becomes greater than the critical angle, a new phenomenon occur called reflection.

Satellite transmission: Satellite transmission is much like line-of-sight microwave transmission in which one of the stations is a satellite orbiting the earth.

16.9 Review Questions

1. Which parts of the electromagnetic spectrum are used for communication?
2. Name the two major categories of transmission media.
3. How do guided media differ from unguided media?
4. What are the three major classes of guided media?
5. What is the major advantage of shielded twisted pair over unshielded twisted pair?
6. Why is coaxial cable superior to twisted-pair cable?
7. What happens to a beam of light as it travels to a less dense medium? What happens if it travels to a denser medium?
8. What are the three criteria used to evaluate transmission media?
9. What is reflection?
10. Discuss the modes for propagating light along optical channels.

Answers for Self Assessment Questions

1. (b) 2. (c) 3. (b) 4. (d) 5. (a) 6. (a)

16.10 Further Studies

1. http://ecomputernotes.com/computernetworkingnotes/computer-network/what-are-the-two-network-connection-types
2. http://technet.microsoft.com/en-us/library/cc780142(v=ws.10).aspx

PC Software

(MCA/PGDCA - 102)



Jaipur National University
Directorate of Distance Education

Established by Government of Rajasthan
Approved by UGC under Sec 2(f) of UGC ACT 1956
(**Recognised by Joint Committee of UGC-AICTE-DEC, Govt. of India**)

1

Office Packages

STRUCTURE

1.0 Objectives

1.1 Introduction

1.2 Office Activities and Their Software Requirement

1.3 Word Processing

1.4 Spreadsheet

1.5 Presentation Graphics

1.6 Microsoft Office

1.7 Lotus Office

1.8 Star Office

1.9 Open Office

1.10 Summary

1.11 Keywords

1.12 Review Questions

1.13 Further Studies

1.0 Objectives

After studying this chapter, you will be able to:

- Define the office activities and their software requirement
- Discuss the word processing
- Explain about the spreadsheet
- Discuss the presentation graphics
- Explain the database

1.1 Introduction

Office software forms a critical link between the primary systems in your day to day work. The initial choice of the office package has far reaching consequences; both for the future selection of additional software in the future and for the ease with which documents and information can be shared throughout the organization.

- Application software uses the computer system to perform useful work or provide entertainment functions beyond the basic operation of the computer itself.
- System software is designed to operate the computer hardware, to provide basic functionality, and to provide a platform for running application software. System software includes:
- Operating system, an essential collection of computer programs that manages resources and provides common services for other software. Supervisory programs, boot loaders, shells and window systems are core parts of operating systems. In practice, an operating system comes bundled with additional software (including application software) so that a user can potentially do some work with a computer that only has an operating system.
- Device driver, a computer program that operates or controls a particular type of device that is attached to a computer. Each device needs at least one corresponding device driver; thus a computer needs more than one device driver.
- Utilities, software designed to assist users in maintenance and care of their computers.
- Malicious software or malware, computer software developed to harm and disrupt computers. As such, malware is undesirable. Malware is closely associated with computer-related crimes, though some malicious programs may have been designed as practical jokes.

1.2 Office Activities and Their Software Requirement

Administration and supports are personnel in a financial services company. They carry out functions like settlements, clearances, record maintenance, regulatory compliance, and accounting. When order processing is slow due to high volume, it is commonly referred to as "back office crunch."

1.3 Word Processing

Using a computer to create, edit, and print documents of all computer applications, word processing is the most common. To perform word processing, you need a computer, a special program called a word processor, and a printer. A word processor enables you to create a document, store it electronically on a disk, display it on a screen, modify it by entering commands and characters from the keyboard, and print it on a printer.

The great advantage of word processing over using a typewriter is that you can make changes without retyping the entire document. If you make a typing mistake, you simply back up the cursor and correct your mistake. If you want to delete a paragraph, you simply remove it, without leaving a trace. It is equally easy to insert a word, sentence, or paragraph in the middle of a document. Word processors also make it easy to move sections of text from one place to another within a document, or between documents. When you have made all the changes you want, you can send the file to a printer to get a hardcopy.

Word processors vary considerably, but all word processors support the following basic features:
***Insert text*:** Allows you to insert text anywhere in the document.
Delete text: Allows you to erase characters, words, lines, or pages as easily as you can cross them out on paper.
Cut and paste: Allows you to remove (cut) a section of text from one place in a document and insert (paste) it somewhere else.
Copy: Allows you to duplicate a section of text.
***Page size and margins*:** Allows you to define various page sizes and margins, and the word processor will automatically readjust the text so that it fits.

Search and replace: Allows you to direct the word processor to search for a particular word or phrase. You can also direct the word processor to replace one group of characters with another everywhere that the first group appears.
Word wrap: The word processor automatically moves to the next line when you have filled one line with text, and it will readjust text if you change the margins.
Print: Allows you to send a document to a printer to get hardcopy.

Word processors that support only these features (and maybe a few others) are called **text editors.** Most word processors, however, support additional features that enable you to manipulate and format documents in more sophisticated ways. These more advanced word processors are sometimes called full-featured word processors. Full-featured word processors usually support the following features:

File management: Many word processors contain file management capabilities that allow you to create, delete, move, and search for files.
Font specifications: Allows you to change fonts within a document. For example, you can specify bold, italics, and underlining. Most word processors also let you change the font size and even the typeface.
Footnotes and cross-references**:** Automates the numbering and placement of footnotes and enables you to easily cross-reference other sections of the document.
Graphics: Allows you to embed illustrations and graphs into a document. Some word processors let you create the illustrations within the word processor; others let you insert an illustration produced by a different program.
Headers, footers, and page numbering: Allows you to specify customized headers and footers that the word processor will put at the top and bottom of every page. The word processor automatically keeps track of page numbers so that the correct number appears on each page.
Layout: Allows you to specify different margins within a single document and to specify various methods for indenting paragraphs.
Macros: A macro is a character or word that represents a series of keystrokes. The keystrokes can represent text or commands. The ability to define macros allows you to save yourself a lot of time by replacing common combinations of keystrokes.
Merges**:** Allows you to merge text from one file into another file. This is particularly useful for generating many files that have the same format but different data. Generating mailing labels is the classic example of using merges.
Spell checker: A utility that allows you to check the spelling of words. It will highlight any words that it does not recognize.
Tables of contents and indexes**:** Allows you to automatically create a table of contents and index based on special codes that you insert in the document.
Thesaurus: A built-in thesaurus that allows you to search for synonyms without leaving the word processor.
Windows: Allows you to edit two or more documents at the same time. Each document appears in a separate window. This is particularly valuable when working on a large project that consists of several different files.
WYSIWYG *(what you see is what you get):* With WYSIWYG, a document appears on the display screen exactly as it will look when printed.

The line dividing word processors from desktop publishing systems is constantly shifting. In general, though, desktop publishing applications support finer control over layout, and more support for full-color documents.

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.

Ex1: Create the hyperlink to an attachment and create PDF in WordPerfect.

……..……

……..……

……...

1.4 Spreadsheet

A table of values arranged in rows and columns. Each value can have a predefined relationship to the other values. If you change one value, therefore, you may need to change other values as well.

Spreadsheet applications (sometimes referred to simply as spreadsheets) are computer programs that let you create and manipulate spreadsheets electronically. In a spreadsheet application, each value sits in a cell. You can define what type of data is in each cell and how different cells depend on one another. The relationships between cells are called formulas, and the names of the cells are called labels.

Once you have defined the cells and the formulas for linking them together, you can enter your data. You can then modify selected values to see how all the other values change accordingly. This enables you to study various what-if scenarios.

A simple example of a useful spreadsheet application is one that calculates mortgage payments for a house. You would define five cells:

1. Total cost of the house
2. Down payment
3. Mortgage rate
4. Mortgage term
5. Monthly payment

Once you had defined how these cells depend on one another, you could enter numbers and play with various possibilities. For example, keeping all the other values the same, you could see how different mortgage rates would affect your monthly payments.

There are a number of spreadsheet applications on the market, Lotus 1-2-3 and Excel being among the most famous. The more powerful spreadsheet applications support graphics features that enable you to produce charts and graphs from the data.

Most spreadsheet applications are multidimensional, meaning that you can link one spreadsheet to another. A three-dimensional spreadsheet, for example, is like a stack of spreadsheets all connected by formulas. A change made in one spreadsheet automatically affects other spreadsheets.

Did You Know?

Dan Bricklin also known as the "father" of the electronic spreadsheet.

1.5 Presentation Graphics

A type of business software that enables users to create highly stylized images for slide shows and reports. The software includes functions for creating various types of charts and graphs and for inserting text in a variety of fonts. Most systems enable you to import data from a spreadsheet application to create the charts and graphs.

Examples of presentation graphics software include PowerPoint, Harvard Graphics, Corel Draw and 3D studio, among others.

The following examples were selected to illustrate principles, not exhaust possibilities. It is organized by the configuration and number of variables, categorical and continuous, that we wish to graph.

- One Variable
- Two variables
- Three variables

Exercise: Check Your Progress 2
Note: i) Use the space below for your answer.
Ex1: Create a database using Microsoft Excel.

……..……

……..……

……...

1.6 Microsoft Office

Office packages are bundles of programs that help you with daily 'office' chores, such as writing documents, making extensive calculations, preparing presentations, maintaining databases etc. While there is a number of competing Office suites, Microsoft Office is the unchallenged market leader and is used by the majority of companies around the world.

1.6.1 The Office button

At the very top left corner of the Office window, you will see what is referred to as the Office button, mainly because it has the Office logo on it, but also because it gives you quick access to many of Office's most important tasks. Among these tasks: open a document, save your work, print your document, and publish your work to a shared work space, and a lot more. From this button, you can also access a list of the most recent dozen and other documents you have work done. The Office button also takes all of the non-document related activities and puts them in one spot. By non-document, It mean tasks that do not directly relate to the editing task at hand. These items include Open, Save, Print, Close, and more. From the button, you can also configure overall product options. In Word, for example, you can set your proofing options, save options and more. See Figure 1.1 for an example of what you will find on the Office button.

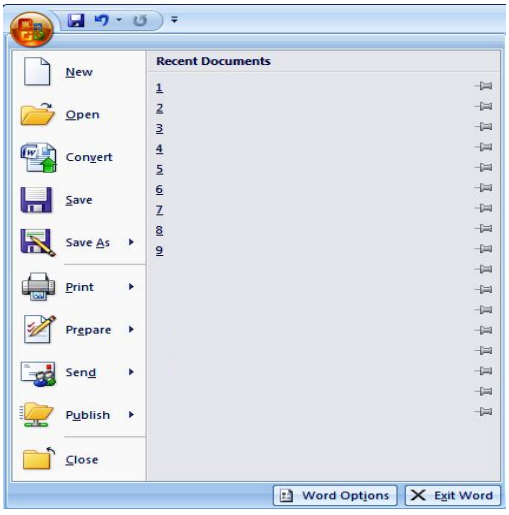


Figure 1.1: Office button.

Items on the Office menu that have arrows to the right of the entry have sub-options. For example, in Excel 2007, when you click on the Office button and hover over Save As, you are provided with a list of the possible save options, as shown in Figure 1.2.

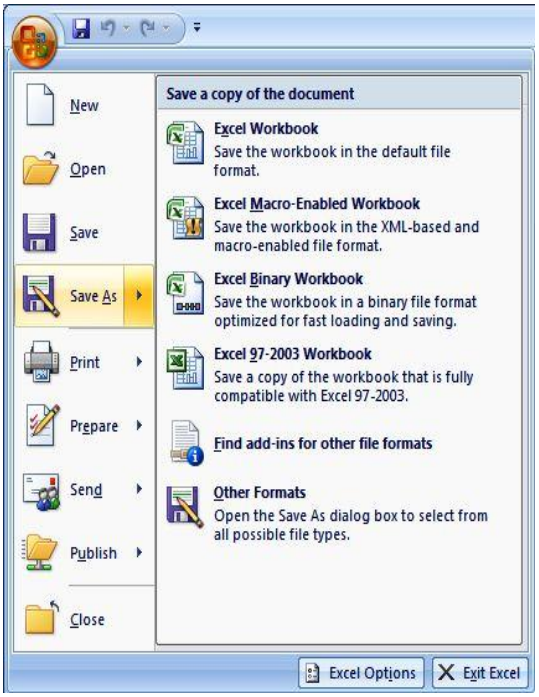


Figure 1.2: The Save As menu in Excel 2007.

1.6.2 The Ribbon

In most of the Office 2007 products, Microsoft has foresworn these tried and true interface objects in favor of something more streamlined: Ribbon takes up a good chunk of the top portion of the screen the section once occupied by the menu bar and various tool bars. Your initial use of

Office 2007 with the new Ribbon may make you wonder why Microsoft would use interface real estate in this way but, after using the Ribbon for a while, you will probably see how its use can result insignificant improvements to the way you work. However, it will take some time, especially if you are an Office power user.

Users that are very familiar with the old Office interface will have the hardest time adjusting to the new system. See Figure 1.7 for a look at the Ribbon in Word 2007. Figure 1.4 shows you the Ribbon used in Excel 2007. Notice that the Ribbon is broken down into a number of tabs, including the Insert tab, from which you can add visual elements, such as tables, charts and more, to your Word document. The Page Layout tab replaces the Page Setup dialog and provides a place for you to change your document's margins, page size, indentation, and more.

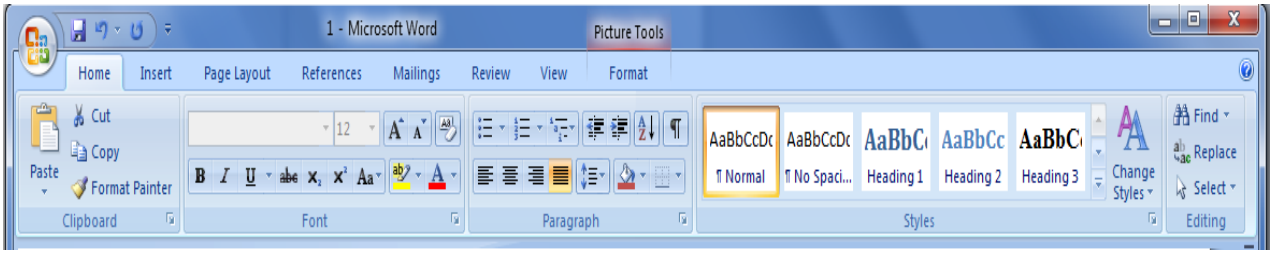


Figure 1.3: The Word Ribbon puts the most necessary items on the Home tab.

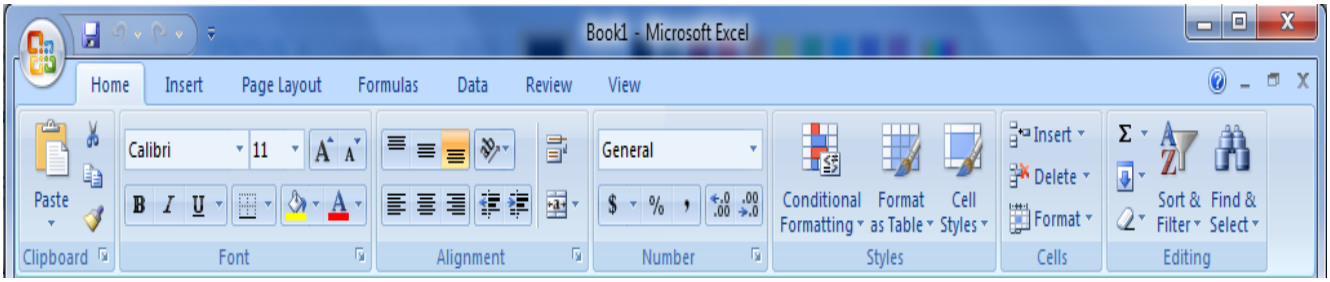


Figure 1.4: The Excel ribbon houses Excel-specific tasks.

The Ribbon provides a contextual experience for your users. By that, it means that the tabs that are available on the Ribbon change based on the document context. If a user is working with a table, for example, a Table Tools section is added to the Ribbon with Design and Layout tabs. These new tabs are visible only when your insertion point is within a table, and stay out of your way at other times. Figure 1.5 shows you an example of the Table Tools context sensitive tabs.

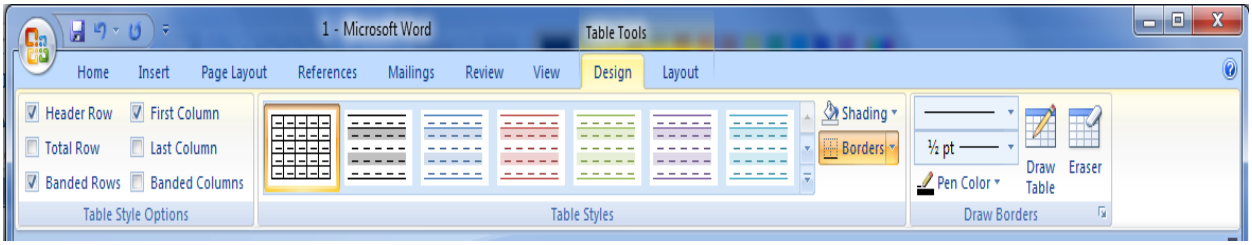


Figure 1.5: Table Tools context sensitive tabs.

Context sensitive tabs keeps the clutter out of your interface when it is not needed. If you are more comfortable working with a more traditional dialog box, these have not been eliminated from Office. In fact, many of the most common dialog boxes are accessible via a single click of the mouse. Take a look back at Figures 1.3, 1.4, and 1.5. In the lower right-hand corner of most of the various sections of the Ribbon, take note of the small arrow pointing down and to the right. These icons open up the associated traditional dialog box. For example, if you click on the arrow icon in the Font section of the Ribbon in Word, the Font dialog box will open. Since not every single option will fit on the Ribbon, these dialog boxes remain useful.

In the Ribbon bar, on the Home tab, you can also see the most obvious example of galleries. A gallery is basically an example of what a particular style will look like. Word, Excel and PowerPoint make liberal use of galleries. Word uses them to give you a look at what would happen if you applied a particular style to your document. Excel uses them to apply formatting to your spreadsheets and PowerPoint uses them so you can get a look at what a particular template might look like.

To use a gallery, just hover your mouse pointer over one of the representations in the Ribbon. In all Office programs that have a gallery, hovering the mouse pointer over the sample actually temporarily applies that style to your work. As you move across the gallery, you can see each style in turn. To apply a particular style to your work, click the style.

Did You Know?

Office 3.0 was the first version of the Microsoft office came in existence in 1992. It was one of the first Microsoft products to come out on both CD-ROM and on floppy disk, adding to the convenience of the package.

Self Assessment Questions

1……..also make it easy to move sections of text from one place to another within a document, or between documents.

(a) Word wraps (b) Word processors

(c) Word document (d) Word Ribbon.

2. ……..is word processor software produced by IBM's Lotus Software group for use on Microsoft Windows-compatible computers and on IBM OS/2 Warp?

(a) Lotus Word Pro (b) Word processors

(c) Word document (d) Word Ribbon.

3. ……was based upon Ami Pro but was substantially rewritten

(a) Word wraps (b) Word processors

(c) Word Pro (d) Word Ribbon.

4. ……is a part of the Lotus SmartSuite office suite for Microsoft Windows.

(a) Word wraps (b) Lotus Freelance Graphics

(c) Lotus Word Pro (d) Word Ribbon.

5 ……is a personal information manager package.

(a) Word wraps (b) Lotus Freelance Graphics

(c) Lotus Word Pro (d) Lotus Organizer

1.7 Lotus Office

Lotus Software (called Lotus Development Corporation before its acquisition by IBM). Lotus is most commonly known for the Lotus 1-2-3 spreadsheet application, SmartSuite is an office suite from Lotus Software. Lotus made versions for IBM's OS/2, as well as Microsoft Windows versions. SmartSuite is in maintenance mode, and supported with fixes and fix packs on Windows 2000 and Windows XP. SmartSuite is not officially supported by IBM on the Windows Vista operating system, but it does work on the 32-bit version of Vista if the installer and applications are run in XP compatibility mode (this is not needed to install or run Organizer6). IBM has no plans to release specific Vista-compatible versions of SmartSuite or Organizer.

1.7.1 Lotus Word Pro–word processor.

Lotus Word Pro is word processor software produced by IBM's Lotus Software group for use on Microsoft Windows-compatible computers and on IBM OS/2 Warp. Word Pro can be obtained as part of the Lotus SmartSuite office suite.

Word Pro was based upon Ami Pro but was substantially rewritten (including a new native document format). Lotus obtained Ami Pro to round out their office suite by acquiring Samna, and continued to develop Ami Pro further, with version 3 becoming a 32-bit application available for Microsoft Windows and IBM OS/2. Create reports, documents and proposals in a snap with the word processor for today's Internet-centered world. Switching to Word Pro? You will feel comfortable right away — Word Pro offers excellent file compatibility with Microsoft Word, plus a choice of other menu formats, including Lotus Ami Pro, Microsoft Word, and WordPerfect.

1.7.2 Lotus 1-2-3—Spreadsheet

Lotus 1-2-3 is a spreadsheet program from Lotus Software (now part of IBM). It was the IBM PC's first "killer application"; The Lotus Development Corporation was founded by Mitchell Kapor, a friend of the developers of VisiCalc. The 1-2-3 was originally written by Jonathan Sachs, who had written two spreadsheet programs previously while working at Concentric Data Systems, Inc.

Unlike Microsoft Multiplan, it stayed very close to the model of VisiCalc, including the "A1" letter and number cell notation, and slash-menu structure. It was free of notable bugs, and was very fast because it was programmed entirely in x86 assembly language and bypassed the slower DOS screen input/output functions in favor of writing directly to memory-mapped video display hardware.

The name "1-2-3" stemmed from the product's integration of three main capabilities. Along with being a spreadsheet, it also offered integral charting/graphing and rudimentary database operations. Data features included sorting data in any defined rectangle, by order of information in one or two columns in the rectangular area. Justifying text in a range into paragraphs allowed it to be used as a primitive word processor.

Lotus Freelance Graphics is a part of the Lotus SmartSuite office suite for Microsoft Windows. (Previous versions were also released for OS/2.) It allows users to create and compile text, digital images, diagrams, basic drawings, and charts (such as bar charts and pie charts) into a digital slide show.

Lotus Smart Center — a toolbar that let users quickly access programs, their calendar, Internet bookmarks, and other resources

Lotus Approach is a relational database management system included in IBM's Lotus SmartSuite for Microsoft Windows.

Lotus Approach is the award-winning relational database designed to manage, analyze and report on business information. It offers breakthrough ease of use, unprecedented cross-product integration, connectivity, and outstanding power and analysis capabilities. Computing features maximize the sharing of information in the organization. Approach offers tight integration with Lotus Notes, making it an excellent tool for reporting on, analyzing and updating Notes data. Approach lets users seamlessly connect to all data, whether it is stored in dBASE, DB2, Oracle, Lotus Notes or almost anywhere else.

Lotus Organizer is a personal information manager package. It was initially developed by Threads, a small British software house, reaching version 3.0. Organizer was subsequently acquired by Lotus Development Corporation, for whom the package was a Windows-based replacement for Lotus Agenda. For several years it was the unquestioned market leader before it was gradually overtaken by Microsoft's Outlook. It is also the only PIM package recommended by the British Philosophical Association.

It is notable for using the organizer graphical metaphor for its user interface and is often bundled within Lotus SmartSuite.

Organizer was the first and most important software to be used as an agenda and its usability was so good that even now is still appreciated. The so called current version is actually more than 10 years old, as Lotus and IBM never updated the software after version 5, the 6 and 6.x versions are really minor upgrades.

The famous Covey organizer has a current software version that is obviously inspired in the Lotus Organizer. It is surprising how Lotus and IBM let go the chance of filling a market niche that Outlook never could reach.

IBM continues to support and ship Lotus Organizer. Version 6.1 is the most recent version, with support for Windows 2000 and Windows XP. Contents. It is an electronic day planner with tabs for each section and pages that turn. You can quickly see all your calendar, contacts, to does, calls, notes, Web information and more at a glance. No more looking for sticky-note reminders or lost scraps of paper. It is all there, right before your eyes.

Lotus FastSite — web design software - .htm files

Lotus ScreenCam — recording of screen activity for demos and tutorials - .scm, .exe, wav files

Caution
Do not use the 64-bit version of Office SharePoint Server 2007 to crawl Lotus Notes because the Lotus C++ API Toolkit is available only in 32-bit.

1.8 Star Office

Star Office is a full featured office suite that you can use to create text documents and Web pages, spreadsheets, slide presentations, and drawings and images. Its three main distinguishing characteristics are that it handles (opens and creates) Microsoft Office formats, as well as nearly 200 other formats; its price is microscopic compared to Microsoft Office; and it runs on Linux, Windows, and Solaris as well as Microsoft Windows. And for anyone who is experienced Microsoft's licensing fees and regulations, Star Office and OpenOffice.org are a breath of fresh air on that front too.

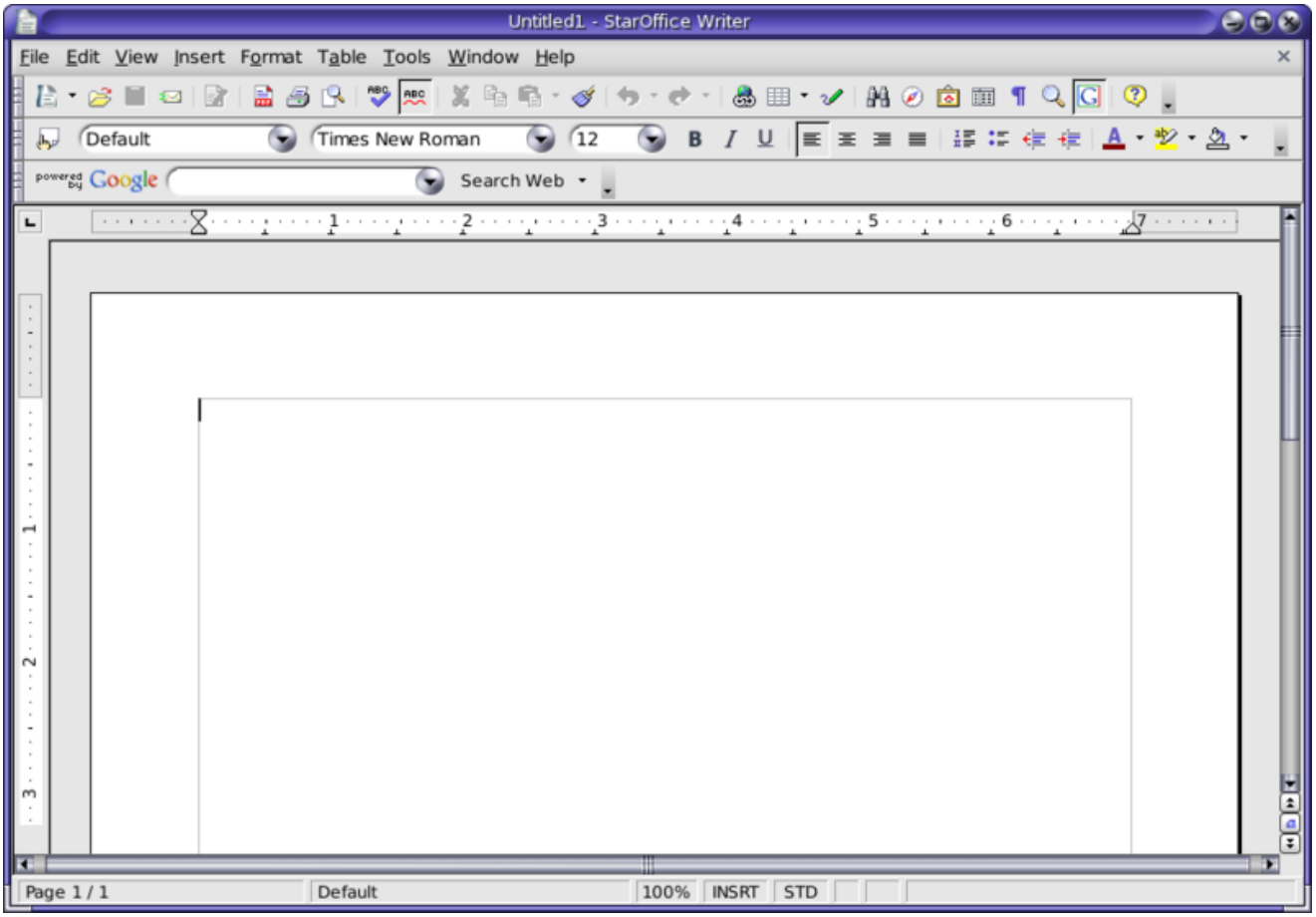


Figure 1.6: Star Office.

1.8.1 Applications and Features

There are four main applications:

- Writer/Web for producing text documents and Web pages
- Calc for spreadsheets
- Impress for slide presentations
- Draw for drawings (vector graphics) and image editing (raster graphics).

Built into some or all of the applications are the following great features and add-ons:

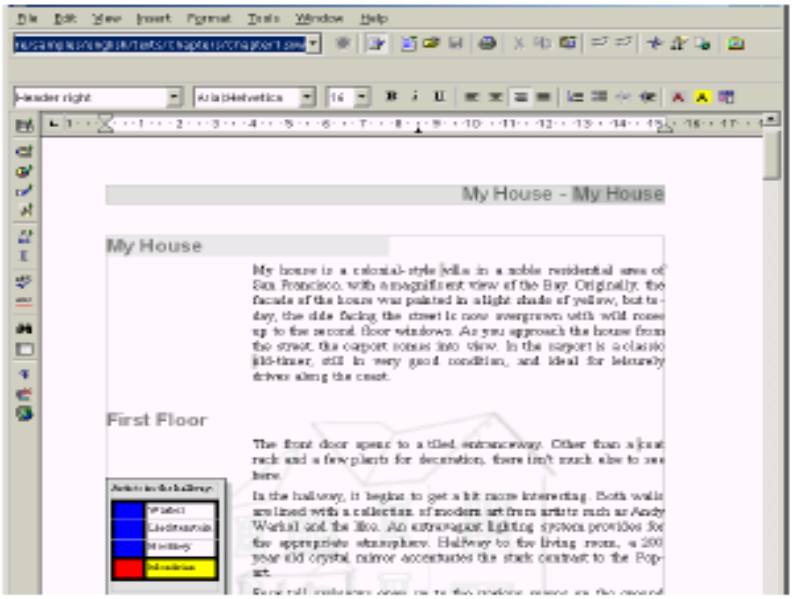


Figure 1.7: Text documents and Web pages.

- Autopilots to guide you through creating new documents and importing data.
- Charts and equations
- Data source connection capabilities for easy mail merges and access to your existing databases
- XML file formats for easy opening by other applications, plus extremely small file sizes

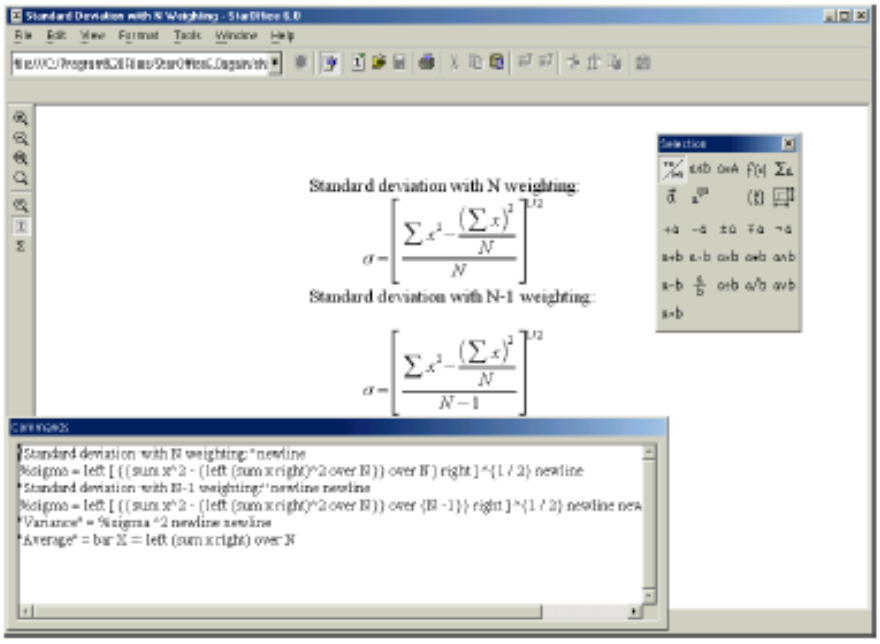


Figure 1.8: Charts and equations.

- Easy, high-quality conversion to and from Microsoft Office and other files
- HTML hotlinks from text or buttons
- A huge gallery of clip art you can use in your documents, modify, and add to
- Animation in presentations, plus animated GIFs
- Available in many languages, plus Asian language support



Figure 1.9: plus animated GIFs.

1.9 Open Office

Open Office is a volunteer-run project. Its aim is to build a world-class office suite, available to all. Everyone is free to redistribute this software thanks to its open source license.

What is "open source"?

In brief, open source software is software which meets the three criteria:

1. Unrestricted redistribution. Open source software can be redistributed either for free or at a profit.
2. Source code. The source code (i.e. "blueprints") for the software must be made available.
3. Derived works. The source code can be used to produce derived works.

For the complete details of what open source is, please refer to the open source definition as provided by the Open Source Initiative (OSI).

OpenOffice.org is reasonably intuitive but is sufficiently different to other suites that a period of familiarization required before use is fluid and effective. A general explanation of the most common control features in OpenOffice.org, such as toolbars, floating toolbars and dockable windows. This is followed by information on functions that can be used in most of the program modules of OpenOffice.org.

1.9.1 The User Interface

OpenOffice.org provides not only the usual control elements found in other applications, such as menus and icons, but a host of other special features besides. For example, many of the special OpenOffice.org features can be accessed via a context menu, opened by right clicking on the selected objects. Other user-friendly functions are available by dragging and dropping with the mouse.

1.9.2 Starting OpenOffice.org

Under Windows, once the program is installed (see the separate setup guide that is part of the download image and is on most CDs), the OpenOffice.org group should be on the Windows® Start menu under Programs. Under *NIX, its location depends upon where it is installed and which window manager is used; however, under Gnome and KDE, the OpenOffice.org group may be found in the Main menu. To create a blank text document in Writer, click Text Document. For a blank spreadsheet in Calc, click Spreadsheet, and so on.

1.9.3 The OpenOffice.org Writer Window

A text document is displayed and edited in the Writer window. Spreadsheet, presentations or drawings are displayed in very similar windows, except the menus and icons change automatically depending on the context.

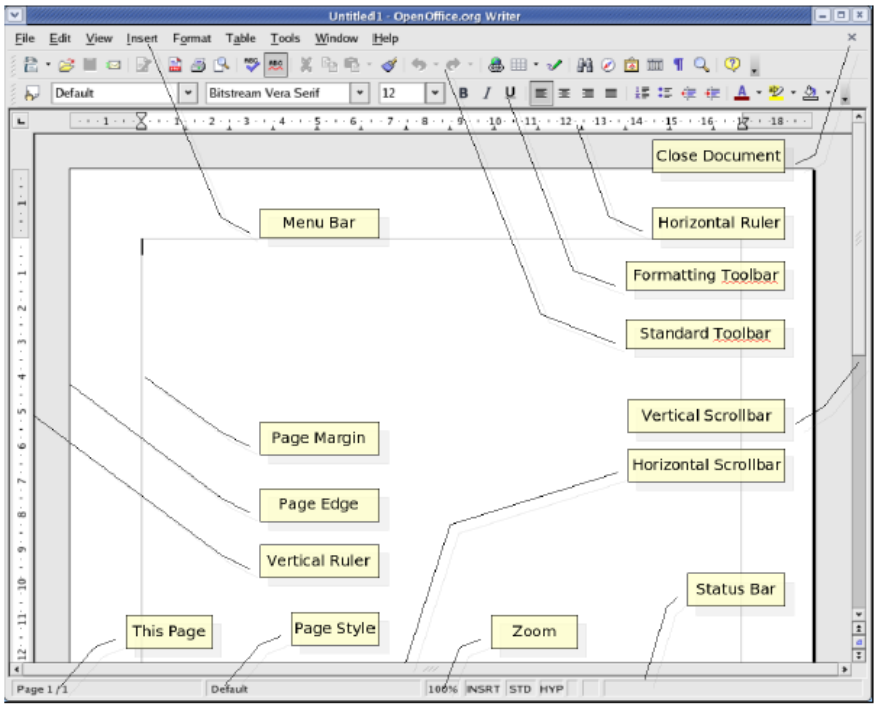
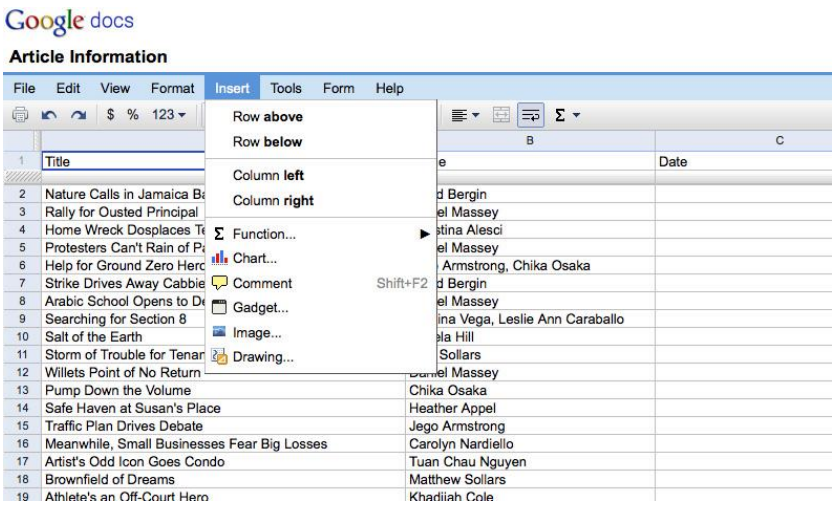


Figure 1.10: OpenOffice.org.

Exercise: Check Your Progress 3

Case Study-Google docs

Google docs offering word processing, spreadsheets, presentations, forms, and drawings, Google Docs provides an easy-to-use, integrated way for teachers and students to work together on projects, reports, and more, and to collect and share information in a secure online environment. Some of the advantages of using Google Docs include:



Anytime, anywhere access Web-based Google Docs safely stores documents online, making them accessible to authorized users from any computer or mobile device, whenever they are needed. No need to save files to a USB thumb drive, you can always access your files from any internet browser.

Collaboration support Google Docs lets users easily invite others to work on the same document, at the same time, without the hassle of attaching and sending documents. Sharing privileges ensure access by only the right people or groups, and allow either editing or read-only access.

Auto save and revision history Continuous auto save ensures that current work stays safe, preserving ongoing drafts and edits. A complete revision history makes it easy to review, compare, or revert to a prior version at any point.

Shared collections Files and docs that are regularly used by teams or groups stay organized and up-to-date without the need to manage and communicate changes.

Templates Ready-made templates covering a wide range of document and report types help jump-start writing projects. You can also create and publish your own document templates to establish assignment structures for your students. Templates can be copied with one click and then modified like any other document.

Questions

1. What was the purpose to develop the Google Docs service?
2. Is there any disadvantage using Google Docs? If yes then discuss in brief.

1.10 Summary

- Word processors make it easy to move sections of text from one place to another within a document, or between documents.
- Making presentation using slides prepared in presentation graphics software is fast becoming one of the modern ways of exchanging ideas between the speaker and his audience.
- A database is a collection of information that is organized so that it easily be accessed, managed, and updated.
- A distributed database is one that dispersed or replicated among different points in a network.
- Lotus is most commonly known for the Lotus 1-2-3 spreadsheet application, SmartSuite is an office suite from Lotus Software.
- Lotus Organizer is a personal information manager package.

1.11 Keywords

Database: A database is a collection of information that is organized so that it can easily be accessed, managed, and updated.

Macros: A macro is a character or word that represents a series of keystrokes. The keystrokes can represent text or commands.

Open Office: It is a volunteer-run project. It is use to build a world-class office suite, available to all.

Spell checker: A utility that allows you to check the spelling of words. It will highlight any words that it does not recognize.

Star Office: It is a full featured office suite that you can use to create text documents and Web pages, spreadsheets, slide presentations, and drawings and images.

Thesaurus: A built-in thesaurus that allows you to search for synonyms without leaving the word processor.

1.12 Review Questions

1. What are the office activities? Discuss their software requirements in brief.
2. What is word processing? How does it perform in windows?
3. Define spreadsheet. Why do we use it?
4. Why do we use the graphics? Give some examples of graphics software.
5. What is the use of database? How many types of databases are used?
6. Write a short note about Microsoft Office.

7. How do we work with Lotus office?
8. What is the Star office? Explain in detail.
9. Discuss the Open office.
10. What is the difference between the Star office and Open office?

Answers for Self Assessment Questions

1 (b) 2 (a) 3 (c) 4 (b) 5 (d)

1.13 Further Studies

1. *Microsoft Office Specialist Study Guide Book/CD Package by Joyce Cox, Joan Lambert*

2

MS Word Basics

STRUCTURE

2.0 Objective

2.1 Introduction

2.2 Introduction to MSWord

2.3 Features and Area of MS Word

2.4 Working with MS Word

2.5 Menus and Commands

2.6 Microsoft Office Template Wizard

2.7 Creating a New Document

2.8 Different Page Views and layouts

2.9 Working with Styles

2.10 Working with Paragraph

2.11 Working with Page Formatting

2.12 Text Editing Using Various Features

2.13 Bullets in Word 2007

2.14 Numbering in Word 2007

2.15 Auto Formatting in Word 2007

2.16 Printing and Various Print Options

2.17 Summary

2.18 Keywords

2.19 Review Questions

2.20 Further Studies

2.0 Objectives

After studying this chapter, you will be able to:

- Discuss the introduction to MSWord
- Explain the menus and commands in MSWord
- Describe about the Microsoft office template wizard
- Explain different page views and layouts
- Working with styles in MSWord

2.1 Introduction

Microsoft Office 2007 Professional Software contains five programs: ***Word*** is the word processing software that has replaced the typewriter. It is commonly used to create letters, mass mailings, resumes, newsletters and so on.

Excel is a program used to create spread sheets. Spread sheets are commonly used to create payroll, balance a check book or track an organization's finances.

PowerPoint is used to create a slideshow that helps address the topics being covered. It is commonly used to help discuss a topic or provide training.

Access is a database management program. It allows large quantity of information to be easily searched, referenced, compared, changed or otherwise manipulated without a lot of work.

Outlook is an e-mail software program that allows users to send and receive e-mail. It also allows you to keep a personal calendar and/or group schedule, personal contacts, personal tasks and has the ability to collaborate and schedule with other users.

Microsoft Works is best described as a less expensive, slimmed down version of Word/Office. It often comes with the purchase of a home computer that contains a Home Edition of Windows XP or Windows Vista. Although some commands are similar in Works and Word, they are different program. Works is not commonly used by professional organizations and there may be compatibility issues if you try to exchange documents with Word users. The same is true for other program such as Word Perfect.

2.2 Introduction to MSWord

Microsoft Word is a word processing program that allows you to create, revise, and save documents for printing and future retrieval. As you type, the words are displayed on a monitor or screen, and are stored in computer memory rather than on paper. Microsoft Word 2007 is a part of a new suite of Office programs that have an enhanced, intuitive design. This course will introduce you to the interface of Microsoft Word 2007, (see Figure 2.1), and guide you through the steps of performing familiar word-processing tasks.

2.2.1 How to Start Word

1. Click the Windows start menu button, and then select all programs.
2. Locate the Microsoft office folder. Then, click the Microsoft Office Word 2007 option.

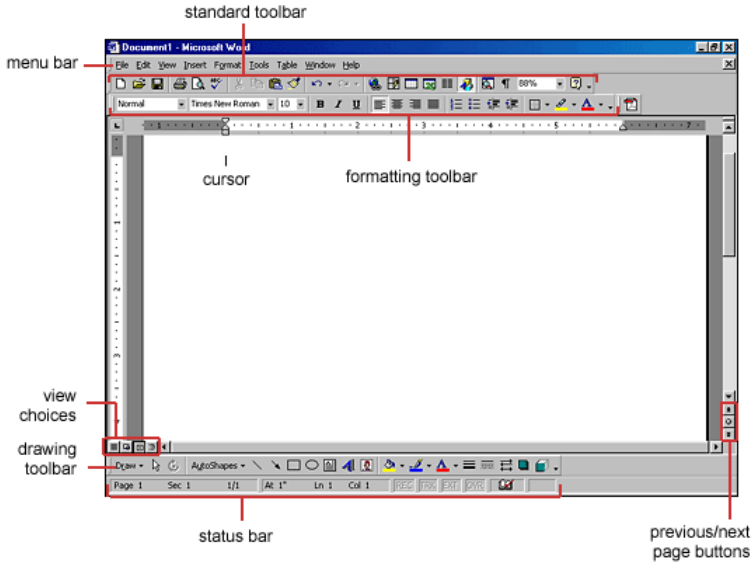


Figure 2.1: Office 2007 Environment.

Table 2.1: Microsoft Office Word 2007 elements

Ribbon	Task-related contextual tabs and menu groups running along the top of the application window.
Office Button	A button in the top left corner of the screen where high-level file controls can be found, such as SAVE, PRINT, and CLOSE.
Quick Access Toolbar	A toolbar located to the right of the Microsoft Office Button, which contains a customized set of command buttons, like SAVE, UNDO, or REDO.
Title Bar	A bar located across the top of the window, which displays the name of the current document and the current application.
Minimize Button	A button located to the left of the Maximize/Restore button in the top right corner of the window, which reduces the window to an icon at the bottom of the screen.
Maximize/ Restore Button	A button located to the left of the Close button in the top right corner of the window, which controls the size of the application and document windows.
Close Button	A button located to the right of the Maximize/Restore button in the top right corner of the window, which displays an X. It is used to close the current window.
Command Tabs	Eight default Command Tabs are located at the top of the Ribbon. They group task-related commands together. Command tab titles include: HOME, INSERT, PAGE LAYOUT, REFERENCES, MAILINGS, REVIEW, VIEW and ADD-INS.
Contextual Tools	These tools appear as additional tabs, beside the Command Tabs, when a particular object is selected. The tab will offer property change options related to the object selected and the tab will disappear when the object is no longer selected.
Command Groups	Commands available for the selected tab that relate to what you are trying to do.
Zoom Controls	A slide scale located in the bottom right corner of the window, which controls the magnification of the document. Slide the marker closer to the plus (+) or minus (-) to increase or decrease the appearance of the document.
Document Layout Controls	A toolbar located in the lower right corner of the window to the left of the Zoom controls, which provides different ways to view your document.
Status Bars	A toolbar located in the bottom left corner of the window, which displays page and word counts.
Scroll Bar(s)	Scroll bars are located on either the far right or bottom of the window, and allow you to navigate up or down and left or right to view different portions of the document.

Did You Know?

The first version of Microsoft Word was developed by Charles Simonyi and Richard Brodie, former Xerox programmers hired by Bill Gates and Paul Allen in 1981.

2.3 Features and Area of MS Word

Word 2007 is full of new tools and options, expanded capabilities, and significant changes. Find out what to focus on and what you can leave for another day (or month, or year…).

2.3.1 The Font dialog box

The new Themes feature that you can skip working with the individual effects that go into controlling the format of your document. Themes include predesigned settings for colors, fonts, and effects, and things like sidebars and quotes have their own styles as well.

Just choose the Page Layout tab, click the Themes arrow, and select the theme you want from the gallery, shown in Figure 2.2. The design will automatically be applied. If you want to change the theme later, simply return to the Themes gallery and make another choice. (If you want more granular control, you can still display

the Font dialog box by clicking the dialog launcher in the lower-right corner of the Font group in the Home tab.)

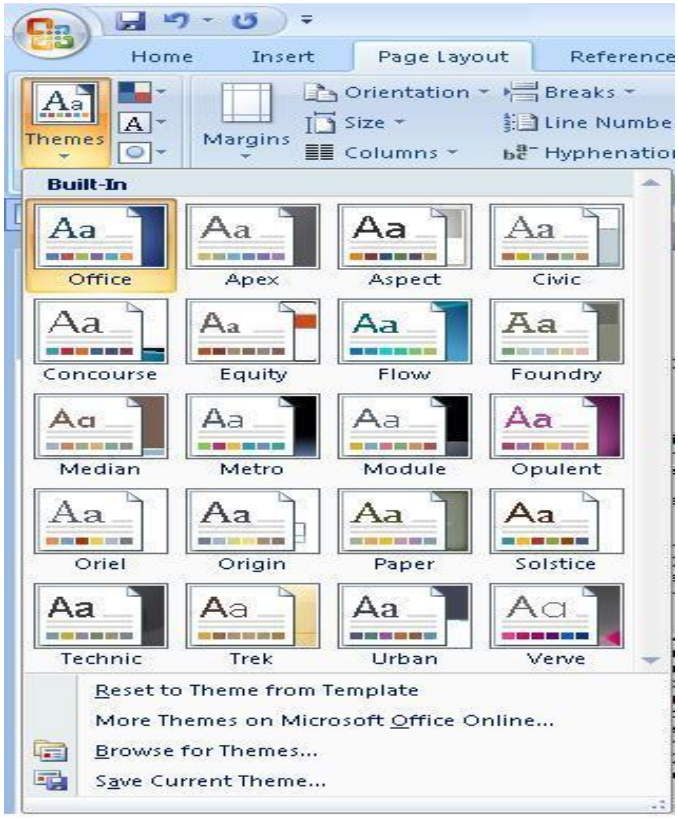


Figure 2.2: The Page Layout tab.

2.3.2 Mark as Final

The Mark as Final feature in Word 2007 (available when you point to Prepare after clicking the Office button) has potential, but in this version of Word, it does not pack a lot of punch. Even though you can mark your current document as final, when someone else receives the document, he or she can simply make a copy and then edit it as usual. If you want to ensure that nobody is going to change your content or alter your formatting, save the document as a PDF or XPS file instead.

2.3.3 Save as Word XML

XML in Word 2003 was kind of a patch job. As XML became more widely used, developers knew they needed to include a way for Word users to save files in XML format and use XML data in their Word documents. The attempt to add XML to Word 2003 created huge, clunky files that you had to save intentionally in XML format by using the Save as XML command. Although you can still, if you choose, save a file in that old Word XML format, trust me you would not want to do that. Today is Word is completely built on Office Open XML, so now all the files you create and save in Word 2007 are actually saved in XML. What does this mean to you? The short answer is that resulting files are much smaller and cleaner. The longer answer includes better security and reliability and an easier exchange with other applications (including nearly seamless integration with SharePoint sites). The only downside to the new format is that it is incompatible with pre-2007 versions of Word, so to send a document to a Word 2003 user, you still have to use Save As and save the file in the Word 97-2003 format (or simply send your coworker a link to the Office Compatibility Pack, available free of charge on the Microsoft Downloads site).

2.3.4 Mail Merge Wizard

If you like programs to walk you through every step of a process, you may be fond of wizards. But in Word 2007, the whole mail merge process has been given higher visibility (thanks to lots of feedback from Word users about mail merge frustrations) and assigned its own tab in the Ribbon. Now you can work through your mail merge projects, whether they are simple or complex, by following the selections on the Mailings tab. If you still want to consult the wizard for a little extra support, you will find it by clicking the Mailings tab, choosing Start Mail Merge, and selecting Step By Step Mail Merge Wizard (Figure 2.3) at the bottom of the command list.

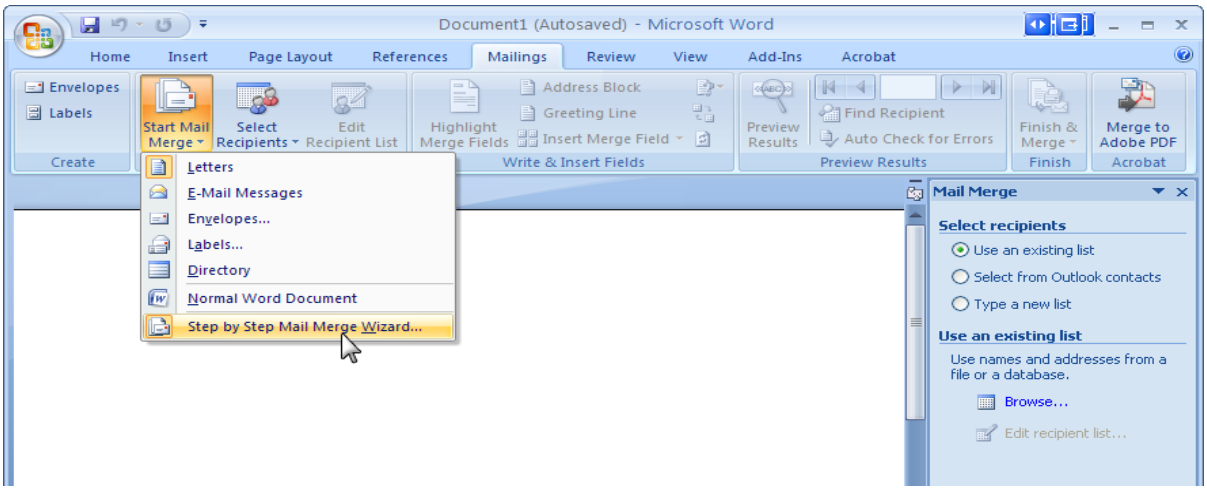


Figure 2.3: Mail Merge Wizard.

2.3.5 Drawing Canvas

The Drawing Canvas was a big deal in earlier versions of Word, but it is a bit of a dinosaur in Word 2007. In this version, it is so easy to draw, arrange, group, and work with shapes and images in your Word documents that the Drawing Canvas really is not needed to tie everything together the way it once was. Now you can assemble your shapes directly in the document, group them, and work with them as a single object. The Drawing Canvas has an annoying way of bumping everything in your document to another place on the screen, and for most custom drawings we do in Word, somebody prefers to draw directly on the page.

2.3.6 Signature Line

Tucked away in the Text group of the Insert tab in Word 2007, you will find the Signature Line command. This is another one of those features that has potential but does not carry a lot of clout. When you first click Signature Line, a popup message box appears, warning you that although you can use a digital signature to help identify the authenticity of a document, this type of signing may or may not hold up in court. The message points you to digital signature services on Windows Marketplace, where you can find digital services that pack a bit more punch.

2.3.7 Web Tools

Are you wondering where Web support went in Word 2007? Although Web Layout view still exists in Word 2007, a number of the Web tools have been removed. For example, there is no longer support for adding movies or sound to your Web page (those commands used to be on the Web Tools toolbar), and many of the browsing buttons you could use while working in Word have been removed. (If you want, however, you can

add in some navigation controls by displaying Word Options, choosing Customize, and selecting All Commands.) You can still save a document as a Web page and view it in Web Layout view so that you know what it will look like when you save it. But for more detailed Web work, you may want to check out Microsoft Expression Web, the new replacement for FrontPage. Another alternative is to use the (awesome) new blogging feature to publish your thoughts directly to a blog. You will find Word 2007's blogging capability tucked away in the Publish options. Click the Office button, choose Publish, and click Blog to get started.

2.3.8 Document Map

This one is a bit of a red herring. It would not suggest that you skip the Document Map if it is a feature you use often, because it is actually one of useful way to navigate (and review) long documents in Word. But if you like quick and easy methods for reviewing your long documents, you will love the new Thumbnails view (Figure 2.4). Now you can look over the section headings in Document Map and then switch to Thumbnails view (click the View tab and click Thumbnails in the Show/Hide group) to get a miniature view of each page in layout form. This enables you to see how your document flows, check the placement of tables and pictures, and just generally get a feel for whether your document is ready to roll.

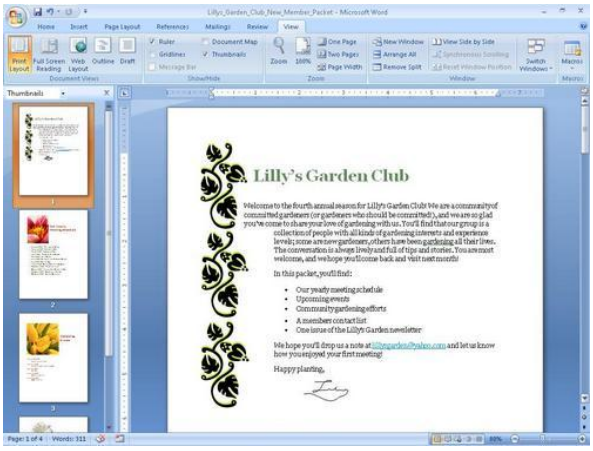


Figure 2.4: Document Map.

2.3.9 Microsoft Clip Organizer

This may be personal opinion shining through, but it is hard to imagine that the Microsoft Clip Organizer really makes anybody's life easier. The process for searching for and adding pictures and clipart is seamless in Word 2007, and the increased prevalence of DSL and TL connections makes grabbing images online easier than ever. Unless your job (or your hobby) requires you to gather huge collections of images that you are sure you will use again later, why take the time to set up, sort through, and manage all those image files in Microsoft Clip Organizer? (If you are a media file maven, consider switching to Windows Vista the media file management features alone make it worth the stress of upgrading.)

2.3.10 Document Properties

By clicking the Office button, choosing Prepare, and clicking Properties in Word 2007, you can display the Document Properties area at the top of your document and enter information about the file such as your name, the title, the status, and a brief description. That is fine if you have a lot of time to kill and do not expect other people to actually read the document. But if you simply want to make sure the author's name appears on the doc and that it is tagged appropriately so you can find it again later in a search, you can do that quickly and easily when you save the file. To add the author's name and tags, simply click in the respective boxes in the

Save As dialog box, shown in Figure 2.5, name the doc, and save it by clicking Save. Easier and faster, with fewer keystrokes.

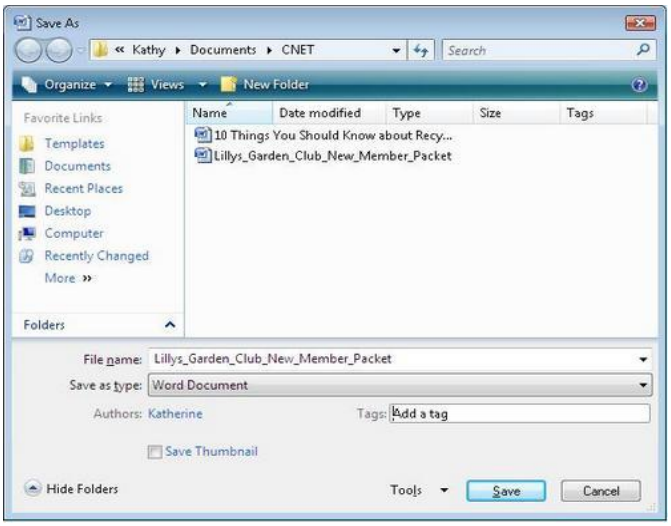


Figure 2.5: Save As dialog box.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: With the help of document map prepare list of headings in MS Word.

……..……

……..……

……...

2.4 Working with MS Word

Microsoft Word is word processing software. It is used to create and edit texts, letters, reports, and graphics. The aim of this section is to provide you with step-by-step instruction on how to use Microsoft word 2007. Click on Microsoft word icon on your desktop, to launch the Microsoft word window that looks like this (see Figure 2.6):

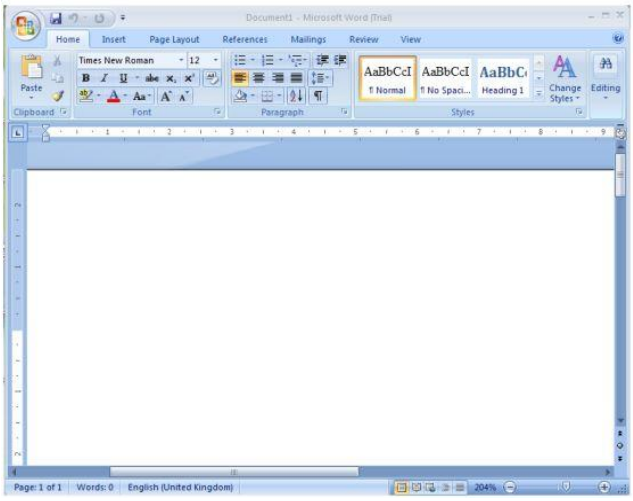


Figure 2.6: Microsoft word window.

2.4.1 The Microsoft Office Button

At the upper-left of the Word 2007 window is the Microsoft Office button. Click on the button, and a drop down menu appears.



With this menu, you can create a new document or file, open an existing document or file, save a file, and perform many other tasks like printing etc.

2.4.2 The Ribbon

The Ribbon (see Figure 2.7) is located at the top of the Microsoft word window. It is more or less the control Panel; this is where you use to issue commands. It provides numerous tabs; clicking on a tab displays drop down menus where you find a number of command groups.

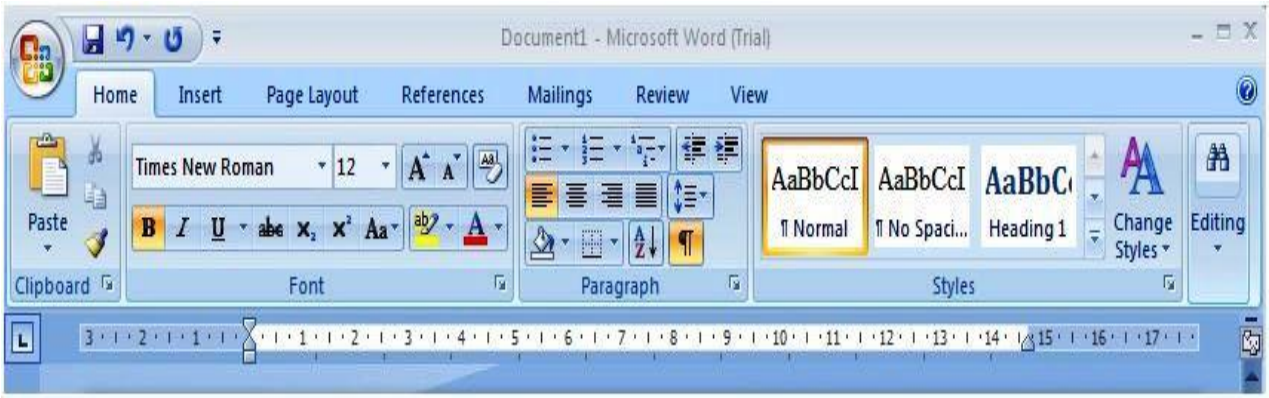


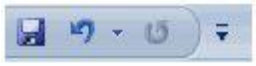
Figure 2.7: Ribbon.

Did You Know?

The RibbonXML is validated against the schema definition for ribbon extensibility. When you begin to create your own RibbonXML script with the Custom UI Editor it is a good practice to always validate your script against the schema definition before closing the editor. The schema definition provides the rules for creating valid hierarchical instructions for customizing the ribbon.

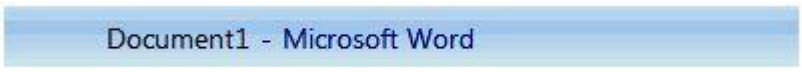
2.4.3 The Quick Access Toolbar

Next to the Microsoft Office button is the Quick Access toolbar. This enables you to perform your frequently use commands like to Save, Undo, and Redo.



2.4.4 The Title Bar

Next to the Quick Access toolbar is the Title bar. This is where the title of the current document you are working on appears. Initially, Word names the first document you open Document1. When the document is saved the Document 1 changes to the name you assigned to it.



2.4.5 Below the Ribbon is the ruler

You can use the ruler to change the size or format of your document.



If the ruler (see Figure 2.8) is not visible, it can be located and inserted by given these steps:



Figure 2.8: The ruler.

2.4.6 Font bar

This is where you manage and format your fonts. You can do the next see Figure 3.9:

- Change the name of font
- Change font size
- Change font color
- Bold text and numbers
- Change text to italics
- Underline texts etc



Figure 2.9: Font bar.

2.4.7 The paragraph bar

This part of the ribbon is where you format or apply changes to paragraphs in your documents. Here you can see Figure 2.10:

- Align text to left
- Align texts to right
- Centre texts
- Justify texts
- Apply bullets and numbers
- Indent your paragraphs etc

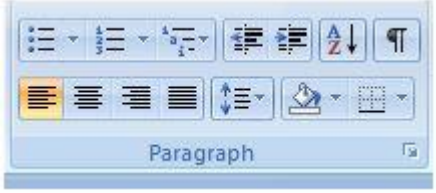


Figure 2.10: The paragraph bar.

2.4.8 The Text Area

The text area (see Figure 2.11) is basically where you type in your texts (letters and numbers). It is the open area with white background (depending on your chosen color) the blinking vertical line in the upper-left corner of the text area is the cursor. It marks the insertion point. As you type, your text displays at the cursor location.

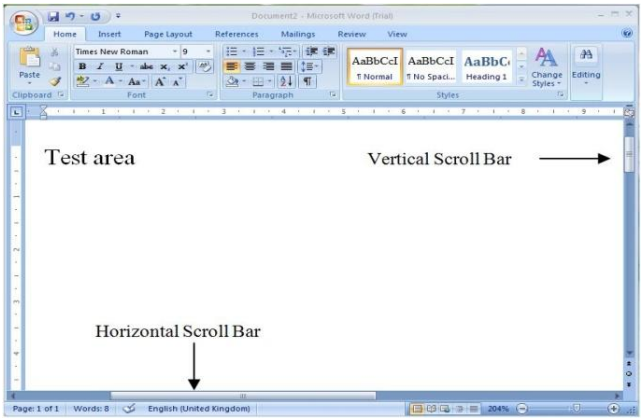


Figure 2.11: The text area.

2.4.9 The Vertical and Horizontal Scroll Bars

On the right side just below your Microsoft window is the vertical and horizontal scroll bars. They enable you to move up, down, and across your window, to do this, you have to click on the arrows those points up or down, left or right at the end of each bar. The vertical scroll bar is located along the right side of the screen. The horizontal scroll bar is located just below your document. The horizontal scroll bar is only visible when your document is larger than your screen.

2.5 Menus and Commands

2.5.1 Menus

If you are familiar with previous versions of Word you will notice a significant change in the menu structure. The menus in Word 2000 display only the commands you have recently used (collapsed form). To view all options (expanded form) in each menu, you must click the double arrows at the bottom of the menu. The images in Figure 2.12 show the Format menu in collapsed form (Figure 2.12a) and in expanded form (Figure 2.12b).

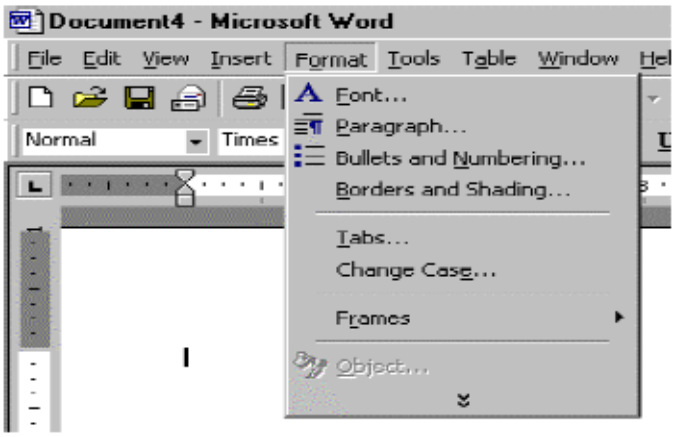


Figure 2.12a: Collapsed Format menu

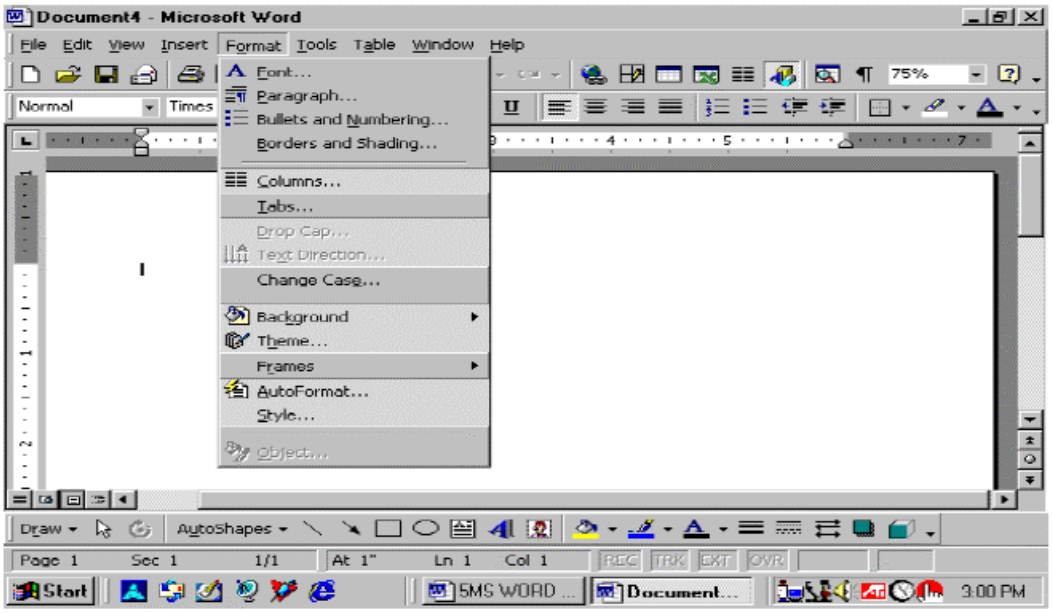


Figure 2.12b: Expanded Format menu

We can also display menus similar to previous versions of Word (like MS Word 97) with all the choices listed initially:

1. Select View Toolbars Customize commands on the menu bar from the menu bar. The Customize dialog (see Figure 2.13) box will appear.
2. Click on the Options tab.
3. Uncheck the Menus show recently used commands first check box.
4. Click on Close to close the Customize dialog box.

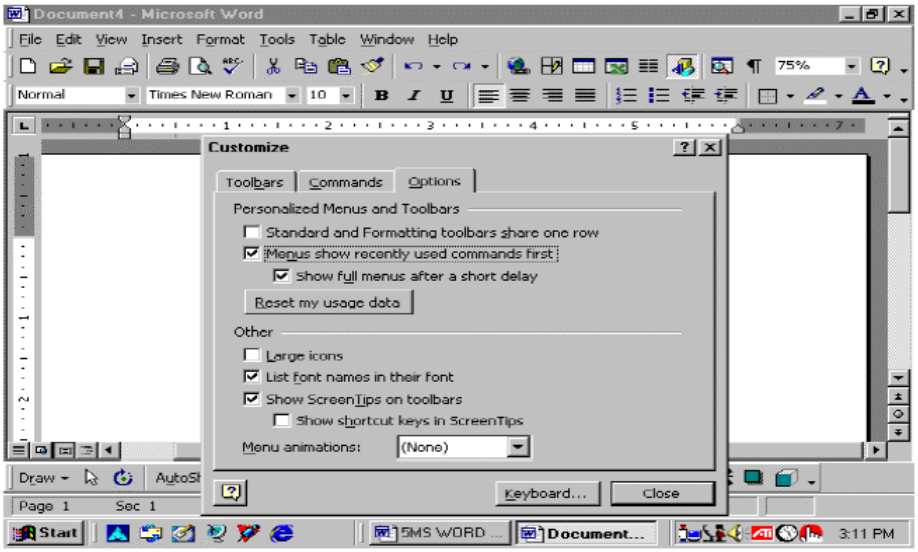


Figure 2.13: Customized dialogue box.

Shortcut Menus

These features allow us to access various Word commands faster than using the options on the menu bar. View shortcut menus by right-clicking the mouse. The options on this menu will vary depending on the element that was right-clicked. For example, the shortcut menu below is produced by right-clicking on a bulleted list.

Actions such as “Decrease Indent” and “Increase Indent” are only applicable to lists and therefore only appear on the list shortcut menu. The shortcut menus are helpful because they display only those options that can be applied to the item that was right-clicked and, therefore, prevent searching through the many menu options see Figure 2.14.

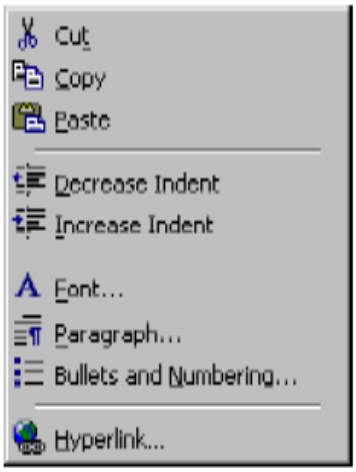


Figure 2.14: Shortcut menus.

Toolbars

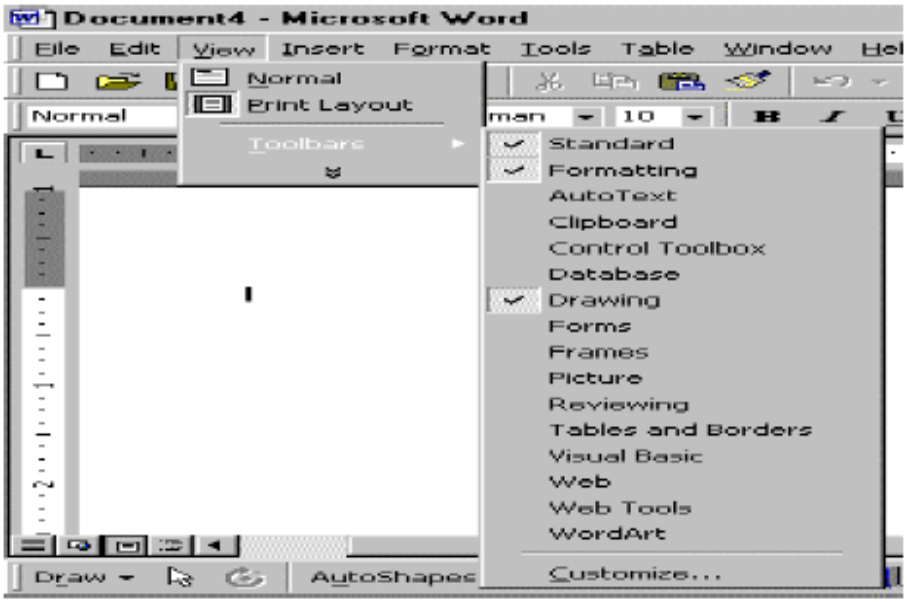


Figure 2.15: The toolbars.

Many toolbars displaying shortcut buttons are also available to make editing and formatting quicker and easier see Figure 2.15. Select View Toolbars command on the menu bar from the menu bar to select the toolbars. The toolbars that are already displayed on the screen are checked. Add a toolbar by simply clicking on the name.

Rulers

The rulers display horizontal and vertical scales that reflect the width and height of your typing area. The horizontal scale is invaluable when you want to quickly set tabs, margins, and indents. If you do not see the

rulers select View Ruler. If you are in normal view, you will see only horizontal ruler. To see both the rulers you should be in Page Layout view. If you do not want to see the ruler selects View Ruler to turn off the ruler.

2.5.2 Word 2007 Commands

Since Word 2007 not only introduces a new user interface, but also a variety of new capabilities, we have listed some of the new Natural Language Commands specific to Word 2007 below.

Common Tasks:

- Go to/hide/open/display/switch to/show thumbnails
- Turn/set side by side view off/on
- Add/create/insert [a] special character
- Add/create/insert [a] hyperlink/chart

Sample Page Layout menu commands:

- View themes; set colors of theme; change the font of [that/the/this] theme
- Create/add line numbering continuously; erase/delete/remove line numbering
- Create/add/insert a watermark
- Set the page color

Sample Reference menu commands:

- Previous footnote/endnote
- Create/add/insert [a] citation/caption
- Create/add/insert index
- Add [this/it/that] in the index

Sample Review menu commands:

- Show/view/open/hide/close [the] research [task] pane
- Translate the first sentence/word; translate the entire page
- Create/insert/add [a] new comment
- Erase/delete/clear/remove all comments shown
- [Move to/go to] [the] next/previous comment
- Hide/close [the] review pane
- Accept/reject [the/this/that] change and go/move to next

Sample Office Button menu commands:

- Open/show document information/document information [panel]
- Quick print [file/document]
- Save [document] as PDF
- Encrypt file
- Add digital signature
- Mark [the] file/document as final
- View/open/show [the] document management [task] pane
- Share [the/that/this] document/file

Sample Mailings menu commands:

- Create blank/new envelope/label
- How/view/hide merge preview

Sample Blog Post menu commands:

- Create a new blog account
- Open a blog post
- Publish [this/that] post to [the] blog account

2.6 Microsoft Office Template Wizard

In your Microsoft Word documents and PowerPoint presentations it is desirable too, to use the Coprorate Identity requirements. To make this easier for your employees, Milestre has developed the so called Microsoft Office Template Wizard. Your Corporate Identity is defined in a set of Microsoft Word templates, Microsoft PowerPoint presentations and macro's. Those macros are responsible for filling in the correct information in the correct position of the template or presentation. In the first instance the employee is able to define employee and department dependent information in a user profile. After the definition of one or more profiles (you can define a profile for each role you fulfill in the organization) the generation of the document or presentation can be started.

2.6.1 Using the Office Template Wizard

The wizard can be used in Microsoft Office 2003 as well as Microsoft Office 2007. When you have started Word or PowerPoint, a new Office Template Wizard (OTW) icon appears in the toolbar see Figure 2.16.



Figure 2.16: Office Template Wizard icon.

When you start OTW for the first time, you have to define at least one profile in the user profile window. This is the information that will be used during the generation of the Word document or PowerPoint presentation when the OTW has been finished successfully.

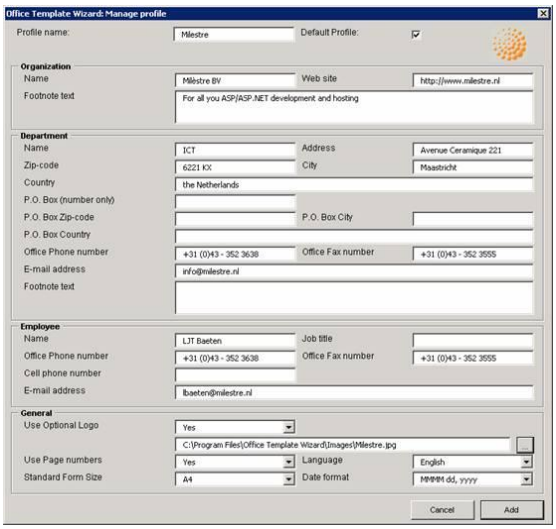


Figure 2.17: Managing profile in Office Template Wizard.

After the definition and selection of the user profile see Figure 2.17, a Word template or PowerPoint presentation must be selected. Dependent on the application that is used, Microsoft Word or PowerPoint, one will see Word templates or PowerPoint presentations (see Figure 2.18).

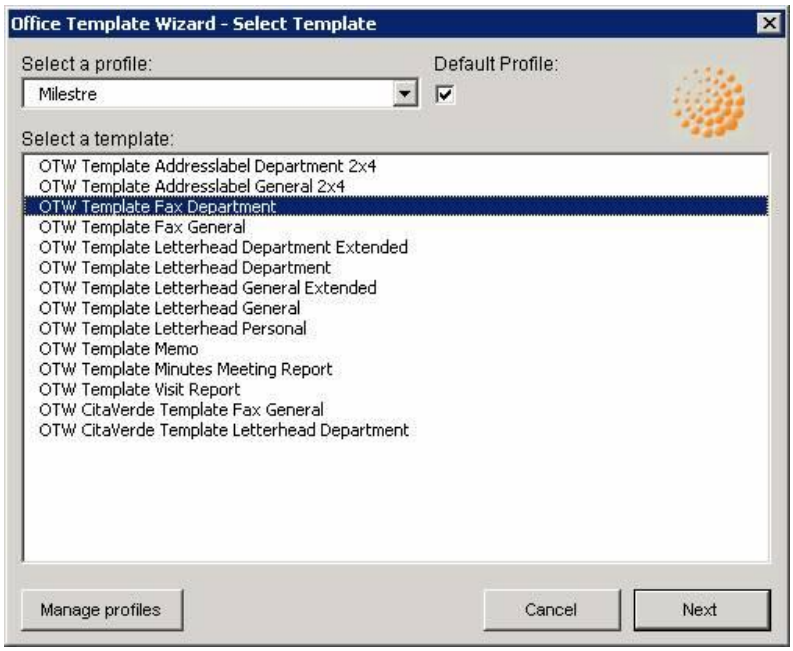


Figure 2.18: Selection in Office Template Wizard.

After selection of the template or presentation you are still able to make temporary changes to the profile definitions see Figure 2.19.

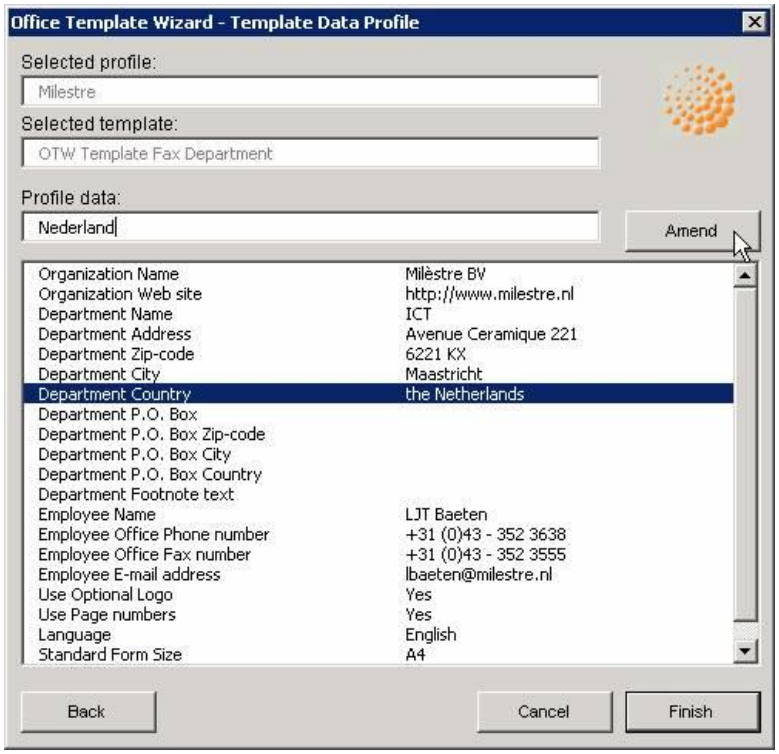


Figure 2.19: Template data profile.

After these changes the wizard is closed and the Word document or PowerPoint presentation is generated automatically conform the Corporate Identity (CI) requirements see Figure 2.20.

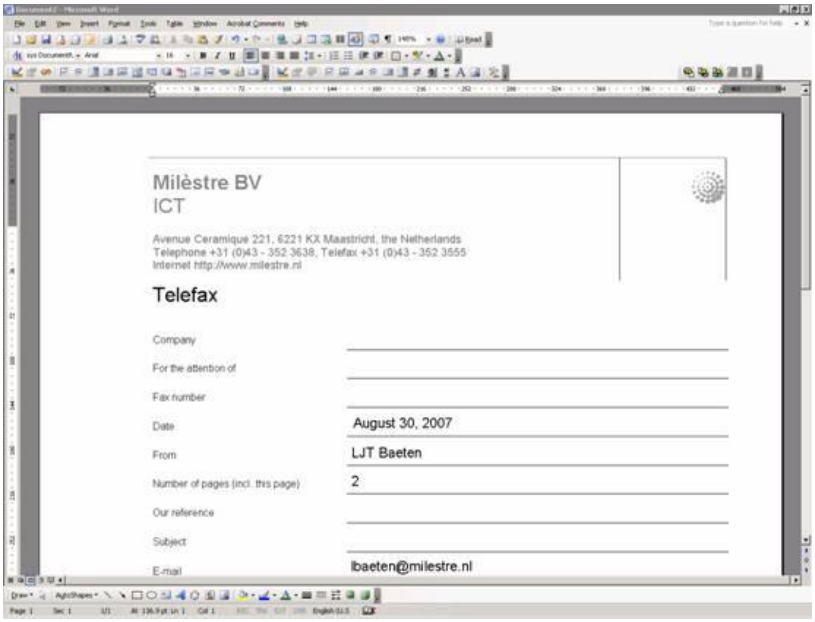


Figure 2.20: Profile wizard.

You are also able to select the languages in which the document or presentation must be defined. Dependent on employee's role, the language can be selected and defined in different user profiles. In the Figure 2.21 the same Word template is chosen but now defined in the Dutch language.

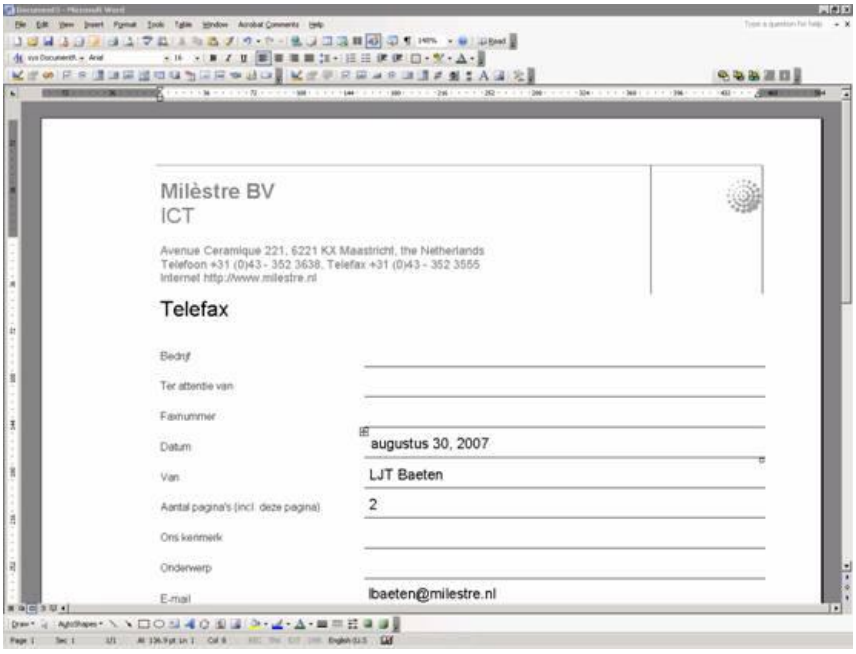


Figure 2.21: Word template in the Dutch language.

You can choose between English, German, Spanish, French, Italian, Dutch, Portugese, Japanese and Chinese so that your Corporate Identity is warranted on international level also. See Figure 2.22 for general options in templates.

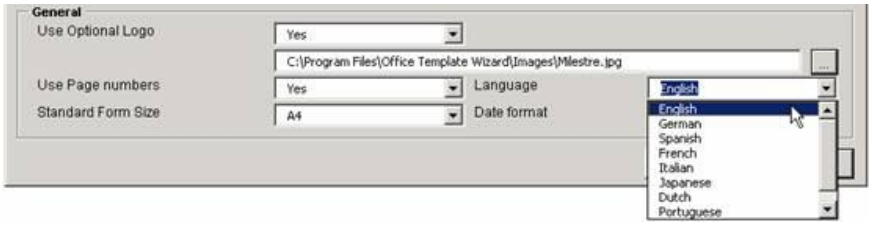


Figure 2.22: General options in templates.

The language wherein the OTW is presented is dependent on the language of your Microsoft Operating System: if you have a Dutch operating system, then OTW will be presented in Dutch; do you have a French operating system, then OTW will be presented in French.

Caution

Be careful to restore the document to normal mode after adding merge fields, if you forgot to restore the document to normal node, the Document Server can stop responding when correspondence is generated.

2.7 Creating a New Document

2.7.1 Keyboard Option

1. Press [Ctrl] + [N]

Notes: A new blank document appears. This can be done under any command tab.

2.7.2 Menu Option

1. From the OFFICE BUTTON,  select New
   The *New Document* dialog box appears (see Figure 2.23).

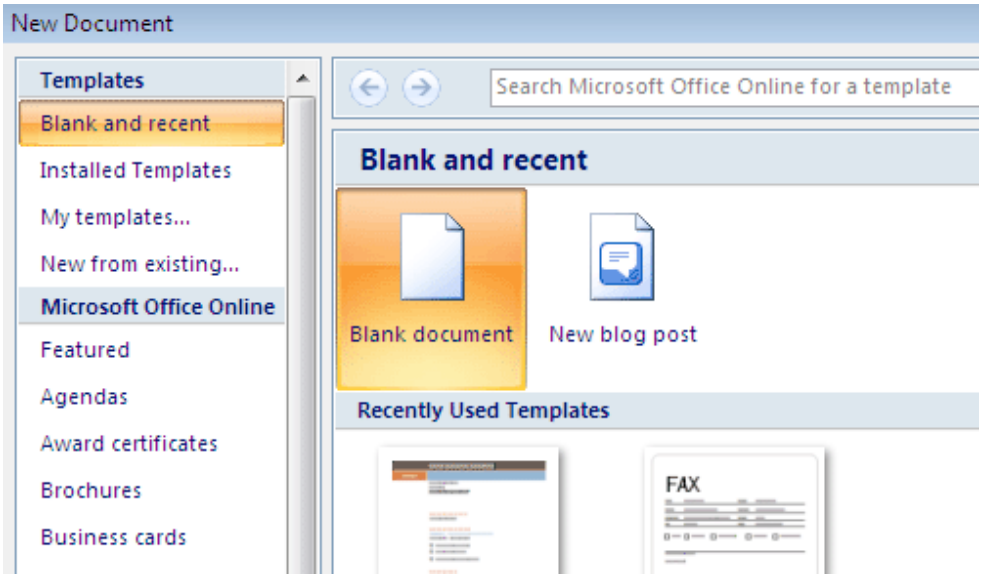


Figure 2.23: New Document dialog box.

2. From the *Blank and recent* section, click BLANK DOCUMENT

A new blank document appears.

2.8 Different Page Views and layouts

The view tab in Word 2007 is where you can change the way your document looks on your computer screen. You are able to see grid lines, zoom in and out, or look at pages in a book view see Figure 2.24.



Figure 2.24: The view tab.

When you click on the Print Layout button in the Document Views section this will change the view of the document you are working on to look just like the document will print. The next button, Full Screen Reading, changes the view of the document to a larger view that takes up most of the screen and removes the buttons at the top to maximize the view for easy reading and editing. If you choose this view click the close button at the top right corner to return to the normal view. The Web Layout button will change the view of the document to appear as it would if the pages were turned into a web page. The outline button will show your document as an outline then give you another tab with more outlining tools. The last button, Draft, will give you a chance to view your document as a draft for quick editing. This view removes elements of the document such as headers and footers for easy editing. In the Word 2007 Page Layout tabs. We will go through step by step explaining how all of the buttons in each section work. The Page layout tab is where you can change the appearance of the entire Word document. Open your greeting card we were working on then click the Page Layout tab and we will get started.



Figure 2.25: Themes menu.

The first section of the Page Layout tab is themes see Figure 2.25. Themes is a great feature if you are typing an elaborate document and want to use a variety of fonts and colors and then duplicating those fonts and colors on another document or throughout a long document. A document theme is a set of formatting choices that include a set of theme colors, a set of theme fonts that you can specify a heading and body text font, and a set of theme effects you can choose lines and fill effects. We are not going to use the themes section on our greeting card but we want you to understand what the feature does. Click the down arrow under themes see Figure 2.26.

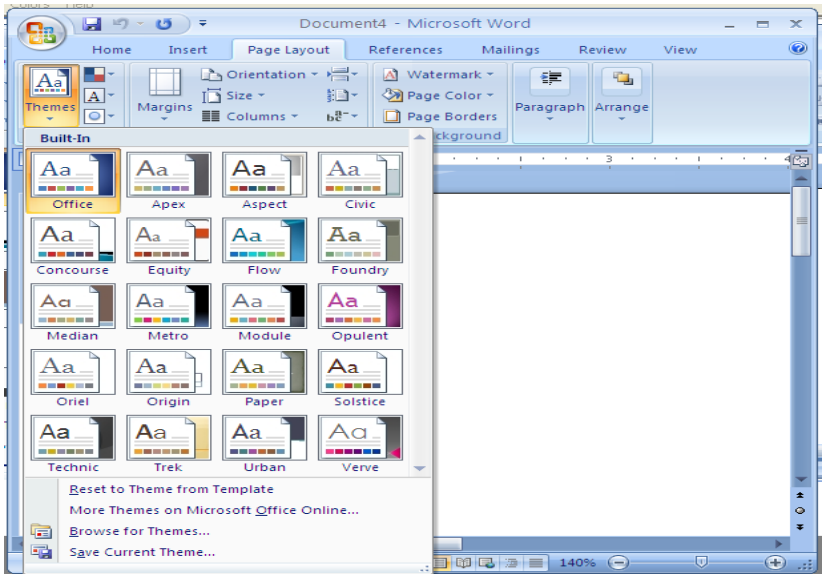


Figure 2.26: Built in themes.

You will get a list of pre-designed themes you can apply to your document. Each theme will include font colors, font styles, font sizes and effects including lines, fill effects, and colors. If you already selected a theme for your document and no longer want to use it click on the Reset to Theme from Template option. If you do not like any of the built-in themes you can click More Themes on Microsoft Office Online and there will be many more to choose from. Or you can create your own theme with the other options in the Theme section. Then click back on Themes and at the bottom click Save Current Theme then you will be prompted for a file name. Once you give your theme a name it will be available to use on other documents you create. Now we will go over how to create custom theme with the other features in this category. Now click on the down arrow next to the square made up of 4 colors to learn how to change the theme color.

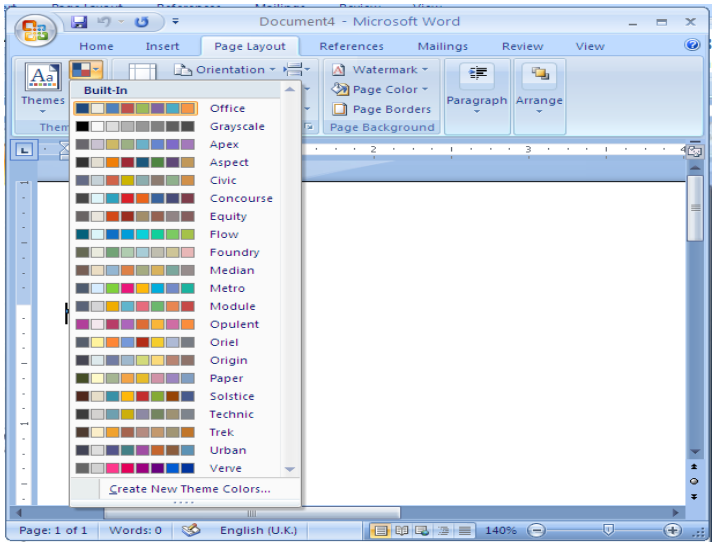


Figure 2.27: Built in theme colors.

When you click on the custom color theme drop down arrow you will get a list of Built-In Color themes for your document see Figure 2.27. These are colors are for a variety of things including heading, body, and accent colors. Now click on Create New Theme Colors see Figure 2.28.

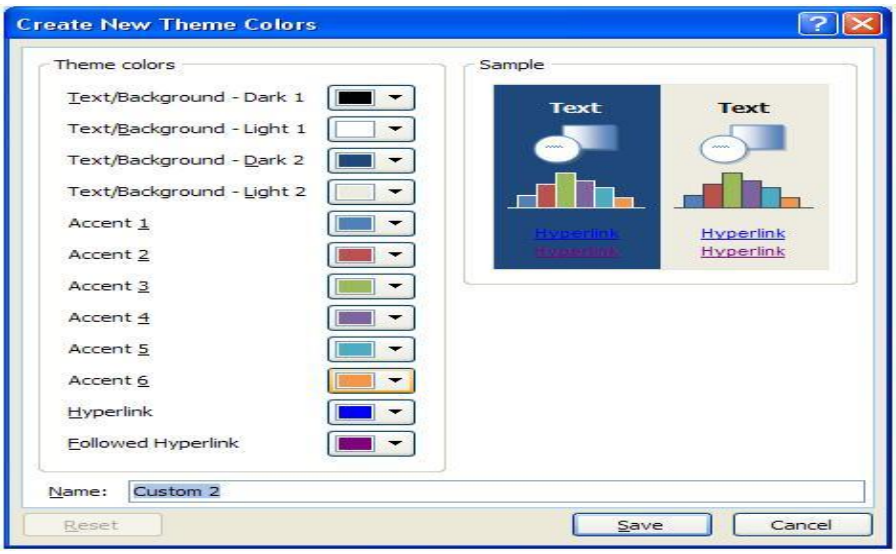


Figure 2.28: Create New Theme Colors window.

The Create New Theme Colors window will appear. Here you get a better idea of what each line of colors is going to do. Not only do you now understand what each color is for but you can modify the colors to your liking. Once you are finished modifying the colors by clicking the dropdown arrow next to the color you want to change and selecting a new color then type a theme color name in the Name section the click Save. Your new modified color will appear in the list of Built-In Color Themes see Figure 2.29.
Next is the Font theme selector. Click the dropdown arrow next to the box with an A.

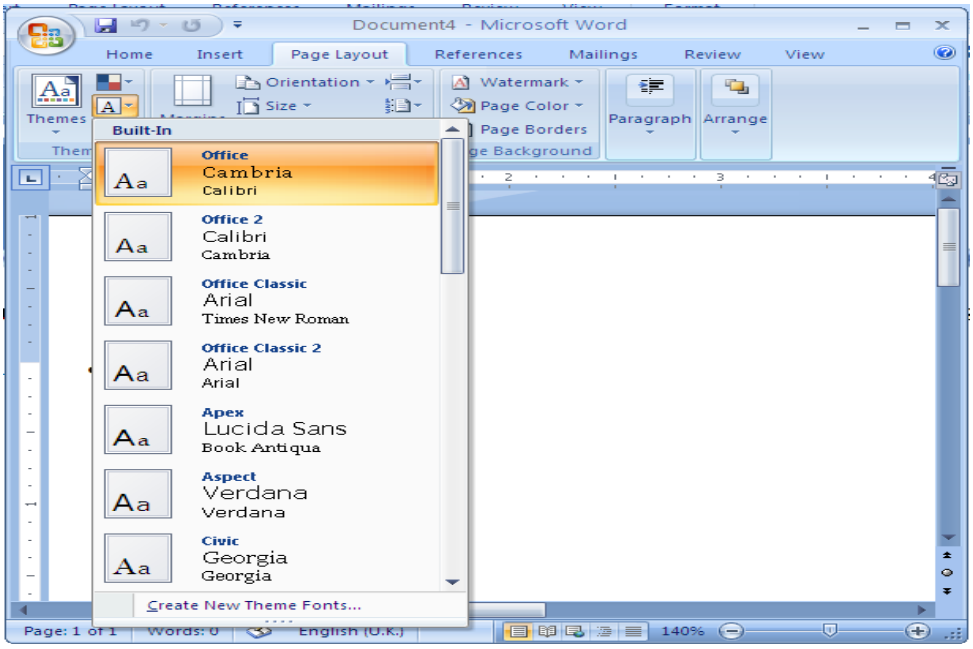


Figure 2.29: Font theme selector.

This menu works just like the colors but you are changing the fonts. It has the same Built-In selections Microsoft Word 2007 has provided for you and also the Create New Theme Fonts option. Click the Create New Theme Fonts.

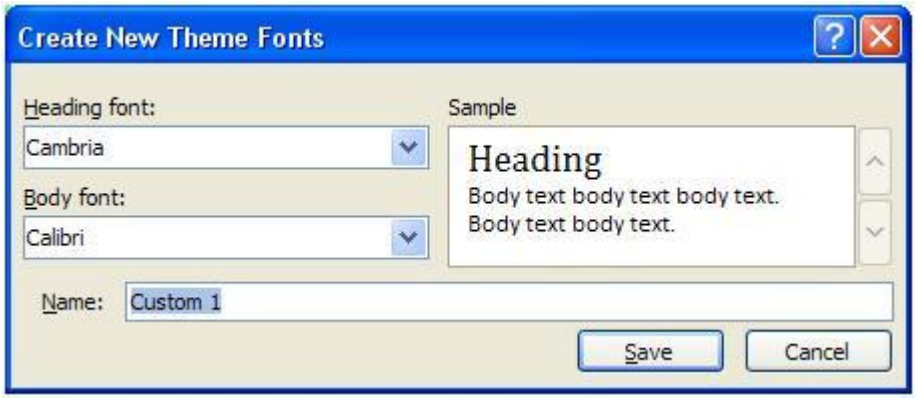


Figure 2.30: Create New Theme Fonts window.

When the Create New Theme Fonts window (see Figure 3.30) opens go ahead and play around with Heading font and the Body font by clicking the dropdown arrow next to the font names. It will show you a preview of your selections in the Sample section. If you want to save your selections type a name in the Name section and click save.

The last button in the Themes is the Effects button see Figure 2.31. Theme effects are sets of lines and fill effects used on shapes and graphics you use in your document. Click the drop down arrow to see your list of choices.

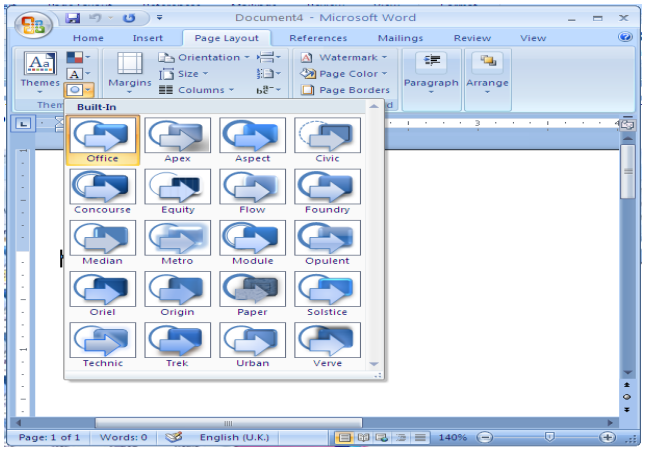


Figure 2.31: Theme Effects button.

The effects do not let you customize your own but it gives you a wide variety of choices of boarders and fills effects to choose from.

Self Assessment Questions

1. The pasteboard is similar to the work space traditionally used by graphic artists.

(a) True (b) False

2. The CorelDraw application is not a part of CorelDraw Graphics Suite 12.

(a) True (b) False

3. Refers to the solid outer rectangle in the.....

(a) Toolbox (b) Drawing page

(c) Drawing window (d) Document Navigator

4. The .....displays the most commonly used functions that are relevant to the active tool or to the task you're performing.

(a) Property bar (b) Toolbox

(c) Menu bar (d) Title bar

2.9 Working with Styles

The Word 2007 Styles section is used to quickly format an entire document. A style is a set of formatting characteristics such as font size, color, paragraph alignment, spacing, and shading.

Open one of your existing letters or document and press ctrl key and A. This will select the entire document. Or if you are just beginning a document you can make these selections before you start typing. Scroll through the default styles with the up and down arrows to the right. Click on one of the default styles and it will change your documents if you already have one typed or start typing and see how Word 2007 is automatically making the changes to the style. If you do not like the looks of it simply make sure the entire document is highlighted and clicks another style.

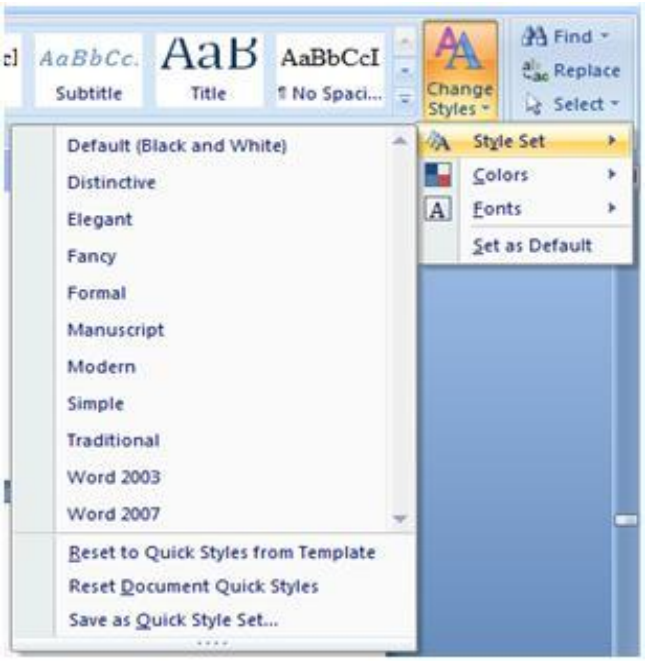


Figure 2.32: Change Styles.

The next button is Change Styles see Figure 3.32. This button gives you the opportunity to customize the style you choose. Click the down arrow to see the options. The first selection is Style Set. A style set is the combination of formatting changes you make to a document. Place your mouse over Style Set to see a list of options. These options will change the style selections you can choose from. If you have made style changes to your document you can click the selections at the bottom of the list to reset changes made to a template, reset the document to quick styles, or save your customized style as a Quick Style set.

The next selection in the Change Styles button is Colors. Place your mouse over Colors. You will see a list of preselected color combinations. These color combinations are for different text colors throughout your document. If you select the Create New Theme Color at the bottom of the menu you will see a list of the different types of text you can change the colors see Figure 2.33.

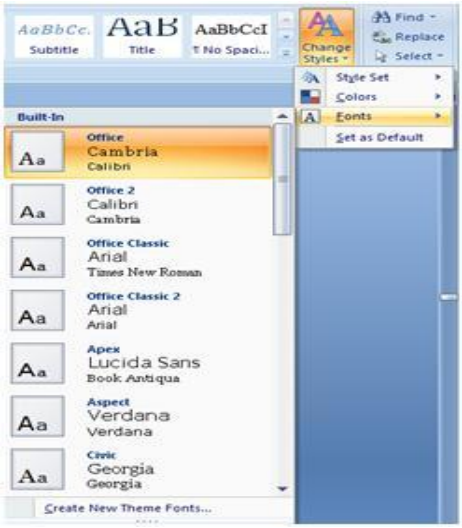


Figure 2.33: Change Styles button.

Next is the Fonts selection in the Change Styles button. Click on the Fonts selection and see a list of default combination of Fonts. The top is the Heading font and the bottom is the text for the body of the document. Use the arrow slide bar on the right hand side to scroll through the selections. At the bottom of the Fonts menu you will see Create New Theme Fonts. Click on this option. The Create new Theme Fonts window will open see Figure 2.34.

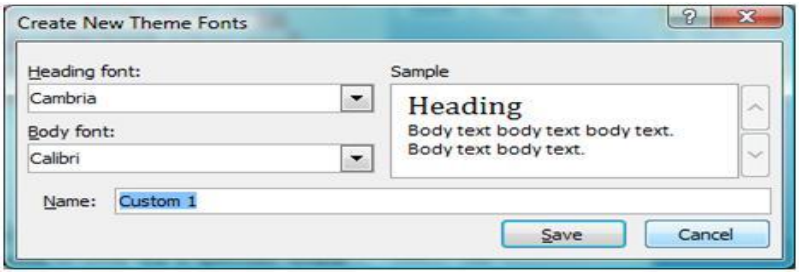


Figure 2.34: New theme font's window.

In the Create new Theme Fonts window you can customize a font theme. Simply use the down arrows to select a font for the Heading and or the Body. Name your font theme and click Save. The last option in the Change Styles button is the set as default selection. This will take the current theme of your document and set it to default so every time you start a new document the theme you have created will be used.

2.9.1 Text Enhancements

Selecting text

1. Click before letter/word/sentence
2. Hold down left mouse button
3. Drag across text (text should be highlighted)

Did You Know?

Text cannot be changed unless it has been highlighted first

Select ALL Text (shortcut – ctrl/A)

Letter Correct

1. Click after the wrong letter
2. Press backspace keyboard key to delete
3. Type new letter

Word/sentence Correct

1. Select word/sentence
2. Type new word/sentence

2.10 Working with Paragraph

The best way to describe the Paragraph section is to say it formats the location, color, or view of your text or objects on the page. The first three buttons  are list buttons. The first button is a bulleted list. Click on the dropdown menu to see you selections for you bullets see Figure 2.35.

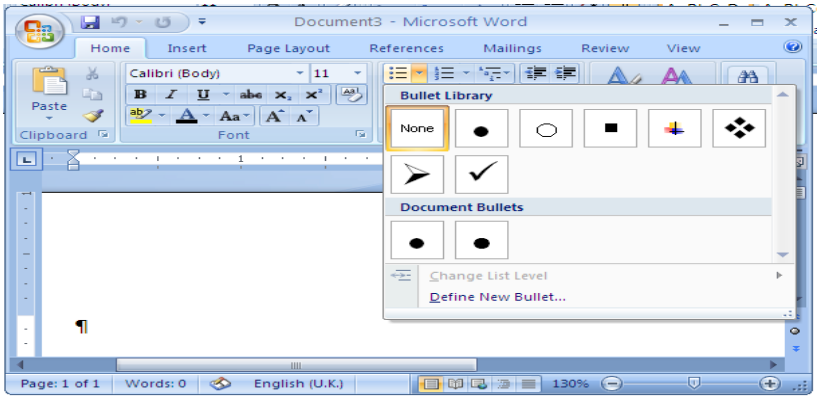


Figure 2.35: Bulleted list.

These are just a few of your options if you click on the Define New Bullet you have endless options to create your own look and feel.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Create a document and use the themes and different tools.

……..……

……..……

……...

2.11 Working with Page Formatting

Word 2007 comes with many formatting options for all types of documents. This document describes page formatting options can affect a page, a section, or your entire document.

- Adjusting Document Margins

- Numbering Pages
- Inserting Page Breaks

2.11.1 Adjusting Document Margins

The *Ruler* is used as a quick way to adjust margins. Margins may also be adjusted by using a preset option provided by Word, or through the Page Setup dialog box.

Adjusting Document Margins: Using a Margin Preset Option

1. From the *Page Layout* command tab (see Figure 2.36), in the *Page Setup* group, click **MARGINS**

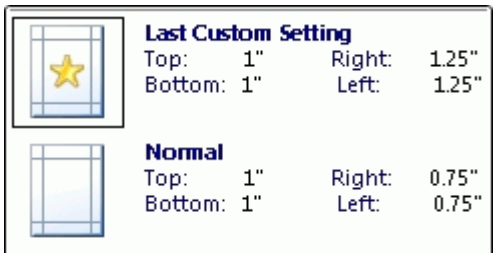


Figure 2.36: *Page Layout* command tab.

2. Select one of the preset margin options
   EXAMPLE: ***Wide*** for one-inch vertical margins and two-inch horizontal margins

Adjusting Document Margins: Dialog Box Option

To set your own margins, use the *Page Setup* dialog box.

1. From the *Page Layout* command tab, in the *Page Setup* group, click **PAGE SETUP** the *Page Setup* dialog box appears (see Figure 2.37).

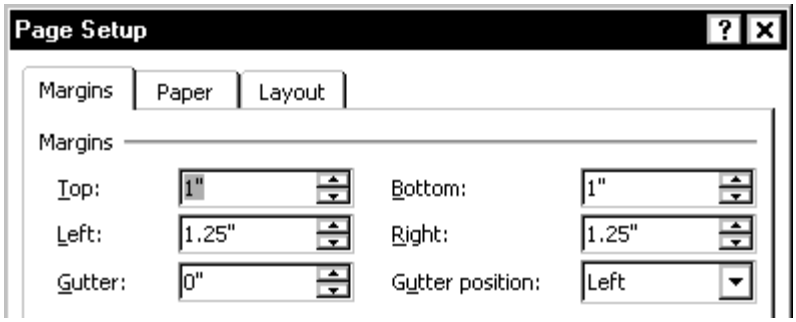


Figure 2.37: *Page setup* command tab.

2. On the *Margins* tab, in the *Margins* section, adjust the margins as needed
3. Click **OK**

2.11.2 Numbering Pages

Word lets you easily enter page numbers in your document by using the *Insert* command tab. Page numbers become part of the header or footer. For information on modifying headers and footers, or removing first page formatting, refer to Creating Headers and Footers.



1. From the *Insert* command tab, within the *Header and Footer* group, click **PAGE NUMBER**
2. Select ***Top of Page*** or ***Bottom of Page*** » select a pre-formatted page number placement
3. OPTIONAL: To specifically format the look of your page numbers, select ***Format Page Numbers***

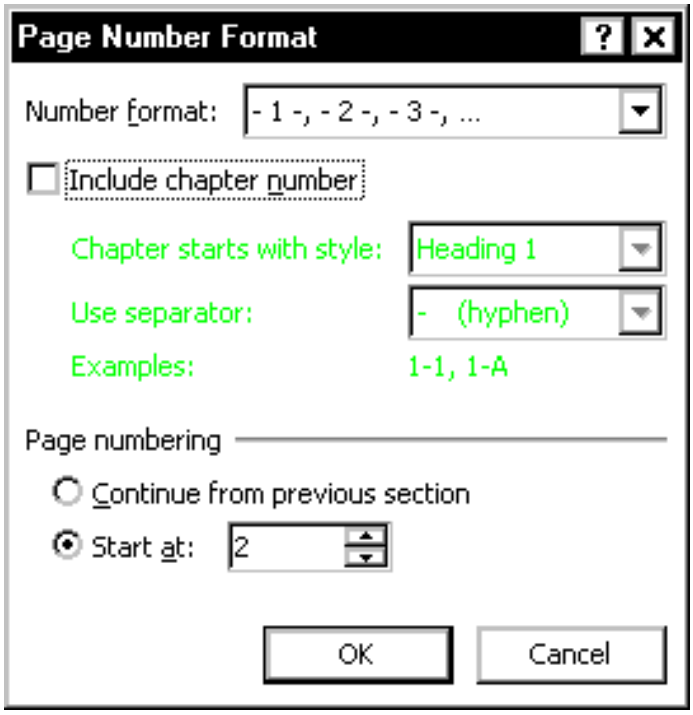


Figure 2.38: *Page Number Format* dialog box.

The *Page Number Format* dialog box appears.

a. From the *Number format* pull-down list, select the desired format

b. To continue numbering, under the *Page numbering* section, select ***Continue from previous section***

To specify the starting page number, select ***Start at*** and type the number

c. Click **OK.**

2.11.3 Inserting Page Breaks

Word has two types of page breaks. The first is a natural page break. This occurs when the information has filled an entire page and needs to flow onto the next page. The second type is forced, often referred to as a hard page break. Forced page breaks occur only when the user inserts a hard page break.

Additional pages can also be started using section breaks. For more information, refer to Working with Section Breaks.

1. Place the insertion point where the new page is to be inserted
2. From the *Insert* command tab, within the *Pages* group, click **PAGE BREAK**  **OR**

Press [**Ctrl**] + [**Enter**]

Removing Page Breaks

1. Select the page break
2. Press [**Delete**]

Preventing Extra Blank Pages

While there is no Word feature that prevents extra blank pages, you can help reduce blank pages by reviewing your document. If your document ends in a forced page break and there are blank lines after it, another page of the document will print. To prevent this, delete the blank lines between the page break and the end of the document.

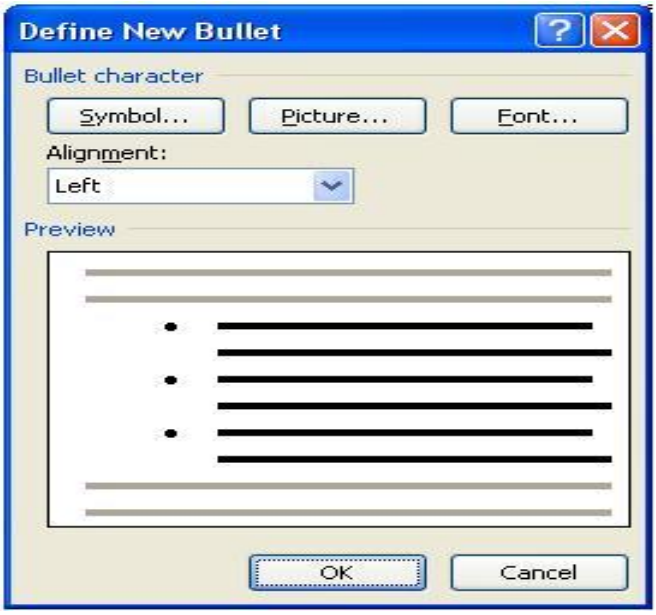
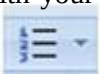


Figure 2.39: Define New Bullet window.

When you click on the Define New Bullet the window (see Figure 2.39) pops up and you can create your own bullet by using the Symbol, Picture or font button. The Alignment dropdown menu lets you choose where on the page you would like your list to appear and the Preview section will let you know what your bullet will look like before you click OK and start your bulleted list. Once you select your bullet style it will automatically place a bullet on your document. Type your text after your bullet and hit enter to create the next item in your list. If you are finished with your list hit enter twice and the bullets will be discontinued. The same rules apply with the numbered list  Click the drop down menu on the numbered list button to get your options see Figure 2.40.

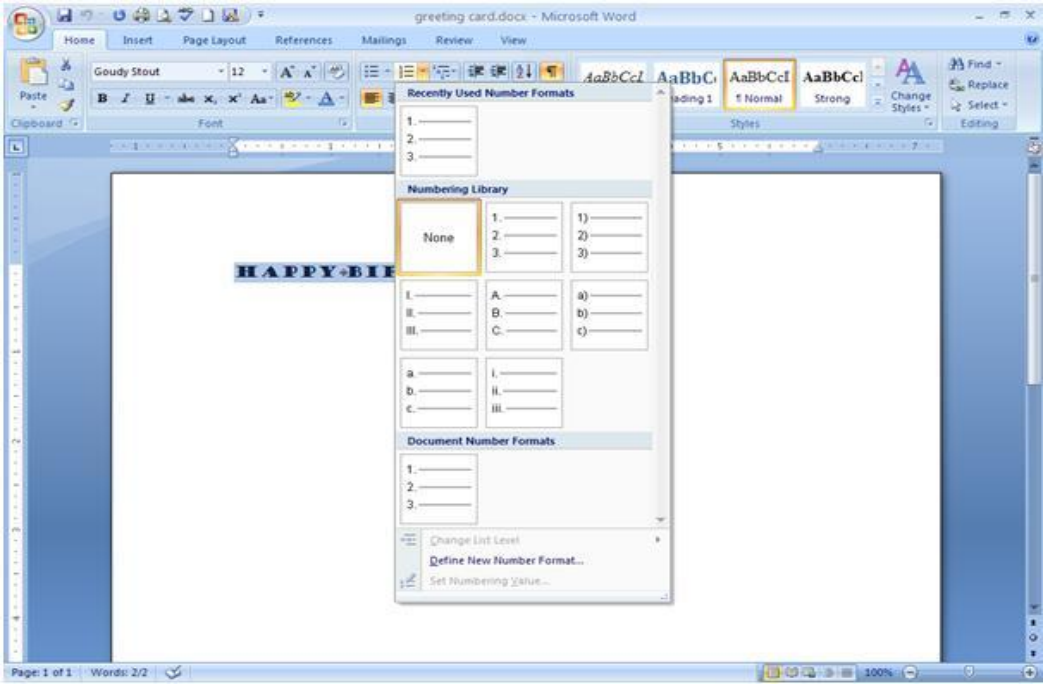


Figure 2.40: Numbered list button

You can choose numbers with a dot or a bracket, roman numerals, letters or define your own format see Figure 2.41.

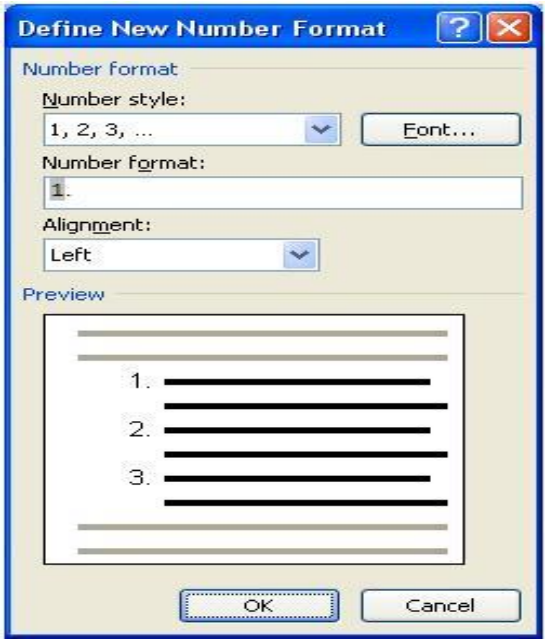


Figure 2.41: Define new number format window.

When you click Define your own format the above window appears and you can customize your own format by instead of using a predefined one. Just enter your customizations in each category, preview it in the preview pane until you are satisfied and click OK.
The numbers will work the same way as the bullets, after typing your text hit enter and the next number will appear. When your list is complete press enter twice and your numbered list will end. The last list button  is the multilevel list. This is great for outlines. It works the same as the lists above with all of the option and customizations but it has one difference; to get to the next level in the list press tab and your list item will tab over and change to the second level format.

Once you have tabbed over to the second, third, etc level when you hit enter it will stay at that level until you hold down the Shift key and press tab. This keyboard command will bring your list up a level. To discontinue the list simply hit enter twice and you can continue your document without continuing the list. The next two buttons  Increase and Decrease an Indent at the beginning of a paragraph. Notice the hourglass at the top left of your word document in the ruler see Figure 2.42.



Figure 2.42: Word document in the ruler.

Now click on the Increase Indent button  did you notice how the hourglass moved to the right? This is setting a tab. Each time you finish a paragraph and press tab on your keyboard the tab will move to the location of the hourglass. By clicking the Increase Indent button more than once the hourglass moves further to the right. The Decrease Indent button will move the tab back.

The  button will alphabetize a list of words or sort numbers for you. To use this features simply select the text you would like to organize by clicking in front of the text you want to highlight and dragging the mouse to the end of the text then click the  button It will alphabetize the list of words for you.

The next button we use all of the time to help me with my formatting. The  Button will give you formatting symbols on your document. It allows you to see if you have an extra space between words by the dots or an extra line break by the paragraph symbol.

These 4 buttons  you will use to justify your text on the page. Either clicks the button before you start typing or highlight text that has already been typed and click the justification you want.

The  button is your line spacing. To use this feature you can either click on the drop down menu and select your options or highlight your text then click and select. Most users will only need to select one of the default options listed.

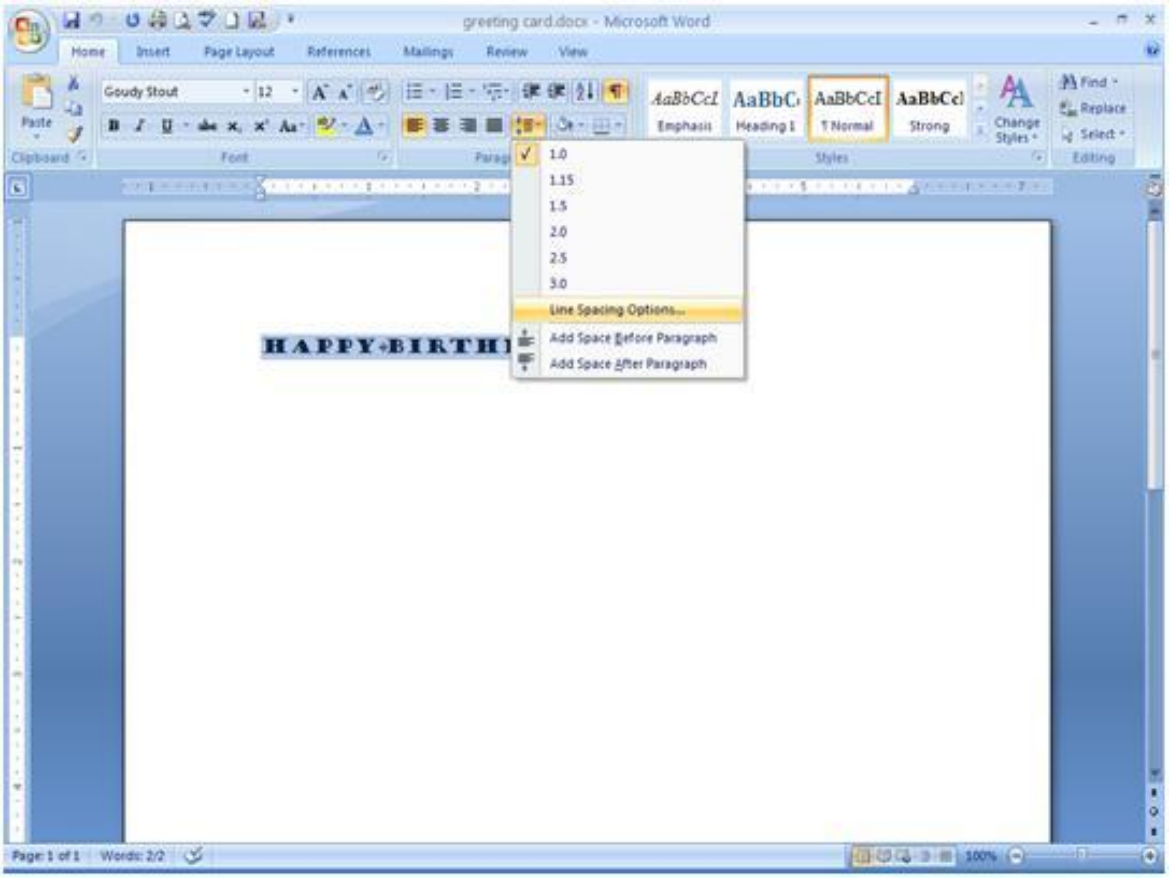


Figure 2.43: Line spacing window.

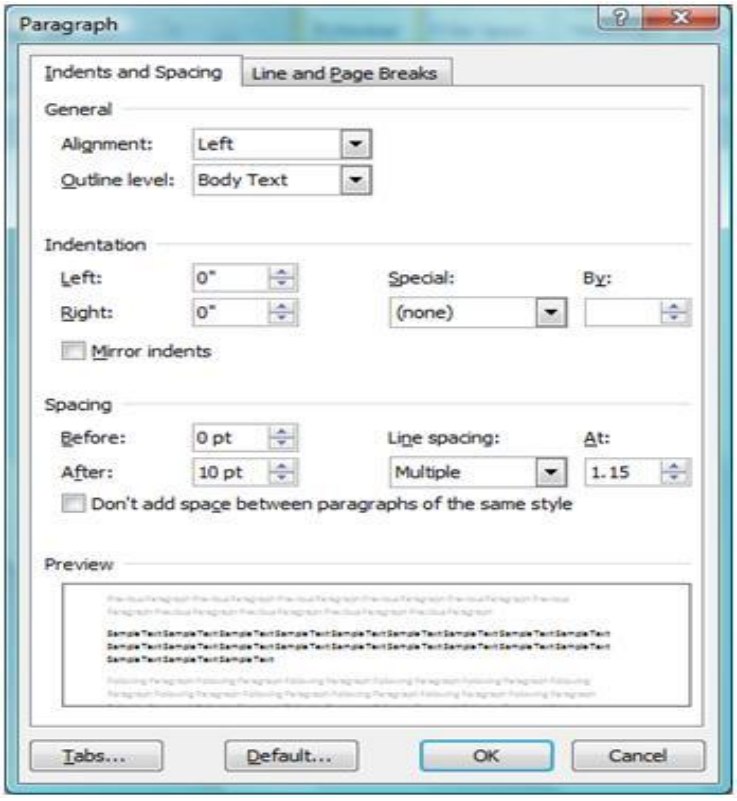
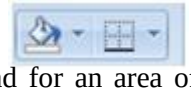


Figure 2.44: Paragraph window.

If you are interested in other options besides the default line spacing selections click on Line Spacing Options see Figure 2.43. This will open the Paragraph window see Figure 2.44. In this window you can do the same things we have been using the buttons for.

Under Line spacing you have a drop down menu where you can select from single, double, 1.5 lines, At Least, Exactly, or Multiple. The At selection is where you can input your custom settings.

The Preview section at the bottom will let you see what the spacing will look like in your document. Click the OK button to return to your document and make the changes.

The fill button and Boarders button  can be used on individual lines of text. If you would like to change the color of the background for an area of text in your document click the button before you start typing and choose a color from the drop down menu. When you are finished typing click the fill button again and the background will go back to the original color see Figure 2.45.

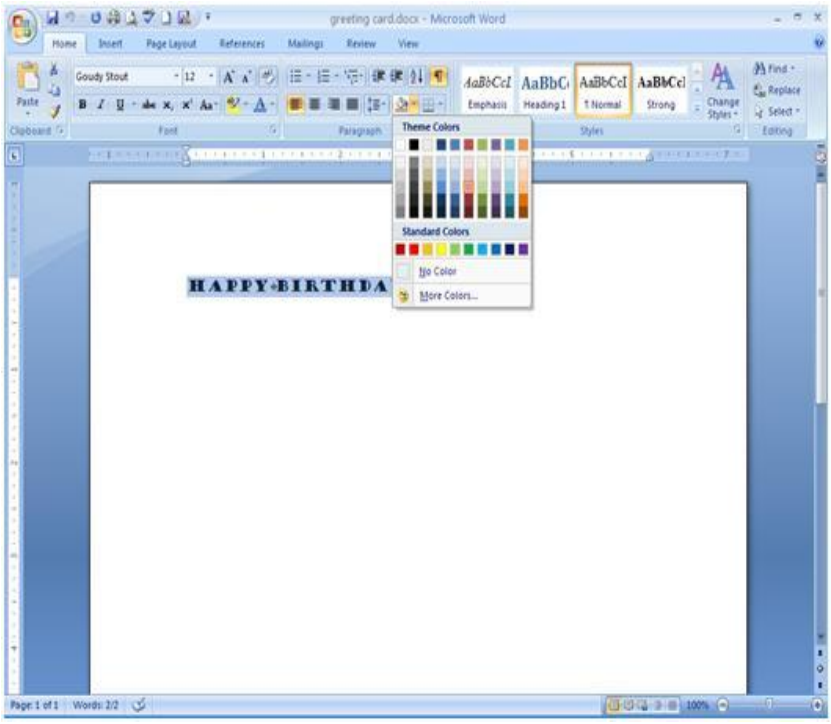


Figure 2.45: Text background color.

Highlight the text you would like a different background and select a color from the dropdown menu and the background of selected text will change.

The Boarders Button works the same way as the background fill but will place a boarder around the text. Click the dropdown menu to see your entire options in Figure 2.46.

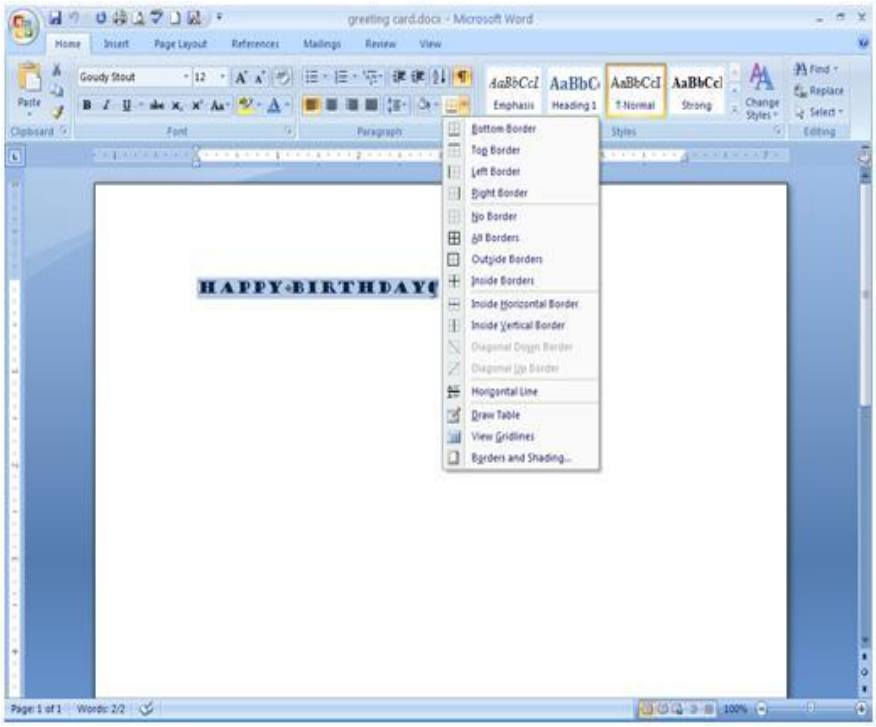


Figure 2.46: Boarders button option.

2.12 Text Editing Using Various Features

A blank document starts out with a flashing insertion point, which looks like a small vertical bar. In addition, if you are working in Draft or Outline view, a horizontal bar (not flashing) appears as an end-of-file marker (see Figure 2.47). Initially the two markers are together because there is nothing in the file, but the end-of-file marker moves further down on the page as you add more text to your document.

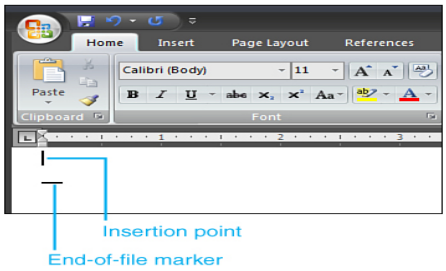


Figure 2.47: The insertion point is a flashing vertical line; the end-of-file marker (appearing only in Draft or Outline view) is a horizontal, non-flashing line.

Text you type always appears at the insertion point. To enter text, just type as you would in any program. The given keys have specific functions:

- **Enter:** Press this key to start a new paragraph.
- **Shift+Enter:** Press this key combination to start a new line within the same paragraph.
- **Ctrl+Enter:** Press this key combination to start a new page.
- **Tab:** Press this key to move to the next tab stop (by default every 0.5").
- **Backspace:** Press this key to delete a single character to the left of the insertion point.
- **Delete:** Press this key to delete a single character to the right of the insertion point.

You can also delete a text selection of any size, including text and/or other objects, by pressing the Delete or Backspace key.

Line Breaks versus Paragraph Breaks: A surprising number of people have trouble understanding the difference between a new paragraph and a new line. Yes, starting a new paragraph does also start a new line, so on the surface they seem to be doing the same thing. But if you turn on the Show/Hide ¶ feature (on the Home tab), you will see that two completely different symbols are inserted.

A paragraph break (¶ symbol) creates a whole new paragraph, which can have its own indentation, bullets and numbering, line spacing, and other paragraph-level settings.

A line break (↵ symbol) is like any other character of text within the paragraph, except instead of printing a letter on the screen, it moves the insertion point to the next line. The text after the line break has the exact same paragraph-level formatting as the text before the break, because it is all one paragraph.

Line breaks come in handy whenever you do not want the stylistic attributes of multiple paragraphs. For example, suppose you want to create a bulleted list of mailing addresses, with each complete address as a separate bullet point. If you press Enter between the lines of each address, each line will have its own bullet character, like this:

- John Smith
- 240 W. Main Street
- Macon, IL 62544

By using line breaks instead, you can create a single bulleted item with multiple lines, like this:

- John Smith
  240 W. Main Street
  Macon, IL 62544

2.12.1 Switching Between Insert and Overtype Modes

When editing text, Insert mode is on by default, meaning that any text you type to the left of existing text will cause the existing text to scoot over to the right to make room for it. The alternative, Overtype mode, types over any existing text to the right of the insertion point.

To toggle between Insert and Overtype mode, follow these steps:

1. Choose Office, Word Options.
2. Click Advanced.
3. Under Editing Options, mark or clear the Use Overtype Mode check box.
4. Click OK.

If you find yourself frequently switching between Insert and Overtype, you might want to set up an easier method for performing the switch. There are two such methods available: remapping the Insert key, and adding an Insert/Overtype indicator to the status bar.

By default, the Insert key works as a shortcut for the Paste command on the Home tab. If you prefer, you can change its mapping so that it instead switches between Insert and Overtype modes.

To make the Insert key toggle between Insert and Overtype views, follow these steps:

1. Choose Office, Word Options.
2. Click Advanced.
3. Under Editing Options, mark the Use the Insert Key to Control Overtype Mode check box.
4. Click OK.

Now the Insert key functions as a toggle between Insert and Overtype modes. To make it more obvious which mode you are in, you might want to turn on the Insert/Overtype mode indicator on the status bar.

To add the indicator to the status bar:

1. Right-click the status bar.
2. Click to place a check mark next to Overtype.

Insert (or *Overtype*) appears in the status bar. You can then click that word to toggle between them.

Caution

Be careful, even if the active document does not have Track Changes enabled, you still cannot access that check box if *any* open document is tracking changes.

2.12.2 Undoing, Redoing, and Repeating

Whenever you make a mistake, such as accidentally deleting or overwriting something, you can easily reverse it with Word's Undo feature. To undo, press Ctrl+Z, or click the Undo button on the Quick Access toolbar. The Undo feature retains a list of actions you have recently taken, and you can undo any number of them. The effect is cumulative. In other words, you can undo, for example, the last five actions you took, but you cannot pick-and-choose among those five; you must undo the intervening four in order to undo the fifth one. To undo multiple levels, repeat Ctrl+Z or repeatedly click the Undo button on the Quick Access toolbar, or click the down arrow to the right of the Undo button to open a menu and then select the actions to undo from that list. After you have undone one or more actions, the Redo button becomes available on the Quick Access toolbar.

It reverses undo operations, and comes in handy when you accidentally undo too much. Ctrl+Y is its keyboard shortcut. Figure 2.48 shows the Undo and Redo buttons.

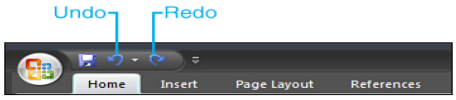


Figure 2.48: The Undo button undoes the last action when clicked; it also has a drop-down list from which you can choose to undo multiple actions at once.

The Repeat feature enables you to repeat an operation such as typing, formatting, inserting, and so on. The Repeat button looks like a U-turn arrow, and appears in place of the Redo button on the Quick Access toolbar, when available. Its shortcut is also Ctrl+Y; this works because Repeat and Redo are not available at the same time (see Figure 2.49).

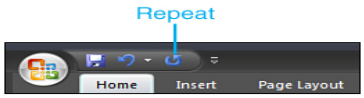


Figure 2.49: The Repeat button makes it easy to repeat the last action you took.

2.12.3 Inserting Symbols and Special Characters

The computer keyboard is very limited in the characters it can produce, and people often need other characters to produce typeset-quality documents. For example, the copyright (©) and trademark (™) symbols are frequently used in business documents, and an attractively typeset page uses em dashes (—) rather than two hyphens together (--) to represent dashes in sentences.

Inserting Symbols with Keyboard Shortcuts or AutoCorrect

Some of the most popular symbols have keyboard shortcuts and/or AutoCorrect shortcuts. AutoCorrect is a feature used most often for correcting common spelling errors, but it is also useful for generating certain common symbols on the fly. To use an AutoCorrect shortcut, type the text shown and press the spacebar once, and Word converts the shortcut text to the specified symbol. Table 3.2 summarizes both the keyboard shortcuts and the AutoCorrect entries for some common symbols.

Table 2.2: Keyboard and AutoCorrect Shortcuts for Symbols

Symbol	Keyboard Shortcut	AutoCorrect Shortcut
—(em dash)	Ctrl+Alt+Num – (minus sign on the numeric keypad)	
–(en dash)	Ctrl+Num – (minus sign on the numeric keypad)	
©(copyright)	Ctrl+Alt+C	(c)
®(registered trademark)	Ctrl+Alt+R	(r)
™(trademark)	Ctrl+Alt+T	(tm)
…(ellipsis)	Ctrl+Alt+. (period)	...
‘(single opening quote)	Ctrl+‘,‘ Hold down Ctrl and press the grave accent key (‘) twice. It is above the Tab key.	
’ (single closing quote)	Ctrl+',' Hold down Ctrl and press the apostrophe	

Symbol	Keyboard Shortcut	AutoCorrect Shortcut
	key twice. It is to the left of the Enter key.	
“(double opening quote)	Ctrl+‘," Hold down Ctrl and press the grave accent key (‘) once, and then type a quotation mark.	
”(double closing quote)	Ctrl+'," Hold down Ctrl and press the apostrophe key once, and then type a quotation mark.	
←(typographical left arrow)	None	<—
→ (typographical right arrow	None	—>
⬅(thick typographical left arrow)	None	<==
➔(thick typographical right arrow)	None	==>
↔ (double-headed arrow)	None	<=>

The single and double quotation marks in Table 2.2 are typographical—that is, they differ depending on whether they are at the beginning or end of the quoted phrase. This is different from the straight quotation marks and apostrophes that you can directly type from the keyboard.

There are no AutoCorrect entries for the dashes and the quotation marks. That is because they're not needed. Word automatically converts straight quotes to typographical ones (Word calls these "smart quotes") and two hyphens in a row to a dash. If you do not want that change to occur, using Undo (Ctrl+Z) immediately after Word makes the change to reverse it. Undo also reverses any of the AutoCorrect conversions as well if you catch them immediately after they occur.

- To disable an AutoCorrect entry.
- To learn how to disable the automatic conversion of straight quotes to smart quotes, or two hyphens to a dash.

Inserting Symbols with the Symbol Dialog Box

Another way to insert a symbol is with the Symbol button on the Insert tab. Click Symbol to open a drop-down list of some common symbols (see Figure 3.50). (This list has some overlap with the ones in Table 2.2, but is not the same list. There are more math symbols here, for example.)

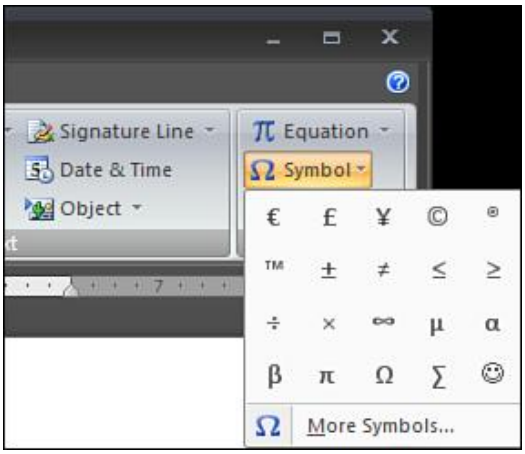


Figure 2.50: Symbols can be inserted from the Symbol drop-down list on the Insert tab.

If the symbol you want does not appear, click More Symbols to open the Symbol dialog box, shown in Figure 2.51. From here you can select any character from any installed font, including some of the alternative characters that do not correspond to a keyboard key, such as letters with accent symbols over them.

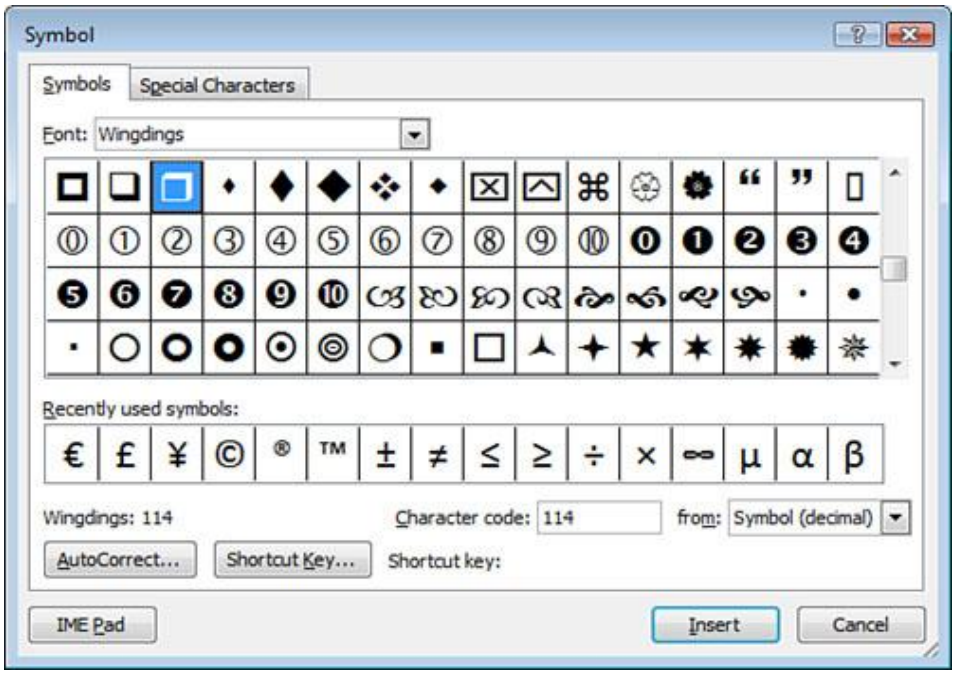


Figure 2.51: The Symbol dialog box can be used to insert any character from any font.

For a wide choice of interesting and unique symbols, check out the Wingdings fonts, which you can select from the Font drop-down menu.

You can also find a symbol by its character code, which is a numeric identifier of a particular symbol in a particular coding system. The two main coding systems are ASCII and Unicode. ASCII is the older system, and characters can be identified using either decimal or hexadecimal numbering in it. Unicode is the Windows standard for character identification, and it uses only hex numbering. Select the desired coding system from the From drop-down list and then type the character code in the Character Code box.

On the Special Characters tab of the dialog box are some of the most common typographical characters, along with reminders of their keyboard shortcuts. If you need to insert one of these common characters, finding it on the Special Characters tab can be easier than trying to wade through all the characters in a font for it.

Did You Know?

If you want a special character such as an accented letter or copyright symbol to blend in smoothly with the rest of the paragraph, make sure (normal text) is selected from the Font drop-down list in the Symbol dialog box. You would not always be able to do this, though, because not all symbols are available in all fonts. When you select symbols from the Special Characters tab, they are automatically in the (normal text) font.

Automating Symbol Entry : To make it easier, insert the same symbol again later. You might want to set up an AutoCorrect entry or a shortcut key combination for it.

To create an AutoCorrect entry, follow these steps:

1. From the Symbol dialog box, click the symbol for which you want to create the entry.
2. Click AutoCorrect. The AutoCorrect dialog box opens with a new entry already started.

3. Type the text that should represent the symbol. It is customary to enclose one or two characters in parentheses for AutoCorrect symbol insertion, but this is not required. For example, to create an entry for the ± sign, you might choose (+) as the text to enter (see Figure 2.52).

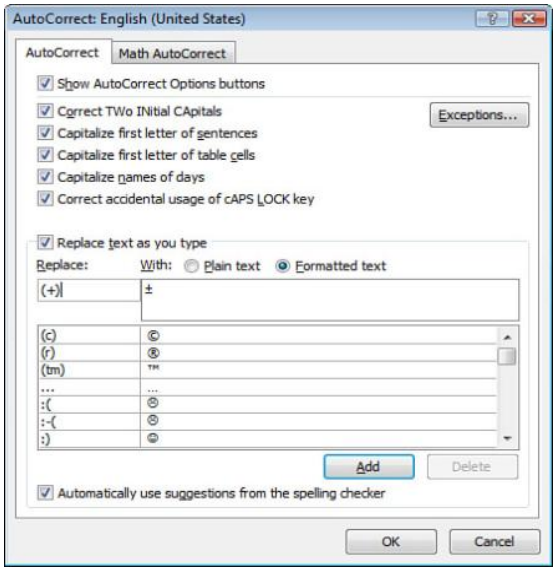


Figure 2.52: Add an AutoCorrect entry for a symbol.

4. Press Enter. The new entry appears on the list.
5. Click OK to return to the Symbol dialog box.

To assign a shortcut key combination to a symbol, follow these steps:

1. From the Symbol dialog box, click the symbol for which you want to create the shortcut.
2. Click Shortcut Key. The Customize Keyboard dialog box appears.
3. Click in the Press New Shortcut Key text box and then type the key combination you want to use. If that key combination is currently assigned to something else, a Currently Assigned To line will appear, as in Figure 2.53. (You can overwrite a default shortcut key assignment if desired.)

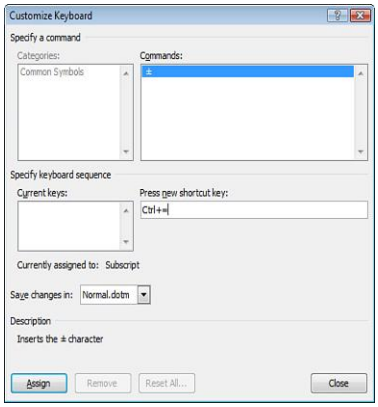


Figure 2.53: Map a keyboard shortcut to a symbol.

4. By default, the change will be saved to the Normal.dotm template; if you want it saved only to the open document, open the Save Changes In list and choose the document.

5. Click the Assign button.
6. Click Close to return to the Symbol dialog box.

2.12.4 Change Text Attributes

Font Change (Ctrl/shift/F)

Note: Default font is Calibri

1. Select text
2. Fonts button
3. Choose the desired font



Type Size Change

Note: Default font size is 11

1. Select text
2. Size button
3. Choose desired size or type a size into size button box

Type Style Change

1. Select text
2. Click the Home tab (if necessary)
3. Click desired style change- bold, italic, underline

Other available styles

1. Click the Font Dialog launcher
2. Select desired style change
    a. Superscript (example X2)
    b. Subscript (example H2O)
    c. Strike through
    d. Double Strike through
    e. Shadow
    f. Emboss
    g. Engrave
    h. SMALL CAPS
    i. ALL CAPS

Self Assessment Questions

5. A .....is a collection of colour swatches.

(a) Dockers (b) Property Bar
(c) Colour Palette (d) Toolbox

6.....display the same types of controls as a dialog box, such as command buttons, options, and list boxes.

(a) Dockers (b) Property Bar
(c) Colour Palette (d) Toolbox

2.13 Bullets in Word 2007

When you are writing a document, you need to make it easy to read. Professional writers sometimes talk about things called "Entry Points"… points where a reader can quickly and easily start reading your document and pick up what you are trying to say. Bullet points (see Figure 3.54) allow you to quickly structure information so your reader can easily interpret exactly what you are going on about!

2.13.1 How Do You Use Bullets In Word 2007?

1. Select the text you want to apply bullets to (you can do this by clicking and dragging across the text you want)
2. On the "Home" tab of the ribbon, look for the "Paragraph" group
3. In the top left hand corner of the paragraph group, you will find a button that looks like three lines with bullets. Click on it.

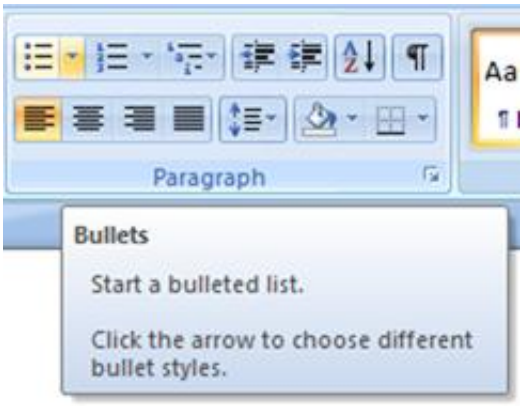


Figure 2.54: Bullet points.

That is the easy way to use bullets in your document. But what if you do not want to use the standard black dot bullet? Maybe you want to use something with a little more flair?

2.13.2 How to Use Custom Bullets in Word 2007

1. Again, select the test you want to apply bullets to, then find the bullets button in the "Paragraph" group on the "Home" tab of the ribbon
2. Instead of clicking on the bullets button, click on the little down arrow beside it. This will display a library of different bullets you can use. Feel free to use any of those, or….
3. Click on "Define New Bullet…" This will allow you to create any bullet you like, whether it be based on a letter, a number, any character out of any font you have installed on your computer, or even any picture that you have on your computer!!!
4. Select the style of bullet that you want… then click ok.

It is the quick and easy way to include bullets in your Word 2007 document

2.14 Numbering in Word 2007

Once you are done with your titles, style them with the appropriate heading style. Forget about setting the numbering in the heading style see Figure 2.55. We will number them in a better way.



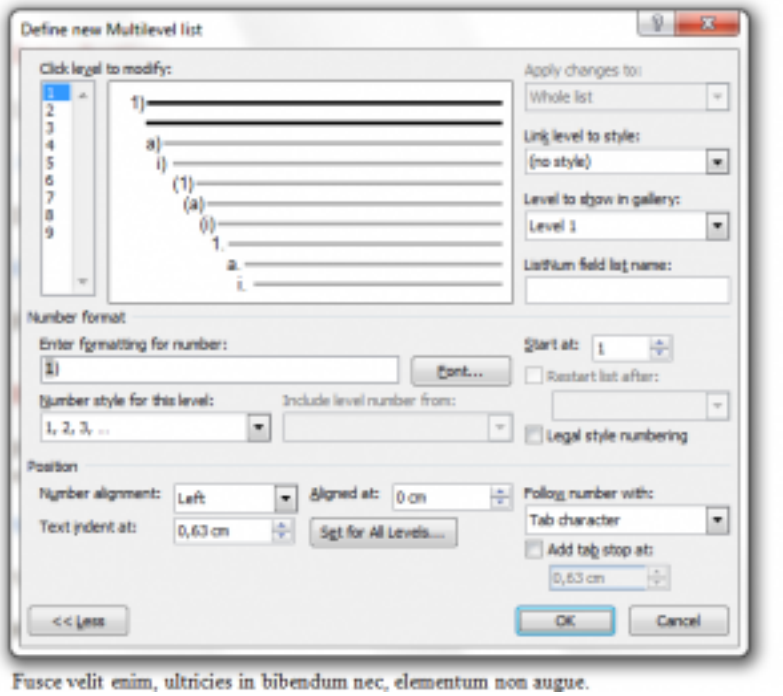
Figure 2.55: Example with heading styles.

Now, select your first heading and click on the "Multilevel List" in the "Home" tab and choose "Define New Multilevel List…." see Figure 2.56.



Figure 2.56: Define New Multilevel List.

On the bottom left of the "Define new Multilevel list" window click on the "More >>" button. This is what you will see in Figure 2.57:

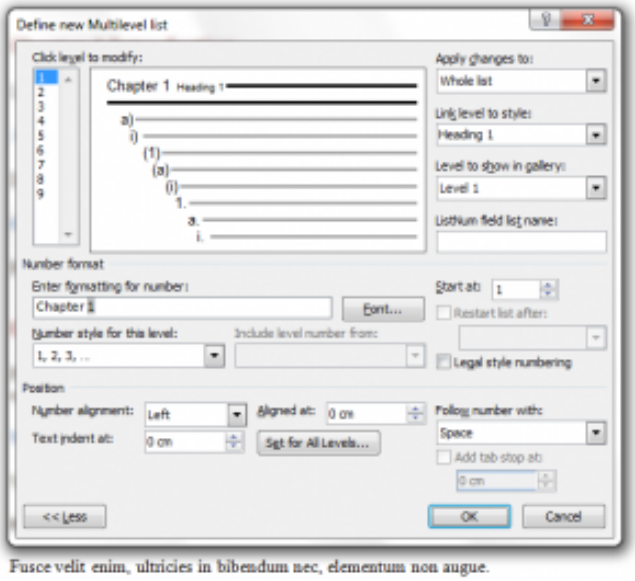


Fusce velit enim, ultricies in bibendum nec, elementum non augue.

Figure 2.57: Multilevel List Dialog.

Here we can select a level of the list (in the top left) and change its style.

Let us say you want each Heading 1 to include "Chapter". Simply enter "Chapter" before the number in the "Enter formatting for number:" text field. But aside from styling your list, the important step is to link this style to your header. You do this by selecting "Heading 1″ to the "Link level to style:" drop down list see Figure 2.58.



Fusce velit enim, ultricies in bibendum nec, elementum non augue.

Figure 2.58: Link level to style.

For levels other than 1, you can include the number of its parent level. This is very useful when you want sub sections to look like "1.2.1 Title". Simply edit those levels and select the level you want in the "Include level number from:" Do not forget to link each level to the appropriate headings style see Figure 2.59.

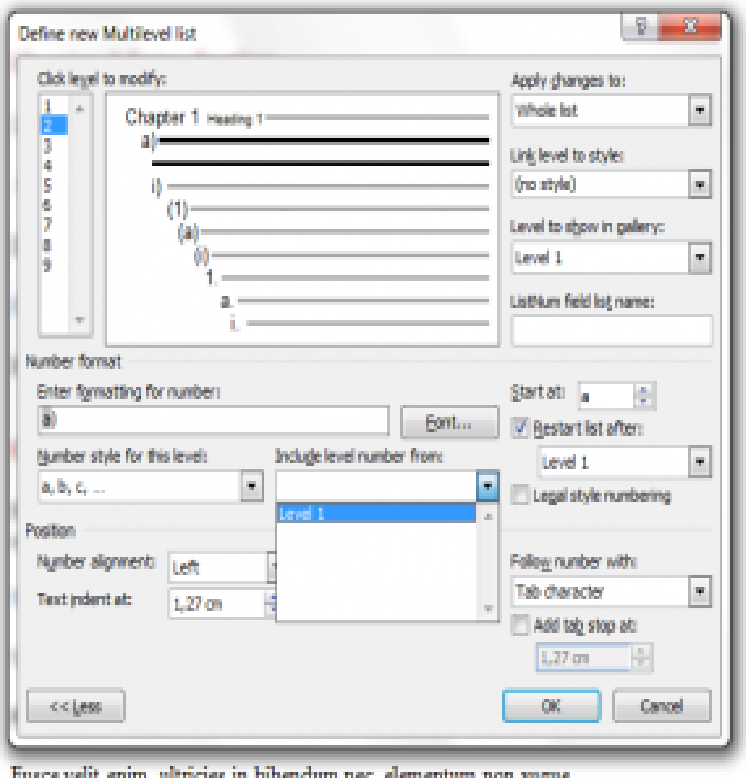


Figure 2.59: Include Level Number.

You should now have each heading numbered correctly! (see Figure 2.60.)



Figure 2.60: After Numbering.

But what about appendices.? What if you want Appendix A, Appendix B etc? Well, we know that appendices appear in the Table of Contents as the same level as Heading 1, so clearly they have to be in the first level of the list.

So, one way of setting this is by applying the Heading 1 style to each. Then, select your first appendix title and define a new style. The settings of the current active list style are already set, so you do not need to set everything again. This time you have the option to select how to apply these changes. Choose "This point forward" in the "Apply changes to:" drop down list and then make your changes. Set, for instance, the "Enter formatting for number:" to Appendix and the "Number style for this level:" to "A, B, C…" and then the "Start at:" to "A" see Figure 2.61.

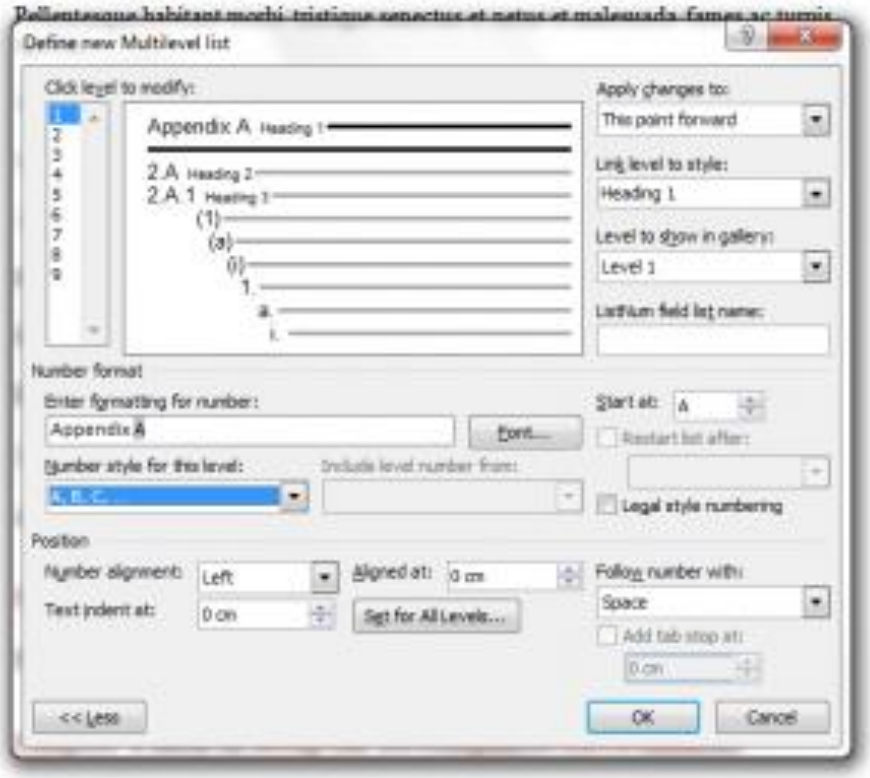


Figure 2.61: Appendix Multilevel List Style.

While this will do the trick, you have to manually set the new style. Create a new style by opening the "Styles" windows and clicking "New Style". Name it "Appendix", set the "Style type:" to "Linked (paragraph and character)" and "Style for given paragraphs:" to "Normal" see Figure 2.62.

Now, this is where the magic happens: set the "Style based on:" to "Heading 1″ and click "OK". This will link this style with Heading 1, which means two things: a) any changes in Heading 1 will be set to Appendix as well and b) the title of this heading will be set to the same level of Heading 1 in Table of Contents.

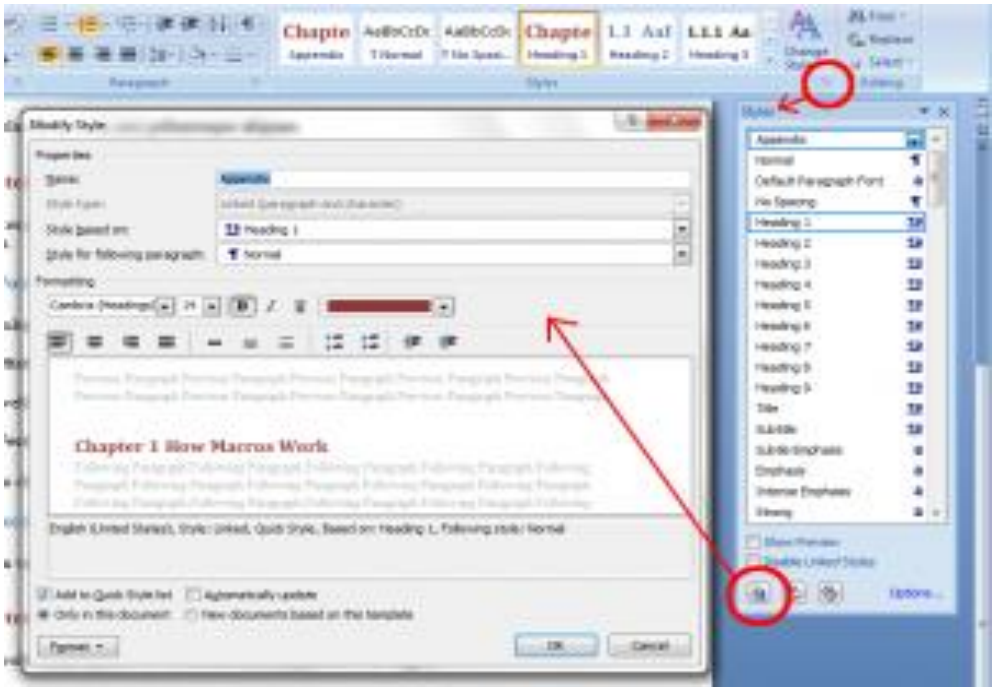


Figure 2.62: Appendix Style.

If you now apply the Appendix style to your titles, you will see that "Chapter" appears before them. To set this to "Appendix", select your title and define a new multi level list. Next, select the fourth level and set the "Link level to style:" to "Appendix". Make any changes to the style you want (like setting "Appendix") and it done! See Figure 2.63.



Figure 2.63: Appendix.

When you will generate your Table of Contents, you will see that Appendices are at the same level as Heading 1. If you change the Appendix style to be based on Heading 2, this will be visible to the Table of Contents as well see Figure 2.64.

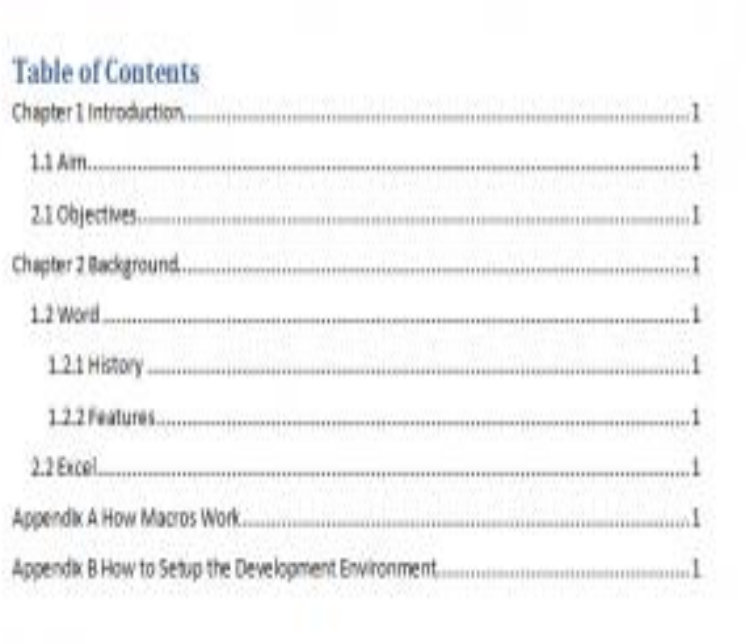


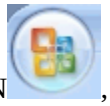
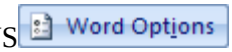
Figure 2.64: Table of Contents.

2.15 Auto Formatting in Word 2007

The AutoFormat tab applies formatting options in conjunction with the AutoFormat as You Type tab. This document shows you how *AutoFormat* can be turned on/off and how its various options affect your Word projects.

- Accessing the AutoFormat Tab
- Unique Options on the AutoFormat Tab

2.15.1 Accessing the AutoFormat Tab

1. From the OFFICE BUTTON  , click WORD OPTIONS 

The *Word Options* dialog box opens see Figure 2.65.

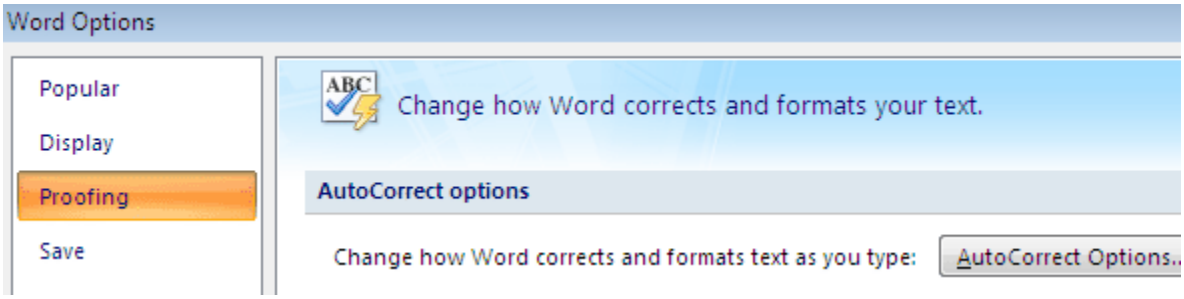


Figure 2.65: Word Options dialog box.

2. In the *Categories* pane, select ***Proofing***
3. In the *AutoCorrect options* section, click **AUTOCORRECT OPTIONS...**

The *AutoCorrect* dialog box appears.

4. Select the ***AutoFormat*** tab see Figure 2.66.

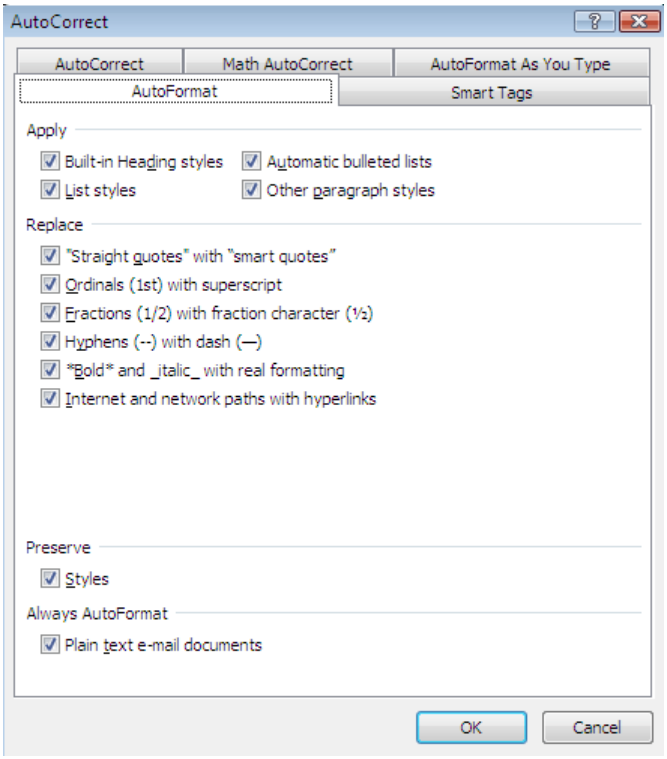


Figure 2.66: AutoFormat tab.

5. Select the desired options

An option is selected and active when a checkmark appears before it.

6. When finished, click **OK**

Note: As with any *AutoCorrect* option, you can override the formatting if you wish.

7. To exit the *Application Settings* dialog box, click **OK**

2.15.2 Unique Options on the AutoFormat Tab

Many *AutoFormat* tab options are identical to those found on the *AutoFormat As You Type* tab. The list below describes options that are unique to the *AutoFormat* tab.

For information on *AutoFormat* tab options that are also found in the *AutoFormat As You Type* tab.

List styles: Applies list styles to numbered, bulleted, outlines, and other lists. It replaces any numbers or bullets that were inserted manually

Other paragraph styles: Applies styles other than for headings and lists (e.g., body text)

Preserve styles: Retains the styles you have already applied in your document

Plain text e-mail documents: Formats e-mail messages when they are opened

2.16 Printing and Various Print Options

Word 2007 provides plenty of options for printing your document. For example, you can opt not to print colored backgrounds and images. Or, you can print your document in draft quality or in duplex, if your printer supports it. This can add up to big savings on ink and paper.

Unfortunately, though, not all of Word's printing options are immediately accessible. You will need to do some searching to find all of the printing options in word 2007.

You can access printing options when you print your document. Press CTRL + P to open the Print dialog box see Figure 2.67.

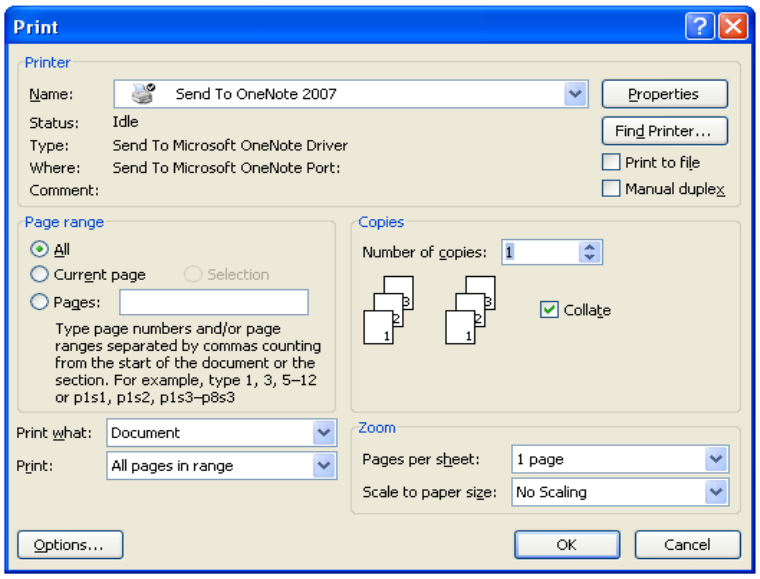


Figure 2.67: Print dialog box.

Click Options button in the bottom left corner of the dialog box. This opens the Word Options dialog box see Figure 2.68.

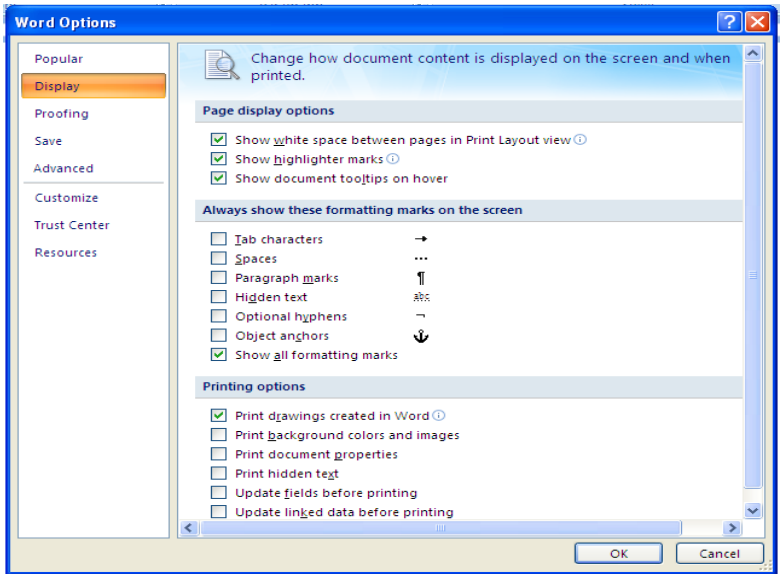


Figure 2.68: Word Options box.

You will see the Display section of the Word Options box. Here you will find a number of options. You can choose to print hidden text, backgrounds, drawings, and properties. You can also have word update links and form fields before printing. If you want more options, you will need to open the advanced section of the Word Options box see Figure 2.69. Then, scroll down to Print.

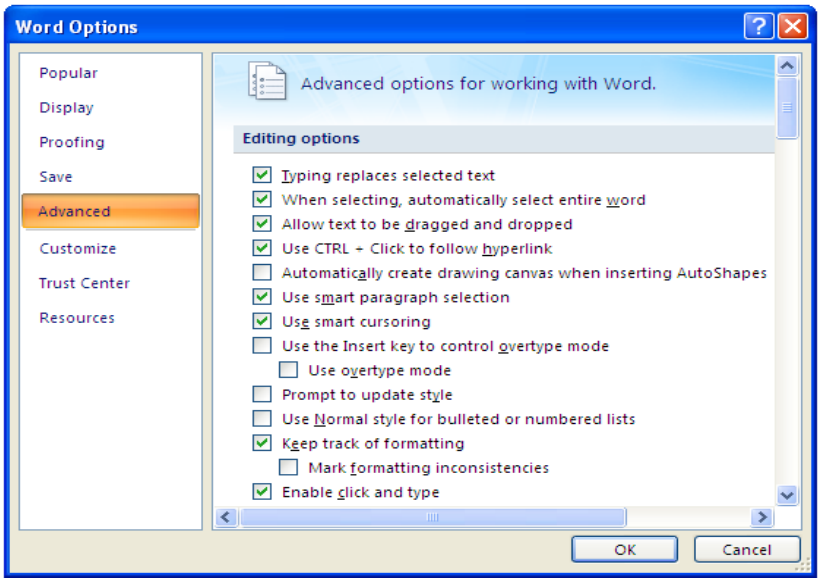


Figure 2.69: Advanced section of the Word Options box.

You can set options for print order, duplex printing and also print quality. When you are done, click OK. The options you select will be kept until you change them again.

2.17 Summary

- Word is the word processing software that has replaced the typewriter. It is commonly used to create letters, mass mailings, resumes, newsletters and so on.
- Word 2007 is full of new tools and options, expanded capabilities, and significant changes.
- Themes include predesigned settings for colors, fonts, and effects, and things like sidebars and quotes have their own styles as well.
- Microsoft Word is word processing software. It is used to create and edit texts, letters, reports, and graphics.
- The vertical scroll bar is located along the right side of the screen. The horizontal scroll bar is located just below your document. The horizontal scroll bar is only visible when your document is larger than your screen.
- The shortcut menus are helpful because they display only those options that can be applied to the item that was right-clicked and, therefore, prevent searching through the many menu options
- The rulers display horizontal and vertical scales that reflect the width and height of your typing area.
- The Word 2007 Styles section is used to quickly format an entire document.

2.18 Keywords

ASCII: It is the older system, and characters can be identified using either decimal or hexadecimal numbering in it.

Microsoft Word: It is a word processing program that allows you to create, revise, and save documents for printing and future retrieval.

Paragraph break: A paragraph break creates a whole new paragraph, which can have its own indentation, bullets and numbering, line spacing, and other paragraph-level settings.

Ruler*:* The *Ruler* is used as a quick way to adjust margins. Margins may also be adjusted by using a preset option provided by Word, or through the Page Setup dialog box.

Style: A style is a set of formatting characteristics such as font size, color, paragraph alignment, spacing, and shading.

Text area: The text area is basically where you type in your texts (letters and numbers). It is the open area with white background (depending on your chosen color) the blinking vertical line in the upper-left corner of the text area is the cursor.

Unicode: It is the Windows standard for character identification, and it uses only hex numbering.

2.19 Review Questions

1. Write a brief description of Microsoft Word 2007.
2. What are the features and area of MS Word 2007?
3. Write a brief notes on working with MS Word.
4. What are the menus and commands using in MS Word?
5. What is the Microsoft office template wizard? How do we use it?
6. How do we creating a new document in MS Word 2007?
7. What are the use of page views and page layouts?
8. How do we work with styles in MS Word 2007?
9. What is the use of paragraph section in MS Word 2007?
10. What use of page formatting option?

Answers for Self Assessment Questions

1 (a) 2 (b) 3 (c) 4 (a) 5 (c) 6 (a)

2.20 Further Studies

1. *Microsoft Office Word by Torben Lage Frandsen*

3

Advanced Features of MS-Word

STRUCTURE

3.0 Objective

3.1 Introduction

3.2 Spell Check

3.3 Thesaurus

3.4 Find and Replace

3.5 Headers and Footers

3.6 Working with Columns

3.7 Tabs and Indents

3.8 Creation and Working with Tables

3.9 Margins and Space Management in Document

3.10 Adding References and Graphics

3.11 Mail Merge

3.12 Envelops and Mailing Labels

3.13 Summary

3.14 Keywords

3.15 Review Questions

3.16 Further Studies

3.0 Objectives

After studying this chapter, you will be able to:

- Discuss the spell check
- Explain the thesaurus
- Discuss about the find and replace
- Explain the headers and footers
- Explain the working with columns
- Discuss the tabs and indents
- Explain the creation and working with tables

3.1 Introduction

Some of the advanced features of Microsoft Office Word 2007 offer ways in which you can automate and streamline the way you work. You can use macros in Word 2007 to easily automate repetitive, complex tasks.

A macro is a set of instructions that can group a series of actions and keystrokes as a single command. You can also simplify the management of your Word 2007 files by using master documents. Master documents divide large files into related subdocuments through a series of links. This course demonstrates how to create, edit, copy and delete macros through the use of the macro recorder. In addition, aspects of master and subordinate documents are explained, including outline levels, rearranging and restructuring subordinate documents in the master, and converting, deleting, merging, locking subdocuments and checking spellings and etc.

3.2 Spell Check

Use the built-in Word 2007 spell checker after you create a document to proof it for typos, spelling errors, and repeated words. Then decide on a word-by-word case or globally whether you want Word 2007 to make spelling corrections.

To check the spelling and grammar of a document: Place the cursor at the beginning of the document or the beginning of the section that you want to check

Click the Review Tab on the Ribbon

Click Spelling and Grammar on the Proofing Group.

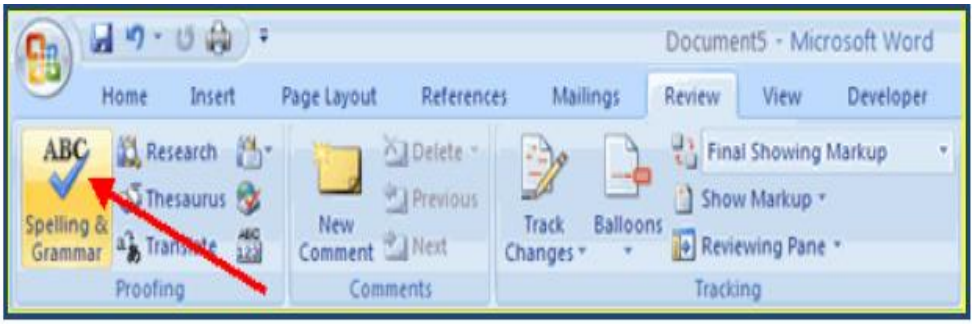


Figure 3.1: Spelling and Grammar.

Any errors will display a dialog box that allows you to choose a more appropriate spelling or phrasing.

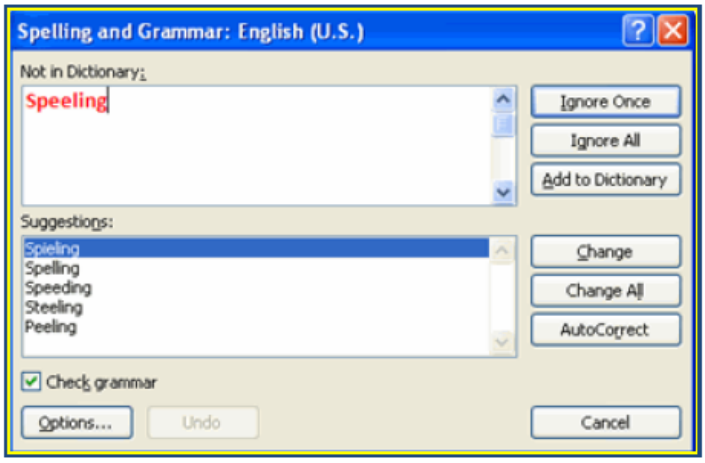


Figure 3.2: Spelling errors display a dialog box.

If you wish to check the spelling of an individual word, you can right click any word that has been underlined by Word and choose a substitution.

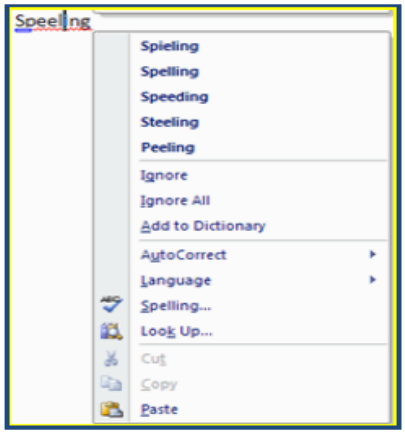


Figure 3.3: Suggested spelling list.

3.3 Thesaurus

The Thesaurus allows you to view synonyms. To use the thesaurus:

Click the Review Tab of the Ribbon

Click the Thesaurus Button on the Proofing Group.

The thesaurus tool will appear on the right side of the screen and you can view word options.

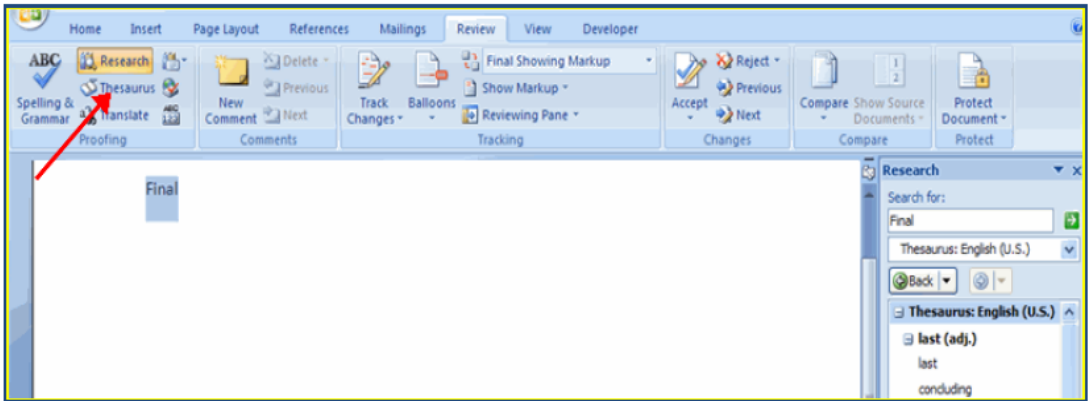


Figure 3.4: Thesaurus menu.

You can also access the thesaurus by right-clicking any word and choosing **Synonyms** on the menu.

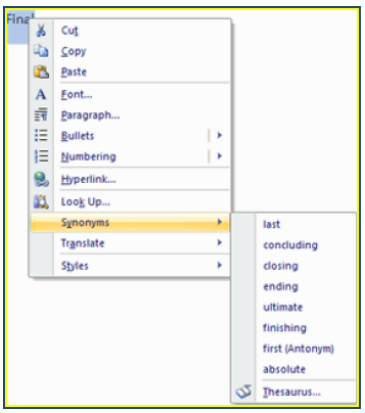


Figure 3.5: Synonyms list.

3.4 Find and Replace

These features are great for large documents. If you need to look for a specific word or section of your document click on the find button

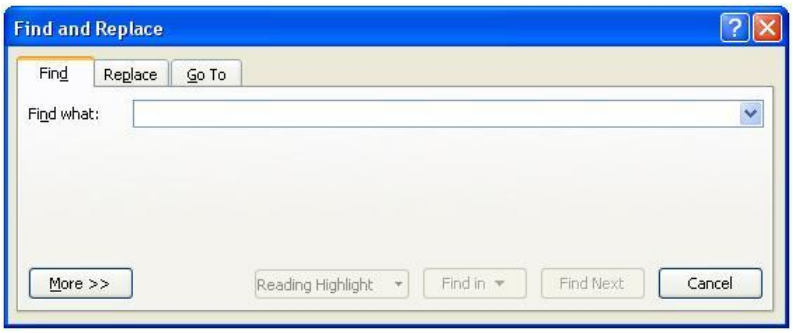


Figure 3.6: Find and replace.

When the Find and Replace window opens type the word you are looking for and the Find Next button will no longer be grayed out. Click on Find Next and it will bring you to the word in your document you typed in the Find what text box.

Now click on the Replace the button in the Editing section.

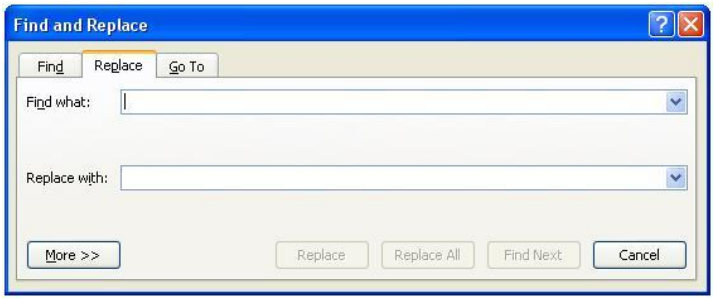


Figure 3.7: Find and replace option.

The same Find and Replace window pops up except the Replace tab is selected. This feature is useful if you have dates or names in a form letter you need to change. Type the word you would like to change in the Find what text box and Type the word you would like to change it to in the Replace with text box.

The Replace, Replace All and the Find Next buttons will no longer be grayed out. If you would like to replace the words one at a time click Find Next and if it is a word you want replaced click Replace and continue that way through the document.

If you know you want every word replaced click Replace All and each word in the document will be replace.

3.5 Headers and Footers

Microsoft Office Word 2007 includes many predesigned headers or footers that you can insert into your document. Or you can create a header or footer from scratch.

Insert a Predefined Header or Footer

- On the Insert tab, in the Header and Footer group, click Header or Footer.

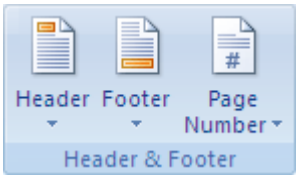


Figure 3.8: Header and Footer menu.

- Click the header or footer design that you want.

If you want to switch to a different predefined header or footer, repeat these steps, and choose a different Header or footer from the gallery.

Did You Know?

You can format the text in the header or footer by selecting the text and using the formatting options on the Mini toolbar, which is a part of the Microsoft Office Fluent interface.

Insert a Custom Header or Footer

- On the Insert tab, in the Header and Footer group, click Header or Footer.

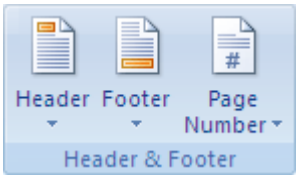


Figure 3.9: Header and Footer.

- Click Edit Header or Edit Footer. Type text or insert graphics and other content by using the options in the Insert group on the Design tab, under the Header and Footer Tools tab. If you do not see a gallery of header or footer designs, there might be a problem with the Building Blocks template on your computer.

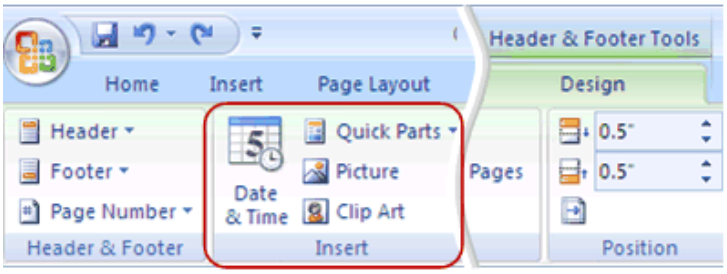


Figure 3.10: Header and Footer Tools tab.

To save the header or footer that you created to the gallery of header or footer options, select the text or graphics in the header or footer, and then click Save Selection as New Header or Save Selection as New Footer.

Make the first page header or footer different from the rest of the pages

- On the first page of the document, double click the header or footer area.

- Under Header and Footer Tools, on the Design tab, in the Options group, select the Different First Page check box.

If your document includes a cover page from the gallery of cover pages in Office Word 2007, the **Different First Page** option is already turned on. Inserting or editing a header or footer on this page does not affect the other pages in the document.

Create a header or footer, or make changes to the existing header or footer, on the first page.

Use no header or footer on the first page

On the first page of the document, double click the header or footer area.

Under Header and Footer Tools, on the Design tab, in the Options group, select the Different First Page check box.

If the Different First Page check box is already checked, do not clear it. Go on to the next step.

- In the First Page Header or First Page Footer area, delete the contents of the header or footer.

Make the header or footer different for odd and even pages

For example, you can use the title of the document on odd-numbered pages, and the chapter title on even-numbered pages. Or, for a booklet, you can place page numbers on odd-numbered pages to be on the right side of the page and page numbers on even-numbered pages to be on the left side of the page. This way, the page numbers are always on the outside edge when the pages are printed on both sides of the paper.

Create Odd and Even Headers or Footers in A Document That Does Not Yet Use Headers or Footers

Click an odd-numbered page, such as the first page of your document.

On the Insert tab, in the Header and Footer group, click Header or Footer.

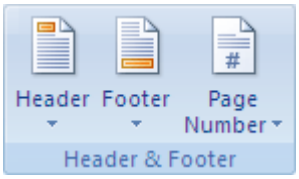


Figure 3.11: Header or Footer.

In the gallery of headers or footers, click a design labeled (Odd Page), such as Austere (Odd Page).

If you do not see a gallery of header or footer designs, there might be a problem with the Building Blocks template on your computer

Under Header and Footer Tools, on the Design tab, in the Options group, select the Different Odd and Even Pages check box.

Under Header and Footer Tools, on the Design tab, in the Navigation group, click Next Section to advance the cursor to the header or footer for even-numbered pages.

Under Header and Footer Tools, on the Design tab, in the Header and Footer group, click Header or Footer.

In the gallery of headers or footers, click a design labeled (Even Page), such as Austere (Even Page).

If necessary, you can format text in the header or footer by selecting the text and using the formatting options on the Office Fluent Mini toolbar.

- If you want to switch to a different predefined header or footer, repeat these steps, and choose a different header or footer from the gallery.

Create odd and even headers or footers in a document that already has headers or footers

- Double-click in the header or footer area.
- Under Header and Footer Tools, on the Design tab, in the Options group, select the Different Odd and Even Pages check box.
- The existing header or footer is now configured for odd-numbered pages only.
- Under Header and Footer Tools, on the Design tab, in the Navigation group, click Next Section to advance the cursor to the header or footer for even-numbered pages, and then create the header or footer for even-numbered pages.

Did You Know?

You can create different first-page or odd-and-even headers or footers, or if there are sections that are not linked, be sure to remove the headers or footers from each different page or section.

Make the header or footer different in each section or chapter

If your document is divided into sections, you can vary the headers and footers so that they display different content for each section. For example, if your document is divided into chapters by using section breaks, the chapter title can be displayed in the header of each chapter.

Tip-If you are unsure whether your document has sections, you can search for them.

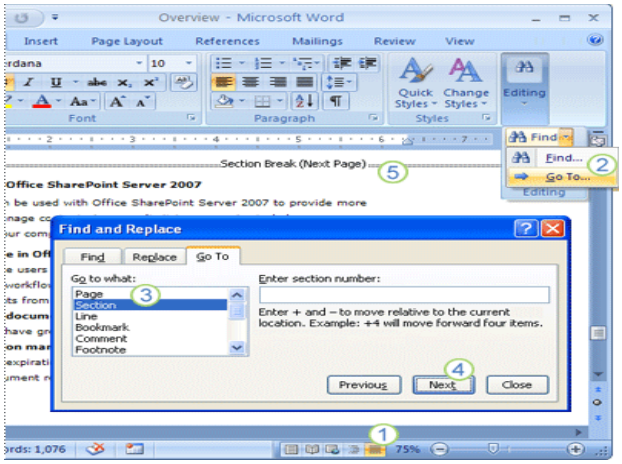


Figure 3.12: Making the header or footer different section.

Click Drafts on the status bar.

On the Home tab, in the Find group, click Go To.

Click Section.

Click Next to find section breaks in the document.

Draft view makes it easy to see the section breaks in the document.

Add section breaks to a document

If your document is not divided into sections, you can insert section breaks where you want the header or footer to vary.

- Starting at the beginning of the document, place the cursor at the beginning of the page where you want to start varying the header or footer.

- On the Page Layout tab, in the Page Setup group, click Breaks, and then under Section Breaks, click Next Page.
- Place the cursor at the beginning of the next page where you want to vary the header or footer, such as the first page of a new chapter.
- On the Page Layout tab, in the Page Setup group, click Breaks, and then under Section Breaks, click Next Page.
- Repeat steps 3 and 4 for every section break that you want in the document.

Use a document's section breaks to vary the header or footer

If your document is already divided into sections, you can use the section breaks to configure headers and footers.

- Starting at the beginning of the document, click in the first section for which you want to vary the header or footer.
- On the Insert tab, in the Header and Footer group, click Header or Footer.

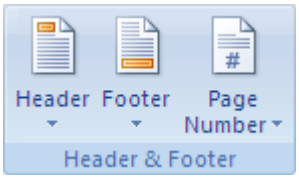


Figure 3.13: Header and Footer group.

- Click Edit Header or Edit Footer.
- On the Headers and Footers tab, in the Navigation group, click Link to Previous to break the connection between the header or footer in this section and the previous section.
- Change the existing header or footer, or create a new header or footer for this section.
- In the Navigation group of the Design tab (Header and Footer contextual tab), click Next Section to advance the cursor to the header or footer of the next section.
- On the Headers and Footers tab, in the Navigation group, click Link to Previous to break the connection between the header or footer in this section and the previous section.
- Change the existing header or footer, or create a new header or footer for this section.
- Repeat the previous three steps for all of the sections in the document.

Use the same header or footer across section boundaries

In a document where the header or footer varies by section, you can make the header or footer the same across section boundaries.

- Double-click the header or footer that you want to preserve across section boundaries.
- On the Headers and Footers tab, in the Navigation group, click Next Section.
- Click Link to Previous.
- Office Word 2007 will ask if you want to delete the header and footer and connect to the header and footer in the previous section. Click Yes.

Change the contents of a header or footer

On the Insert tab, in the Header and Footer group, click Header or Footer.

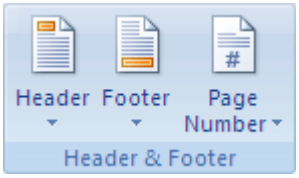


Figure 3.14: Header and Footer group.

Click Edit Header or Edit Footer.

Make your changes to the header or footer by selecting the text and revising it or by using the options on the Fluent Mini toolbar to format the text. For example, you can change the font, apply bold format, or apply a different font color.

Insert a page number

If your document already has content in the header or footer, you can add the page number to the header or footer.

If you want to use one of the preformatted page number designs, but you want more information in the header or footer than just the page number, it is easiest to start with a page number and then add other text or graphics to the header or footer.

Many of the footer designs in the gallery of footers already include the page number as part of the content.

For more information about inserting page numbers,

Add a page number to an existing header or footer

Double-click the header or footer area.

Position the cursor where you want to insert the page number.

- You may need to press the TAB key, adjust paragraph alignment, or make other changes to the header or footer content to position the cursor where you want it.
- Many header and footer designs are laid out in a table, and pressing TAB moves the cursor without inserting a tab stop. To insert a tab stop within a table cell, press CTRL+TAB. To determine whether the header or footer that you are working with is laid out in a table, look for the **Table Tools** contextual tab at the top of the Ribbon while the cursor is in the header or footer.
- Many header and footer designs contain content controls for storing content such as the document title and the date. When you click to edit the header or footer, the cursor might automatically select the content control. To move the cursor away from the content control (so that you can insert the page number in the header or footer without deleting the content of the content control), press the LEFT ARROW or RIGHT ARROW key.

- On the Insert tab, in the Header and Footer group, click Page Number.

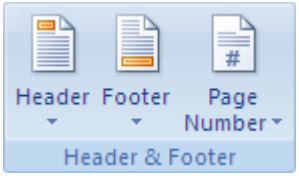


Figure 3.15: Header and Footer group.

- Click Current Position.
- Choose a page number design from the gallery of designs.

Add header or footer content to a preformatted page number design

On the **Insert** tab, in the **Header and Footer** group, click **Page Number**.

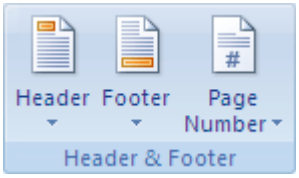


Figure 3.16: Header and Footer group.

Click Top of Page, Bottom of Page, or Page Margins, depending on where you want page numbers to appear in your document.

Choose a page number design from the gallery of designs.

Choose a design that positions the page number where you want it. For example, if you want some header content aligned on the left margin and the page number aligned on the right margin, choose a right-aligned page number design.

Do one of the following:

- To insert header or footer content before the page number, press the HOME key, enter the content, and then press TAB to position the content.
- To insert header or footer content after the page number, press the END key, press TAB, and then enter the content.
- If you chose a design from the Page Margins designs, click in the header or footer, and add the content that you want.

Insert the file name of the document

- Place the cursor where you want to insert the file name in the header or footer.
- Under Header and Footer Tools, on the Design tab, in the Insert group, click Quick Parts, and then click Field.
- In the Field names list, click Filename. If you want to include the path as part of the file name, select the Add path to filename check box.

Securities- Because field codes can be visible to anyone reading your document, be sure that the information you place in field codes is not information that you want kept private.

Insert the document title, author's name, or other document property

Place the cursor where you want to insert the document property in the header or footer.

Under Header and Footer Tools, on the Design tab, in the Insert group, clickQuick Parts.

Point to Document Property, and then click the document property that you want.

Insert the current date

Place the cursor where you want to insert the date in the header or footer.

Under Header and Footer Tools, on the Design tab, in the Insert group, click Date and Time. In the Available formats list, click the format in which you want the date (and time, if you want) to appear.

Remove the header or footer

Click anywhere in the document.

On the **Insert** tab, in the **Header and Footer** group, click **Header** or **Footer**.

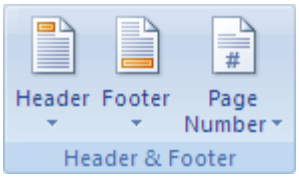


Figure 3.17: Header and Footer group.

Click Remove Header or Remove Footer.

The headers or footers are removed from the entire document.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Create odd and even headers in your Project document.

……..……

……..……

……...

3.6 Working with Columns

When you lay out page text in columns you will generally fit a little more text on the page than if you laid it out so the lines stretch full width of the page. The shorter lines of text are also easier to read, as the reader's eye does not need to travel so far across the page before returning to read the next line of text. A document laid out in columns can look more approachable too, as there is white space within the page, which offers readers a place to rest their eyes. Columns can help to separate sections of your document and make them look more inviting to read. Columns are a good way to separate sections of your document on one page. For example, when creating a newsletter or bulletin, columns can be used to give the document a more professional look. The following instructions show you how to create, modify, and delete columns in your document.

- Creating Columns
- Modifying Columns
- Deleting Columns

3.6.1 Creating Columns

By creating columns, you are creating a separate section in your document. Therefore, any formatting changes such as margins, indents, or headers and footers that you make to the column text affect only the section, not the entire document. You can create columns that span a single page or the entire document. Using the button option to create columns is quick and easy, whereas the Columns dialog box requires more steps but offers more options for modification.

Creating Columns: Button Option

Columns are easy to apply using the presets found on the COLUMNS button.

1. Place the insertion point in the document that you want formatted into columns

OR

Select the text to be formatted into columns



2. Windows: From the *Page Layout* command tab, within the *Page Setup* section, click **COLUMNS**

3. Select the desired number and style of columns

Creating Columns: Dialog Box Option

1. To create columns on selected text only, select the *More Columns...* option within the *Columns* submenu, which brings you to the *Columns* dialog box. The *Columns* dialog box gives you more options for modifying columns. With the dialog box, you can adjust the width and spacing of columns. Also, the dialog box allows you to create up to 45 columns depending on the size of your paper; however, standard paper allows only 12 columns.

2. Place the insertion point in the document that you want formatted into columns

OR

Select the text to be formatted into columns

Windows: From the *Page Layout* command tab, within the *Page Setup* section, click ***Columns*** » select ***More Columns...***

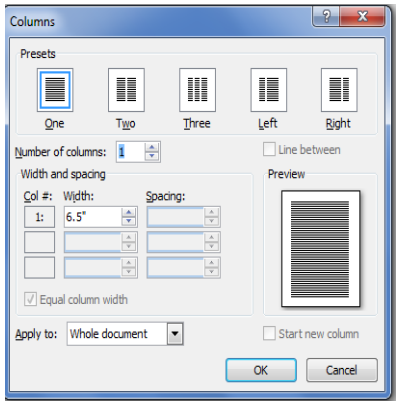


Figure 3.18: Column Window.

1. To select the number of columns, in the *Number of columns* text box, use the nudge buttons or type the desired number of columns

OR

Within the *Presets* section, select the desired option

4. From the *Apply to* pull-down list, select **Selected Text**

Your choices will depend upon whether you selected the text or placed your insertion point in the text to create columns.

5. Click **OK**

The columns are applied to the selected text only.

3.6.2 Modifying Columns

There are several different options you can use to modify your columns to make them appear exactly how you want them. The following section provides instructions on how to adjust column width, add lines, insert column breaks, and level column endings.

1. Adjusting Column Width: Ruler Option

Place the insertion point in any column of text

2. Move the insertion point over any of the column margin boundaries on the ruler so that it turns into a double arrow. 

3. Click and drag the column boundary for the appropriate column width

Adjusting Column Width: Column Dialog Box Option

Place the insertion point in the document that is formatted into columns **OR**

Select the text that is formatted into columns

Windows: From the *Page Layout* command tab, within the *Page Setup* section, click ***Columns*** » select ***More Columns...***

OPTIONAL: To create columns of unequal width, make sure that **Equal column width** is not selected

NOTE: The option is not selected when no checkmark appears

Under *Width and spacing*, use the nudge buttons or type values for the column attributes you want to change

NOTE: The *Width* will alter the width of the column and the *Spacing* will alter the space between the columns.

Click **OK**

Adding Lines

Adding lines between columns can add an element of design to your document. You may want to add lines to your column if you are following a style similar to that of a newsletter or bulletin. The following feature automatically adds lines between all columns.

1. Place the insertion point within the column text

2. Windows: From the *Page Layout* command tab, within the *Page Setup* section, click ***Columns*** » select

More Columns...

3. Select ***Line between***

4. Click **OK**

Inserting Column Breaks

Insert a column break when you want to force the end of a column and the beginning of another.

Place the insertion point at the point in the text where you want the column to break

Windows: From the *Page Layout* command tab, within the *Page Setup* section, select ***Breaks*** » select **Column**

Balancing Column Endings

When using columns, often the text in the last column is of uneven length with the previous column. Inserting a continuous column break will balance the column lengths, giving your document a finished, professional look.

Place the insertion point after the last character in the last column

Windows: From the *Page Layout* command tab, within the *Page Setup* section, select ***Breaks*** » select ***Continuous***

Caution

Be careful when sizing columns if a cell is selected in a column and if you attempt to drag the sizing tool to change the column width, only the width of the row holding selected cell will change. Make sure no cells are selected if you want to size the entire column.

3.6.3 Deleting Columns

You can choose to delete all columns in a document or only a section of columns.

Deleting Columns: Button Option

Place the insertion point in the document that is formatted into columns

OR

Select the text that is formatted into columns

Windows: From the *Page Layout* command tab, within the *Page Setup* section, click **COLUMNS**

Deleting Columns: Dialog Box Option

Deleting All Columns

Windows: From the *Home* command tab, within the *Editing* section, click **Select** » select **Select All**

OR

Windows: Press [**Ctrl**]+[**A**]

Windows: From the *Page Layout* command tab, within the *Page Setup* section, click ***Columns*** » select ***More Columns***

Within the *Presets* section, select **One**

From the *Apply To* pull-down list, select **Whole document**

Click **OK**

Deleting Columns from a Section

Select the text that you want changed to one column

Windows: From the *Page Layout* command tab, within the *Page Setup* section, click ***Columns*** » The *Columns* dialog box opens.

Within the *Presets* section, select **One**

From the *Apply To* pull-down list, select **This section**

Click **OK**

Self Assessment Questions

1. To use the thesaurus by……

(a) Click the review tab on the ribbon (b) Click the review tab of the ribbon

(c) Click spelling and grammar on the proofing group (d) Click drafts on the status bar.

2. ……..includes many predesigned headers or footers that you can insert into your document.

(a) Microsoft office word 2003 (b) Microsoft office word 2010

(c) Microsoft office word 2007 (d) Microsoft office word 97

3. Using the button option to ……..is quick and easy, whereas the Columns dialog box requires more steps but offers more options for modification.

(a) Create columns (b) Modifying columns

(c) Deleting columns (d) Inserting columns

3.7 Tabs and Indents

Using Tabs and Indents ensures that when your document prints, paragraphs and text are all lined up in the way you intended. We have seen many people use the space bar to indent text and while everything appears

lined up on the screen, once printed, text rarely lines up appropriately. To effectively work with Tabs and Indents, you will need to have your ruler displayed.



Figure 3.19: Ruler.

If you do not see the ruler at the top of your document workspace (below the Ribbon), you will want to turn it on.

Turning On the Ruler Using the Ribbon
Check the box next to Ruler in the Show/Hide group on the View tab

Turning On the Ruler Using the Jump Button 

Click the Ruler Jump Button at the top of the vertical scroll bar ()

3.7.1 Tabs

Tabs are set, by default at every ½ inch between your margins (until you start setting custom tabs). You do not have to do anything special to use the default tabs except press the [Tab] key on your keyboard.
When you do, you can see tabs in the form of arrows (→ → → ¶) on your screen. (You may need to click the Show/Hide button in the Paragraph group on the Home tab to see the [Tab] characters.)
If you look closely at the ruler bar, you can see small tick marks at every ½ inch along the bottom. Those at the default tab stops. However, when you begin to set custom tabs, any default tabs to the left of (or before) the custom tab stop are automatically deleted. The tab stop indicator is located on the very left edge of your ruler, just under the Ribbon. Typically, it displays the Left-Aligned Tab symbol (). In addition to [Tab] stops that align along the left, you can create [Tab] stops that align in the center, at the right or with a decimal.

Symbol	Means
	Left-Aligned Tab
	Center-Aligned Tab
	Right-Aligned Tab
	Decimal-Aligned Tab
	Bar Tab

Setting a Left-Aligned Tab

- Select the paragraphs that will be affected by the new [Tab] stop
- Verify the tab stop indicator displays the symbol for a Left-Aligned Tab ()
- Click in the Ruler Bar at the location of the new [Tab] stop

Upon setting a new [Tab], you should see:



The “L” indicates a custom, left-aligned tab. You can see the default tabs stops before the custom stop have been removed. But, the tab stops after the custom stop remains.

Setting a Center-Aligned Tab

- Select the paragraphs that will be affected by the new [Tab] stop
- Click the tab stop indicator until you see the symbol for a Center-Aligned Tab ( )
- Click in the Ruler Bar at the location of the new [Tab] stop

Setting a Right-Aligned Tab

- Select the paragraphs that will be affected by the new [Tab] stop
- Click the tab stop indicator until you see the symbol for a Right-Aligned Tab ( )
- Click in the Ruler Bar at the location of the new [Tab] stop

Setting a Decimal-Aligned Tab

- Select the paragraphs that will be affected by the new [Tab] stop
- Click the tab stop indicator until you see the symbol for a Right-Aligned Tab ( )
- Click in the Ruler Bar at the location of the new [Tab] stop

Moving a Custom Tab

- Select the paragraphs that will be affected by changing the custom [Tab] stop
- Click and drag the custom tab stop to its new location

Deleting a Custom Tab

- Select the paragraphs that will be affected by changing the custom [Tab] stop
- Click and drag the custom tab stop DOWN off the ruler

3.7.2 Indents

Whether you know it or not, you create and work with indents every time you create a bulleted or numbered list.

While you can set your indents in the Paragraph Dialog Box, it is certainly easier to set them using the Ruler.

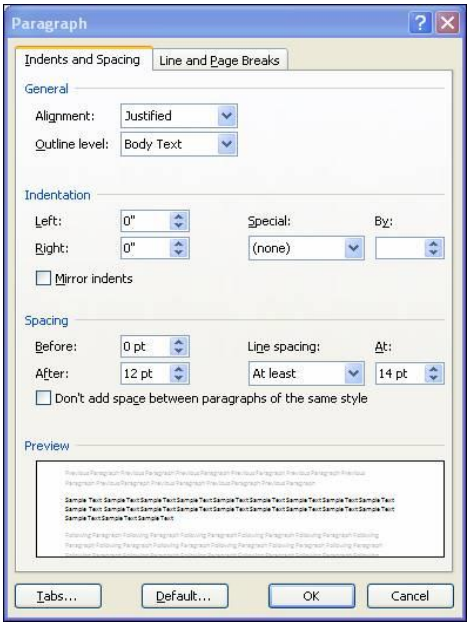


Figure 3.20: Paragraph Dialog Box.

Symbol…	Means
	First Line Indent
	Left Indent
	Subsequent Lines Indent
	Right Indent

Modifying the Left Indent

- Select the paragraphs that will be affected by changing the Indent
- Drag the Left Indent marker left or right to a new location ()

Modifying the First Line Indent

- Select the paragraphs that will be affected by changing the Indent
- Drag the First Line Indent marker left or right to a new location ()

Modifying the Subsequent Lines Indent

- Select the paragraphs that will be affected by changing the Indent
- Drag the Subsequent Lines Indent marker left or right to a new location ()

Modifying the Right Indent

- Select the paragraphs that will be affected by changing the Indent
- Drag the Right Indent marker left or right to a new location ()

Exercise: Check Your Progress 2
Note: i) Use the space below for your answer.
Ex1: Create a header and a trailer page, formatting data, conditional formatting, and output.

……..……

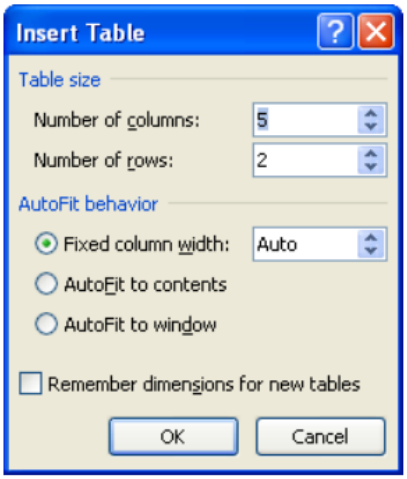
……..……

……...

3.8 Creation and Working with Tables

A table is made up of rows and columns. The intersection of a row and column is called a cell. Tables are often used to organize and present information, but they have a variety of uses as well. You can use tables to align numbers and create interesting page layouts.

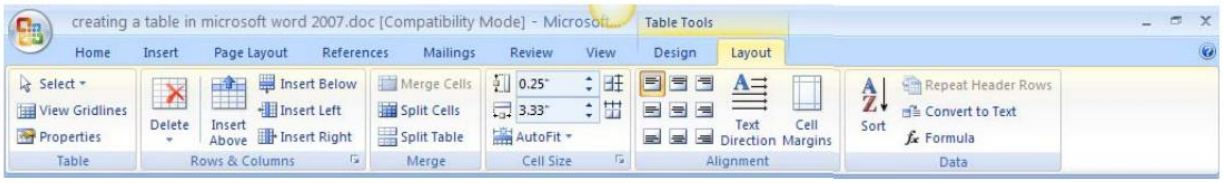
Click the New File icon to create a new document if necessary.
To Create a Table:
1) Click the Insert tab on the Ribbon
2) Click the Table button
3) Select Insert Table



4) Click the arrows to select the desired number of columns
5) Click the arrows to select the desired number of rows
6) Click OK

Navigating in a Table
Please see below to learn how to move around inside a table.

Action	Description
Key	To move from one cell in the table to another. When you reach the last cell in a table, pressing the Tab key will create a new row.
Shift+	To move one cell backward in a table
Arrow Keys	Allow you to move left, right, up and down



To Insert a Row:
1) Position the cursor in the table where you would like to insert a row
2) Select the Layout tab on the Ribbon
3) Click either the Insert Row Above or the Insert Row Below button in the Rows & Columns group

To Insert a Column
1) Position the cursor in the table where you would like to insert a column
2) Select the Layout tab on the Ribbon
3) Click either the Insert Columns to Left button or the Insert Columns to Right button in the Rows & Columns group

To Delete a Row
1) Position your cursor in the row that you would like to delete

2) Select the Layout tab on the Ribbon
3) Click the Delete button in the Rows & Column group
4) Select Delete Rows

To Delete a Column
1) Position your cursor in the column that you would like to delete
2) Select the Layout tab on the Ribbon
3) Click the Delete button in the Rows & Column group
4) Select Delete Columns

Formatting a Table
Using Microsoft Word you are able to format a table by changing table lines and colors, shading tables, adjusting row and column size as well as alignment.



To Shade the Cells/Table:
1) Position your cursor in the table
2) Click the Layout tab on the Ribbon
3) Click the Select button in the Table group
4) Choose the item you wish to format
5) Click the Design tab on the Ribbon
6) Click the Shading button
7) Select a shading color

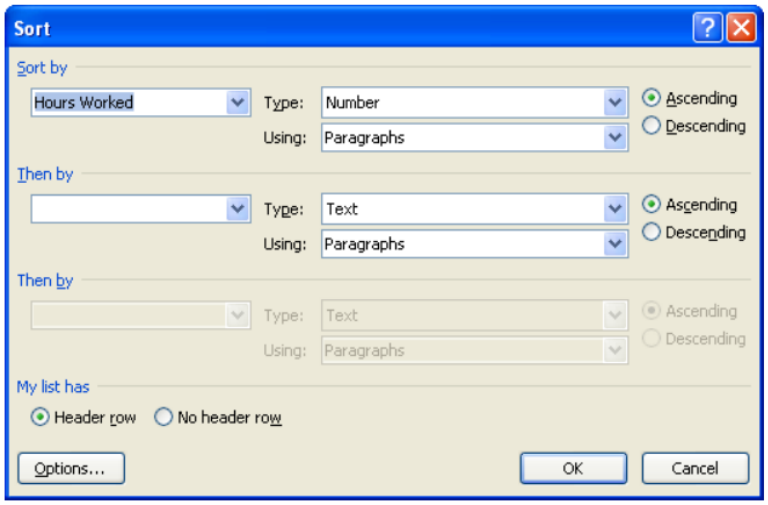
To Adjust the Line Style of a Cell/Table:
1) Position your cursor in the table
2) Click the Layout tab on the Ribbon
3) Click the Select button in the Table group
4) Choose the item you wish to format
5) Click the Design tab on the Ribbon
6) Select the Line Style drop-Down Arrow
7) Select a style
8) Click your mouse in the table where you want to apply the line style to 9) Press ESC key on the keyboard when you are finished applying the style

Note: You are able to format data in a table the same way you format it in a document
To Merge Cells in a Table
1) Select the cells that you would like to merge in the table
2) Click on the Layout tab on the ribbon
3) Click the Merge Cells button in the Merge group

Sorting Data in a table
You can easily sort data in a table in either ascending or descending order.
To Sort Data in a Table:

1) Position your cursor in the table
2) Select the Layout tab
3) Click the Sort button in the Data group
4) Select the field to sort by
5) Select the order (ascending, descending)
6) Choose if your table has a header row (In the example below, the table has a header row)

Name	Hours Worked
Mike	10
Ann	16
Steve	25
Michele	50



7) Click OK

3.9 Margins and Space Management in Document

Page margins are the blank space around the edges of the page. In general, you insert text and graphics in the printable area between the margins. However, you can position some items in the margins for example, headers, footers, and page numbers. If you need to change margins for portion of text, visit this GROK article Microsoft Word 2007: Creating margin for portion of text.

1. On the Page Layout tab, click Margins.

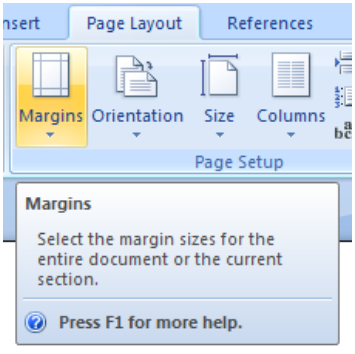


Figure 3.32: Margins.

2. Click the margin type that you want. For the most common margin width, click Normal.

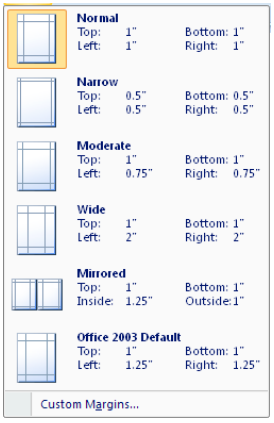


Figure 3.33: margin type.

3. When you click the margin type that you want, your entire document automatically changes to the margin type that you have selected.

You can also specify your own margin settings. Click Margins, click Custom Margins, and then in the Top, Bottom, Left, and Right boxes, enter new values for the margins.

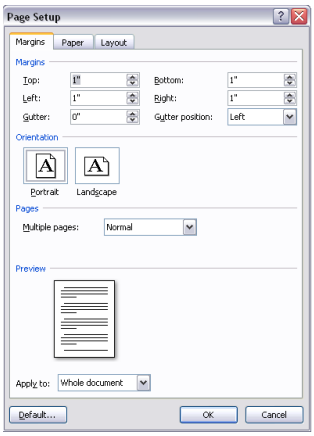


Figure 3.34: Page setup.

To change the default margins, click Margins after you select a new margin, and then click Custom Margins. In the Page Setup dialog box, click the Default button, and then click Yes. The new default settings are saved in the template on which the document is based. Each new document based on that template automatically uses the new margin settings.

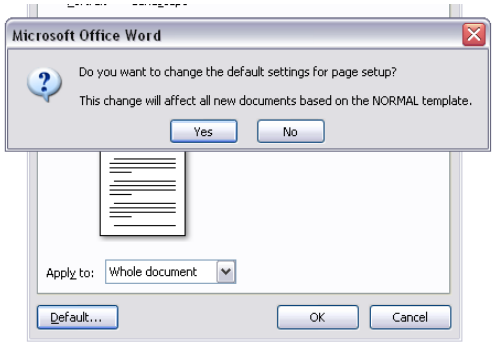


Figure 3.35: margin settings.

3.10 Adding References and Graphics

3.10.1 References and Citations

Word 2007 offers great tools for citing sources, creating a bibliography, and managing the sources. The first step to creating a reference list and citations in a document is to choose the appropriate style that you will be using for formatting the citations and references.

Style

To choose a publishing style:

- Click the **References** Tab on the Ribbon
- Click the drop down box next to **Style** in the Citations and Bibliography Group
- Choose the appropriate style.

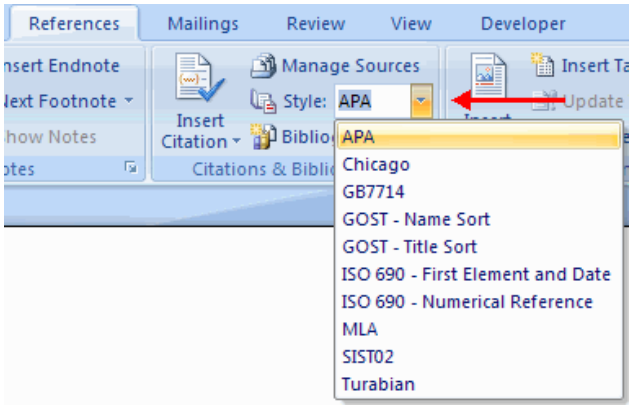


Figure 3.36: Style.

Citations

To insert a citation in the text portion of your document:

- Click the **References** Tab on the Ribbon
- Click the **Insert Citation** Button on the Citations and Bibliography Group
- If this is a new source, click **New Source**
- If you have already created this source, it will in the drop down list and you can click on it

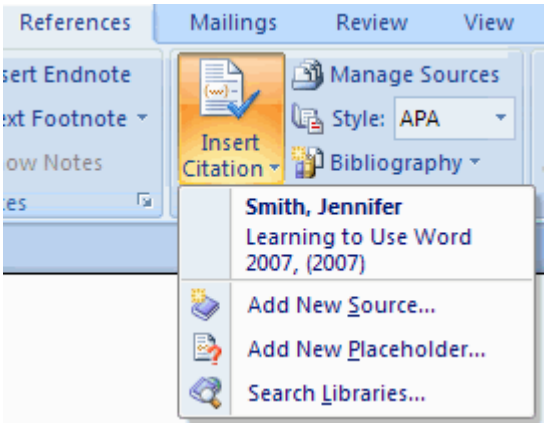


Figure 3.37: Insert Citation Button.

- If you are creating a **New Source**, choose the **type of source** (book, article, etc.)

- Complete the **Create Source Form**
- If you need additional fields, be sure to click the **Show All Bibliography Fields** check box
- Click **OK**

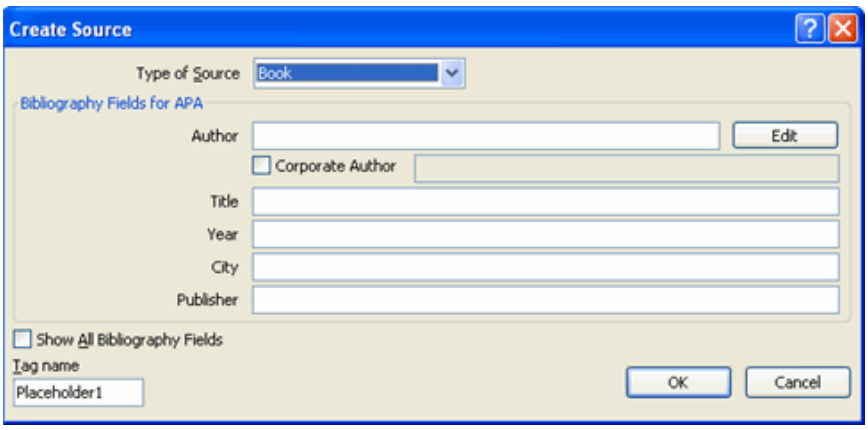


Figure 3.38: Create Source.

Placeholders

Placeholders can be utilized when there is a reference to be cited, but you do not have all of the information on the source. To insert a Placeholder:

- Click **Insert Citation**
- Click **Add New Placeholder**

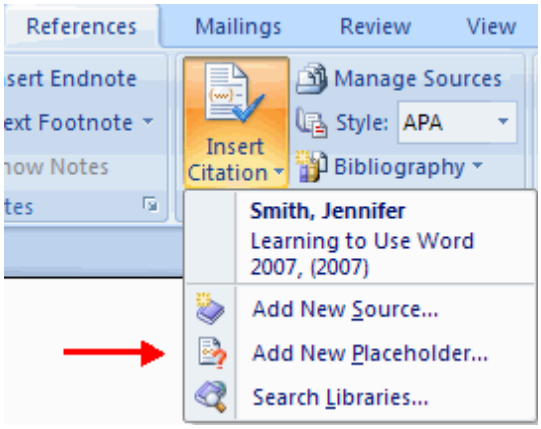


Figure 3.39: Add New Placeholder.

Manage Sources

Once you have completed a document you may need to add or delete sources, modify existing sources, or complete the information for the placeholders. To Manage Sources:

- Click the **References** Tab on the Ribbon
- Click the **Manage Sources** Button on the Citations and Bibliography Group
- From this menu you can **Add, Delete, and Edit Sources** (note, you can preview the source in the bottom pane of the window

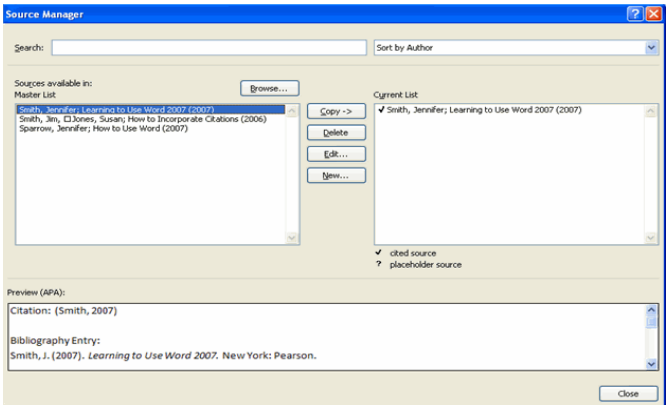


Figure 3.40: Source Manager.

Bibliography

To add a Bibliography to the document:

- Place the cursor in the document where you want the bibliography
- Click the **References** Tab on the Ribbon
- Click the **Bibliography** Button on the Citations and Bibliography Group
- Choose **Insert Built-in Bibliography/Works Cited** or **Insert Bibliography**



Figure 3.41: Bibliography button.

3.10.2 Insert Footnote

Some types of academic writing utilize footnotes. To insert a footnote:

- Click the **References** Tab on the Ribbon
- Click **Insert Footnote** (or Insert Endnote depending on your needs)
- Begin typing the footnote

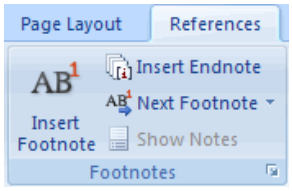


Figure 3.42: References.

3.10.3 Illustrations, Pictures, and SmartArt

Word 2007 allows you to insert illustrations and pictures into a document. To insert **illustrations**:

- Place your cursor in the document where you want the illustration/picture
- Click the **Insert** Tab on the Ribbon
- Click the **Clip Art** Button
- The dialog box will open on the screen and you can search for clip art.
- Choose the illustration you wish to include

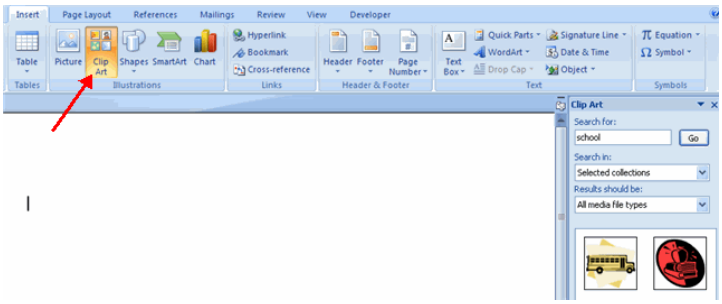


Figure 3.43: Clip Art menu.

To insert a **picture**:

- Place your cursor in the document where you want the illustration/picture
- Click the **Insert** Tab on the Ribbon
- Click the **Picture** Button
- Browse to the picture you wish to include
- Click the **Picture**
- Click **Insert**

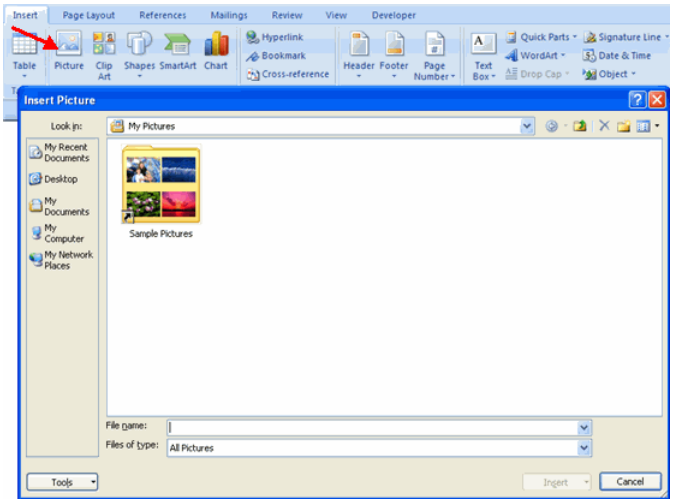


Figure 3.44: Inserting image.

Smart Art is a collection of graphics you can utilize to organize information within your document. It includes timelines, processes, or workflow. To insert SmartArt

- Place your cursor in the document where you want the illustration/picture
- Click the **Insert** Tab on the Ribbon

- Click the **SmartArt** button
- Click the **SmartArt** you wish to include in your document
- Click the arrow on the left side of the graphic to insert text or type the text in the graphic.

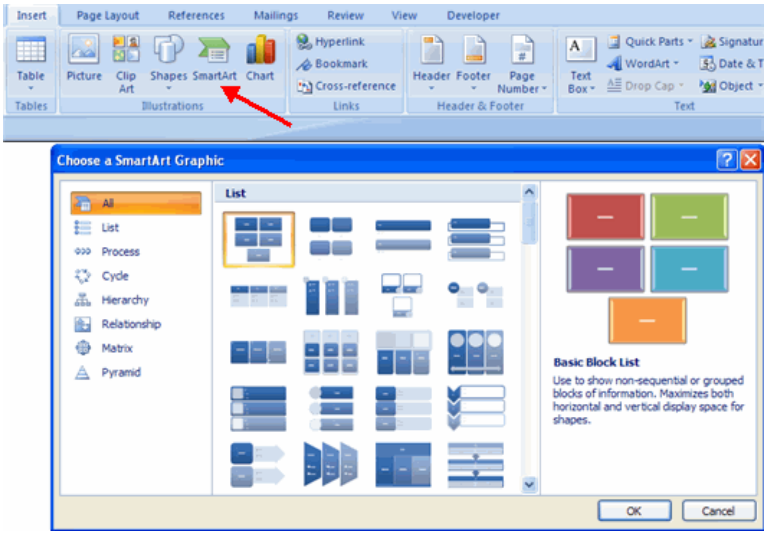


Figure 3.45: Choosing smart art graphics.

Resize Graphics

All graphics can be resized by clicking the image and clicking one corner of the image and dragging the cursor to the size you want the picture.

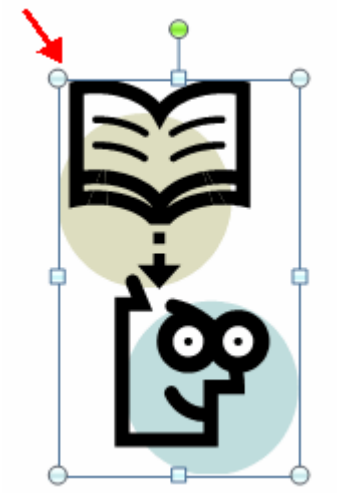


Figure 3.46: Resize Graphics.

Watermarks

A watermark is a translucent image that appears behind the primary text in a document. To insert a watermark:

- Click the **Page Lay**out Tab in the Ribbon
- Click the **Watermark** Button in the Page Background Group
- Click the **Watermark** you want for the document or click **Custom Watermark** and create your own watermark
- To remove a watermark, follow the steps above, but click **Remove Watermark**

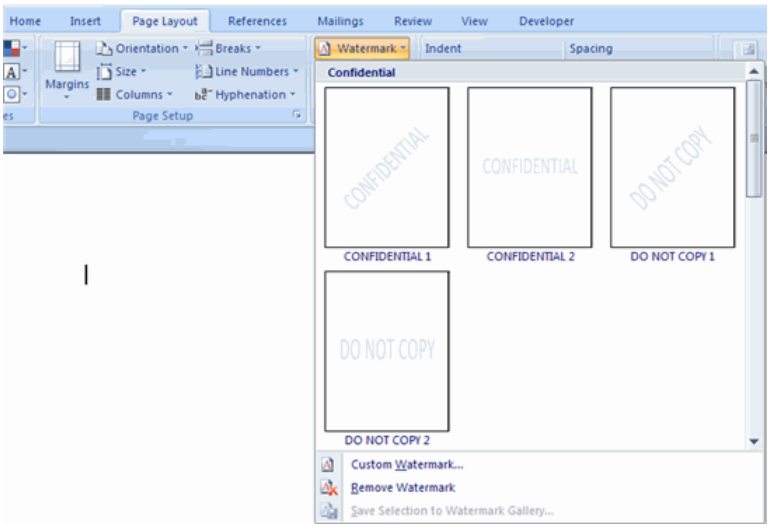


Figure 3.47: Watermarks.

3.11 Mail Merge

Mail Merge is an automated feature of MS Word that enables you to merge a data source(a file that stores fields and records of information, like first name, last name, etc...) into a copy of a document to customize or personalize the document. The Word 2007 Mail Merge toolbar looks as follows:

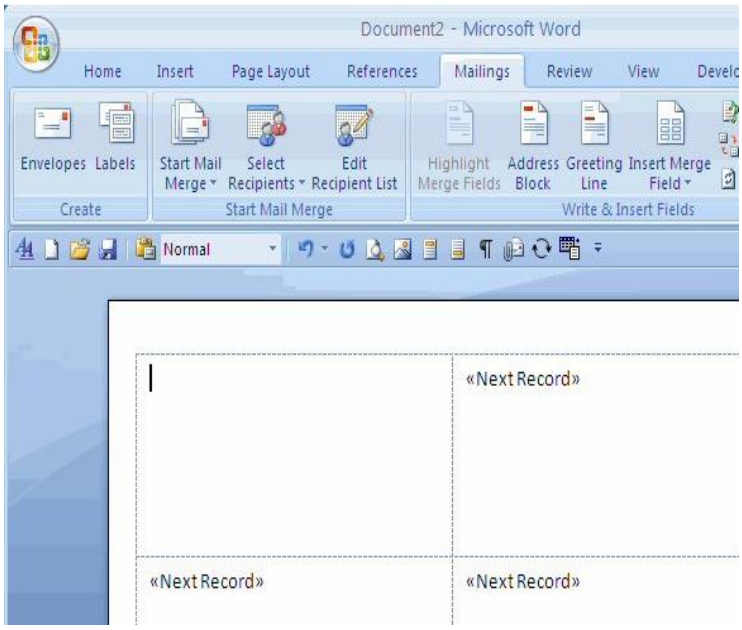


Figure 3.48: Mail Merge toolbar.

Mail Merge in Word is accomplished by the following steps:

Step 1: Set the data document type.

The data document holds the text that repeats for all merged documents. In the Mailings tab, click Start Mail Merge and select the document type.

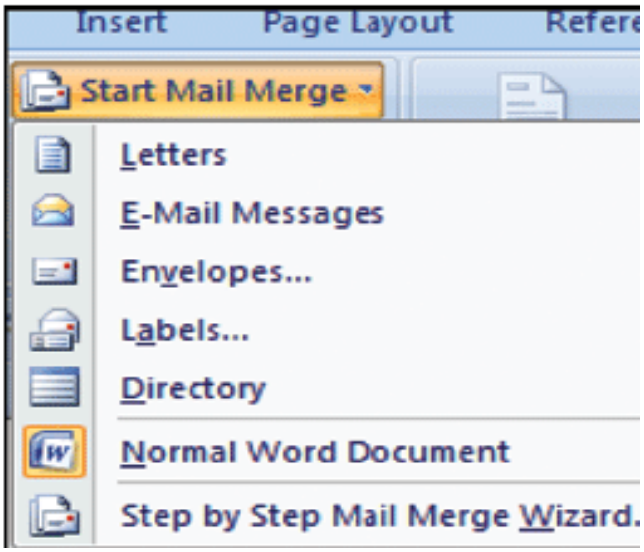


Figure 3.49: Start Mail Merge.

Word offers the following types of documents:

Letters: Use this option for composing and designing mass mailings for which only the recipient information varies. This document type is also used when sending out a form letter or invoice.

E-mail Messages: Using e-mail merge, multiple addresses in the To, Cc or Bcc fields can be avoided. Each recipient can receive a personalized e-mail.

Envelopes: Use this option for producing envelopes.

Labels: Use this option to print sheets of labels. Many addresses can be printed on the same page, in different label formats.

Directory: Use this option when printing a catalog or any other document that requires printing multiple records per page.

Step 2: Associate a data source with the document:

In the Mailings tab, choose Select Recipients.

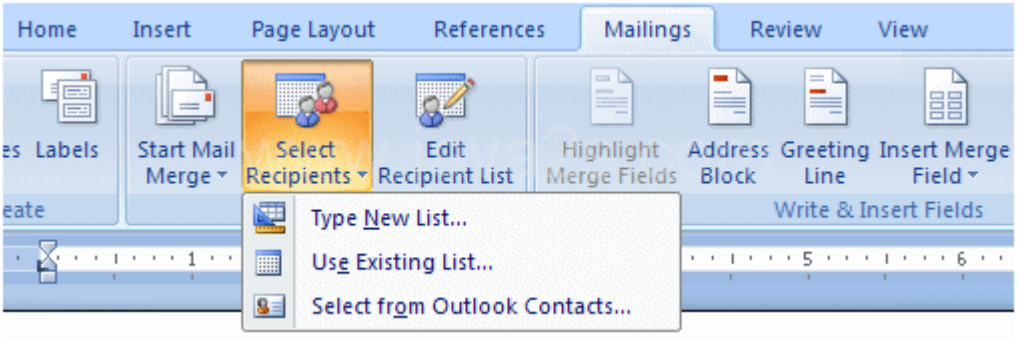


Figure 3.50: Select Recipients.

The various options are:

Use Existing List:

The Select Data Source dialog box appears when you choose this option. Navigate to and select a data source file. Typically, the data source is created in Excel or Access.

Type New List:

If you choose this, click Create, and then use the dialog box that appears to enter names and addresses. If you do not plan to use the entire database, you can use the Mail Merge Recipients dialog box, to select just the recipients you want. To open the dialog box, click Edit Recipient list in the Start Mail Merge group of the Mailings tab.

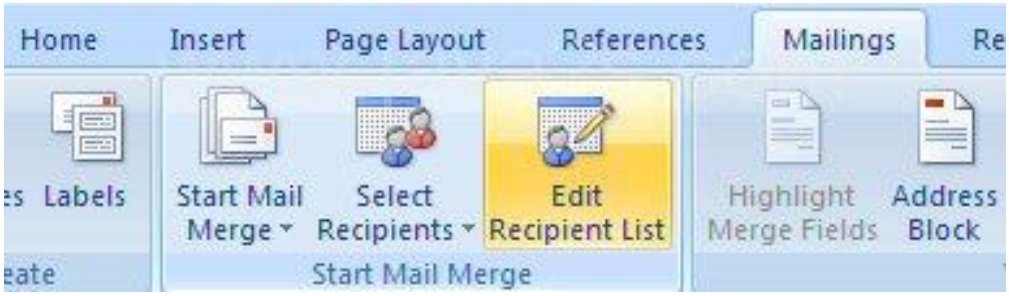


Figure 3.51: Edit recipients list.

The recipient list can be refined by Sort, Filter, Find Duplicates, Find Recipient and Validate addresses options.

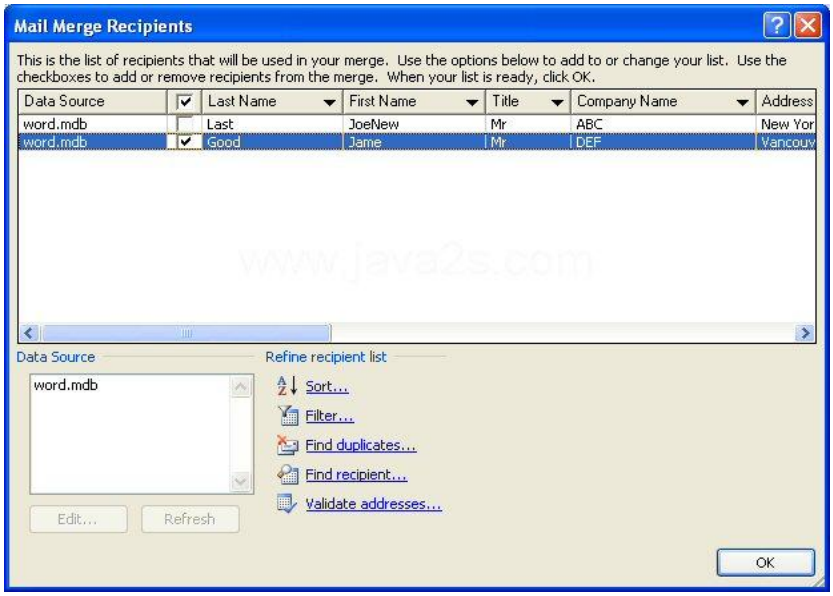


Figure 3.52: Find Recipient and Validate addresses options.

Select from Outlook Contacts.

Step 3: Design your data document by combining ordinary document features with Word merge fields.
Placeholders can be used when designing the data document for information pertaining to the intended recipient. When you are done, edit your document and substitute Merge Fields for the placeholders. To insert a merge field, position the insertion point where you want the field to appear. In the Mailings tab, choose Insert Merge Field in the Write and Insert Fields group. Click on the field you want to insert. Special sets of merge fields like Address Block and Greeting Line can be inserted to save time!



Figure 3.53: Inset Merge field.

Step 4: Preview the finished document by testing to see how it looks with different data records.
Click the Preview Results button in the Preview Results group of the Mailings tab. Navigation buttons help you to traverse through the records.

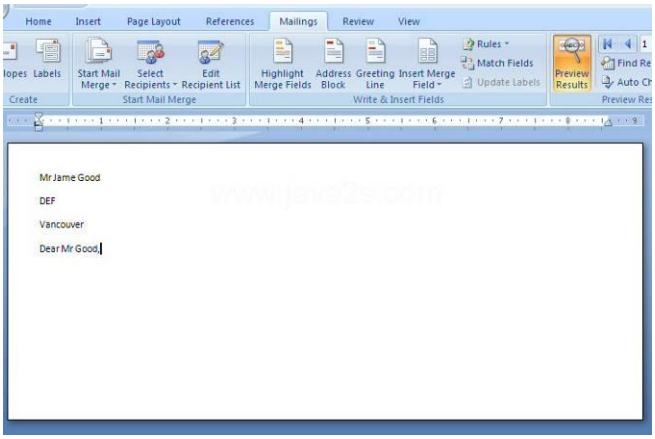


Figure 3.54: Mailings tab Navigation.

Step 5 Finish the process. Merge the data document with the data source, creating a printed result, a saved document or an e-mailed document.



Figure 3.55: Edit individual documents.

Your other option is to use the Mail Merge Wizard! In the Start Mail Merge group of the Mailings tab, click the Start Mail Merge button and choose Step by Step Mail Merge Wizard.

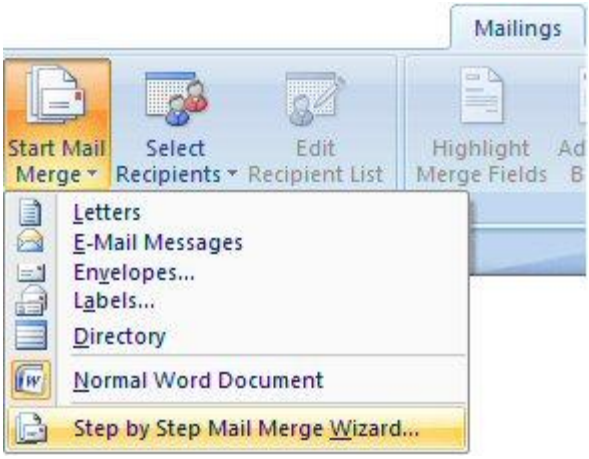


Figure 3.56: Mail Merge Wizard.

3.12 Envelops and Mailing Labels

Address labels come in different shapes and sizes from standard rectangle white ones to upscale color ovals. Usually complete with self-adhesive backs, you can use create your own sender and recipient labels with Microsoft Word 2007. Word comes with several tools for making mailing labels, including creating a sheet of the same or different addresses. By using Word's illustrations group, you can add a company logo or a graphic to your label and modify the font style.

3.12.1 Making Basic Labels

1 Open a new document in Word 2007. Click the "Mailings" tab, and then click "Labels" in the "Create" group. The "Envelopes and Labels" dialog box appears.

2 Click "Options" to choose which type of label you are using and your printer type. Click on the "Continuous-feed Printers" or the "Page Printers" radio button. Continuous feed printers are usually used with dot matrix printers, while page printers are typically laser jets or ink jets.

3 Click on the "Label Vendors" drop-down box to select Avery, Office Depot or another manufacturer that matches your labels. Scroll through the "Product Number" selections, and select your label. Click "OK."

4 Type the shipping address in the "Address" field. Select the text and right-click with your mouse. Click on "Font" to select the font and size for the label text.

5 Click on the "Full Page of the Same Label" or "Single Label" radio button. Put the label paper into your printer, and click "Print."

3.12.2 Making Personalized Labels

1 Follow similar directions as above including opening a new Word document, clicking on the "Mailings" tab, and clicking "Labels." Click "Options" and choose your label type. Click "OK" when you have completed your label selections.

2 Click "New Document" in the "Envelopes and Labels" dialog box to open a sheet of blank labels. You will see a table with the labels outlined in blue. Type individual label information in the label outline. To go to the next label, press "Tab" or use your mouse.

3 Modify the font by selecting the text and clicking on the "Home" tab. Make selections in the "Font" group. Add an image by clicking on the "Insert" tab and clicking either "Clip Art" or "Picture." To make an image

larger or smaller, drag the handles on the picture. Change text wrapping around the illustration by right-clicking on the image and selecting "Text Wrapping."
4 Print the personalized label sheet by clicking on the "Microsoft Office" button. Click "Print," and put your labels into the printer's paper tray. Click "OK" to print the labels.

3.12.3 Mailing Labels

There are two parts to a mass mailing. The list or database with contact information and the letter or address labels you wish to send out. Using the Mail Merge feature of MS Word you can create pages of mailing labels or letters addressed to each individual listed in an Excel spreadsheet.

The list

If you do not already have a list created to pull the names and addresses from you will need to create one. Click the link for instructions How to build a list in Excel 2007.

Importing a list to Word

Once you have your list of addresses ready you can start the process of transferring the information to Microsoft Word to make labels.

In Microsoft Word click on the **Mailings** tab to bring up the Mailings ribbon.

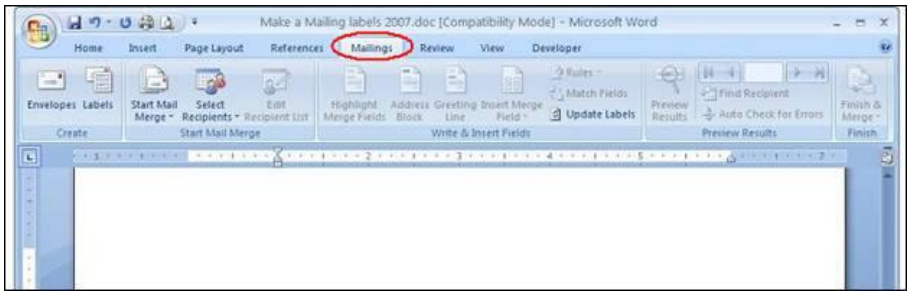


Figure 3.57: Mailings menu.

Click the Start Mail Merge button and from the drop down list click Labels… to make a page of mailing labels. The Letters option is also available to you at this point if you are planning on having a form letter with a personalized salutation.

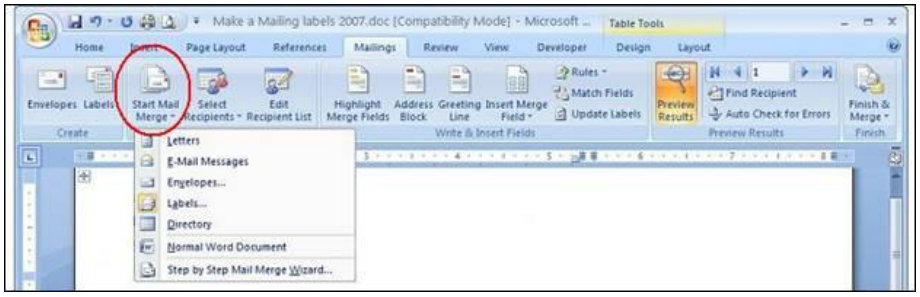


Figure 3.58: Start mail merge.

For mailing labels; From the Label Options window choose the Label vendor and product number of the labels you will be using. The label number will appear on the outside of the box of labels you purchase from the store. After your selection click **OK**.

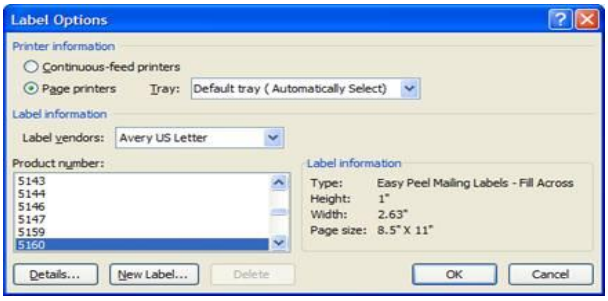


Figure 3.59: Label option.

You are now ready to get the information from the database with the contact information. Click the Select Recipients button and pick Use Existing List from the drop down menu.

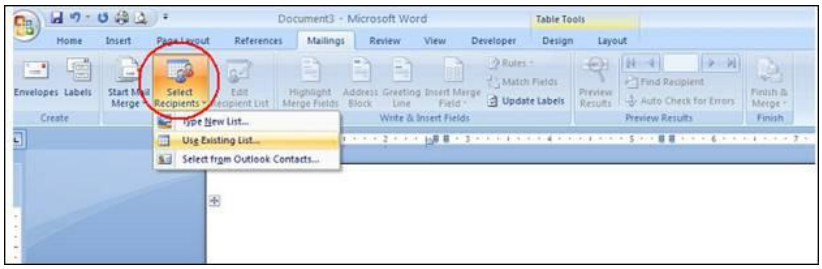


Figure 3.60: Use Existing List.

You will be prompted to select the source of your database. You will need change the Look in: field to the folder where your list is stored and then select your list and choose Open.

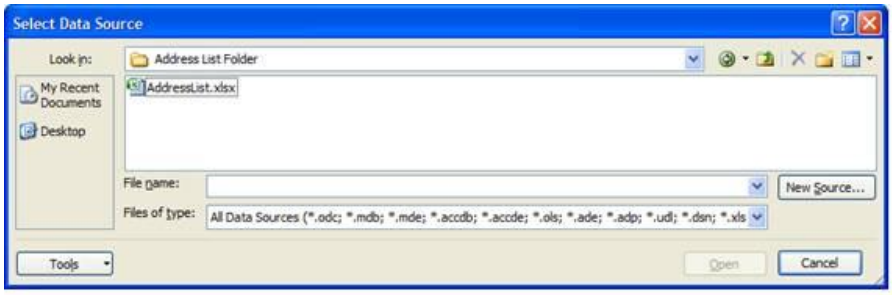


Figure 3.61: Selecting data source.

Once you have selected your Excel list you will be prompted to select the table. The table is the specific sheet that holds your information. (Most likely your table will be on sheet one if you did not specify a specific page.

Did You Know?

Box First row of data contains column headers should be check for properly built lists.

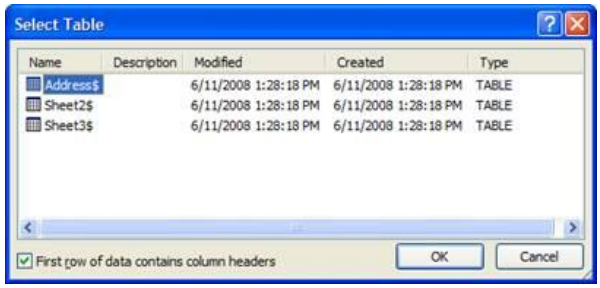


Figure 3.62: Selecting table.

3.12.4 Choose the import fields

Now Microsoft Word knows WHERE to find the information. Next you need to tell word WHAT information to put on our labels. Click on the **Insert Merge Field** button.

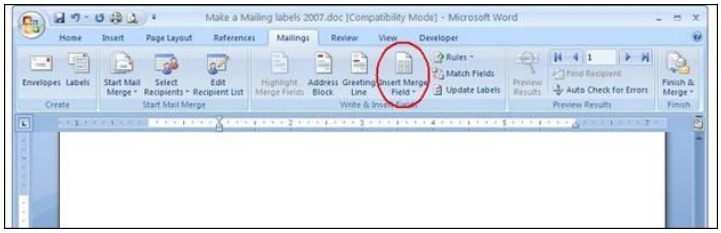


Figure 3.63: Insert Merge Field button.

Word will bring up an Insert Merge Field window. This will have a list of all the fields you have entered into your database. We are going to use First Name, Last Name, Address, City, State, and Zip. If your database has extra fields like phone number, or e-mail address we will not select those for the mailing label. To add a field to a label double click on the field name or click on the field name and then press the **Insert** button. After all the required fields have been selected you can close the window by pressing the **Close** button.



Figure 3.64: Insert button.

3.12.5 Format Fields

Your field names now appear in your Word document; however the fields are not formatted. You need to format the fields. Put a space between the First and Last name, enter after the last name to have the address appear on the second line… etc. Once the first label is formatted correctly you can then press the Update Labels button to apply the formatting to all subsequent labels.

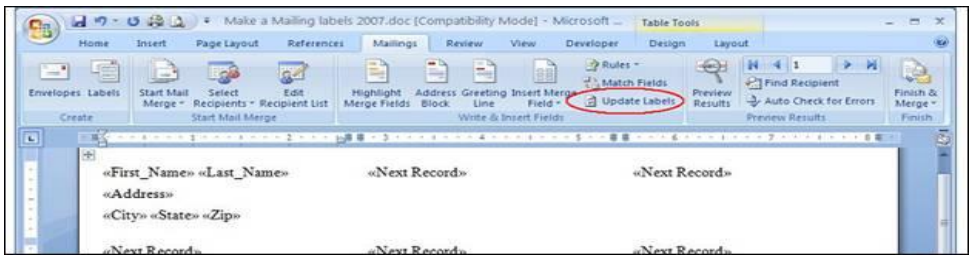


Figure 3.65: Update label.

To preview your newly created labels click the **Preview Results** button.

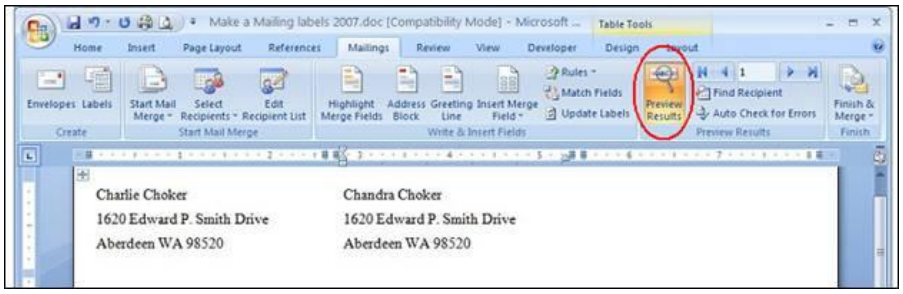


Figure 3.66: Preview Results button.

3.12.6 Merge to new document

You can now print or merge your new labels to a new document by pressing the **Finish and Merge** button. Merging your document will allow you to save and access your labels at a later date without having to access your Excel list again.

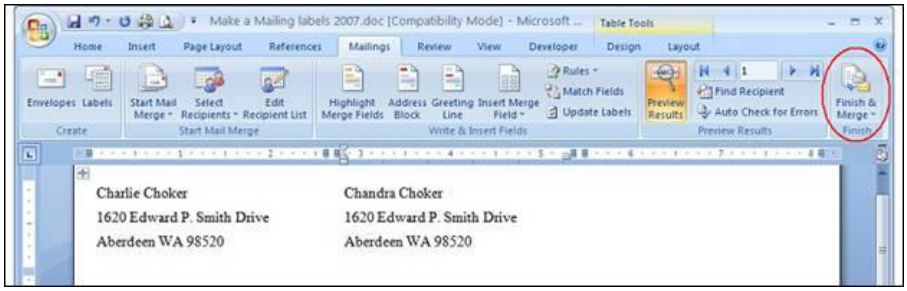


Figure 3.67: Finish and Merge button.

Self Assessment Questions

4. The new default settings are saved in the template on which the document is based. Each new document based on that template automatically uses the new margin settings.

(a) True (b) False

5. Continuous feed printers are usually used with dot matrix printers, while page printers are not typically laser jets or ink jets.

(a) True (b) False

3.13 Summary

- Smart Art is a collection of graphics you can utilize to organize information within document.
- A watermark is a translucent image that appears behind the primary text in a document.
- The existing header o
- Columns are a good way to separate sections of your document on one page r footer is now configured for odd-numbered pages only.
- Mail Merge is an automated feature of MS Word that enables you to merge a data source into a copy of a document to customize or personalize the document.
- Many header and footer designs are laid out in a table, and pressing TAB moves the cursor without inserting a tab stop.

3.14 Keywords

Bibliography: Bibliography is an AJAX-based multi-user open source on-line bibliographic data manager and bibliography tool.

Header and Footer: Headers and footers are those little identifiers that run across the top and bottom of your document, providing important background information about it.

Mail Merge: Mail merge is a software function which allows to create multiple (and potentially large numbers of) documents from a single template form and a structured data source.

Master documents: A Master Document is a document that contains a set of related documents called Sub Documents.

Page Layout: Page layout is the process of placing and arranging and rearranging text and graphics on the page to produce documents such as newsletters, brochures, books, etc.

3.15 Review Questions

1. How to check the spelling and grammar of a document?
2. What is the use of thesaurus?
3. How to create a header or footer for your documents?
4. What are uses of headers and footers in a document?
5. How to create, modify, and delete columns in your document?
6. What are tabs and indents?
7. How to insert table window on the word document?
8. What is step of mail merge in word?
9. Describe the mailing labels.

Answers for self Assessment Questions

1 (b) 2 (c) 3 (a) 4 (a) 5 (b)

3.16 Further Studies

1. http://litwict.pbworks.com/f/ADVANCEDWord+2007.pdf

4

MS Excel

STRUCTURE

4.0 Objective

4.1 Excel Working Area

4.2 Working with Excel's Windows

4.3 Working with Rows and Columns

4.4 Concepts of Workbooks and Worksheets

4.5 Moving Around a Worksheet

4.6 Creating Your First Excel Worksheet

4.7 Different Views of Worksheets

4.8 Cell Formatting

4.9 Summary

4.10 Keywords

4.11 Review Questions

4.0 Objectives

After studying this chapter, you will be able to:

- Discuss Excel working area
- Explain working with Excel's windows
- Understand working with rows and columns
- Discuss concepts of workbooks and worksheets
- Explain moving around a worksheet
- Understand creating your first excel worksheet
- Discuss different views of worksheets
- Define cell formatting

4.1 Excel Working Area

Excel, as you probably know, is the world's most widely used spreadsheet program and is part of the Microsoft Office suite. Other spreadsheet programs are available, but Excel is by far the most popular and has become the world standard. Much of the appeal of Excel is due to the fact that it is so versatile. Excel's forte, of course, is performing numerical calculations, but Excel is also very useful for non-numerical applications.

Here are just a few of the uses for Excel: Number crunching: Create budgets, analyze survey results, and perform just about any type of financial analysis you can think of.

- **Creating charts**: Create a wide variety of highly customizable charts.
- **Organizing lists**: Use the row-and-column layout to store lists efficiently.
- **Accessing other data**: Import data from a wide variety of sources.
- **Creating graphics and diagrams**: Use Shapes and the new SmartArt to create professional-looking diagrams.
- **Automating complex tasks**: Perform a tedious task with a single mouse click with Excel's macro capabilities.

What's New in Excel 2007?

If you have used a previous version of Excel, this section is for you. Here you will find a quick overview of what is new and what is changed in Excel 2007.

4.2.1 A New User Interface

The first thing you notice about Excel 2007 is its new look. The time-honoured menu-and-toolbar user interface has been scrapped and replaced with a new "tab-and-ribbon" interface. Although the new interface kind of resembles menus and toolbars, you will find that it is radically different.

Long-time Excel users have probably noticed that, with each new version, the menu system has gotten increasingly complicated. In addition, the number of toolbars had become almost overwhelming. After all, every new feature must have a way to be accessed. In the past, access meant adding more items to the menus and building new toolbars. The Microsoft designers set out to solve the problem, and the new Ribbon interface is their solution.

Did You Know?

Having more rows and columns does not mean that you can actually use them all. If you attempted to fill up all cells in a worksheet, you would soon run out of memory.

Table 4.1: By the Numbers: Excel 2003 versus Excel 2007

	Excel 2003	**Excel 2007**
Number of rows	65,536	1,048,576
Number of columns	256	16,384
Amount of memory used by Windows	1GB	Maximum allowed
Number of colours	56	4.3 billion
Number of conditional formats per cell	3	Unlimited
Number of levels of sorting	3	64
Number of levels of undo	16	100
Number of items shown in the Auto-Filter dropdown	1,000	10,000
The total number of characters that can display in a cell	1,000	2,000
Number of unique styles in a workbook	4,000	64,000
Maximum number of characters in a formula	1,000	8,000
Number of levels of nesting in a formula	7	64
Maximum number of function arguments	30	255

4.2.2 New File Formats

Over the years, Excel is XLS file format has become an industry standard. Excel 2007 still supports that format, but it now uses new default "open" file formats that are based on XML (Extensible Markup Language).

4.2.3 Worksheet Tables

Excel, of course, has always been able to deal with tables. A table is just a rectangular range of cells that (usually) contain column headers. The designers of Excel 2007 realized that such tables are widely used in Excel, and they have taken the concept to a new level. Working with tables is easier than ever.

Once you designate a particular range to be a table (using the Insert⇨Tables⇨Table command), Excel provides you with some very efficient tools that work with the table. For example:

- You can apply attractive formatting with a single click.
- You can easily insert summary formulas in the table's total row.
- If each cell in a column contains the same formula, you can edit one of the formulas, and the others change automatically.
- You can easily toggle the display of the table's the header row and totals row.
- Removing duplicate entries is easy.
- AutoFiltering and sorting options have been expanded.
- If you create a chart from a table, the chart will always reflect the data in the table—even if you add new rows.
- If you scroll a table downwards so that the header row is no longer visible, the column headers now display where the worksheet column letters would be.

4.3 Working with Excel's Windows

An Excel workbook file can hold any number of sheets, and these sheets can be either worksheets (sheets consisting of rows and columns) or chart sheets (sheets that hold a single chart). A worksheet is what people usually think of when they think of a spreadsheet. You can open as many Excel workbooks as necessary at the same time.

One of the windows is minimized and appears near the lower-left corner of the screen. (When a workbook is minimized, only its title bar is visible.) Worksheet windows can overlap, and the title bar of one window is a different colour. That is the window that contains the active workbook.

The workbook windows that Excel uses work much like the windows in any other Windows program. Each window has three buttons at the right side of its title bar. From left to right, they are Minimize, Maximize (or Restore), and Close. When a workbook window is maximized, the three buttons appear directly below Excel's title bar.

Excel's windows can be in one of the following states:

- **Maximized:** Fills Excel's entire workspace. A maximized window does not have a title bar, and the workbook's name appears in Excel's title bar. To maximize a window, click its Maximize button.
- **Minimized:** Appears as a small window with only a title bar. To minimize a window, click its Minimize button.
- **Restored:** A non-maximized size. To restore a maximized or minimized window, click its Restore button.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Work with more than one workbook simultaneously and move, resize, and switch among the workbook windows.

……..……

……..……

……...

4.3.1 Moving and Resizing Windows

To move a window, make sure that it is not maximized. Then drag its title bar with your mouse.

To resize a window, drag any of its borders until it is the size that you want it to be. When you position the mouse pointer on a window's border, the mouse pointer changes to a double-sided arrow, which lets know that you can now drag to resize the window. To resize a window horizontally and vertically at the same time, drag any of its corners.

If you want all your workbook windows to be visible (that is, not obscured by another window), you can move and resize the windows manually, or you can let Excel do it for you. Choosing View➪Window➪Arrange All displays the Arrange Windows dialog box. This dialog box has four window-arrangement options. Just select the one that you want and click OK. Windows that are minimized are not affected by this change.

4.4 Working with Rows and Columns

This section discusses worksheet operations that involve rows and columns. Rows and columns make up an Excel worksheet. Every worksheet has exactly 1,048,576 rows and 16,384 columns, and these values cannot be changed.

If you open a workbook that was created in a previous version of Excel, the workbook is opened in "compatibility mode". These workbooks have 65,536 rows and 256 columns. To increase the number of rows and columns, save the workbook as an Excel 2007 XLSX file and then reopen it.

4.4.1 Inserting Rows and Columns

Although the number of rows and columns in a worksheet is fixed, you can still insert and delete rows and columns if you need to make room for additional information. These operations do not change the number of rows or columns. Rather, inserting a new row moves down the other rows to accommodate the new row.

The last row is simply removed from the worksheet if it is empty. Inserting a new column shifts the columns to the right, and the last column is removed if it is empty.

If the last row is not empty, you cannot insert a new row. Similarly, if the last column contains information, Excel does not let you insert a new column. Attempting to add a row or column displays a warning dialog box shown. Click OK and then move or eliminate the contents of the nonblank cells to continue.

To insert a new row or rows, you can use any of these techniques:

- Select an entire row or multiple rows by clicking the row numbers in the worksheet border. Right click and choose Insert from the shortcut menu.
- Move the cell pointer to the row that you want to insert and then choose Home➪Cells➪Insert➪Insert Sheet Rows. If you select multiple cells in the column, Excel inserts additional rows that correspond to the number of cells selected in the column and moves the rows below the insertion down.

4.4.2 Deleting Rows and Columns

You may also want to delete rows or columns in a worksheet. For example, your sheet may contain old data that is no longer needed.

To delete a row or rows, use either of these methods:

- Select an entire row or multiple rows by clicking the row numbers in the worksheet border. Right click and choose Delete from the shortcut menu.
- Move the cell pointer to the row that you want to delete and then choose Home➪Cells➪Delete➪Delete Sheet Rows. If you select multiple cells in the column, Excel deletes all rows in the selection.

Deleting columns works in a similar way. If you discover that you accidentally deleted a row or column, select undo from the Quick Access Toolbar (or press Ctrl+Z) to undo the action.

4.4.3 Hiding Rows and Columns

In some cases, you may want to hide particular rows or columns. Hiding rows and columns may be useful if you do not want users to see certain information or if you need to print a report that summarizes the information in the worksheet without showing all the details.

To hide rows or columns in your worksheet, select the row or rows that you want to hide by clicking in the row or column header. Then right-click and choose Hide from the shortcut menu. Or, you can use the commands on the Home➪Cells➪Format drop-down list.

You also can drag the row or column's border to hide the row or column. You must drag the border in the row or column heading. Drag the bottom border of a row upward or the border of a column to the left.

A hidden row is actually a row with its height set to zero. Similarly, a hidden column has a column width of zero. When you use the arrow keys to move the cell pointer, cells in hidden rows or columns are skipped.

Unhiding a hidden row or column can be a bit tricky because selecting a row or column that is hidden is difficult. The solution is to select the columns or rows that are adjacent to the hidden column or row. (Select at least one column or row on either side.) Then right-click and choose Unhide. For example, if column G is hidden, select columns F and H.

Another method is to choose Home➪Find and Select➪Go To (or its F5 equivalent) to select a cell in a hidden row or column. For example, if column A is hidden, you can press F5 and specify cell A1 (or any other cell in column A) to move the cell pointer to the hidden column. Then you can choose Home➪Cells➪Format➪Hide and Unhide➪Unhide Columns.

4.4.4 Enhanced Conditional Formatting

Conditional formatting refers to the ability to format a cell based on its value. Conditional formatting makes it easy to highlight certain values so that they stand out visually. For example, you may set up conditional formatting so that if a formula returns a negative value, the cell background displays green.

In the past, a cell could have at most three conditions applied. With Excel 2007, you can format a cell based on an unlimited number of conditions. But that is the least of the improvements. Excel 2007 provides a number of new data visualizations: data bars, colour scales, and icon sets.

4.4.5 SmartArt

Excel 2007 still includes a wide assortment of Shapes that you can use to create visual diagrams, such as flow charts, org charts, or diagrams that depict relationships. But the new SmartArt feature is a much better tool for such tasks. You can quickly add shadows, reflection, glow, and other special effects.

4.4.6 Formula AutoComplete

Entering formulas in Excel 2007 can be a bit less cumbersome, thanks to the new Formula AutoComplete feature. When you begin typing a formula, Excel displays a continually updated drop-down list of matching items, including a description of each item. When you see the item you want, press Tab to enter it into your formula. The items in this list consist of functions, defined names, and table references.

4.4.7 Other New Features

Other new features in Excel 2007 worth noting are:

- **Trust Centre:** Protecting yourself from malicious macros is a bit easier with Excel 2007. For example, you can disable all macros, except those in workbooks that are stored in trusted locations on your computer.
- **PDF add-in:** You can create an industry-standard Adobe PDF file directly from Excel using an add-in available from Microsoft. Search Excel's Help system for "PDF" to learn more.
- **Improved zooming:** Use the Zoom control or Zoom slider on the right side of the status bar to quickly zoom in or zoom out on your worksheet.
- **More control over the status bar:** You can now control the type of information that appears in the status bar.
- **Colour Schemes:** Change the appearance of Excel by applying one of three colour schemes that ship with Excel (Blue, Silver, or Black).
- **Resizable formula bar:** When editing lengthy formulas, you can increase the height of the formula bar so that it does not obscure your worksheet. Just click and drag on the bottom border of the formula bar.
- **Lots of new templates:** Why reinvent the wheel? Choose Office Button➪New, and you can choose from a variety of templates. One of them may be exactly (or at least close) to what you need.

Self Assessment Questions

1. Sheets that hold a single chart is known as.....

(a) chart sheets (b) charge sheet

(c) worksheet (d) None of these.

2. Worksheet has exactly 1,048,576 rows and..... columns.

(a) 15,684 (b) 16,384

(c) 12,587 (d) None of these.

3. Workbooks have ..... rows and 256 columns.

(a) 48,576 (b) 16,384

(c) 65,536 (d) None of these.

4. XML stands for.....

(a) Extensible Markup Language (b) Xtensible Markup Language

(c) Extensible Memory Learning (d) None of these.

5. By default, Excel 2007 workbooks use an file extension.

(a) XLS (b) XML (c) EML (d) XLSX.

4.5 Concepts of Workbooks and Worksheets

The work you do in Excel is performed in a workbook file, which appears in its own window. You can have as many workbooks open as you need. By default, Excel 2007 workbooks use an XLSX file extension.

Each workbook is comprised of one or more worksheets, and each worksheet is made up of individual cells. Each cell contains a value, a formula, or text. A worksheet also has an invisible draw layer, which holds charts, images, and diagrams. Each worksheet in a workbook is accessible by clicking the tab at the bottom of the workbook window. In addition, workbooks can store chart sheets. A chart sheet displays a single chart and is also accessible by clicking a tab.

Newcomers to Excel are often intimidated by all the different elements that appear within Excel's window.

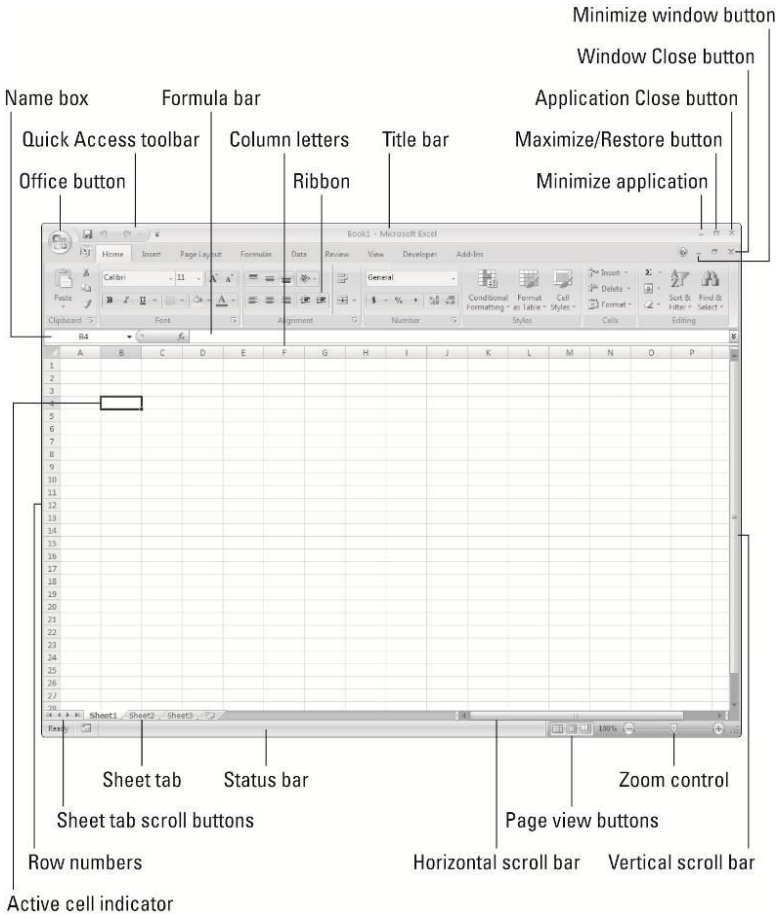


Figure 4.1: The Excel screen has many useful elements that you will use often.

Table 4.2 Parts of Excel Screen

Parts of the Excel Screen That You Need to Know

Name	Description
Active cell indicator	This dark outline, also called the cell selector, indicates the currently active cell (one of the 17,179,869,184 cells on each worksheet).
Application close button	Clicking this button closes Excel.
Window close button	Clicking this button closes the active workbook window.
Column letters	Letters range from A to XFD — one for each of the 16,384 columns in the worksheet. You can click a column heading to select an entire column of cells.
Office button	This button leads to lots of commands for working with your document, or Excel in general.
Formula bar	When you enter information or formulas into Excel, they appear in this line.
Horizontal scrollbar	Enables you to scroll the sheet horizontally.
Maximize/Restore button	Clicking this button increases the workbook window's size to fill your monitor's workspace. If the window is already maximized, clicking this button Restores Excel's window down to its previous window size so that it no longer fills the entire screen.
Minimize application button	Clicking this button minimizes Excel's window down to the Windows taskbar.
Minimize window button	Clicking this button minimizes the workbook window.
Name box	Displays the active cell address or the name of the selected cell, range, or object.
Page view buttons	Change the way the worksheet is displayed by clicking one of these buttons.
Quick Access Toolbar	A toolbar that you customize to hold commonly-used commands
Ribbon	The main location to find Excel's commands. Clicking a tab changes the Ribbon buttons that appear.
Row numbers	Numbers range from 1 to 1,048,576 — one for each row in the worksheet. You can click a row number to select an entire row of cells.
Sheet tabs	Each of these notebook-like tabs represents a different sheet in the workbook. A workbook can have any number of sheets, and each sheet has its name displayed in a sheet tab. By default, each new workbook that you create contains three sheets. Add a new sheet by clicking the Insert Worksheet button (which is displayed after the last sheet tab).
Sheet tab scroll buttons	These buttons let you scroll the sheet tabs to display tabs that aren't visible.
Status bar	This bar displays various messages as well as the status of the Num Lock, Caps Lock, and Scroll Lock keys on your keyboard. It also shows summary information about the range of cells that is selected. Right-click the status bar to change the information that's displayed
Title bar	All Windows programs have a title bar, which displays the name of the program and the name of the current workbook and also holds some control buttons that you can use to modify the window.
Vertical scrollbar	Lets you scroll the sheet vertically.
Zoom control	A slider control that lets you zoom your worksheet in and out.

4.6 Moving Around a Worksheet

This section describes various ways to navigate through the cells in a worksheet. Every worksheet consists of rows (numbered 1 through 1,048,576) and columns (labelled A through XFD). After column Z comes column AA, which is followed by AB, AC, and so on. After column AZ comes BA, BB, and so on. After column ZZ is AAA, AAB, and so on.

The intersection of a row and a column is a single cell. At any given time, one cell is the active cell. You can identify the active cell by its darker border. Its address (its column letter and row number) appears in the Name box. Depending on the technique that you use to navigate through a workbook, you may or may not change the active cell when you navigate.

Notice that the row and column headings of the active cell appear in different colours to make it easier to identify the row and column of the active cell.

4.6.1 Navigating with Keyboard

As you probably already know, you can use the standard navigational keys on your keyboard to move around a worksheet. These keys work just as you'd expect: The down arrow moves the active cell down one row, the right arrow moves it one column to the right, and so on. PgUp and PgDn move the active cell up or down one full window. (The actual number of rows moved depends on the number of rows displayed in the window.)

The Num Lock key on your keyboard controls how the keys on the numeric keypad behave. When Num Lock is on, Excel displays Num Lock in the status bar, and the keys on your numeric keypad generate numbers.

Most keyboards have a separate set of navigational (arrow) keys located to the left of the numeric keypad. The state of the Num Lock key does not affect these keys.

Table 4.3 Movement keys of excel worksheet

Excel's Worksheet Movement Keys

Key	Action
Up arrow	Moves the active cell up one row
Down arrow	Moves the active cell down one row
Left arrow or Shift+Tab	Moves the active cell one column to the left
Right arrow or Tab	Moves the active cell one column to the right
PgUp	Moves the active cell up one screen
PgDn	Moves the active cell down one screen
Alt+PgDn	Moves the active cell right one screen
Alt+PgUp	Moves the active cell left one screen
Ctrl+Backspace	Scrolls the screen so that the active cell is visible
Up arrow*	Scrolls the screen up one row (active cell does not change)
Down arrow*	Scrolls the screen down one row (active cell does not change)
Left arrow*	Scrolls the screen left one column (active cell does not change)
Right arrow*	Scrolls the screen right one column (active cell does not change)

* With Scroll Lock on

4.6.2 Navigating with Mouse

To change the active cell by using the mouse, click another cell; it becomes the active cell. If the cell that you want to activate is not visible in the workbook window, you can use the scrollbars to scroll the window in any

direction. To scroll one cell, click either of the arrows on the scrollbar. To scroll by a complete screen, click either side of the scroll bar's scroll box. You also can drag the scroll box for faster scrolling.

Press Ctrl while you use the mouse wheel to zoom the worksheet. If you prefer to use the mouse wheel to zoom the worksheet without pressing Ctrl, choose Office Button→Excel Options and select the Advanced section. Place a check mark next to the Zoom On Roll With Intellimouse checkbox.

Using the scroll bars or scrolling with your mouse does not change the active cell. It simply scrolls the worksheet.

Caution

To change the active cell, you must click a new cell after scrolling.

Did You Know?

If your mouse has a wheel on it, you can use the mouse wheel to scroll vertically. Also, if you click the wheel and move the mouse in any direction, the worksheet scrolls automatically in that direction. The more you move the mouse, the faster the scrolling.

4.7 Creating Your First Excel Worksheet

This section presents an introductory hands-on session with Excel. If you have not used Excel, you may want to follow along on your computer to get a feel for how this software works.

4.7.1 Getting Started on Your Worksheet

Start Excel and make sure that you have an empty workbook displayed. To create a new, blank workbook, press Ctrl+N.

The sales projection will consist of two columns of information. Column A will contain the month names, and column B will store the projected sales numbers. You start by entering some descriptive titles into the worksheet. Here is how to begin:

1. Move the cell pointer to cell A1 by using the direction keys. The Name box displays the cell's address.
2. Enter Month into cell A1**.** Just type the text and then press Enter. Depending on your setup, Excel either moves the cell pointer to a different cell, or the pointer remains in cell A1. (You can change this behaviour in the advanced category of the Excel Options dialog box. Appendix A, "Customizing Office," explains how to find and work with options.)
3. Move the cell pointer to B1, type Projected Sales, and press Enter.

4.7.2 Filling in the Month Names

In this step, you enter the month names in column A.

1. Move the cell pointer to A2 and type Jan (an abbreviation for January). At this point, you can enter the other month name abbreviations manually, but we will let Excel do some of the work by taking advantage of the AutoFill feature.
2. Make sure that cell A2 is selected. Notice that the active cell is displayed with a heavy outline. At the bottom-right corner of the outline, you will see a small square known as the *fill handle.* Move your mouse pointer over the fill handle, click, and drag down until you have highlighted from A2 down to A13.
3. Release the mouse button, and Excel will automatically fill in the month names.

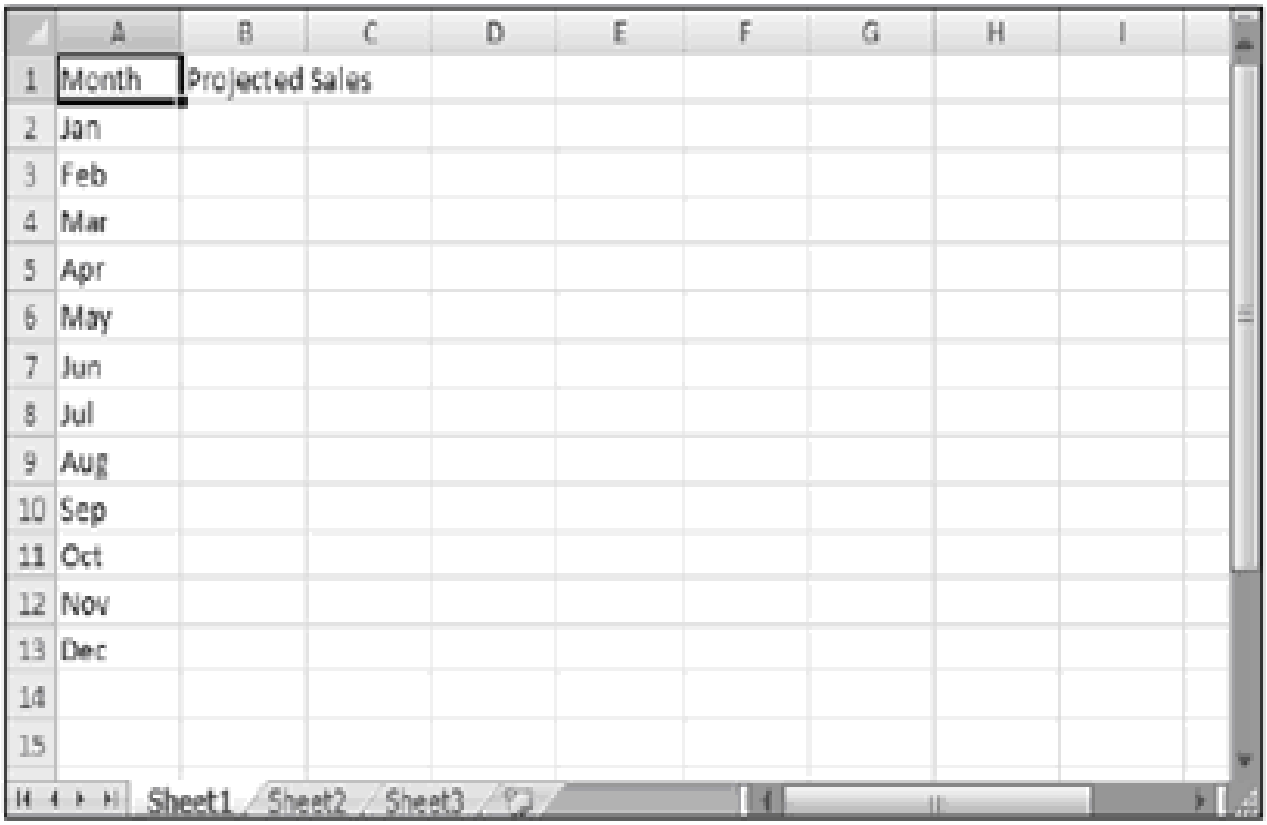


Figure 4.2: Worksheet, after entering the column headings and month names.

4.6.3 Entering the Sales Data

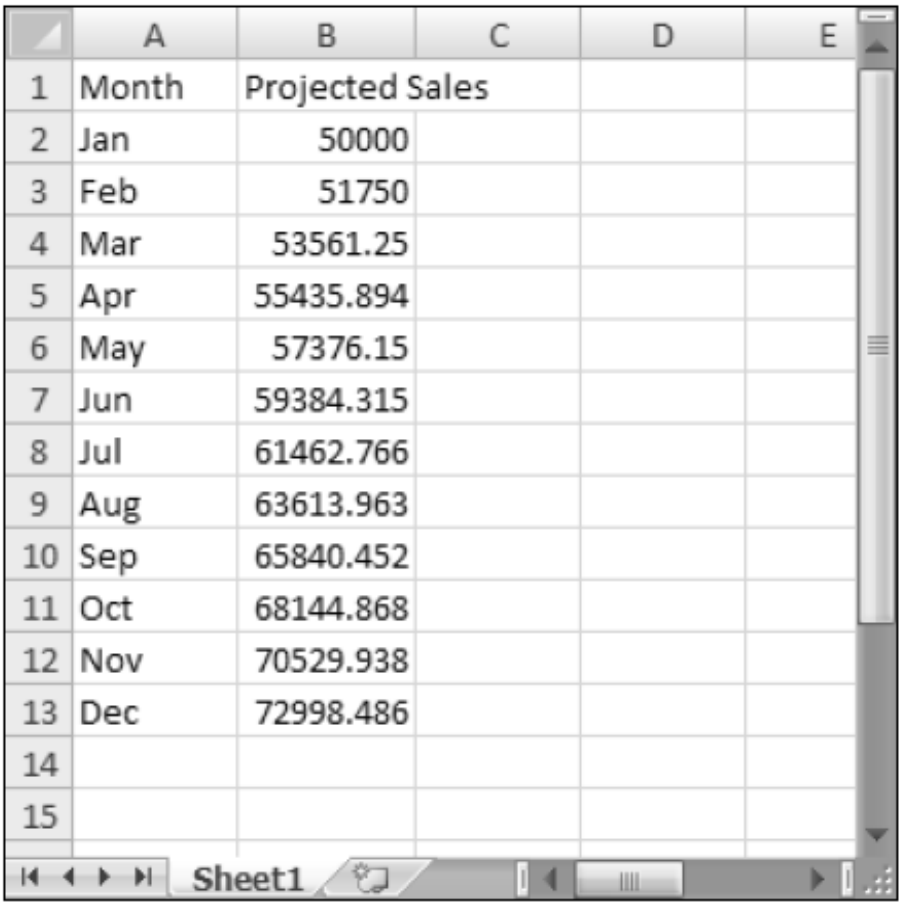
Next, you provide the sales projection numbers in column B. Assume that January's sales are projected to be 50,000, and that sales will increase by 3.5% in each of the subsequent months.

1. Move the cell pointer to B2 and type 50000, the projected sales for January.

2. To enter a formula to calculate the projected sales for February, move to cell B3 and enter the following: =B2*103.5%**.** When you press Enter, the cell will display 51750. The formula returns the contents of cell B2, multiplied by 103.5%. In other words, February sales are projected to be 3.5% greater than January sales.

3. The projected sales for subsequent months will use a similar formula. But rather than retyping the formula for each cell in column B, once again take advantage of the AutoFill feature. Make sure that cell B3 is selected. Click the cell's fill handle, drag down to cell B13, and release the mouse button.

At this point, your worksheet should resemble the one shown in Figure 5.3. Keep in mind that, except for cell B2, the values in column B are calculated with formulas. To demonstrate, try changing the projected sales value for the initial month, January (in cell B2). You will find that the formulas recalculate and return different values. But these formulas all depend on the initial value in cell B2.



	A	B	C	D	E
1	Month	Projected Sales			
2	Jan	50000			
3	Feb	51750			
4	Mar	53561.25			
5	Apr	55435.894			
6	May	57376.15			
7	Jun	59384.315			
8	Jul	61462.766			
9	Aug	63613.963			
10	Sep	65840.452			
11	Oct	68144.868			
12	Nov	70529.938			
13	Dec	72998.486			
14					
15					

Figure 4.3: Your worksheet, after creating the formulas.

4.7.4 Formatting the Numbers

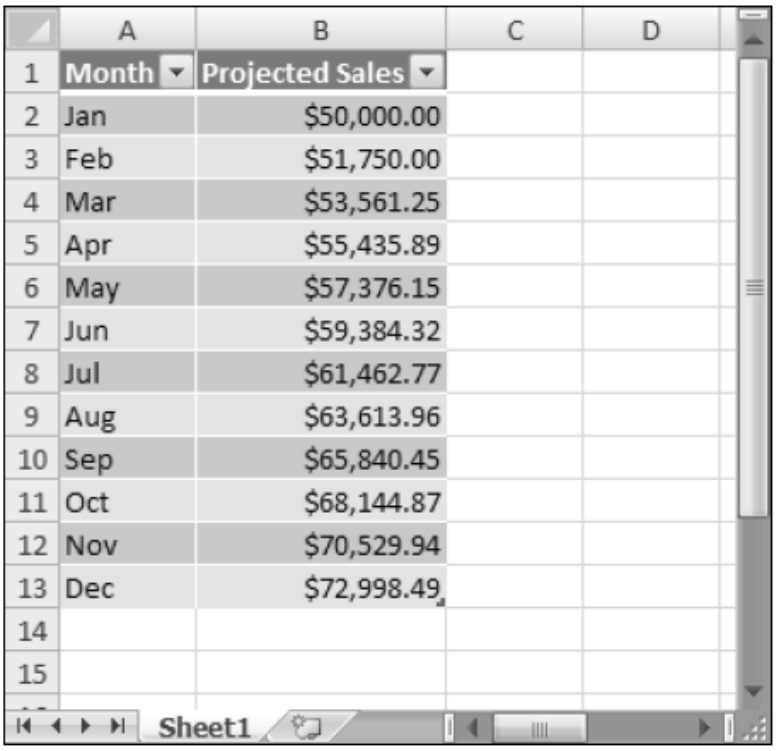
The values in the worksheet are difficult to read because they are not formatted. In this step, you apply a number format to make the numbers easier to read and more consistent in appearance:

1. Select the numbers by clicking cell B2 and dragging down to cell B13.
2. Choose Home➪Number, click the drop-down Number Format control (it initially displays General), and select Currency from the list. The numbers now display with a currency symbol and two decimal places. Much better!

4.7.5 Making Worksheet Look Bit Fancier

At this point, you have a functional worksheet—but it could use some help in the appearance department. Converting this range to an "official" (and attractive) Excel table is a snap:

1. Move to any cell within the range.
2. Choose Insert➪Tables➪Table. Excel displays its Create Table dialog box to make sure that it guessed the range properly.
3. Click OK to close the Create Table dialog box. Excel applies its default table formatting and also displays its Table Tools➪Design contextual tab.
4. If you do not like the default table style, just select another one from the Table Tools ➪Design ➪Table Styles group.



	A	B	C	D
1	Month	Projected Sales		
2	Jan	$50,000.00		
3	Feb	$51,750.00		
4	Mar	$53,561.25		
5	Apr	$55,435.89		
6	May	$57,376.15		
7	Jun	$59,384.32		
8	Jul	$61,462.77		
9	Aug	$63,613.96		
10	Sep	$65,840.45		
11	Oct	$68,144.87		
12	Nov	$70,529.94		
13	Dec	$72,998.49		
14				
15				

Figure 4.4: Your worksheet, after converting the range to a table.

4.7.6 Summing the Values

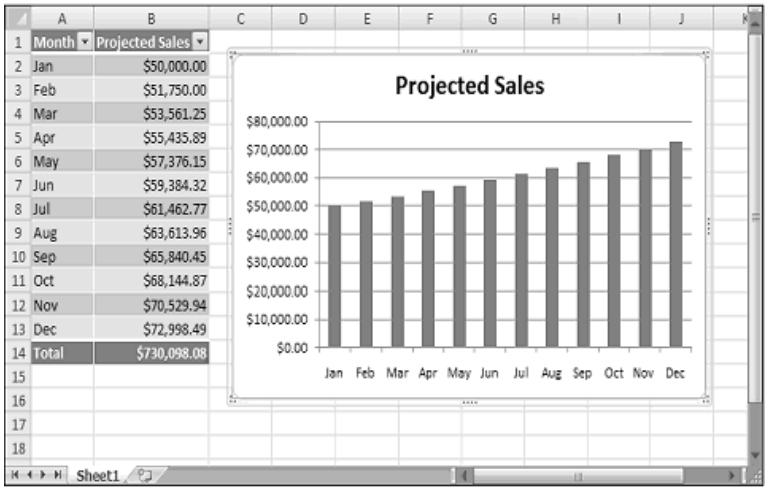
The worksheet displays the monthly projected sales, but what about the total sales for the year? Because this range is a table, it is simple:

1. Activate any cell in the table.
2. Choose Table Tools➪Design➪Table Style Options➪Totals Row. Excel automatically adds a new row to the bottom of your table, including a formula that calculates the total of the Projected Sales column.
3. If you'd prefer to see a different summary formula (for example, average), click cell B14 and choose a different summary formula from the drop-down list.

4.7.7 Creating a Chart

How about a chart that shows the projected sales for each month?

1. Activate any cell in the table.
2. Choose Insert➪Charts➪Column and then select one of the 2-D column chart types. Excel inserts the chart in the center of your screen.
3. To move the chart to another location, click its border and drag it.
4. To change the appearance and style of the chart, use the commands in the Chart Tools context tab.



Month	Projected Sales
Jan	$50,000.00
Feb	$51,750.00
Mar	$53,561.25
Apr	$55,435.89
May	$57,376.15
Jun	$59,384.32
Jul	$61,462.77
Aug	$63,613.96
Sep	$65,840.45
Oct	$68,144.87
Nov	$70,529.94
Dec	$72,998.49
Total	$730,098.08

Figure 4.5 The table and chart.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Create a simple monthly sales projection table along with a chart in MS Excel.

……..……

……..……

……...

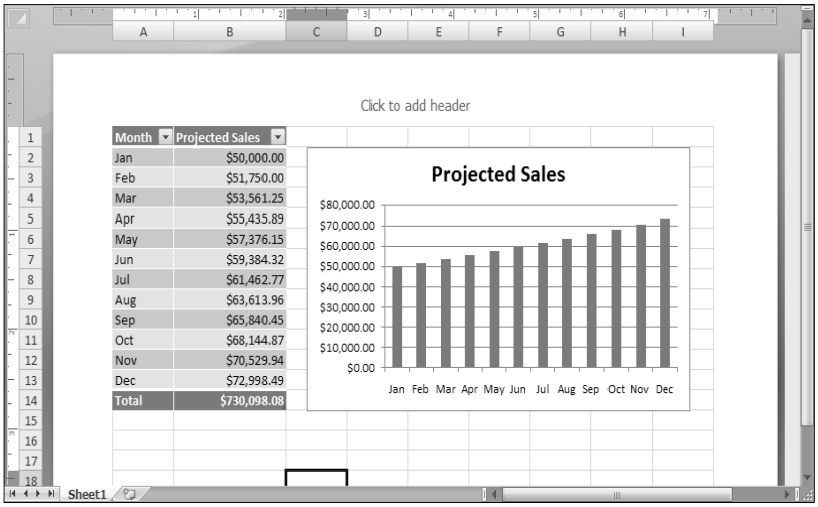
4.7.8 Printing Worksheet

Printing your worksheet is very easy (assuming that you have a printer attached and that it works properly).

1. First, make sure that the chart is not selected. If a chart is selected, it will print on a page by itself. To deselect the chart, just press Esc or click any cell.

2. To make use of Excel's handy new page layout view, click the Page Layout View button on the right side of the status bar. Excel will then display the worksheet page by page so that you can easily see how your printed output will look. For example, you can tell immediately if the chart is too wide to fit on one page. If the chart is too wide, click and drag its lower-right corner to resize it.

3. When you are ready to print, choose Office Button➪Print➪Quick Print.



Month	Projected Sales
Jan	$50,000.00
Feb	$51,750.00
Mar	$53,561.25
Apr	$55,435.89
May	$57,376.15
Jun	$59,384.32
Jul	$61,462.77
Aug	$63,613.96
Sep	$65,840.45
Oct	$68,144.87
Nov	$70,529.94
Dec	$72,998.49
Total	$730,098.08

Figure 4.6: Viewing the worksheet in Page Layout mode.

4.7.9 Saving Workbook

Until now, everything you have done has occurred in your computer's memory. If the power should fail, all may be lost—unless Excel's AutoRecover feature happened to kick in. It is time to save your work to a file on your hard drive.

1. Click the Save button on the Quick Access Toolbar. (This button looks like an old-fashioned floppy disk.) Because the workbook has not been saved yet and still has its default name, Excel responds with the Save As dialog box.
2. In the box labeled File Name, enter a name such as Monthly Sales Projection, and then click Save or press Enter. Excel saves the workbook as a file. The workbook remains open so that you can work with it some more.

Did You Know?

By default, Excel saves a copy of your work automatically every 10 minutes. To adjust this setting (or turn it off), use the Save tab of the Excel Options dialog box. To display this dialog box, choose Office Button➪Excel Options. However, you should never rely on Excel's AutoRecover feature. Saving your work frequently is a good idea.

If you have followed along, you may have realized that creating this workbook was not at all difficult. But, of course, you have barely scratched the surface. The remainder of this book will cover these tasks (and many, many more) in much greater detail.

4.8 Different Views of Worksheets

4.8.1 Use of Text Entries

Most worksheets also include text in their cells. You can insert text to serve as labels for values, headings for columns, or instructions about the worksheet. Text is often used to clarify what the values in a worksheet mean.

Text that begins with a number is still considered text. For example, if you type 12 Employees into a cell, Excel considers the entry to be text rather than a value. Consequently, you cannot use this cell for numeric calculations. If you need to indicate that the number 12 refers to employees, enter 12 into a cell and type Employees into the cell to the right.

4.8.2 Use of Formulas

Formulas are what make a spreadsheet a spreadsheet. Excel enables you to enter powerful formulas that use the values (or even text) in cells to calculate a result. When you enter a formula into a cell, the formula's result appears in the cell. If you change any of the values used by a formula, the formula recalculates and shows the new result.

Formulas can be simple mathematical expressions, or they can use some of the powerful functions that are built into Excel. The worksheet contains values, text, and formulas. The cells in column A contain text. Column B contains four values and two formulas. The formulas are in cells B6 and B10. Column D, for reference, shows the actual contents of the cells in column B.

4.9 Cell Formatting

4.9.1 Modifying Cell Contents

After you enter a value or text into a cell, you can modify it in several ways:

- Erase the cell's contents
- Replace the cell's contents with something else
- Edit the cell's contents

4.9.2 Erasing the Contents of a Cell

To erase the contents of a cell, just click the cell and press **Delete**. To erase more than one cell, select all the cells that you want to erase and then press **Delete**. Pressing Delete removes the cell's contents but does not remove any formatting (such as bold, italic, or a different number format) that you may have applied to the cell.

For more control over what gets deleted, you can choose Home⇨Editing⇨Clear. This command's dropdown list has four choices:

- **Clear All:** Clears everything from the cell
- **Clear Formats:** Clears only the formatting and leaves the value, text, or formula
- **Clear Contents:** Clears only the cell's contents and leaves the formatting
- **Clear Comments:** Clears the comment (if one exists) attached to the cell

Caution

Clearing formats does not clear the background colours in a range that has been designated as a table, unless you have replaces the table style background colours manually.

4.9.3 Replacing the Contents of a Cell

To replace the contents of a cell with something else, just activate the cell and type your new entry, which replaces the previous contents. Any formatting that you previously applied to the cell remains in place and is applied to the new content.

4.9.4 Editing the Contents of a Cell

If the cell contains only a few characters, replacing its contents by typing new data usually is easiest. But if the cell contains lengthy text or a complex formula and you need to make only a slight modification, you probably want to edit the cell rather than re-enter information. When you want to edit the contents of a cell, you can use one of the following ways to enter cell-edit mode:

- **Double-clicking the cell** enables you to edit the cell contents directly in the cell.
- **Selecting the cell and pressing F2** enables you to edit the cell contents directly in the cell.
- **Selecting the cell that you want to edit and then clicking inside the Formula bar** enables you to edit the cell contents in the Formula bar.

The following are the number-format categories, along with some general comments:

- **General:** The default format; it displays numbers as integers, as decimals, or in scientific notation if the value is too wide to fit in the cell.
- **Number:** Enables you to specify the number of decimal places, whether to use a comma to separate thousands, and how to display negative numbers (with a minus sign, in red, in parentheses, or in red and in parentheses).
- **Currency:** Enables you to specify the number of decimal places, whether to use a currency symbol, and how to display negative numbers (with a minus sign, in red, in parentheses, or in red and in parentheses). This format always uses a comma to separate thousands.

- **Accounting:** Differs from the Currency format in that the currency symbols always line up vertically.
- **Date:** Enables you to choose from several different date formats.
- **Time:** Enables you to choose from several different time formats.
- **Percentage:** Enables you to choose the number of decimal places and always displays a percent sign.
- **Fraction:** Enables you to choose from among nine fraction formats.
- **Scientific:** Displays numbers in exponential notation (with an E): 2.00E+05 = 200,000; 2.05E+05 = 205,000. You can choose the number of decimal places to display to the left of E.
- **Text:** When applied to a value, causes Excel to treat the value as text (even if it looks like a number). This feature is useful for such items as part numbers.
- **Special:** Contains four additional number formats (Zip Code, Zip Code +4, Phone Number, and Social Security Number).
- **Custom:** Enables you to define custom number formats that are not included in any other category.

Self Assessment Questions

6. A ..... is just a rectangular range of cells that (usually) contain column headers.

(a) table (b) column

(c) row (d) None of these.

7. The ..... key on your keyboard controls how the keys on the numeric keypad behave.

(a) PgUp (b) Num Lock

(c) PgDn (d) None of these.

8. A hidden row is actually a row with its height set to .....

(a) one (b) two

(c) zero (d) five.

4.10 Summary

- Excel is the world's most widely used spreadsheet program and is part of the Microsoft Office suite.
- Excel is very useful for non-numerical applications.
- Every worksheet has exactly 1,048,576 rows and 16,384 columns, and these values cannot be changed.
- To increase the number of rows and columns, save the workbook as an Excel 2007 XLSX file and then reopen it.
- A chart sheet displays a single chart and is accessible by clicking a tab.

4.11 Keywords

Automating complex tasks: Perform a tedious task with a single mouse click with Excel's macro capabilities.

Chart Sheet: It displays a single chart and is also accessible by clicking a tab.

Conditional Formatting: It refers to the ability to format a cell based on its value

Table: It is a rectangular range of cells that contain column headers.

Workbook: It is comprised of one or more worksheets, and each worksheet is made up of individual cells.

4.12 Review Questions

1. Excel is for non-numerical applications. Explain.
2. How Excel 2007 is different from previous version of excel?
3. What is the importance of table in worksheet?
4. What is the difference between worksheet and workbook?
5. Explain the process of moving and resizing the windows.
6. Discuss the process to increase the number of rows and columns in a worksheet.
7. How can we hide the rows and columns? Explain.
8. Define SmartArt.
9. What are the new features in Excel 2007? Discuss.
10. What are the tools through which you can move around a worksheet? Discuss each of them.

Answers for Self Assessment Questions

1 (a)	2 (b)	3 (c)	4 (a)
5 (d)	6 (a)	7 (b)	8 (c)

4.13 Further Studies

1 Microsoft Office Excel 2007 in Depth by Bill Jelen

5

MS PowerPoint

STRUCTURE

5.0 Objectives

5.1 Introduction

5.2 Area and Working with MS PowerPoint

5.3 Starting and Exiting PowerPoint

5.4 Opening a New Display Window

5.5 Creating a New Presentation

5.6 Saving Your Work

5.7 Closing and Reopening Presentations

5.8 Creating New Slides

5.9 Inserting Content from External Sources

5.10 Importing Image Files into PowerPoint

5.11 Animation and Transitions

5.12 Handouts

5.13 Creating Handouts

5.14 Adding Graphics, Sounds and Movies to a Slide

5.15 Summary

5.16 Keywords

5.17 Review Questions

5.18 Further Studies

5.0 Objectives

After studying this chapter, you will be able to:

- Understand area and working with MS PowerPoint
- Define starting and exiting PowerPoint
- Define creating a new presentation
- Discuss closing and reopening presentations
- Define creating a new slide
- Understand inserting content from external sources

5.1 Introduction

A presentation is any kind of interaction between a speaker and audience, but it usually involves one or more of the following visual aids: 35 mm slides, overhead transparencies, computer-based slides (either local or at a Web site or other network location), hard-copy handouts, and speaker notes. PowerPoint 2007 can create all of these types of visual aids, plus many other types that you learn about as we go along. Like other programs in the Office 2007 suite, PowerPoint 2007 takes a radical and innovative new approach to its user interface. Although it is very convenient to use once you master it, even experienced users of earlier versions might need some help getting started.

5.2 Area and Working with MS PowerPoint

5.2.1 Tabs and the Ribbon

Instead of using a complex menu system, PowerPoint 2007 relies on a graphical Ribbon with multiple tabbed pages (referred to as tabs). Each tab is like a toolbar, with buttons and lists you can select or open. Tabs are not easily customizable as toolbars were in earlier versions, but the Quick Access Toolbar provides a home for any custom buttons or shortcuts you would like to keep readily available. You can add almost any button or command to the Quick Access Toolbar by right-clicking it and choosing Add to Quick Access Toolbar.

5.2.2 Office (File) Menu

One menu remains: the Microsoft Office menu (abbreviated in this book as Office menu). You can access it by clicking the Microsoft Office Button (again, abbreviated in this book as Office Button), the big round button in the top-left corner of the screen. The commands on the Office menu are for working with the file itself: saving, printing, opening, closing, and so on. These commands were on the File menu in earlier PowerPoint versions (and in fact, some people actually still call the Office menu the “File menu”).

5.2.3 Styles for Graphics and Text

If you have worked with Microsoft Word, you are probably familiar with the concept of styles. A style is a saved formatting specification that you can apply to multiple blocks of text in Word, to ensure consistency. For example, to ensure that all of your headings are formatted the same way, apply a Heading style to each one. PowerPoint 2007 extends the concept of styles to cover graphic objects such as pictures, drawn lines, and shapes as well as text. For example, suppose you want each photo to have a bevelled edge effect; you can simply apply a picture style that contains the desired edge type to each picture. No more manual formatting of multiple graphic objects! Styles are found in various parts of PowerPoint, depending on the object type. For example, Figure 5.1 shows the Picture Styles section of the Format tab, available when a picture is selected. Shape Styles (for drawn lines and shapes) and WordArt Styles (for text) are also available.

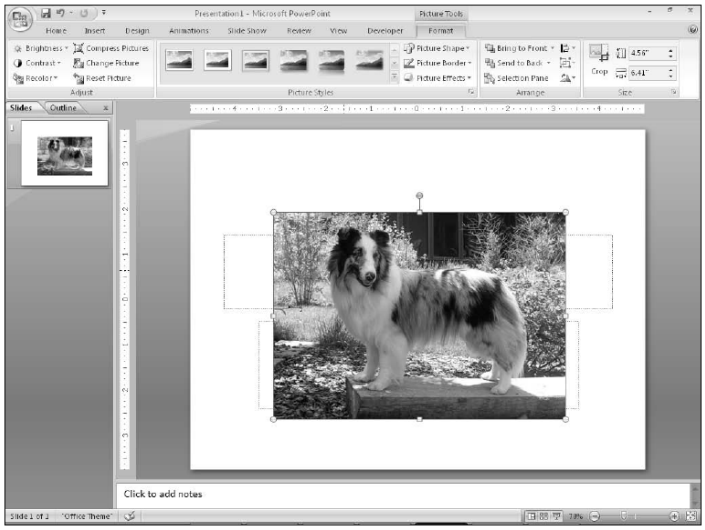


Figure 5.1: You can apply styles too easily and consistently format graphic objects.

5.2.4 Drawing and Photo Formatting Choices

Drawn objects (formerly called AutoShapes, now called Office Art graphics) are much improved in PowerPoint 2007. You can not only apply basic colours and fills to them, but you can add shadows, glows, surfaces, and 3-D tilt and rotation. In earlier versions, the 3-D option simply enabled you to extend perspective to create "sides" on a flat object. In PowerPoint 2007, the 3-D option now enables you to tilt the entire object. Figure 6.2 shows examples of some of the new effects. All of the new effects for drawn objects can also be applied to the borders of other graphics, such as imported pictures. There are also some new tools for working with photos, including applying a tint to a picture and changing the shape of the photo frame.

5.2.5 Text Formatting Options

PowerPoint 2007 adds several new text formatting capabilities to help users further polish their work. For example, you can now control character spacing and kerning, use different underline styles and colours, and make all characters in a line equal height.

Perhaps the most significant improvement in text formatting, however, is the ability to format any text using the full range of WordArt formatting tools. WordArt (a.k.a. shaped text) has been around in Office programs for years, but there has always been a strict differentiation between regular text and WordArt. Regular text (that is, text appearing in the presentation outline) could not receive WordArt formatting such as reshaping, stretching, and distortion.

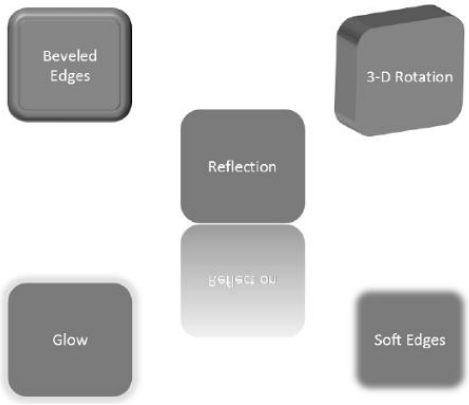


Figure 5.2: Many more effects are available for drawn lines and shapes.

PowerPoint 2007 loses the differentiation between WordArt text and regular text, so the full gamuts of formatting features are available to all text, regardless of position or usage. You can format individual words as separate pieces of WordArt, or entire text boxes by using a common WordArt style. In Figure 5.3, the slide title "Green Hill Shelties" is regular text, and appears on the presentation outline, but it also benefits from WordArt formatting effects.

5.2.6 Colour, Font, and Effect Themes

Styles can automate the formatting of individual objects, but you can also apply overall themes to the entire presentation to change all of the formatting at once. A theme is a set of formatting specifications that are applied to objects and text consistently throughout the presentation (except in cases where an object has manual formatting applied that overrides the theme). There are three elements to a theme: the colours, the fonts, and the effects. Colours are applied via a set of placeholders, as they were in PowerPoint 2003, but now you can apply tints or shades of a colour much more easily. Whenever you open a list or menu that contains a colour picker, you select from a palette like the one in Figure 5.4. The top row contains swatches for the colours in the current theme, and beneath them are various tints (lighter versions) and shades (darker versions) of the colours. By applying theme colours instead of fixed colours, you enable objects to change colour automatically when you switch to a different theme.



Figure 5.3: WordArt can now be applied to regular text, including slide titles.

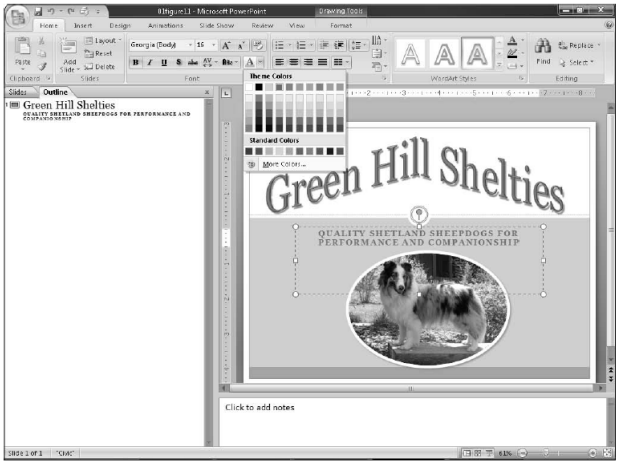


Figure 5.4: Choose colours for text and graphic objects from a colour picker that focuses on theme-based colour choices.

Font themes apply one font for headings and another for body text. In PowerPoint 2007 it is usually best not to apply a specific font to any text, but instead to apply either (Body) or (Heading) to it. Then you can let the font theme dictate the font choices, so that they will update automatically when you chose a different theme. On the Font drop-down list, the top choices are now (Body) and (Heading). The font listed next to them is the font that happens to be applied with the current theme.

Effect themes apply shadows and 3-D effects to graphic objects. PowerPoint 2007's new gallery of effects are impressive, and can make plain lines and shapes appear to pop off the screen with textures that simulate glass, metal, or other surfaces.

SmartArt

SmartArt uses groups of lines and shapes to present text information in a graphical, conceptually meaningful way. Experts have been saying for years that people respond better to information when it is presented graphically, but the difficulty in constructing attractive diagrams has meant that most people used plain bulleted lists for everything. SmartArt can convert a bulleted list into a conceptual diagram in just a few clicks. Figure 5.5 shows a plain bulleted list (left) and a SmartArt diagram constructed from it. The SmartArt is not only more interesting to look at, but it also conveys additional information—it shows that the product life cycle repeats continuously.

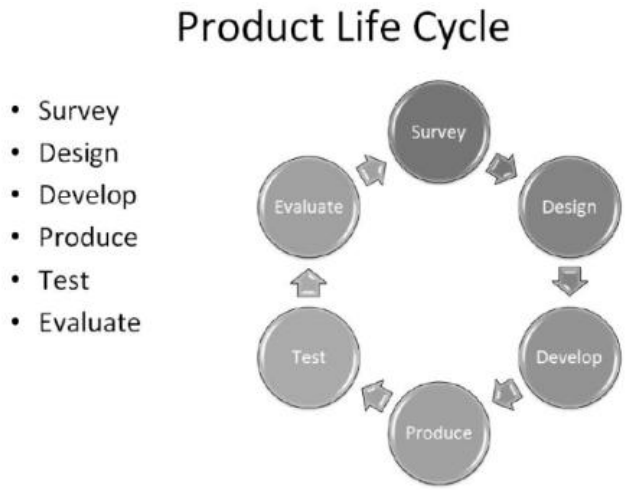


Figure 5.5: SmartArt diagrams are easy to create and make information more palatable and easy to understand.

Did You Know?

SmartArt is similar to the Diagrams feature found in PowerPoint 2003, but is based on a new graphics engine from Microsoft called Escher 2.0.

5.2.8 Better Charting Tools

In earlier Office versions, Excel had a great charting feature, but the other applications suffered along with an inferior tool called Microsoft Graph. PowerPoint 2007 includes a great, all-new charting tool, the same one that is in Excel 2007. Like SmartArt, it is integrated with the new Escher 2.0 graphics engine, and you can format the charts with a variety of theme effects. Figure 5.6 shows a typical chart.

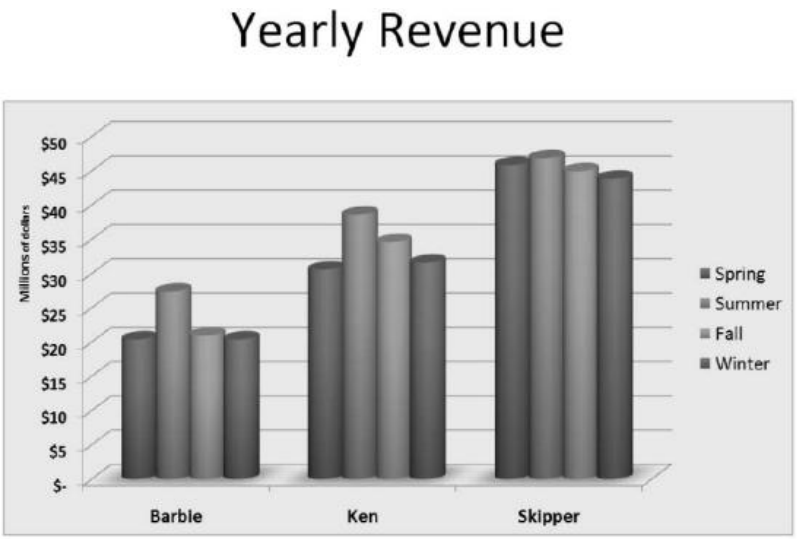


Figure 5.6: You can construct charts more easily with better formatting and layout options in PowerPoint 2007

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: Create a presentation using SmartArt and Charting features.

……..……

……..……

……...

5.2.9 Custom Slide Layouts

In earlier versions of PowerPoint, you were stuck with the slide layouts that came with PowerPoint. In PowerPoint 2007, you can create your own slide layouts, complete with content placeholders, and apply them as easily as you can apply the built-in ones. This makes it easy to reuse complex layouts without having to resort to manually created text frames each time. Custom layouts are configured from within the expanded and improved Slide Master view.

5.3 Starting and Exiting PowerPoint

You can start PowerPoint just like any other program in Windows: from the Start menu. Follow these steps:

1. Click the Start button. The Start menu opens.

2. Click All Programs.

3. Click Microsoft Office.

4. Click Microsoft Office PowerPoint 2007. The program starts.

If you have opened PowerPoint before, a shortcut to it might appear in the Recently Used Programs list, which is directly above the All Programs command on the Start menu. If you use other applications more frequently than PowerPoint, PowerPoint may scroll off this list and you therefore have to access it via the All Programs menu.

When you are ready to leave PowerPoint, select Office Button➪Exit or click the Close (X) button in the top-right corner of the PowerPoint window. (The Office button is the round button in the top left corner.) If you

have any unsaved work, PowerPoint asks if you want to save your changes. Because you have just been playing around in this chapter, you probably do not have anything to save yet. Otherwise, click No to decline to save your changes, and you are outta there.

5.3.1 Changing the View

A view is a way of displaying your presentation on-screen. PowerPoint comes with several views because at different times during the creation process, it is helpful to look at the presentation in different ways. For example, when you add a graphic to a slide, you need to work closely with that slide, but when you rearrange the slide order, you need to see the presentation as a whole.

PowerPoint offers the following views:

Normal: A combination of several resizable panes, so you can see the presentation in multiple ways at once. Normal is the default view.

Slide Sorter: A light-table-type overhead view of all the slides in your presentation, laid out in rows, suitable for big-picture rearranging.

Notes Page: A view with the slide at the top of the page and a text box below it for typed notes.

*Slide Show***:** The view you use to show the presentation on-screen. Each slide fills the entire screen in its turn.

There are two ways to change a view: click a button on the View tab, or click one of the view buttons in the bottom-right corner of the screen.

See Figure 5.7. All of the views are available in both places except Notes Page, which you can access only from the View tab.

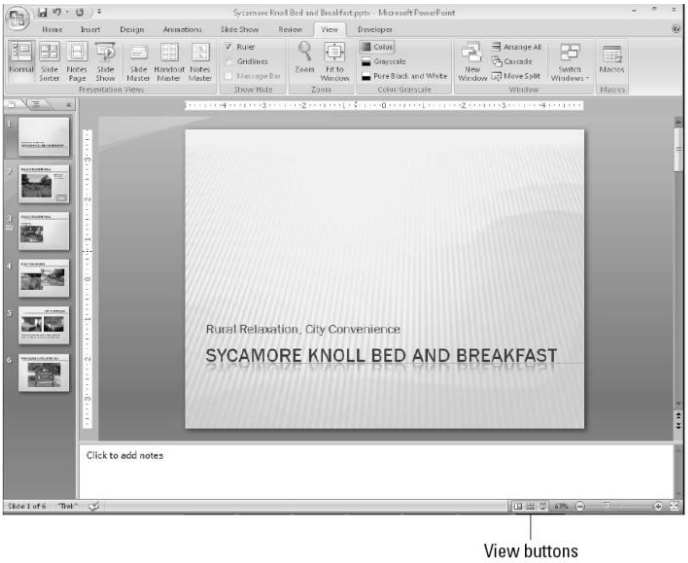


Figure 5.7 Select a view from the View tab or from the viewing controls in the bottom-right corner of the screen.

5.3.2 Normal View

Normal view, shown in Figure 5.8, is a very flexible view that contains a little of everything. In the center is the active slide, below it is a Notes pane, and to its left is a dual-use pane with two tabs: Outline and Slides. (Figure 5.7 shows Slides, and Figure 5.8 shows Outline.)

When the Outline tab is selected, the text from the slides appears in an outline form. When the Slides tab is selected, thumbnail images of all the slides appear.

Each of the panes in Normal view has its own scroll bar, so you can move around in the outline, the slide, and the notes independently of the other panes. You can resize the panes by dragging the dividers between the panes. For example, to give the notes area more room, point the mouse pointer at the divider line between it and the slide area so that the mouse pointer becomes a double-headed arrow, and then hold down the left mouse button as you drag the line up to a new spot.

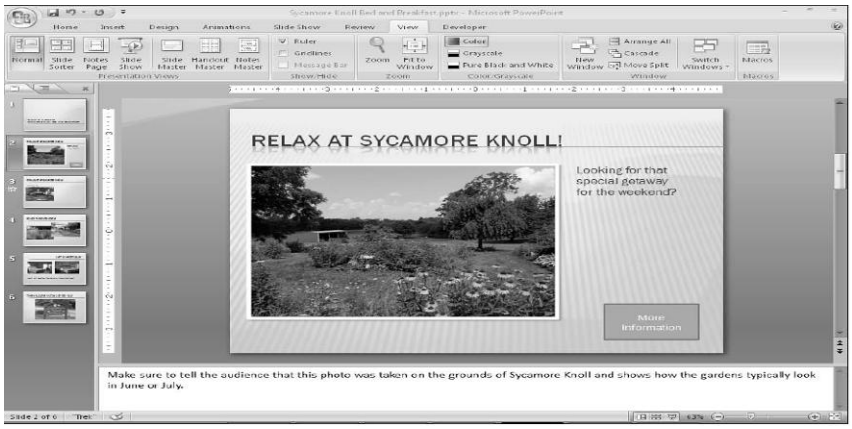


Figure 5.8: Normal view, the default, offers access to the outline, the slide, and the notes all at once.

The Slides/Outline pane is useful because it lets you jump quickly to a specific slide by clicking on it. For example, in Figure 5.7 you can click on any of the slide thumbnails on the Slides tab to display it in the Slide pane. Or in Figure 6.8 you can click some text anywhere in the outline to jump to the slide containing that text. You can turn the Slides/Outline pane off completely by clicking the X button in its top-right corner. This gives maximum room to the Slides pane. When you turn it off, the Notes pane disappears too; they cannot be turned on/off separately. To get the extra panes back, reapply Normal view.

Did You Know?

In earlier versions of PowerPoint, an Outlining toolbar was available when working with the Outline tab. In PowerPoint 2007, you can right-click anywhere in the outline to access some of those same tools on a context menu.

5.3.3 Slide Sorter View

If you have ever worked with 35mm slides, you know that it can be helpful to lay the slides out on a big table and plan the order in which to show them. You rearrange them, moving this one here, that one there, until the order is perfect. You might even start a pile of backups that you will not show in the main presentation, but will hold back in case someone asks a pertinent question. That is exactly what you can do with Slide Sorter view, shown in Figure 5.9. It lays out the slides in miniature, so you can see the big picture.

You can drag the slides around and place them in the perfect order. You can also return to Normal view to work on a slide by double-clicking the slide.

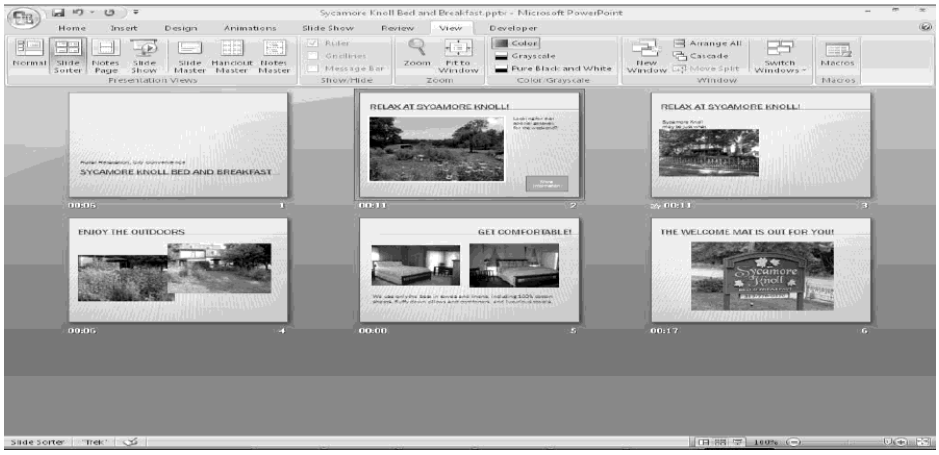


Figure 5.9: Use Slide Sorter view for a birds-eye view of the presentation.

5.3.4 Slide Show View

When it is time to rehearse the presentation, nothing shows you the finished product quite as clearly as Slide Show view does. In Slide Show view (Figure 5.10), the slide fills the entire screen. You can move from slide to slide by pressing the Page Up or Page Down keys, or by using one of the other movement methods available. You can right-click in Slide Show view to display a menu that enables you to control the show without leaving it. To leave the slide show, choose End Show from the menu or just press the Esc key.

5.3.5 Working with Notes

When you give a presentation, your props usually include more than just your brain and your slides. You typically have all kinds of notes and backup material for each slide—figures on last quarter's sales, sources to cite if someone questions your data, and so on. In the old days of framed overhead transparencies, people used to attach sticky notes to the slide frames for this purpose, and hope that nobody asked any questions that required diving into the four-inch-thick stack of statistics they brought. Today, you can type your notes and supporting facts directly in PowerPoint. As you saw earlier, you can type them directly into the Notes pane below the slide in Normal view. However, if you have a lot of notes to type, you might find it easier to work with Notes Page view instead. Notes Page view is accessible only from the View tab. In this view, you see a single slide (uneditable) with a text area, called the notes placeholder, below it for your notes. See Figure 5.10.



Figure 5.10: Notes Page view offers a special text area for your notes, separate from the slides.

5.3.6 Zooming In and Out

If you need a closer look at your presentation, you can zoom the view in or out to accommodate almost any situation. For example, if you have trouble placing a graphic exactly at the same vertical level as some text in a box next to it, you can zoom in for more precision. You can view your work at various magnifications on-screen without changing the size of the surrounding tools or the size of the print on the printout. In Normal view, each of the panes has its own individual zoom. To set the zoom for the Slides/Outline pane only, for example, select it first; then choose a zoom level. Or to zoom only in the Slide pane, click it first.

In a single-pane view like Notes Page or Slide Sorter, a single zoom setting affects the entire work area. The larger the zoom number, the larger the details on the display. A zoom of 10% would make a slide so tiny that you could not read it. A zoom of 400% would make a few letters on a slide so big they would fill the entire pane. The easiest way to set the zoom level is to drag the Zoom slider in the bottom-right corner of the PowerPoint window, or click its plus or minus buttons in increment the zoom level. See Figure 5.11.

To resize the current slide so that it is as large as possible while still fitting completely in the Slides pane, click the Fit Slide to Current Window button, or click the Fit to Window button in the Zoom group on the View tab.

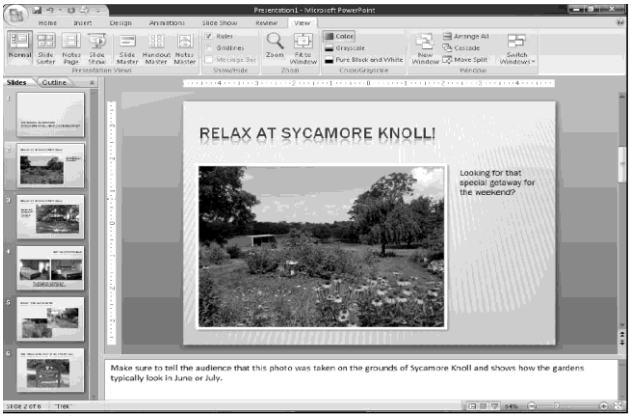


Figure 5.12: Zoom in or out to see more or less of the slide(s) at once.

Another way to control the zoom is with the Zoom dialog box. On the View tab, in the Zoom group, click the Zoom button. (You can also open that dialog box by clicking the % next to the Zoom slider.) Make your selection, as shown in Figure 5.12, by clicking the appropriate button, and then click OK. Notice that you can type a precise zoom percentage in the Percent text box. You can specify any percentage you like, but some panes and views will not go higher than 100%.

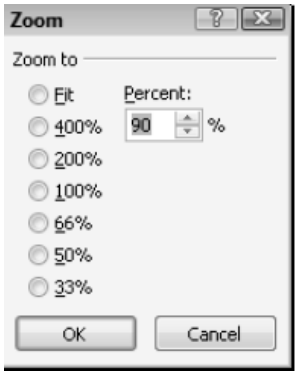


Figure 5.12: You can zoom with this Zoom dialog box rather than the slider if you prefer.

Enabling Optional Display Elements

PowerPoint has a lot of optional screen elements that you may (or may not) find useful, depending on what you are up to at the moment. The following sections describe them.

Ruler

Vertical and horizontal rulers around the slide pane can help you place objects more precisely. To toggle them on or off, mark or clear the Ruler check box on the View tab. Rulers are available only in Normal and Notes Page views. The rulers help with positioning no matter what content type you are working with, but when you are editing text in a text frame they have an additional purpose as well. The horizontal ruler shows the frame's paragraph indents and any custom tab stops, and you can drag the indent markers on the ruler just like you can in Word.

Gridlines

Gridlines are non-printing dotted lines at regularly spaced intervals that can help you line up objects on a slide. Figure 5.13 shows gridlines (and the ruler) enabled.

To turn gridlines on or off, use any of these methods:

Press Shift+F9.

On the View tab, in the Show/Hide group, mark or clear the Gridlines check box.

On the Design tab, in the Arrange group, choose Align➪Show Gridlines.

There are many options you can set for the gridlines, including whether objects snap to it, whether the grid is visible, and what the spacing should be between the gridlines.

To set grid options, follow these steps:

1. On the Home tab, in the Drawing group, choose Arrange➪Align➪Grid Settings, or right click the slide background and choose Grid and Guides. The Grid and Guides dialog box opens (see Figure 6.14).

2. In the Snap To section, mark or clear these check boxes.

Snap Objects to Grid: Specifies whether or not objects will shift automatically align with the grid.

Snap Object to Other Objects: Specifies whether or not objects will automatically align with other objects.

3. In the Grid Settings section, enter the amount of space between gridlines desired.

4. Mark or clear the Display Grid on Screen check box to display or hide the grid.

5. Click OK.

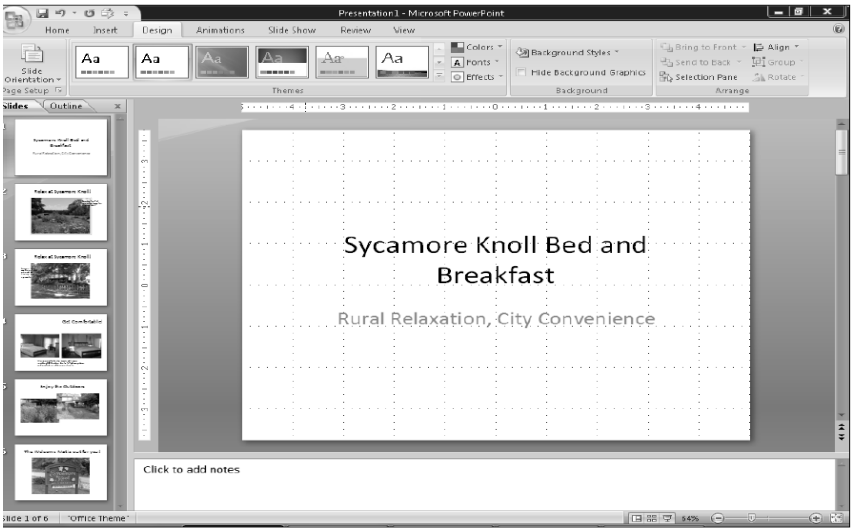


Figure 5.13: Gridlines and the ruler help align objects on a slide.

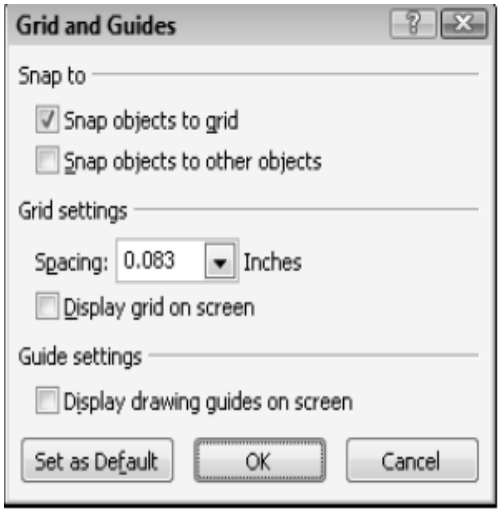


Figure 5.14: Set grid options and spacing.

Guides

Guides are like gridlines except they are individual lines, rather than a grid of lines, and you can drag them to different positions on the slide. As you drag a guide, a numeric indicator appears to let you know the ruler position. See Figure 5.15. Use the Grid and Guides dialog box to turn guides on/off, or press Alt+F9.

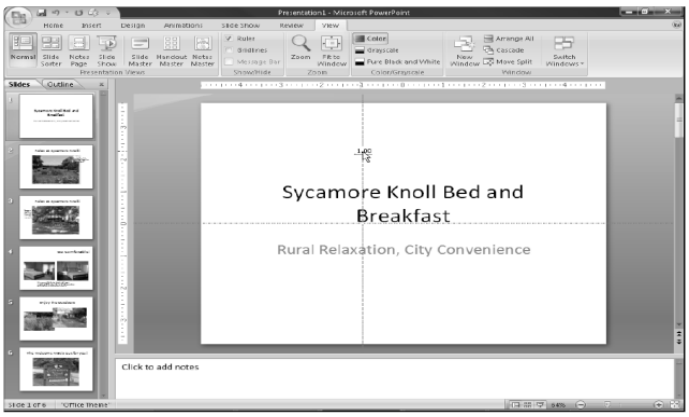


Figure 5.15: Guides are movable, non-printing lines that help with alignment.

You can create additional sets of guide lines by holding down the Ctrl key while dragging a guide (to copy it). You can have up to eight horizontal and vertical guides, all at positions you specify.

5.3.7 Colour/Greyscale/Pure Black And White Views

Most of the time you will work with your presentation in colour. However, if you plan to print the presentation in black and white or greyscale (for example, on overhead transparencies or black-and-white handouts), you should check to see what it will look like without colour.

Click the Greyscale or the Pure Black and White button on the View tab to switch to one of those views. When you do so, a Greyscale or Black and White tab becomes available, as shown in Figure 5.16. From its Setting group, you can fine-tune the greyscale or black-and-white preview. Choose one that shows the object to best advantage; PowerPoint will remember that setting when printing or outputting the presentation to a greyscale or black-and-white source.

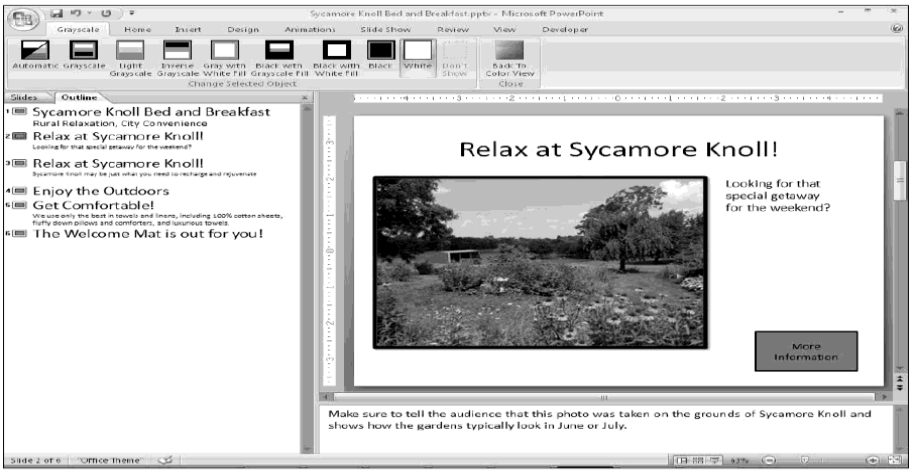


Figure 5.16: Select a grayscale or a black-and-white preview type.

When you are finished, click the Back to Colour View button on the Grayscale tab. Changing the Black and White or Grayscale settings does not affect the colours on the slides; it only affects how the slides will look and print in black and white or grayscale.

5.4 Opening a New Display Window

Have you ever wished you could be in two places at once? Well, in PowerPoint, you actually can. PowerPoint provides a way to view two spots in the presentation at the same time by opening a new window. To display a new window, display the View tab and click New Window in the Window group. Then use Arrange All or Cascade to view both windows at once.

You can use any view with any window, so you can have two slides in Normal view at once, or Slide Sorter and Notes Pages view, or any other combination. Both windows contain the same presentation, so any changes you make in one window are reflected in the other window.

5.4.1 Arranging Windows

When you have two or more windows open, whether they are for the same presentation or different ones, you need to arrange them for optimal viewing. You saw earlier in this chapter how to resize a window, but did you know that PowerPoint can do some of the arranging for you?

When you want to arrange the open windows, do one of the following:

Tile: On the View tab, click Arrange All to tile the open windows so there is no overlap.

Cascade: On the View tab, click Cascade to arrange the open windows so the title bars cascade from upper-left to lower-right on the screen. Click a title bar to activate a window.

These commands do not apply to minimized windows. If you want to include a window in the arrangement, make sure you restore it from its minimized state first.

5.4.2 Switching Among Windows

If you have more than one window open and can see at least a corner of the window you want, click it to bring it to the front. If you have one of the windows maximized, on the other hand, or if another window is obscuring the one you want, click Switch Windows (on the View tab) and select the window you want to view.

5.5 Creating a New Presentation

You can start a blank presentation from scratch, or you can base the new presentation on a template or on another presentation. Using a template or existing presentation can save you some time. However, if you have a specific vision you are going for, starting a presentation from scratch gives you a clean canvas to work from.

5.5.1 Starting a Blank Presentation from Scratch

When you start PowerPoint, a new blank presentation begins automatically with one slide. Just add your content to it, add more slides if needed, change the formatting.

If you need to start another blank presentation, follow these steps:

1. Choose Office Button➪New. The New Presentation dialog box opens. See Figure 5.17.
2. Blank Presentation is already selected. Click Create.

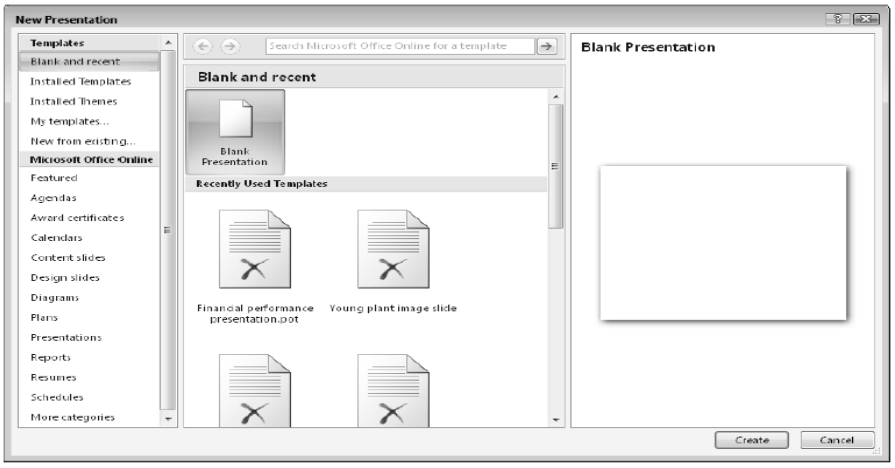


Figure 5.17: Select Blank Presentation from the New Presentation dialog box.

5.5.2 Starting a Presentation from a Template

A template is a file that contains starter settings—and sometimes starter content—on which you can base new presentations. Templates vary in their exact offerings, but can include sample slides, a background graphic, custom colour and font themes, and custom positioning for object placeholders.

When selecting a template, you can choose from these categories:

Installed Templates: Microsoft-provided templates that come preinstalled with PowerPoint

My Templates: Templates that you have created and saved yourself and templates that you previously downloaded from Microsoft Office Online

Microsoft Office Online templates: Microsoft-provided templates that you download from Microsoft on an as-needed basis

5.5.3 Using an installed template

There are only a few installed templates because Microsoft assumes that most people have an always-on Internet connection these days. Each installed template demonstrates a special-purpose type of presentation, such as a photo album, pitch book, or quiz show. There is one Corporate Presentation template as well, but if you are interested in standard corporate presentation templates, you might prefer to look at the online offerings instead.

Follow these steps to start a presentation based on an installed template:

1. Choose Office Button➪New. The New Presentation dialog box opens.
2. In the Templates list, click Installed Templates. A list of the installed templates appears.
3. Click a template to see a preview of it.
4. Select the template you want and click Create. A new presentation opens based on that template.

5.5.4 Using a Saved Template

When you start a new presentation with an online template, as in the preceding section, PowerPoint copies that template to your hard disk so you can reuse it in the future without connecting to the Internet. It is stored, along with any custom template you have created, in the My Templates folder.

To access these downloaded and custom templates, follow these steps:

1. Choose Office Button➪New. The New Presentation dialog box opens (see Figure 5.17).
2. Click My Templates. A different New Presentation dialog box appears containing templates that you have downloaded or created. See Figure 5.18.
3. Click OK. A new presentation opens based on that template.

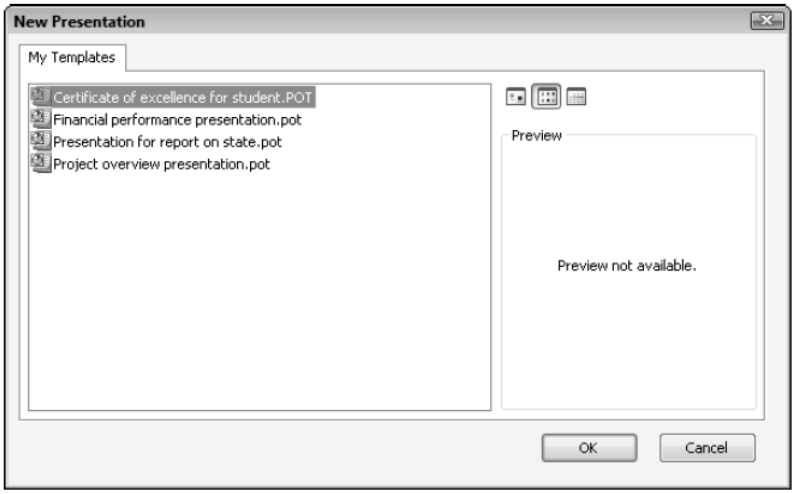


Figure 5.18: Choose a previously used or custom template.

5.5.5 Using an Online Template

The bulk of the templates for presentations are available online. You can access the library of online templates without leaving PowerPoint. Follow these steps:

1. Choose Office Button➪New. The New Presentation dialog box opens.
2. In the Templates list, in the Microsoft Office Online section, click the category of template you want. If you want standard business presentations, click Presentations; most of the other categories have special purposes.
3. Depending on the category you choose, a subcategory list might appear in the center pane.

If it does, click the subcategory that you want.

4. Click a template to see a preview of it.
5. Select the template that you want and click Download. A new presentation opens based on that template.

Basing a New Presentation on an Existing One

If you already have a presentation that is similar to the new one you need to create, you can base the new presentation on the existing one.

Follow these steps to use an existing presentation as a template:

1. Choose Office Button➪New. The New Presentation dialog box opens.
2. Click New from Existing. The New from Existing Presentation dialog box opens. See Figure 6.18.
3. Navigate to the location containing the existing presentation and select it. When you select a presentation, the Open button changes to a Create New button.
4. Click Create New.

Basing a new presentation on content from another application

PowerPoint can open files in several formats other than its own, so you can start a new presentation based on some work you have done elsewhere. For example, you can open a Word outline in PowerPoint. The results might not be very attractive—but you can fix that later with some text editing, slide layouts, and design changes.

To open a file from another application, do the following:

1. Choose Office Button➪Open. The Open dialog box appears.
2. Click the File Type button (or Files of Type in Windows XP) and choose the file type. For example, to open a text file, choose All Outlines. See Figure 5.19.
3. Select the desired file, and then click Open.
4. Save your work as a PowerPoint file by choosing Office Button➪Save As.

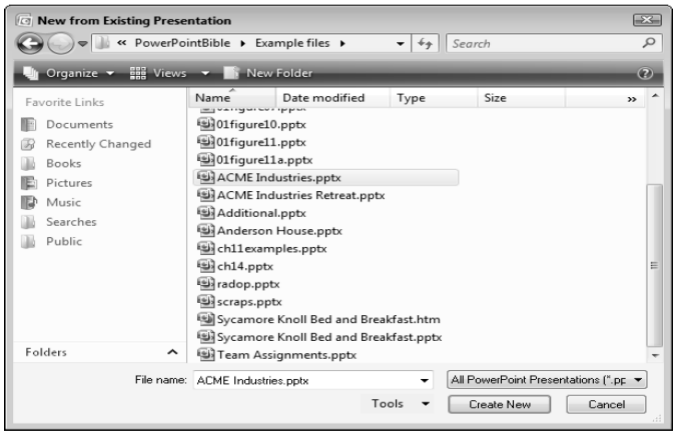


Figure 5.19. Select an existing presentation to use as a template.

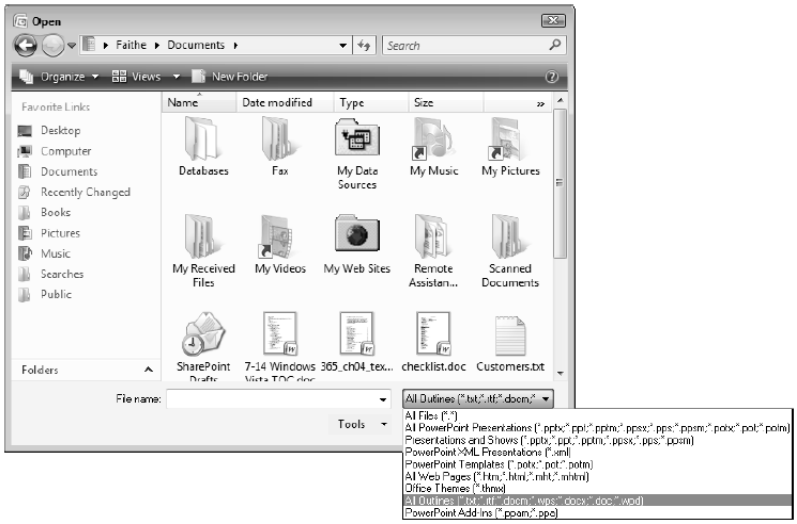


Figure 5.20. Select a data file from some other program as the basis of a new presentation.

Self assessment Questions

1. A .....is a set of formatting specifications that are applied to objects and text consistently throughout the presentation.

(a) Colour (b) Font (c) Themes (d) Text.

2. SmartArt uses groups of lines and shapes to present .....information in a graphical, conceptually meaningful way.

(a) Colour (b) Font (c) Themes (d) Text.

3. .....rulers around the slide pane can help you place objects more precisely.

(a) Vertical and horizontal (b) Left and right

(c) Centre (d) Up and down.

4. .....are non-printing dotted lines at regularly spaced intervals that can help you line up objects on a slide.

(a) View tab (b) Gridlines (c) Guides (d) Ruler.

5.6 Saving Your Work

PowerPoint is typical of most Windows programs in the way it saves and opens files. The entire PowerPoint presentation is saved in a single file, and any graphics, charts, or other elements are incorporated into that single file.

The first time you save a presentation, PowerPoint opens the Save As dialog box, prompting you for a name and location. Thereafter, when you save that presentation, PowerPoint uses the same settings and does not prompt you for them again.

5.6.1 Saving for the First Time

If you have not previously saved the presentation you are working on, Save and Save As do the same thing: They open the Save As dialog box. From there, you can specify a name, file type, and file location.

Follow these steps:

1. Choose Office Button➪Save. The Save As dialog box appears.
2. Enter a filename in the File name box. See Figure 5.21.

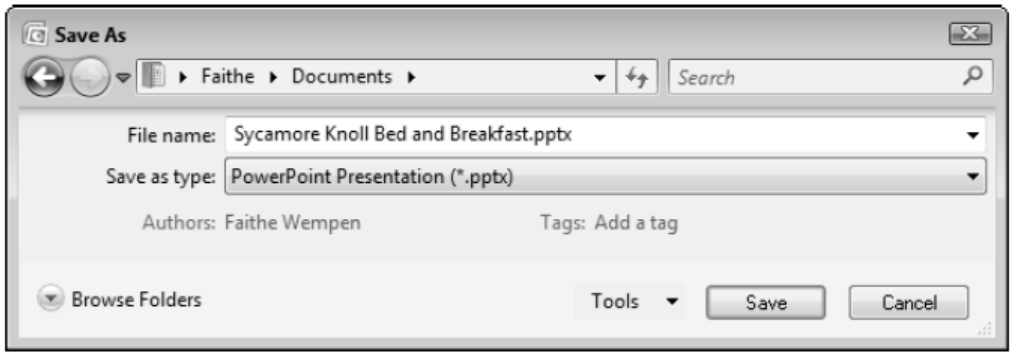


Figure 5.21: Save your work by specifying a name for the presentation file.

3. Click Save. Your work is saved.

Filenames can be up to 255 characters. For practical purposes, however, keep the names short. You can include spaces in the filenames and most symbols except <, >, ?, *, /, and \. However, if you plan to post the file on a network or the Internet at some point, you should avoid using spaces; use the underscore character instead to simulate a space if needed. There have also been problems reported with files that use exclamation points in their names, so beware of that. Generally it is best to avoid punctuation marks in names.

5.6.2 Saving Subsequent Times

After you have once saved a presentation, you can resave it with the same settings (same file type, name, and location) in any of the following ways:

Choose Office Button ➪ Save.

Press Ctrl+S.

Click the Save button on the Quick Access toolbar.

If you need to save your presentation under a different name, as a different file type, or in a different location, use the Save As command instead. This reopens the Save As dialog box, as in the preceding steps, so that you can save differently. The originally saved copy will remain under the original name, type, and location.

5.6.3 Changing Drives and Folders

By default, all files in PowerPoint (and all of the Office applications) are saved to the Documents folder (or My Documents under Windows XP) for the current user. Each user has his or her own version of this folder, so that each person's documents are kept separate depending on who is logged in to the PC.

The Documents folder is a convenient save location for beginners, because they never have to worry about changing the drive or folder. However, more advanced users will sometimes want to save files to other locations.

These other locations can include floppy disks, other hard disks in the same PC, hard disks on other PCs in a network, hard disks on Web servers on the Internet, or writeable CDs. Throughout all of the Office programs, the dialog boxes that save and open files are different depending on the operating system you are using.

5.6.4 Saving In Format

PowerPoint 2007 has a new XML-based file format, like the other Office 2007 applications. XML stands for eXtensible Markup Language; it is a text-based coding system similar to HTML that describes formatting by using inline bracketed codes and style sheets. XML-based data files are smaller than the data files from earlier PowerPoint versions, and they support all of the latest PowerPoint 2007 features. For best results, use this format whenever possible.

There are also several variants of this format for specialty uses. For example, there is a macro-enable version with a .pptm extension. There are also "show" variants (.ppsx and .ppsm) that open in Slide Show view by default, and template variants (.potx and .potm) that function as templates.

However, not everyone has PowerPoint 2007, and only PowerPoint 2007 can open files with these new formats.

(You can download a compatibility pack for earlier PowerPoint versions that will allow them to accept the new files, but you cannot assume that everyone who has an earlier version of PowerPoint will download it.) Therefore you might need to save presentations in other file formats in order to share files with other people.

The available formats are shown in Table 5.1. In the Save As dialog box, open the Save as Type drop-down list and select the desired format. See Figure 5.22.

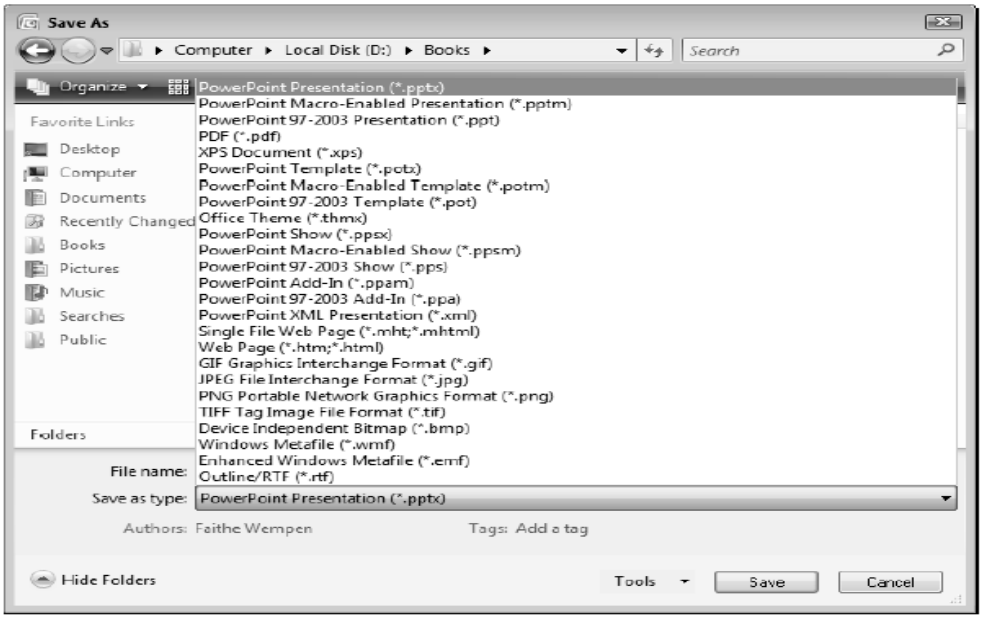


Figure 5.22: Choose a different format, if needed, from the Save As Type drop-down list.

Table 5.1: PowerPoint Save as Formats

Presentations

Format	Extension	Usage Notes
PowerPoint Presentation	.pptx	The default; use in most cases. Can open only in PowerPoint 2007 (or on an earlier version with a conversion add-on installed).
PowerPoint Macro-Enabled Presentation	.pptm	Same as above except it supports the storage of VBA or macro code.
PowerPoint 97-2003 Presentation	.ppt	A backward-compatible format for sharing files with users of PowerPoint 97, 2000, 2002 (XP), or 2003.
PowerPoint Template	.potx	A 2007-format template file.
PowerPoint Macro-Enabled Template	.potm	A 2007-format template file that supports the storage of VBA or macro code.
PowerPoint 97-2003 Template	.pot	A backward-compatible template file, also usable with PowerPoint 97, 2000, 2002 (XP), or 2003.
PowerPoint Show	.ppsx	Just like a regular PowerPoint file except it opens in Slide Show view by default; useful for distributing presentations to the audience on disk.
PowerPoint Macro-Enabled Show	.ppsm	Same as above except it supports the storage of VBA or macro code.

Presentations

Format	Extension	Usage Notes
PowerPoint 97-2003 Show	.pps	Same as a regular backward-compatible presentation file except it opens in Slide Show view by default.
PowerPoint XML Presentation	.xml	A presentation in XML format, suitable for integrating into an XML information storage system.
Single File Web Page	.mht, .mhtml	A complete presentation stored in a single file that can be displayed in most modern Web browsers. Suitable for posting on a Web site or sending via e-mail.
Web Page	.htm, .html	A typical text HTML file with each graphic element in a separate file. Suitable for posting on a Web site.

Graphics/Other

Format	Extension	Usage Notes
PDF	.pdf	Produces files in Adobe PDF format, which is a hybrid of a document and a graphic. It shows each page exactly as it will be printed, and yet allows the user to mark up the pages with comments and to search the document text. Available only after downloading PDF and XPS support from Office Online. You must have a PDF reader such as Adobe Acrobat to view PDF files.
XPS	.xps	Much the same as PDF except it's a Microsoft format. Windows Vista comes with an XPS viewer application.
Office Theme	.thmx	Somewhat like a template, but it contains only theme settings (fonts, colors, and effects). Use this if you want to apply the colors, fonts, and effects from the current presentation to other presentations but you don't want to save any of the content or layout.
PowerPoint Add-In	.ppam	A file containing executable code (usually VBA) that extends PowerPoint's capabilities.
PowerPoint 97-2003 Add-In	.ppa	Same as above except the add-in is backward-compatible.
GIF Graphics Interchange Format	.gif	Static graphic. GIFs are limited to 256 colors.
PNG Portable Network Graphics Format	.png	Static graphic. Similar to GIF except without the color depth limitation. Uses lossless compression; takes advantage of the best features of both GIF and JPG.
JPEG File Interchange Format	.jpg	Static graphic. JPG files can be very small, making them good for Web use. A lossy format, so picture quality may not be as good as with a lossless format.
TIFF Tagged Image File Format	.tif	Static graphic. TIF is a high-quality file format suitable for slides with high-resolution photos. A lossless compression format.
Device Independent Bitmap	.bmp	Static graphic. BMP is the native format for Windows graphics, including Windows background wallpaper.

Graphics/Other

Format	Extension	Usage Notes
Windows Metafile	.wmf	Static graphic. A vector-based format, so it can later be resized without distortion. Not Mac-compatible.
Enhanced Windows Metafile	.emf	Enhanced version of WMF; not compatible with 16-bit applications. Also vector-based and non-Mac-compatible.
Outline/RTF	.rtf	Text and text formatting only; excludes all non-text elements. Only text in slide placeholders will be converted to the outline. Text in the Notes area is not included.

Table 5.1 lists a lot of choices, but do not let that overwhelm you. You have three main decisions to make:

PowerPoint 2007 format or backward-compatible with PowerPoint 97-2003. Unless compatibility is essential, go with the 2007 format because you get access to all of the new features. (See Table 5.2 to learn what you will lose with backward-compatibility.) If you use a backward compatible format, some of the features described in this book work differently or are not available at all.

Macro-enabled or not. Most people will never need to create a macro-enabled presentation. PowerPoint 2007 does not support macro recording, so the only macros you would have would be written in VBA, and most PowerPoint users are not fluent in VBA programming.

Regular presentation or PowerPoint Show. The "show" variant starts the presentation in Slide Show view when it is loaded in PowerPoint; that is the only difference between it and a regular presentation. You can build your presentation in a regular format, and then save in show format right before distribution.

Most of the other choices from Table 5.2 are special-purpose, and not suitable for everyday use.

Table 5.2 PowerPoint 2007 Features Not Supported in Previous PowerPoint Versions

Feature	Issues
SmartArt Graphics	Converted to uneditable pictures
Charts (except Microsoft Graph charts)	Converted to editable OLE objects, but the chart might appear different
Custom Slide Layouts	Converted to multiple masters
Drop Shadows	Soft shadows converted to hard shadows

Feature	Issues
Equations	Converted to uneditable pictures
Heading and body fonts	Converted to static formatting
New effects: ■ 2-D or 21-D text ■ Gradient outlines for shapes or text ■ Strikethrough and double-strikethrough ■ Gradient, picture, and texture fills on text ■ Shadows, soft edges, reflections ■ Most 21-D effects	Converted to uneditable pictures
Themes	Converted to styles
Theme colors	Converted to styles
Theme effects	Converted to styles
Theme fonts	Converted to regular font usage

5.6.5 Saving for use on the Web

To share your presentation on the Web with people who do not have PowerPoint, you can save in one of the Web Page formats.

You have two choices for Web format: Web Page or Single File Web Page. Web Page creates an HTML document that has links to the slides, and then the slides and their graphics are stored in a separate folder. This would be suitable for posting on a Web site. Single File Web Page creates a single .mht document that contains all of the HTML codes and all of the slides. This would be suitable for e-mailing, for example. (In fact, the "M" in the name format is short for "mail," because this format was originally designed for e-mail use.)

However, with both of these Web formats, you lose some of the special effects, so you might prefer to distribute the presentation in a different way on the Web. If keeping the full effect of all the effects is important, consider saving in one of the PowerPoint Show formats and then make the PowerPoint Viewer utility available for free download from the same Web page.

5.6.6 Saving slides as graphics

If you save your presentation in one of the graphic formats shown in the Graphics/Other section of Table 5.1, the file ceases to be a presentation and becomes a series of unrelated graphic files, one per slide. If you choose one of these formats, you are asked whether you want to export the current slide only or all slides. If you choose all slides, PowerPoint creates a new folder in the selected folder with the same name as the original presentation file and places the graphics files in it.

5.6.7 Saving slide text only

If you want to export the text of the slides to some other application, consider the Outline/RTF format, which creates an outline similar to what you see in the Outline pane in PowerPoint. This file can then be opened in

Word or any other application that supports RTF text files. Only text in placeholders is exported, though, not text in manually inserted text boxes.

5.6.8 Specifying Save Options

The Save Options enable you to fine-tune the saving process for special needs. For example, you can employ Save Options to embed fonts, to change the interval at which PowerPoint saves AutoRecover information, and more.

There are two ways to access the Save options:

Choose Office Button➪PowerPoint Options and click Save.

From the Save As dialog box, click Tools➪Save Options.

The PowerPoint Options dialog box appears, as in Figure 5.23.

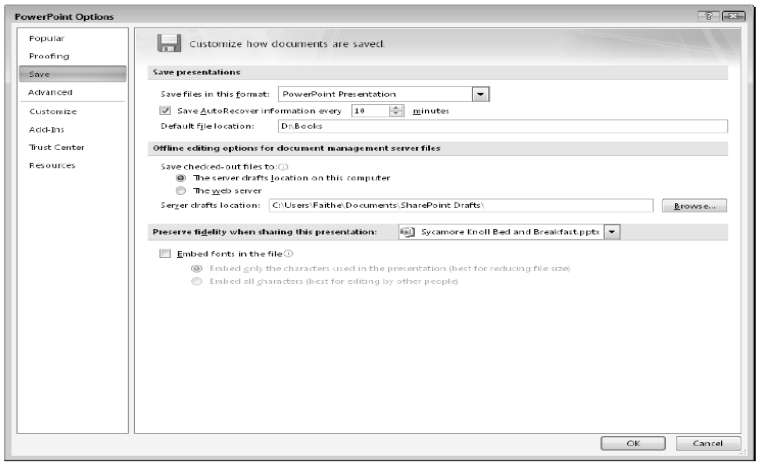


Figure 5.23: Set Save Options to match the way you want PowerPoint to save your work.

Then set any of the options desired. They are summarized in Table 5.3. Click OK when you are finished. One of the most important features described in Table 5.3 is AutoRecover, which is turned on by default.

This means if a system error or power outage causes PowerPoint to terminate unexpectedly, you do not lose all of the work you have done. The next time you start PowerPoint, it opens the recovered file and asks if you want to save it.

Caution

AutoRecover is not a substitute for saving your work the regular way. It does not save in the same sense that the Save command does; it only saves a backup version as PowerPoint is running. If you quit PowerPoint normally, that backup version is erased. The backup version is available for recovery only if PowerPoint terminates abnormally (because a system lockup or a power outage).

5.7 Closing and Reopening Presentations

You can have several presentation files open at once and switch freely between them, but this can bog down your computer's performance somewhat. Unless you are doing some cut-and-paste work, it is best to have only one presentation file open—the one you are actively working on. It is easy to close and open presentations as needed.

5.7.1 Closing a presentation

When you exit PowerPoint, the open presentation file automatically closes, and you're prompted to save your changes if you have made any. If you want to close a presentation file without exiting PowerPoint, follow these steps:

1. Choose Office Button➪Close. (Figure 5.24 shows the Close command.)

If you have not made any changes to the presentation since the last time you saved, you are done.

2. If you have made any changes to the presentation, you are prompted to save them. If you do not want to save your changes, click No, and you are done.

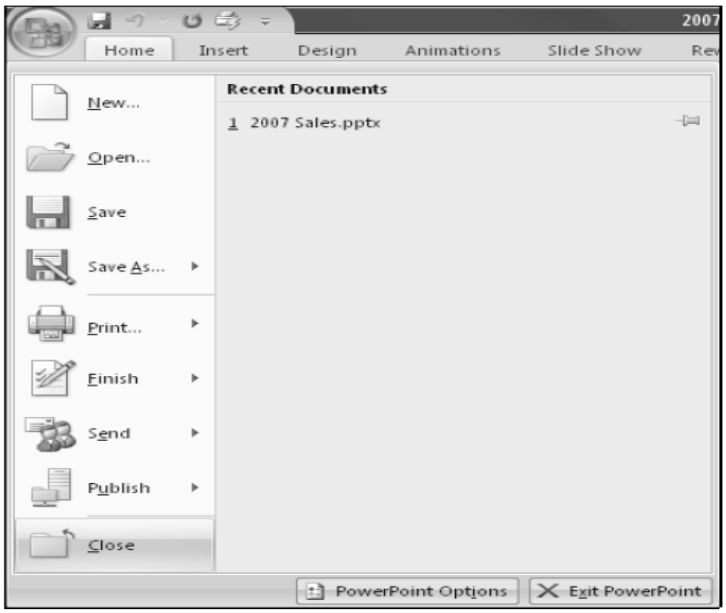


Figure 5.24: Close the presentation via the Office menu.

3. If you want to save your changes, click Yes. If the presentation has already been saved once, you are done.

4. If the presentation has not been saved before, the Save As dialog box appears. Type a name in the File Name text box and click Save.

5.7.2 Opening a presentation

To open a recently used presentation, select it from the right side of the Office menu. Although only one file appears in Figure 5.24, up to nine can appear by default.

If the presentation you want to open does not appear on the Office menu, follow these steps to find and open it:

1. Choose Office Button➪Open. The Open dialog box appears.
2. Choose the file you want. If necessary, change the location to find the file.
3. Click Open. The presentation opens.

To open more than one presentation at once, hold down the Ctrl key as you click each file you want to open. Then, click the Open button and they all open in their own windows.

The Open button in the Open dialog box has its own drop-down list from which you can select commands that open the file in different ways. See Figure 5.24, and refer to Table 5.4 for an explanation of the available options.

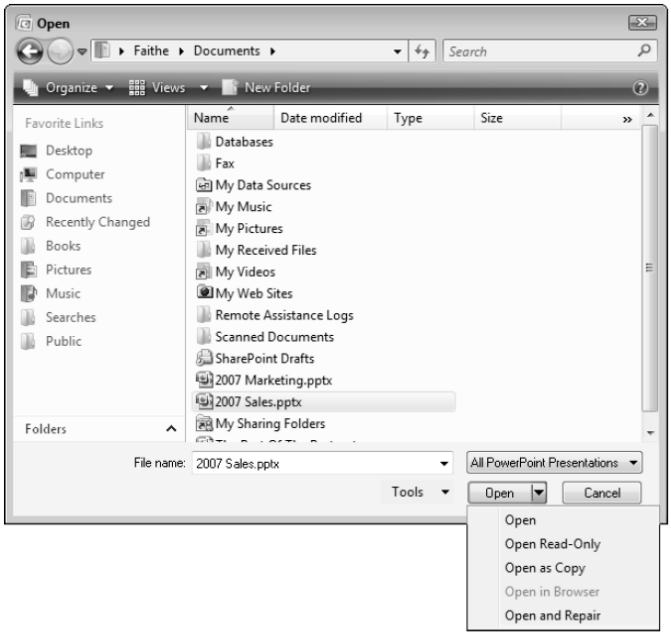


Figure 5.24: The Open button's menu contains several special options for opening a file.

5.7.3 Opening a file from a different program

Just as you can save files in various program formats, you can also open files from various programs. PowerPoint can detect the type of file and convert it automatically as you open it, so you do not have to know the exact file type. (For example, if you have an old PowerPoint file with a .ppt extension, you do not have to know what version it came from.) The only problem is with files that do not have extensions that PowerPoint automatically recognizes. In that case, you must change the File Typesetting in the Open dialog box to All Files so that the file to be opened becomes available on the file list. See Figure 5.24. This change is valid for only this one use of the Open dialog box; the file type reverts to All PowerPoint Presentations, the default, the next time you open it.

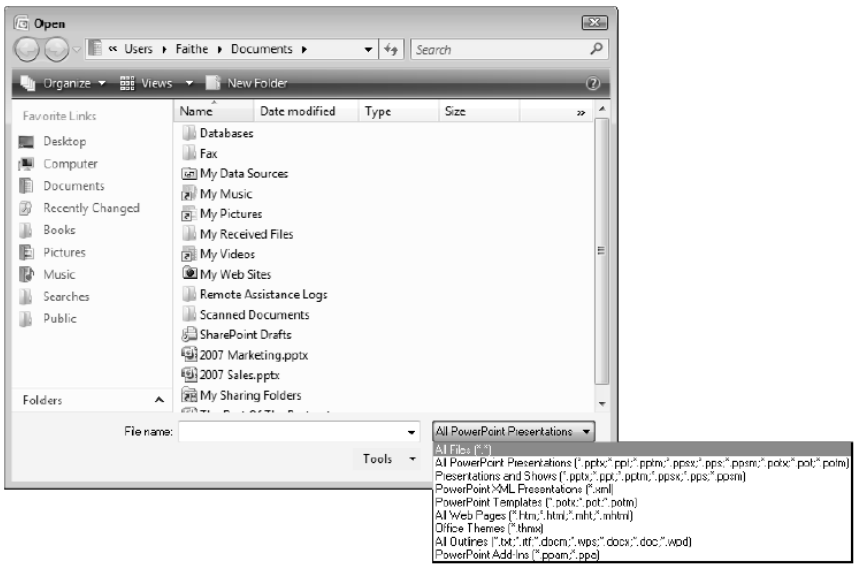


Figure 5.24: To open files from different programs, change the File Type setting to All Files.

5.7.4 Finding a presentation file to open

If you have forgotten where you saved a particular presentation file, you're not out of luck. The Open dialog box (under Windows Vista) includes a Search box that can help you locate it. See Figure. 5.25.

To search for a file, follow these steps:

1. Choose Office Button ➪ Open to display the Open dialog box.
2. Navigate to a location that you know the file is in. For example, if you know it is on the C:
drive, click Computer in the Favourite Links list and then double-click the C: drive.
3. Click in the Search box and type part of the filename (if you know it) or a word or phrase used in the file.
4. Press Enter. A list of files appears that match that specification.
5. Open the file as you normally would.

5.8 Creating New Slides

Different templates start a presentation with different numbers and types of slides. A blank presentation has only a single slide, and you must create any others that you want.

There are several ways to create new slides. For example, you can type new text in the outline and then promote it to slide status, or you can add slides with the New Slide button that is on the Insert tab. You can also copy existing slides, either within the same presentation or from other sources.

5.8.1 Creating new slides from the Outline pane

The outline pane shows the text from the presentation's slides in a hierarchical tree, with the slide titles at the top level (the slide level) and the various levels of bulleted lists on the slides displaying as subordinate levels. Text that you type in the Outline pane appears on the slide, and *vice versa*, as shown in Figure 5.25.

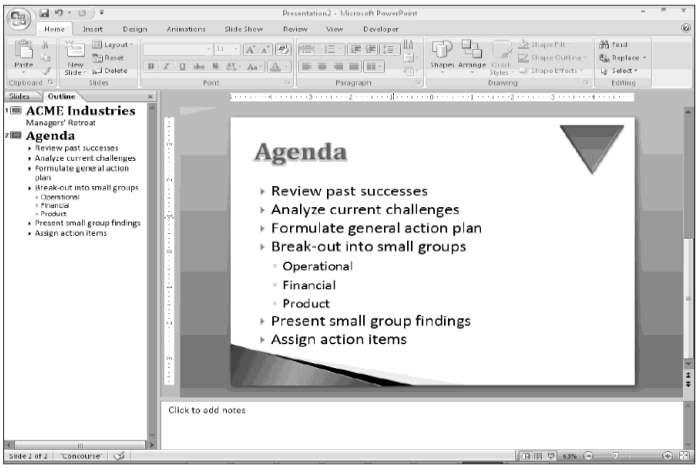


Figure 5.25: When you type text into the Outline pane, it automatically appears on the current slide.

Steps to create a new slide from the Outline pane are:

1. Switch to Normal view.
2. Right-click the existing line on the Outline pane that the new slide should follow.
3. Click New Slide. A new line appears in the Outline pane, with a slide symbol to its left.
4. Type the title for the new slide. The title appears both in the Outline pane and on the slide.

You can also create a new slide by starting a new line in the Outline pane and then promoting it to slide level by pressing Shift+Tab. Follow these steps to insert a new slide in this way:

1. Position the insertion point at the end of the last line of the slide that the new slide should follow, and press Enter to start a new line.
2. Press Shift+Tab to promote the new line to the highest level (press it multiple times if needed), so that a slide icon appears to its left.
3. Type the title for the new slide. The title appears both in the Outline pane and on the slide.

After creating the slide, you can continue creating its content directly in the Outline pane. Press Enter to start a new line, and then use Tab to demote, or Shift+Tab to promote, the line to the desired level. You can also right-click the text and choose Promote or Demote. Promoting a line all the way to the top level changes the line to a new slide title.

5.8.2 Creating a slide from the Slides pane

Here is a very quick method for creating a new slide, based on the default layout. It does not get much easier than this:

1. In Normal view, in the Slides pane, click the slide that the new slide should follow.
2. Press Enter. A new slide appears using the Title and Content layout.

The drawback to this method is that you cannot specify the layout. To choose a layout other than the default one.

5.8.3 Creating a slide from a layout

A slide layout is a layout guide that tells PowerPoint what placeholder boxes to use on a particular slide and where to position them. Although slide layouts can contain placeholders for text, they also contain graphics, charts, tables, and other useful elements. After you create a new slide with placeholders, you can click a placeholder to open whatever controls you need to insert that type of object.

When you create new slides using the outline method described in the preceding section, the new slides use the Title and Content layout, which consists of a slide title and a single, large placeholder box for content. If you want to use another layout, such as a slide with two adjacent but separate frames of content, you must either switch the slide to a different layout after its creation (using the Layout menu on the Home tab), or you must specify a different layout when you initially create the slide.

To specify a certain layout as you are creating a slide, follow these steps:

1. In Normal or Slide Sorter view, select or display the slide that the new one should follow.

You can select a slide by clicking its thumbnail image in Slide Sorter view or on the Slides pane in Normal view. You can also move the insertion point to the slide's text in the Outline pane.

2. On either the Home tab, do one of the following:

To add a new slide using the default Title and Content layout, click the top (graphical) portion of the New Slide button.

To add a new slide using another layout, click the bottom (text) portion of the New Slide button and then select the desired layout from the menu, as shown in Figure 5.26.

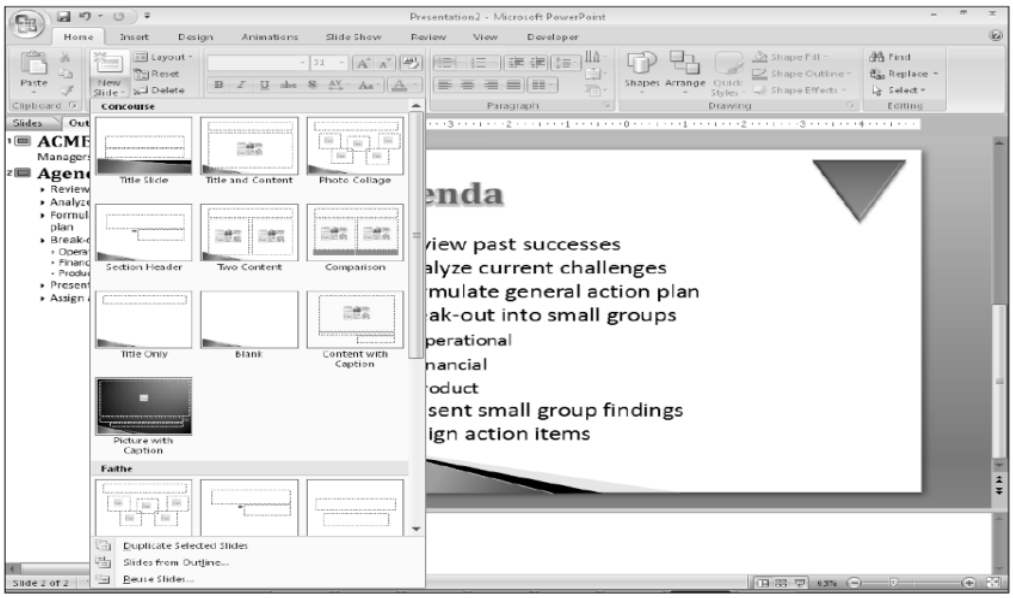


Figure 5.26: Create a new slide, based on the layout of your choice.

5.8.4 Copying Slides

Another way to create a new slide is to copy an existing one in the same presentation. This is especially useful when you are using multiple slides to create a progression because one slide is typically identical to the next slide in a sequence, except for a small change.

There are several ways to copy one or more slides. One way is to use the Windows Clipboard, as in the steps:

1. Select the slide or slides that you want to copy.
2. Press Ctrl+C. You can also click the Copy button on the Home tab, or right-click the selection and click Copy.
3. Select the slide that the pasted slide or slides should follow. Alternately, in the Outline pane, click to place the insertion point where you want the insertion.
4. Press Ctrl+V. You can also click the Paste button on the Home tab, or right-click the destination and click Paste.

PowerPoint also has a Duplicate Slides command that does the same thing as a copy-and-paste command. Although it may be a little faster, it gives you less control as to where the pasted copies will appear:

1. Select the slide or slides to be duplicated.
2. On the Home tab, click the bottom part of the New Slide button to open its menu.
3. Click Duplicate Selected Slides.

PowerPoint pastes the slides immediately after the last slide in the selection. For example, if you selected slides 1, 3, and 6, then the copies are placed after slide 6.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Create a presentation including more than five slides and use different time for all slides.

……..……

……..……

……...

5.9 Inserting Content from External Sources

Many people find that they can save a lot of time by copying text or slides from other programs or from other PowerPoint presentations to form the basis of a new presentation. There is no need to reinvent the wheel each time! The following sections look at various ways to bring in content from external sources.

5.9.1 Copying slides from other presentations

There are several ways to copy slides from other presentations.

You can:

Open the presentation, save it under a different name, and then delete the slides that you do not want, leaving a new presentation with the desired slides ready for customization.

Open two PowerPoint windows side-by-side and drag-and-drop slides between them.

Open two PowerPoint presentations, copy slides from one of them to the Clipboard (Ctrl+C), and then paste them into the other presentation (Ctrl+V).

Use the Reuse Slides feature in PowerPoint, as described next.

To reuse slides from other presentations with the Reuse Slides feature, follow these steps:

1. On the Home tab, click the lower portion of the New Slide button to open its menu.
2. Click Reuse Slides. The Reuse Slides pane appears.
3. Click Open a PowerPoint File

OR

Click the Browse button and then click Browse File.

4. In the Browse dialog box, select the presentation from which you want to copy slides, and click Open. Thumbnail images of the slides in the presentation appear in the Reuse Slides pane, as shown in Figure 5.27.

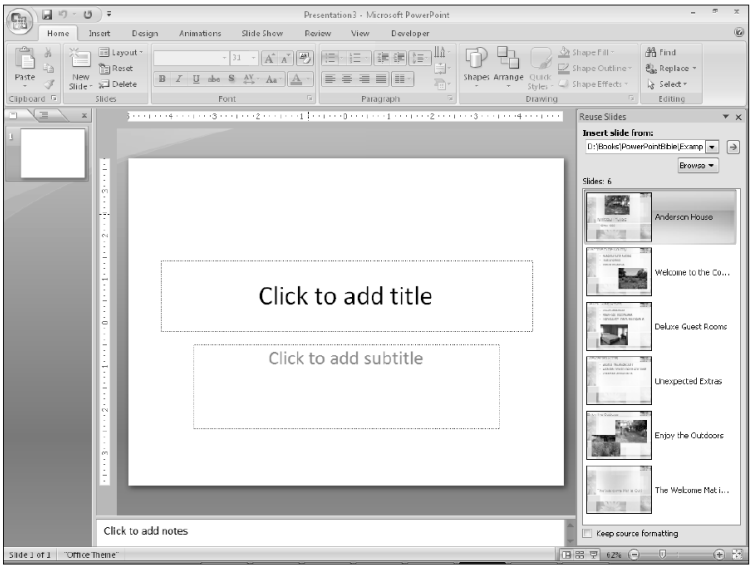


Figure 5.27: Choose individual slides to copy to the current presentation.

5. (Optional) If you want to keep the source formatting when copying slides, select the Keep Source Formatting check box at the bottom of the task pane.
6. (Optional) You can move the cursor over a slide to see an enlarged image of it.
7. Do any of the following:

To insert a single slide, click it.

To insert all slides at once, right-click any slide and choose Insert All Slides.
To copy only the theme (not the content), right-click any slide and choose Apply Theme to All Slides, or Apply Theme to Selected Slides.

Caution
Copying the theme with the Apply Theme to All Slides or Apply Theme to Selected Slides command does not copy the background graphics, layouts, or anything else other than the three elements that are included in a theme: font choices, colour choices, and effect choices. If you want to copy all of the formatting, select the Keep Source Formatting checkbox and insert one or more slides

5.9.2 Inserting new slides from an Outline

All of the Microsoft Office applications work well together, and so it is easy to move content between them. For example, you can create an outline for a presentation in Microsoft Word and then import it into PowerPoint. PowerPoint uses the heading styles that you assigned in Word to decide which items are slide titles and which items are slide content. The top-level headings form the slide titles.
To try this out, open Word, switch to Outline view (from the View tab), and then type a short outline of a presentation. Press Tab to demote, or Shift+Tab to promote, a selected line. Then save your work, go back to PowerPoint, and follow these steps to import it:
1. On the Home tab, click the lower portion of the New Slide button to open its menu.
2. Click Slides from Outline. The Insert Outline dialog box opens.
3. Select the file containing the outline text that you want to import.
4. Click Insert. PowerPoint imports the outline.

If there were already existing slides in the presentation, they remain untouched. (This includes any blank slides, and so you might need to delete the blank slide at the beginning of the presentation after importing.)
All of the Heading 1 lines from the outline become separate slide titles, and all of the subordinate headings become bullet points in the slides.

5.9.3 Importing From Other Text-Based Formats

In addition to Word, PowerPoint also imports from plain-text files, from WordPerfect (5.x or 6.x), from Microsoft Works, and from Web pages. The procedure is the same as in the preceding steps. If the file does not appear in the Insert Outline dialog box, change the file type to the desired file type. If you are setting up a plain-text file for import, you obviously would not have the outlining tools from Word at your disposal. Instead, you must rely on tabs. Each line that should be a title slide should start at the left margin; first-level bullet paragraphs should be preceded by a single tab; second-level bullets should be preceded by two tabs, and so on.

5.9.4 Post-import cleanup

After importing text from an outline, there will probably be a few minor corrections that you need to make.
Run through this checklist:
The first slide in the presentation might be blank. If it is, then delete it.
The Title Slide layout may not be applied to the first slide; apply that layout, if necessary.
A theme may not be applied; choose one from the Design tab, if necessary.

Some of the text might contain manual formatting that interferes with the theme formatting and creates inconsistency. Remove any manual formatting that you notice. (One way to do this is to select all of the text in the Outline pane by pressing Ctrl+A and then stripping off the manual formatting by pressing Ctrl+spacebar.)
If some of the text is too long to fit comfortably on a slide, change to a different slide layout, such as a two-column list, if necessary. You might also need to split the content into two or more slides.
There might be some blank bullet points on some slides. Delete these bullet points.

5.10 Importing Image Files into PowerPoint

Most of the choices you make regarding a raster image's resolution, colour depth, and file type are done outside of PowerPoint. Consequently, by the time you are ready to put them into PowerPoint, the hard part is over.
Assuming you have already acquired the image, use the following steps to insert it into PowerPoint.
1. Display the slide on which you want to place the image.
2. If the slide has a content placeholder for Insert Picture from File, as in Figure 5.28, click it. Otherwise, click Picture on the Insert tab. The Insert Picture dialog box opens.

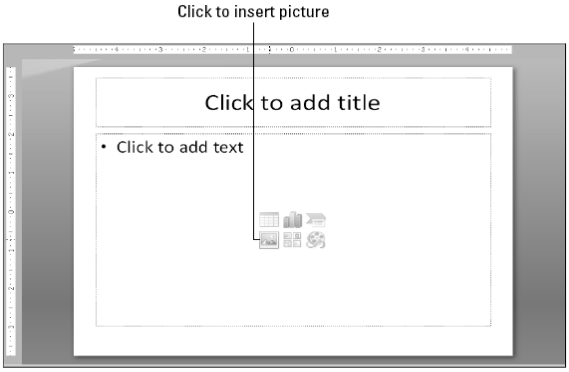


Figure 5.28: You can insert a picture by using the Insert Picture from File content placeholder icon.

3. Select the picture to import. See Figure 5.29. You can switch the view by using the View (or Views) button in the dialog box to see thumbnails or details if either is effective in helping you determine which file is which.
4. Click Insert. The picture is inserted.

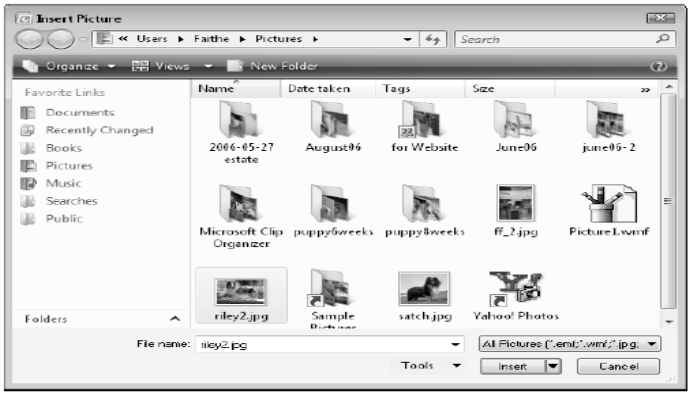


Figure 5.29: Select the picture to be inserted.

5.10.1 Linking to a graphic file

If you have a sharp eye, you may have noticed that the Insert button in Figure 24-5 has a drop-down list associated with it. That list has these choices:

Insert: The default inserts the graphic but maintains no connection.

Link to File: Creates an OLE link to the file, but does not maintain a local copy of it in PowerPoint.

Insert and Link: Creates a link to the file, and also inserts a local copy of its current state, so if the linked copy is not available in the future, the local copy will still appear.

Use Link to File whenever you want to insert a pointer rather than the original. When the presentation opens, it pulls in the graphic from the disk. If the graphic is not available, it displays an empty frame with a red X in the corner in the graphic's place. Using Link to File keeps the size of the original PowerPoint file very small because it does not actually contain the graphics—only links to them. However, if you move or delete the graphic, PowerPoint would not be able to find it anymore.

The important thing to know about this link in the Link to File feature is that it is not the same thing as an OLE link. This is not a dynamic link that you can manage. It is a much simpler link and much less flexible.

You cannot change the file location to which it is linked, for example; if the location of the graphic changes, you must delete it from PowerPoint and reinsert it.

5.11 Animation and Transitions

In PowerPoint, animation is the way that individual objects enter or exit a slide. On a slide with no animation, all of the objects on the slide simply appear at the same time when you display it. (Boring, eh?) However, you can apply animation to the slide so that the bullet points fly in from the left, one at a time, and the graphic drops down from the top afterward.

A transition is another kind of animation. A transition refers to the entry or exit of the entire slide, rather than of an individual object on the slide.

Here are some ideas for using animation effectively in your presentations: Animate parts of a chart so that the data appears one series at a time. This technique works well if you want to talk about each series separately.

Set up questions and answers on a slide so that the question appears first, and then, when you click the question, the answer appears. Dim each bullet point when the next one comes into view, so that you are, in effect, highlighting the current one.

Make an object appear and then disappear. For example, you might have an image of a lightning bolt that flashes on the slide for one second and then disappears, or a picture of a race car that drives onto the slide from the left and then immediately drives out of sight to the right.

Rearrange the order in which objects appear on the slide. For example, you could make numbered points appear from the bottom up for a Top Ten list.

5.12 Handouts

If you are presenting a live show, the center piece of your presentation is your slides. Whether you show them using a computer screen, a slide projector, or an overhead projector, the slides—combined with your own dazzling personality—make the biggest impact. But if you rely on your audience to remember everything you say, you may be disappointed. With handouts, the audience members can follow along with you during the show and even take their own notes. They can then take the handouts home with them to review the information again later.

You probably want a different set of support materials for yourself than you want for the audience. Support materials designed for the speaker's use are called speaker notes. In addition to small printouts of the slides, the speaker notes contain any extra notes or background information that you think you may need to jog your memory as you speak. Some people get very nervous when they speak in front of a crowd; speaker notes can remind you of the joke you wanted to open with or the exact figures behind a particular pie chart.

Presentation professionals are divided about how and when to use handouts most effectively. Here are some of the many conflicting viewpoints. The bottom line is that each of them is an opinion on how much power and credit to give to the audience; your answer may vary depending on the audience you are addressing.

You should give handouts at the beginning of the presentation. The audience can absorb the information better if they can follow along on paper.

This approach makes a lot of sense. Research has proven that people absorb more facts if presented with them in more than one medium. This approach also gives your audience free will; they can listen to you or not, and they still have the information. It is their choice, and this can be extremely scary for less-confident speakers. It is not just a speaker confidence issue in some cases, however. If you plan to give a lot of extra information in your speech that is not on the handouts, people might miss it if you distribute the handouts at the beginning because they're reading ahead.

You should not give the audience handouts because they would not pay as close attention to your speech if they know that the information is already written down for them.

This philosophy falls at the other end of the spectrum. It gives the audience the least power and shows the least confidence in their ability to pay attention to you in the presence of a distraction (handouts). If you truly do not trust your audience to be professional and listen, this approach may be your best option. However, do not let insecurity as a speaker drive you prematurely to this conclusion.

The fact is that people would not take away as much knowledge about the topic without handouts as they would if you provide handouts. So, ask yourself if your ultimate goal is to fill the audience with knowledge or to make them pay attention to you.

You should give handouts at the end of the presentation so that people will have the information to take home but not be distracted during the speech.

This approach attempts to solve the dilemma with compromise. The trouble with it, as with all compromises, is that it does an incomplete job from both angles. Because audience members cannot follow along on the handouts during the presentation, they miss the opportunity to jot notes on the handouts. And because the audience knows that handouts are coming, they might nod off and miss something important. The other problem is that if you do not clearly tell people that handouts are coming later, some people spend the entire presentation frantically copying down each slide on their own notepaper.

5.13 Creating Handouts

To create handouts, you simply decide on a layout (a number of slides per page) and then choose that layout from the Print dialog box as you print. No muss, no fuss! If you want to get more involved, you can edit the layout in Handout Master View before printing.

5.13.1 Choosing a layout

Assuming you have decided that handouts are appropriate for your speech, you must decide on the format for them. You have a choice of one, two, three, four, six, or nine slides per page.

1: Places a single slide vertically and horizontally "centered" on the page.

2: Prints two big slides on each page. This layout is good for slides that have a lot of fine print and small details or for situations where you are not confident that the reproduction quality will be good. There is nothing more frustrating for an audience than not being able to read the handouts!

3: Makes the slides much smaller—less than one-half the size of the ones in the two-slide layout.

But you get a nice bonus with this layout: lines to the side of each slide for note-taking. This layout works well for presentations where the slides are big and simple, and the speaker is providing a lot of extra information that is not on the slides. The audience members can write the extra information in the note-taking space provided.

4: Uses the same size slides as the three-slide layout, but they are spaced out two-by-two without note-taking lines. However, there is still plenty of room above and below each slide, so the audience members still have lots of room to take notes.

5: Uses slides the same size as the three-slide and four-slide layouts, but crams more slides on the page at the expense of note-taking space. This layout is good for presentation with big, simple slides where the audience does not need to take notes. If you are not sure if the audience will benefit at all from handouts being distributed, consider whether this layout would be a good compromise.

This format also saves paper, which might be an issue if you need to make hundreds of copies.

6: Makes the slides very tiny, almost like a Slide Sorter view, so that you can see nine at a time.

This layout makes them very hard to read unless the slide text is extremely simple. We do not recommend this layout in most cases, because the audience really would not get much out of such handouts.

Finally, there is an Outline handout layout, which prints an outline of all of the text in your presentation—that is, all of the text that is part of placeholders in slide layouts; any text in extra text boxes you have added manually is excluded. It is not considered a handout when you are printing, but it is included with the handout layouts in the Handout Master.

5.13.2 Printing handouts

When you have decided which layout is appropriate for your needs, print your handouts as follows:

1. (Optional) If you want to print only one particular slide, or a group of slides, select the ones you want in either Slide Sorter view or in the slide thumbnails task pane on the left.
2. Select Office Button➪Print. The Print dialog box appears.
3. Set options for your printer or choose a different printer. See the “Setting Printer-Specific Options” section later in this chapter for help with this.
4. In the Print Range area, choose one of the following:

All to print the entire presentation.

Current Slide to print whatever slide you selected before you issued the Print command.

Selection to print multiple slides you selected before you issued the Print command. It is not available if you did not select any slides beforehand.

Custom Show to print a certain custom show you have set up. It is not available if you do not have any custom shows.

Slides to print the slide numbers that you type in the accompanying text box. Indicate a contiguous range with a dash. For example, to print slides 1 through 9, type 1-9. Indicate non-contiguous slides with commas. For example, to print slides, 2, 4, and 6, type 2, 4, 6. Or to print slides 2 plus 6 through 10, type 2, 6-10. To print them in reverse order, type the order that way, such as 10-6, 2.

5. Enter a number of copies in the Number of Copies text box. The default is 1. If you want the copies collated (applicable to multipage printouts only), make sure you mark the Collate checkbox.
6. Open the Print What drop-down list and choose Handouts. The Handouts section of the box becomes available, as shown in Figure 6.30.

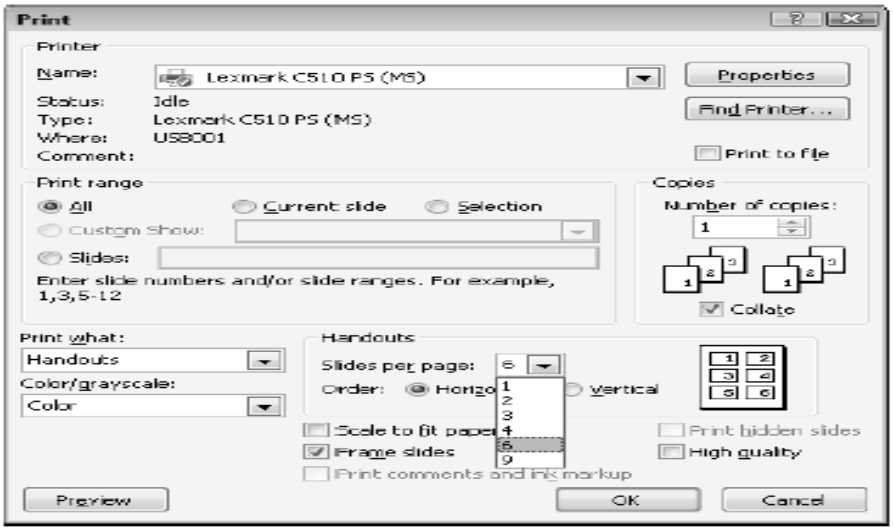


Figure 5.30: Choose Handouts to print and specify which handout layout you want.

7. Open the Slides Per Page drop-down list and choose the number of slides per page you want.
8. If available, choose an Order: Horizontal or Vertical. Not all number-of-slide choices (from Step 7) support an Order choice.
9. Open the Colour/Grayscale drop-down list and select the colour setting for the printouts:
Colour**:** Sends the data to the printer assuming that colour will be used. When you use this setting with a black-and-white printer, it results in slides with grayscale or black backgrounds.
Use this setting if you want the handouts to look as much as possible like the onscreen slides.
Grayscale: Sends the data to the printer assuming that colour will not be used. Coloured backgrounds are removed, and if text is normally a light colour on a dark background, that is reversed. Use this setting if you want PowerPoint to optimize the printout for viewing on white paper.
Pure Black and White: This format hides most shadows and patterns. It is good for faxes and overhead transparencies.
10. Mark any desired checkboxes at the bottom of the dialog box:
Scale to Fit Paper: Enlarges the slides to the maximum size they can be and still fit on the layout
Frame Slides: Draws a black border around each slide image. Useful for those slides being printed with white backgrounds.
Print Comments: Prints any comments that you have inserted with the Comments feature in PowerPoint.
Print Hidden Slides: Includes hidden slides in the printout. This option is not available if you do not have any hidden slides in your presentation.
High Quality: Optimizes the appearance of the printout in small ways, such as allowing text shadows to print.
11. (Optional) Click the Preview button to see a preview of your handouts; then click the Print button to return to the Print dialog box.
12. Click OK. The handouts print, and you are ready to roll!

5.13.3 Setting printer-specific options

In addition to the controls in the Print dialog box in PowerPoint, there are controls you can set that affect the printer you have chosen. In the Printer section of the Print dialog box, you can open the Name drop-down list and choose the printer you want to use to print the job.

5.13.4 Creating Speaker Notes

Speaker notes are like handouts, but for you. Only one printout format is available for them: the Notes Pages layout. It consists of the slide on the top half (the same size as in the two-slides-per-page handout) with the blank space below it for your notes to yourself.

Speaker notes printed in PowerPoint are better than traditional note cards for several reasons. For one thing, you can type your notes right into the computer and print them out on regular paper. There is no need to jam a note card into a typewriter and use messy correction fluid or erasers to make changes. The other benefit is that each note page contains a picture of the slide, so it is not as easy to lose your place while speaking.

5.13.5 Typing speaker notes

You can type your notes for a slide in Normal view (in the notes pane), or in Notes Page view. The latter shows the page more or less as it will look when you print your notes pages; this can help if you need to gauge how much text will fit on the printed page.

To switch to Notes Page view, on the View tab click Notes Page as shown in Figure 5.31. Unlike some of the other views, there is no shortcut button for this view in the bottom-right corner of the PowerPoint window.

Once you are in Notes Page view, you can zoom and scroll just like in any other view to see more or less of the page at once. You can scroll further to move from slide to slide, or you can move from slide to slide in the traditional ways (the Page Up and Page Down keys on the keyboard or the Next Slide or Previous Slide buttons onscreen).

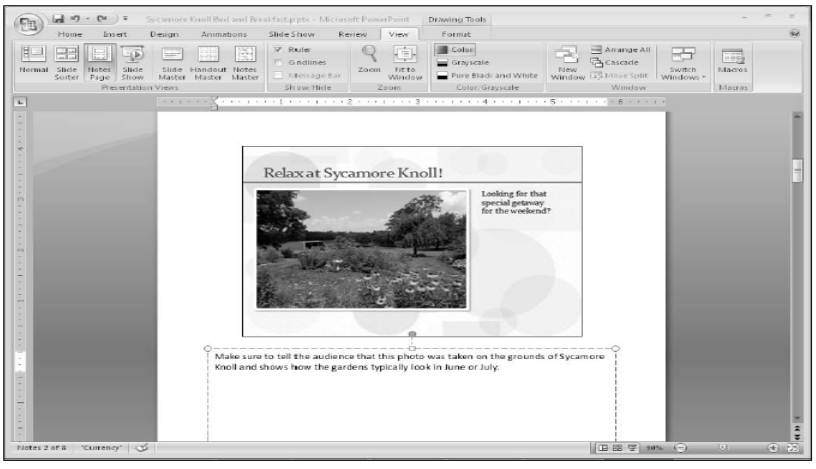


Figure 5.31: Notes Page view is one of the best ways to work with your speaker notes.

5.14 Adding Graphics, Sounds and Movies to a Slide

You can enhance a slide by adding a video clip or sound. Inserting a video clip (also called movie clips or files) or adding sound to a slide can add interest to your presentation, but relying on multimedia effects to carry your presentation often backfires. Too much sound and motion during a presentation is ultimately

confusing to your audience. Used judiciously, however, video and sound effects can quickly catch your viewers' attention.

5.14.1 Insert a video or movie clip

You can insert one or more video clips — that is, movies — into any slide. You might want to play a video quote from your product manager, for example, or run a short documentary movie for a fund-raising event. You could even create a video for product tutorials and educational materials.

Most of the movie clips stored in the Clip Organizer are simply animated .gif files, small files that contain an animated sequence of images. If you imported your own movie files into the Clip Organizer, you will find them there, too, generally stored in .avi format. You might be able to find additional movie clips on your network, intranet, or the Internet.

Before you insert video clips, think about the environment you will be presenting in. Does the computer you will use have the necessary hardware (such as a sound card, speakers, and an enhanced video card) for playing the multimedia items during your presentation? You might consider adding only a few multimedia elements just in case. If the machine you give your presentation on does not support them, plan an alternate way to make your presentation compelling. With this basic detail considered, you will find that adding video and sound is the same as adding any other item to your slides, and the special effects are truly exciting.

To insert a movie into a slide, follow these steps: Insert a movie using Clip Organizer

1. Display the slide in which you want to add a video clip.
2. On the Insert menu, point to Movies and Sounds, and then click Movie from Clip Organizer.

You can now browse the Clip Organizer for a movie clip. PowerPoint will open the Clip Art task pane and will select the Movies option in the Results should be drop-down list. You can use the Clip Art task pane to locate a video clip stored on your computer or on the Office Online Web site.

3. Click the movie you want to add to your slide.

Note some of the layouts you can apply to a slide (using the Slide Layout task pane) add a group of buttons to the slide for inserting various types of objects. These are known as Content Layouts or Text and Content Layouts. If you have applied such a layout to your slide, you can insert a video clip from the Clip Organizer by clicking the Insert Media Clip button and selecting a video clip in the Media Clip dialog box (which displays both movie and sound clips). However, the Clip Art task pane (or the freestanding Clip Organizer program) provides more features for locating and inserting a video clip and usually makes the task easier.

5.14.2 Insert a Movie from a File

You can insert a movie from a movie file on a local disk, a shared network location, or an Internet site.

1. On the Insert menu, point to Movies and Sounds, and then click Movie from File.
2. Select the movie file in the Insert Movie dialog box, and then click the OK button.

If you selected a movie clip in animated .gif format, the animation will run automatically the entire time the slide is displayed in a slide show. If, however, you selected a movie in another format (such as .avi), PowerPoint will display a message box letting you choose when the movie will play.

- To have the movie play automatically when the slide is displayed in a slide show, click the automatically button.
- To have the movie play only after you click the movie image, click the When Clicked button. PowerPoint will then add the clip(?) to your slide.

Resize the video image (if necessary)

Select the movie in the slide and then drag the sizing handles that are displayed around the image.

- To maintain the image's original proportions, drag one of the corner sizing handles.
- To change the original proportions, drag one of the sizing handles in the center of a border.
- To change the image's size without displacing the center of the image (that is, to resize it symmetrically about the center), press CTRL while you drag a sizing handle.
- To move the video image, drag it to a new location.

Modify the way the video plays during a slide show

1. Right-click the video image in your slide.
2. Do any or all of the following:

- To adjust the volume or to change the video object display options, on the shortcut menu, click

Edit Movie Object, and then select the options you want. The Movie Options dialog box also indicates where the movie clip is stored. If it is stored within the presentation file, the dialog box will display the location Contained In Presentation. If it is stored in a separate linked file, it will display the file path. In the latter case, if you are going to present your slide show on another computer, you will need to take the linked file with you. The easiest way to copy a presentation plus all linked files to a portable medium that you can bring to another computer is to use the new Package for CD feature.

- To modify the way the video clip plays during a slide show, on the shortcut menu, click Custom

Animation, and then use the controls in the Custom Animation task pane.

- To control the action that takes place when you either click the video image or move the mouse

pointer over it, on the shortcut menu, click Action Settings.

- To preview the video clip, click Play Movie.

5.14.2 Insert Sound

Sound effects, such as music and voice recordings, can add another level of professionalism to your slide presentations. Music is an effective way to introduce or end a presentation and it gives your audience something to listen to as they enter and leave the presentation room. You could play a movie theme song as background music for several slides, or play a voice recording that contains advertising slogans to insert on a single slide, for example.

You can find sound files in several places— in the Clip Organizer, in the folder where you have chosen to store your audio files, on the Internet, or from a network folder. There are many sound file formats; two common ones are wave files (with the .wav filename extension) and MIDI files (with the .mid, .midi, or .rmi filename extension). After you add a sound to a slide, you will see a Sound icon, as shown below.

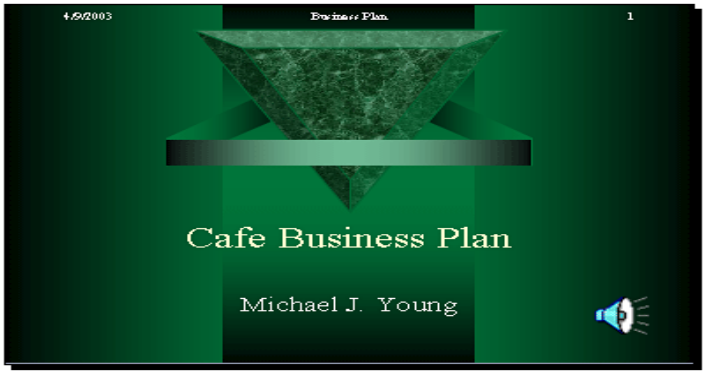


Figure 5.32: Sound icon in slide.

To add a sound clip to your slide show, perform the following steps:

1. Display the slide to which you want to add sound.
2. Use one of the following three methods to insert the sound clip:

- To browse through the Clip Organizer for a sound clip, on the Insert menu, point to Movies and Sounds, and then click Sound from Clip Organizer. PowerPoint will display the Clip Art task pane and will select Sounds in the Results Should Be drop-down list. You can use the Clip Art task pane to locate a sound clip stored on your computer or on the Office Online Web site.
- To insert a sound clip from an existing sound file on a local disk, a shared network location, or an Internet site (if you have not imported the file into the Clip Organizer), on the Insert menu, point to Movies and Sounds, and then click Sound from File. The Insert Sound dialog box will appear. Select the sound file that you want, and then click the OK button.
- To record your own sound clip (you must have a microphone attached to your computer) and add it to the slide, on the Insert menu, point to Movies and Sound, and then click Record Sound. PowerPoint will display the Record Sound dialog box. Use the controls in this dialog box to record your sound and then click the OK button.

Did You Know?

Some of the layouts you can apply to a slide (using the Slide Layout task pane) add a group of buttons to the slide for inserting various types of objects. These are known as Content Layouts or Text and Content Layouts. If you have applied such a layout to your slide, you can insert a sound clip from the Clip Organizer by clicking the Insert Media Clip button and selecting a sound clip in the Media Clip dialog box (which displays both movie and sound clips). However, the Clip Art task pane (or the freestanding Clip Organizer program) provides more features for locating and inserting a sound clip and usually makes the task easier.

If you recorded the sound clip, the sound will play during a slide show only when you click the sound icon. (You can modify this behaviour using the Custom Animation task pane.) If, however, you inserted a sound clip from the Clip Organizer or from a sound file, PowerPoint will display a message box letting you choose when the sound clip will play.

- To have the clip play automatically when the slide is displayed in a slide show, click the automatically button.
- To have the clip play only after you click the sound icon, click the When Clicked button.

Resize the Sound Icon

Select the sound icon in the slide and drag the sizing handles that are displayed around the object.

- To maintain the icon's original proportions, drag one of the corner sizing handles.
- To change the original proportions, drag one of the sizing handles in the center of a border.
- To change the icon's size without displacing the center of the object (that is, to resize it symmetrically about the center), press CTRL while you drag a sizing handle.
- To move the sound icon, drag it to a new location.

Modify the Way the Sound Clip Plays During a Slide Show

1. Right-click the sound icon in your slide.
2. Do any or all of the following:

- To adjust the volume or change the sound object display options, on the shortcut menu, click Edit Sound Object and then select the options you want. The Sound Options dialog box also indicates where the clip is stored. If it is stored within the presentation file, the dialog box will display the location Contained In Presentation. If it is stored in a separate linked file, it will display the file path. (PowerPoint normally stores a sound clip in a separate file if it is larger than 100 KB.) In the latter case, if you are going to present your slide show on another computer, you will need to take the linked file with you.

  The easiest way to copy a presentation plus all linked files to a portable medium that you can bring to another computer is to use the new Package for CD feature.
  - To modify the way the sound clip plays during a slide show, on the shortcut menu, click Custom Animation, and then use the controls in the Custom Animation task pane.
  - To control the action that takes place when you either click the sound icon or move the mouse pointer over it, on the shortcut menu, click Action Settings.
  - To preview the sound clip, on the shortcut menu, click Play Sound.

5.15 Summary

- PowerPoint 2007 adds new text formatting capabilities to help users further polish their work.
- Regular text could not receive WordArt formatting such as reshaping, stretching, and distortion
- SmartArt uses groups of lines and shapes to present text information in a graphical, conceptually meaningful way.
- SmartArt convert a bulleted list into a conceptual diagram in just a few clicks.
- Notes Page view is accessible only from the View tab.
- Graphics and videos can be added to the presentation.

5.16 Keywords

Gridlines: These are non-printing dotted lines at regularly spaced intervals that can help you line up objects on a slide.

***Single File Web Page*:** It creates a single .mht document that contains all of the HTML codes and all of the slides.

***Slide Show*:** The view you use to show the presentation on-screen. Each slide fills the entire screen in its turn

SmartArt: It can convert a bulleted list into a conceptual diagram in just a few clicks.

Template: It is a file that contains starter settings on which you can base new presentations.

5.17 Review Questions

1. Explain working of MS PowerPoint?
2. Explain the basic steps for creating PowerPoint slide.
3. Describe the tools in PowerPoint window.
4. Explain all categories of temples.
5. Discus steps used for save and open PowerPoint files.
6. What is process of close and reopen presentations?
7. How to create new slides from the outline pane?
8. What are uses of animation and transitions?

9. What is a handout? Explain the creation.
10. How we can insert graphics, sounds and movies into a slide?

Answers for Self Assessment Questions

1 (c) 2 (d) 3 (a) 4 (b)

5.18 Review Questions

1 Microsoft Office 2007 Power Point: A Training Book for Microsoft PowerPoint 2007 by W R Mills

6

Outlook Express

STRUCTURE

6.0 Objectives

6.1 Introduction

6.2 Setup E-mail Account with Outlook

6.3 Sending and Receiving Mail through Outlook

6.4 Concepts of Cc and Bcc

6.5 Forwarding Mail

6.6 Draft Messages

6.7 Formatting E-mail Message

6.8 Concept of MIME

6.9 Outlook Protocol

6.10 Attaching Files and Items into Messages

6.11 Inserting Hyperlink Using Outlook

6.12 Creating and using Send and Receive Group E-mails

6.13 Opening Received Messages

6.14 Opening Messages with Attachment

6.15 Forwarding Messages

6.16 Flagging for Further Action

6.17 Setting E-mail Options

6.18 Managing Contacts with Outlook

6.19 Setting up Multiple E-mail Accounts on Single Machine

6.20 Summary

6.21 Keywords

6.22 Review Questions

6.23 Further Studies

6.0 Objectives

After studying this chapter, you will be able to:

- Explain the setup e-mail account with outlook

- Discuss the sending and receiving mail through outlook
- Explain the concepts of Cc and Bcc
- Explain the forwarding mail
- Explain the draft messages
- Discuss about the formatting e-mail message
- Explain the concept of MIME
- Discuss the outlook protocol
- Discuss about the attaching files and items into messages
- Understand the inserting hyperlink using outlook

6.1 Introduction

Outlook Express is a free online communication tool from Microsoft that you can use for e-mail or newsgroups. It is included with Microsoft Internet Explorer 6 for Windows operating systems. With Outlook Express, you can download your e-mail messages from the UH mail server onto your computer's local hard drive. Outlook Express also allows you to view old mail messages and compose new mail messages off-line and simplifies reading and sending attachments. It allows you to receive mail from multiple e-mail accounts, as well as create Inbox rules that allow you to manage and organize your e-mail.

6.2 Setup E-mail Account with Outlook

To Set Up Your E-mail Account in Microsoft Outlook 2007

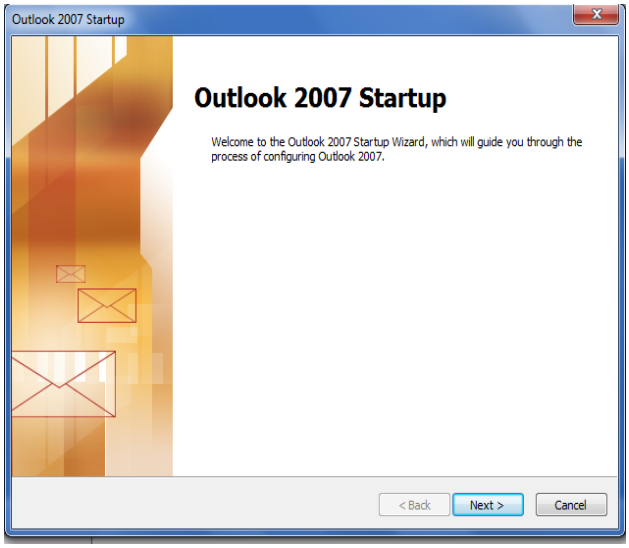
1. Start Microsoft Outlook 2007. When the Startup Wizard displays, click Next.
2. 

Figure 6.1: Microsoft Outlook 2007.

2. On the Account Configuration page, select yes to indicate you want to configure an e-mail account, and then click Next.

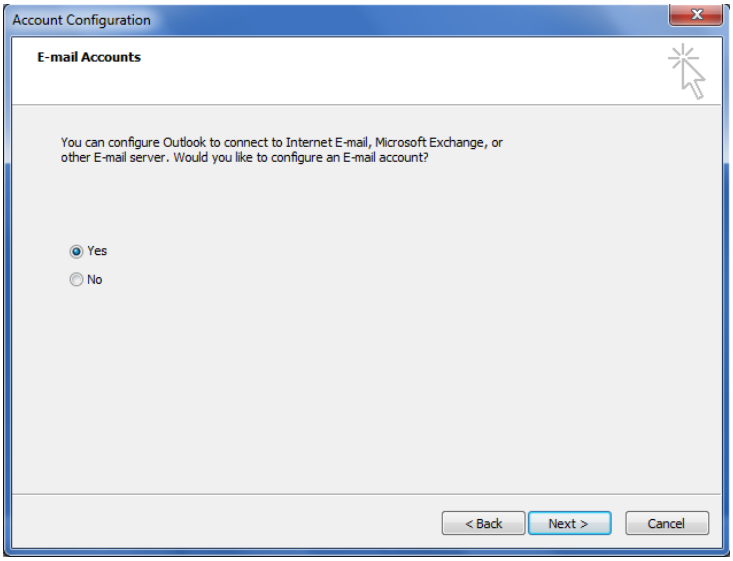


Figure 6.2: Account Configuration page.

3. On the Auto Account Setup page, enter the following:

- Your Name
- Enter your first and last name.
- E-mail Address
- Enter your e-mail address.
- Password
- Enter the password you created for your e-mail account.
- Retype Password
- Enter your password again.

4. At the bottom of the page, select manually configure server settings or additional server types, and then click Next.

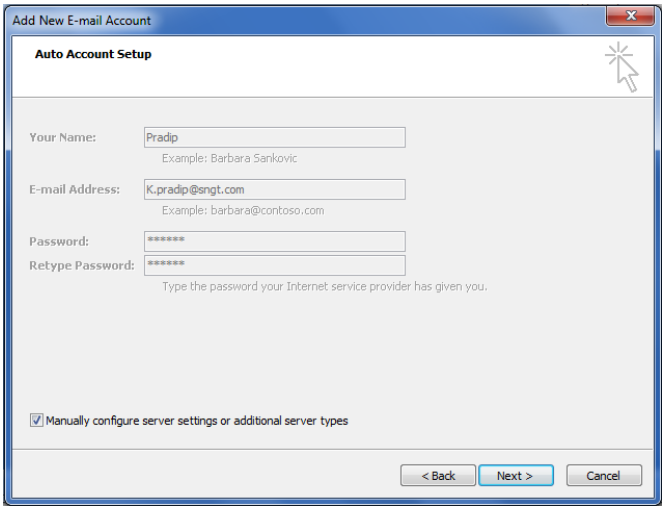


Figure 6.3: Add new e-mail account.

5. On the choose e-mail service page, select Internet e-mail, and then click Next.

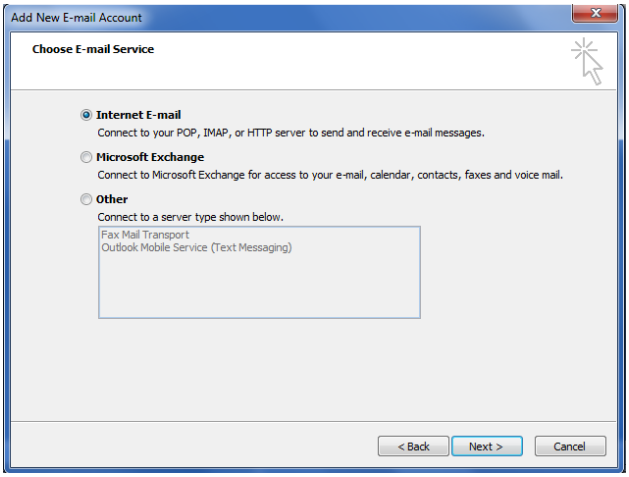


Figure 6.4: Choose e-mail service page.

6. On the Internet e-mail Settings page, enter your e-mail account information as follows:

- Your Name
- Enter your first and last name.
- E-mail Address
- Enter your e-mail address.
- Account Type
- Select POP3.
- Incoming mail server

Type pop-1.mail.vi.net for your incoming mail server. Outgoing mail server (SMTP)
Type smtp-1.mail.vi.net for your outgoing mail server.

- User Name
- Enter your e-mail address again.
- Password
- Enter the password you created for your e-mail account.

7. Select the Remember Password checkbox, and then click More Settings.

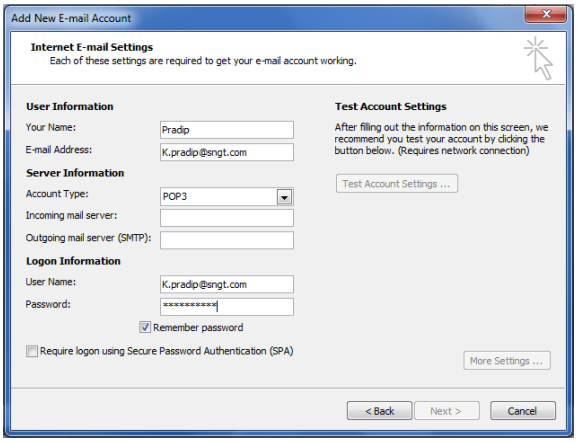


Figure 6.5: Password checkbox.

8. Click Finish.

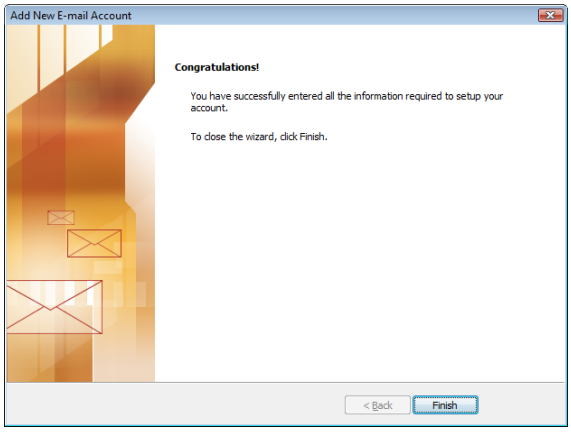


Figure 6.6: Complete account.

Did You Know?

If the Startup Wizard does not automatically display, from the Tools menu, select Account Settings, and then click New. In the Add New e-mail Account dialog box, select Microsoft Exchange, POP3, IMAP, or HTTP, and then click next.

6.3 Sending and Receiving Mail through Outlook

Follow the steps below to sending and receiving mail through Microsoft Outlook 2007:

1. Create a new account by selecting Tools > Account Settings.

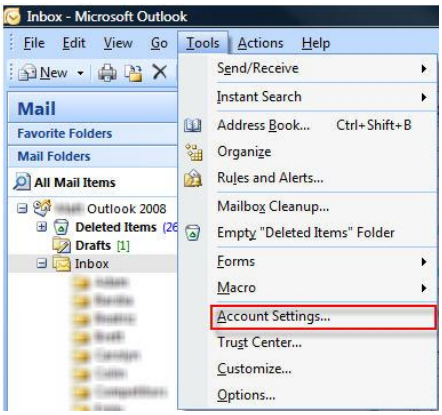


Figure 6.7: Account Settings.

2. The Account Settings window appears. Click New to add a new e-mail account:

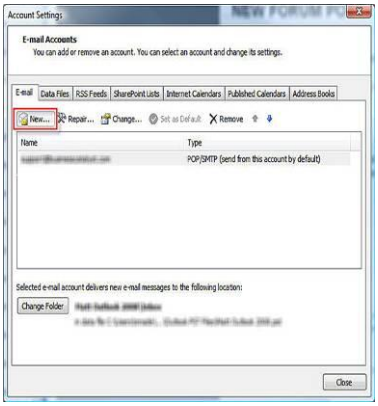


Figure 6.8: new e-mail account.

Select the option: Microsoft Exchange, POP3, IMAP, or HTTP and click Next.
The hosted system supports POP3 e-mail servers, and you cannot use any other protocol in the list.

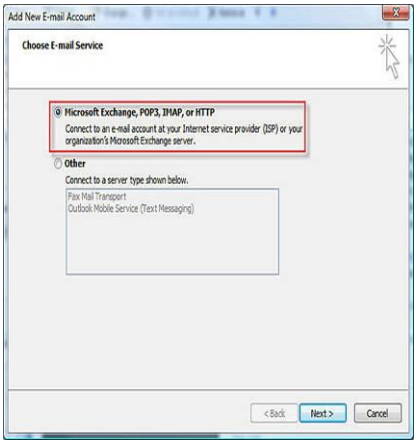


Figure 6.9: Choose e-mail services.

Select the option in the bottom left corner to manually configure server settings or additional server settings. Click Next.

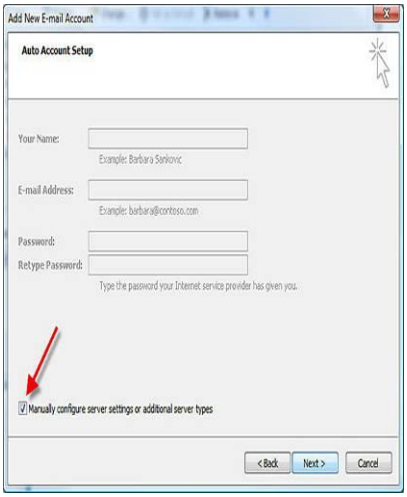


Figure 6.10: manually configure server settings.

Next, fill in the account information, using the following details:

- **Your Name**: Enter your name this is the name that will be displayed to recipients of your sent messages.
- **E-mail address**: This is the POP e-mail address, enter the same e-mail you use for your user name when accessing the Admin Console.
- **Username**: Enter the same user name (e-mail address) you used to create the account in the Admin Console.
- **Password**: Enter the same password you used when creating the account.
- **Incoming mail server (POP3)**: Enter mail.my_domain_name.com (replacing my_domain_name.com with the site's domain name).
- **SMTP or sending server**: Try entering mail.my_domain_name.com (replacing my_domain_name.com with the site's domain name).

However, if you have issues sending e-mail messages after setting this up, the network communication to third-party SMTP servers may blocked by your Internet Service Provider (ISP). This occurs because the ISP is inhibiting SPAM messages from being sent. To resolve this, you may need to enter your ISP's sending (SMTP) server here. To get this information, check one of your existing e-mail accounts. If this is the first e-mail account you have set up, contact your Internet Service Provider and ask them for the SMTP server information.



Figure 6.11: Internet Service Provider.

Click More Settings. In the General tab, enter your name and a reply e-mail address. Usually, this is the same as your e-mail address.

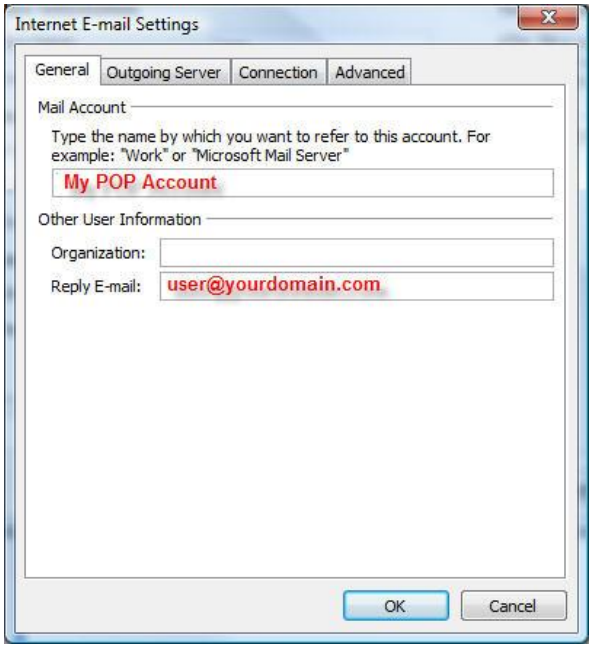


Figure 6.12: General tab.

Click the Outgoing Server tab. If you are using this service's server for sending e-mail, select the option: My outgoing server (SMTP) requires authentication.
If you are using your Internet Service Provider's SMTP server, it is likely you will not need to enable this option, but check with them to verify the settings to use.

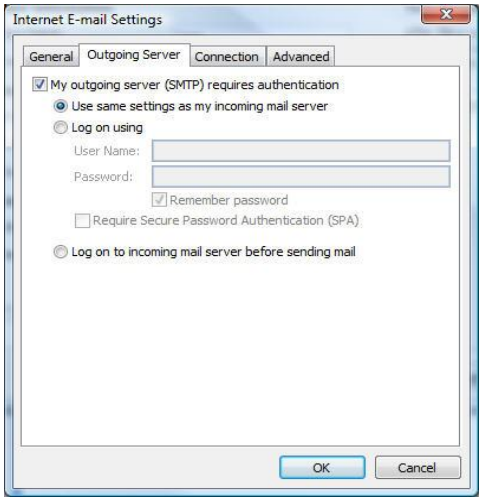


Figure 6.13: Verify the settings to use.

Click the Advanced tab. Verify that the port numbers are set to 110 and 25. Make sure the other options are not selected.
Click OK and then click Finish.

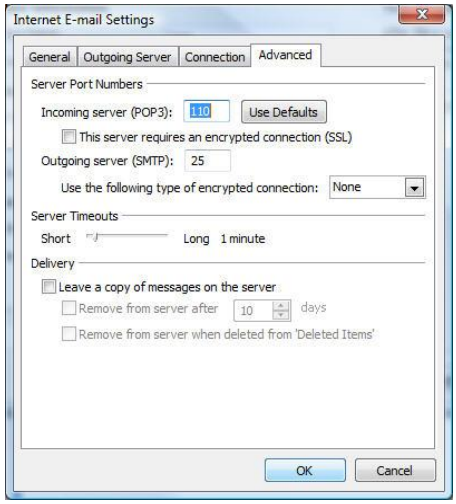


Figure 6.14: OK and then click Finish.

Test the new e-mail account to verify that you can send and receive mail, by clicking Send/Receive.
Did You Know?
The maximum size of one piece of mail is to 10 MB for both transmission and reception.

6.4 Concepts of Cc and Bcc

6.4.1 The Cc Field

When you click the "Create Mail" or "New Mail" button in your e-mail program, whether it is Outlook Express or not, no doubt you have noticed the Cc field. The Cc stands for Carbon Copy. You would normally use this field to include the e-mail address of someone you wanted to receive a copy of the e-mail you are sending. For example, if you were part of a work team and you were e-mailing your boss about something, you would put your boss's e-mail address in the e-mail's "To" field, and if you wanted your workmates to be aware of that e-mail, you might put their e-mail addresses in the "Cc" field. Your e-mail would then be sent to both your boss and your workmates at the same time. All recipients would be able to see exactly who the e-

mail was sent to (in this case the boss) and they would be able to see everyone that received a copy of the e-mail. The picture below demonstrates how Outlook Express displays information about who the e-mail was sent to. In this case sent an e-mail, and copied it to friend. You can clearly see that in the Outlook Express display. The snapshot is taken from the grey bar just above the Outlook Express display window.



Figure 6.15: The Cc Field.

The e-mail's headers also tell me that the message was sent to nav@sngt.com and copied to nav@asngt.com at the same time.
From: "Navneet" editor@sngt.com
To: nav@sngt.com
Cc: k.rah@gmail.com
Subject: test
(You can check an e-mail's headers in Outlook Express by right-clicking on the e-mail in your inbox and left clicking on Properties.)

6.4.2 The Bcc Field

If anyone want to send an e-mail to someone, and copy it to someone else, but he do not want the person, sending it to know that someone else is receiving a copy? That is where the Bcc (Blind Carbon Copy) field is used. Most of the time you would use the Bcc field when you are being sneaky about something. If you were having a dispute with a colleague and you wanted to copy your boss in on your e-mail dealings with that colleague, but you did not want your colleague to know you were doing that, then you would put your boss's e-mail address into the Blind Carbon Copy field (after checking on relevant privacy laws). It use the Bcc field when It dealing with someone and for whatever reason we want to keep a copy of the e-mail on another computer. If so, it would send the e-mail with an e-mail address in the Bcc field that is only checked by the computer we want the copy of the e-mail on. It is easier for me to hide the fact that we doing that than to explain to whoever as to why copying someone else in on the e-mail. We are not being sneaky; it is just more convenient for me not to have to explain.

So, who sees what?

You have to be careful using the Bcc field if you are being sneaky. Plenty of people mix up which e-mail address goes where and end up revealing their sneakiness to the very person they were trying to conceal it from. The Bcc field has legitimate uses and it is up to you to decide when it is appropriate to use. Using the hypothetical example above. If we put nav@sngt.com into my e-mail's "To" field, and k.rah@gmail.com into the Bcc field, then Rob would receive the e-mail, but would not know that Norrie has been copied in. Norrie will receive an e-mail in his inbox that does not have his e-mail address on it. Instead it appears as below.



Figure 6.16: The Bcc Field

Now if Norrie sees that, he should right away be able to tell that his e-mail address has been put into the Bcc field. It is pretty obvious as an e-mail has appeared in his inbox that is clearly addressed to someone else. This

may ring a bell for some of you who have noticed you have received Spam messages that do not appear to be addressed to you. Obviously they are being sent to someone else and your e-mail address has been included in the Bcc field.

Received: by xyz@abc.com
Delivered-To: tqs@nmj.com
Message-ID: <011201c1b34d$48eed570$4d1560cb@ZORNCAT>
From: "nav" editor@abc.com
To: vb@abc.com
Subject: test Bcc

This information tells him that the e-mail was delivered to him even though it was addressed to someone else, a clear sign that his e-mail address was in the Bcc field.

If that is too confusing, then perhaps we can simplify it as follows. If you do not want someone to know that the e-mail to them is being copied to someone else, put their e-mail address in the "To" field. The person whose e-mail address you put into the Bcc field will know what is going on.

Accessing the Bcc Field

Outlook Express does not display the Bcc field for e-mails by default. To get the Bcc field to appear, open up a fresh new e-mail. Click the View Menu and then click a tick next to "All Headers" in the drop down menu. From now on all your new e-mails will have the Bcc field included by default. To stop the Bcc field from appearing, just click the View menu again in a freshly opened e-mail and click to remove the tick.

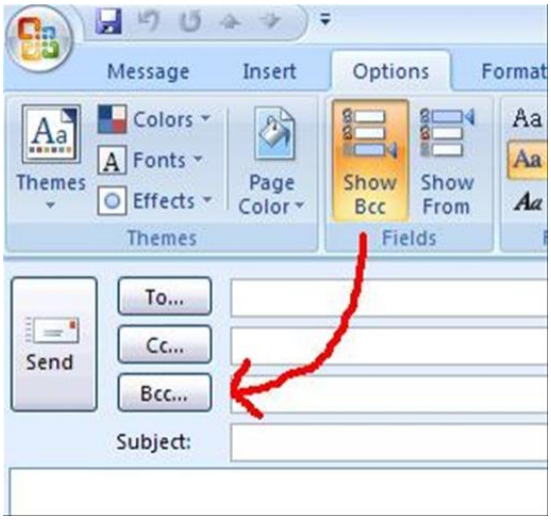


Figure 6.17: Accessing the Bcc Field.

6.5 Forwarding Mail

When you forward a message, you specify the recipients in the To, Cc, and Bcc boxes. The message will include any attachments that arrived with the original message. To add additional attachments, see Attach a file or other item to an e-mail message.



Figure 6.18: Add additional attachments.

1. Do one of the follow:

- When the message is selected but not opened in its own window On the Standard toolbar, click forward.
- When the message is open On the Message tab, in the Respond group, click forward.

2. Compose your message.
3. Enter recipient names in the To box.

To select recipient names from a list, click the To, Cc, or Bcc button. More information about the Cc and Bcc boxes

4. Click Send.

You must have at least one valid e-mail recipient in the To, Cc, or Bcc box to send a message.

Did You Know?

If you want to forward two or more messages, select the first message, press CTRL while selecting each additional message, and then click forward. Each message will be forwarded as attachments in a new message.

6.6 Draft Messages

By default, Microsoft Office Outlook automatically attempts to save any e-mail message that you create but do not send. You can return to Outlook later and find the unsent message. Save a draft of a message that you want to finish later.

To manually save a draft of a message, do one of the following:

Click Save on the Quick Access Toolbar or click the Microsoft Office Button, and then click Save.



Figure 6.19: Click the Microsoft Office Button, and then click Save.

To return to a saved draft so that you can continue composing the message, do the following:

In Mail, in the All Mail Items, looks for a folder called Drafts, and then double-click the message that was saved in that folder.

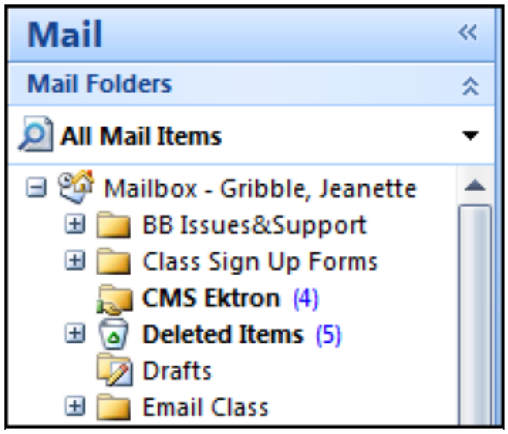


Figure 6.20: All Mail Items.

Change the time interval or location for saving unfinished items

Microsoft Office Outlook automatically saves all unfinished messages for you. By default, unfinished messages are saved to your Drafts folder every three minutes. You can, however, change this time.
On the Tools menu, click Options.
Click Preferences tab, then e-mail Options button, then Advanced e-mail Options button.
You have some choices here but a suggestion is leaving the drafts be saved to the Drafts folder.
You can change how frequently drafts are saved
Under Save messages, in the AutoSave unsent every minute's box, type a number from 1 to 99.

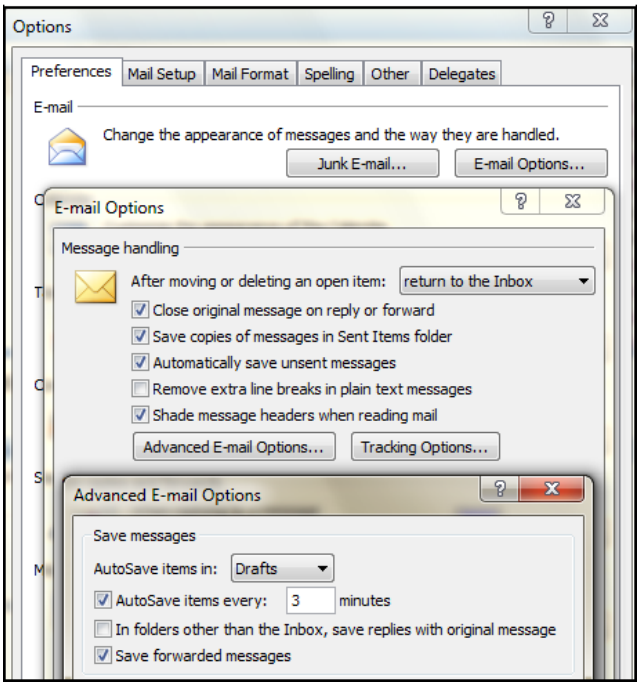


Figure 6.21: Advanced e-mail options button.

Be sure to click OK on each window after making a change.

Caution

When creating a long mail document, save a draft during mail creation. When mail being created remains as is without being saved for 120 minutes, the session may be interrupted, and the created mail may be lost.

Self Assessment Questions

1. ……..is proprietary to Microsoft e-mailing software.

(a) Plain text format (b) Rich text format
(c) HTML format (d) Mail Format.

2. ……is the collaboration application in Microsoft's Office productivity suite.

(a) Microsoft Outlook (b) Microsoft Office
(c) Microsoft Exchange (d) Microsoft Internet Explorer.

3 Microsoft Outlook works with…….Microsoft proprietary mail server for businesses.

(a) Microsoft Outlook (b) Microsoft Office
(c) Microsoft Exchange (d) Microsoft Internet Explorer.

4. Internet……event items may be sent via e-mail as a one-time calendar snapshot.

(a) Hotmail (b) Mail

(c) Microsoft Exchange (d) Calendar.

6.7 Formatting E-mail Message

Microsoft Outlook 2007 offers three formats for sending e-mail messages: HTML, plain text, and rich text. Here is a look at the pros and cons of the different Outlook e-mail formats, and some easy ways to change them.

Microsoft Outlook 2007 e-mail formats come in three flavors:

- **HTML format:** These days, almost all e-mail is transmitted in HTML format, the same format with which Web pages are made. If HTML is the default format you use for creating messages in Outlook — and it is, unless you have tinkered with the default settings — the e-mail messages you send are, in effect, little Web pages. The HTML gives you the most opportunities for formatting text and graphics. In HTML format, you can place pictures in the body of an e-mail message, use a background theme, and do any number of sophisticated formatting tricks.

  However, the HTML format has it share of detractors. First, the messages are larger because they include sophisticated formatting instructions, and being larger, they take longer to transmit over the Internet. Some e-mail accounts allocate a fixed amount of disk space for incoming e-mail messages and reject messages when the disk space allocation is filled. Because they are larger than other e-mail messages, HTML messages fill the disk space quicker. Finally, some e-mail software cannot handle HTML messages. In this software, the messages are converted to plain text format.
- **Plain text format:** In plain text format, only letters and numbers are transmitted. The format does not permit you to format text or align paragraphs in any way, but you can rest assured that the person who receives the message can read it exactly as you wrote it.
- **Rich text format:** The rich text format is proprietary to Microsoft e-mailing software. Note that only people who use Outlook and Outlook Express can see rich text formats. If formatting text in e-mail messages is important to you, choose the HTML format because more people can read your messages.

When someone sends you an e-mail message, you can tell which format it was transmitted in by looking at the title bar, where HTML, Plain Text, or Rich Text appears in parentheses after the subject of the message. Outlook is smart enough to transmit messages in HTML, plain text, or rich text format when you reply to a message that was sent to you in that format.

Follow these instructions if you need to change the format in which your e-mail messages are transmitted:

- **Changing the default format:** Choose Tools --> Options, and in the Options dialog box, select the Mail Format tab. From the Compose in This Message Format drop-down list, choose HTML, Plain Text, or Rich Text.
- **Changing the format for a single e-mail message:** In the Message window, click the Options tab. Then click the Plain Text, HTML, or Rich Text button.
- **Always using the plain text or rich text format with a contact:** To avoid transmitting in HTML with a contact, start in the Contacts folder, double-click the contact's name, and in the Contact form, double-click the contact's e-mail address. You see the e-Mail Properties dialog box. In the Internet Format drop-down list, choose Send Plain Text Only or Send Using Outlook Rich Text Format.

6.8 Concept of MIME

The MIME files are encoded files that are mainly associated with Multi Purpose Internet Mail Extension. The MIME files are used by e-mail applications and they define multiple methods for representing binary data in

US ASCII text format. You can decode MIME files using file decompression applications, such as Corel WinZip 14.5 and Smith Micro Stuff It Deluxe 2010; however, if you have Microsoft Outlook 2007, you can open and view MIME files with without having to download any additional software.

1. Right-click on the MIME file.
2. Click on "Open with."
3. Select the "Choose Default Program" option.
4. 4 Select Microsoft Outlook 2007.
5. Click on "OK." Your MIME file will open.

6.9 Outlook Protocol

Microsoft Outlook is the collaboration application in Microsoft's Office productivity suite. Outlook can handle data served by Microsoft Exchange as well as various non-proprietary e-mail, calendar and syndication protocols.

Mail

- Microsoft Outlook works with Microsoft Exchange, Microsoft's proprietary mail server for businesses. Outlook can also handle popular e-mail protocols included POP3 (Post Office Protocol), IMAP (Internet Message Access Protocol) and SMTP (Send Mail Transfer Protocol.) Users can also access Windows Live Hotmail accounts from Outlook. Active mail accounts will show up in the Mail section of Outlook.

Calendar

- Outlook Calendar can handle calendars served by Microsoft Exchange as well as Internet Calendar (.ics) format files. Internet Calendar event items may be sent via e-mail as a one-time calendar snapshot. Whole calendars can be subscribed to over the Internet. Calendar event items and subscribed calendars will show up in the user's Outlook calendar.

Other

- Outlook can also subscribe to RSS (Real Simple Syndication) feeds to keep track of updates to websites such as news sites and blogs. Subscribed feeds show up in the Mail section of Outlook.

6.10 Attaching Files and Items into Messages

Attachments are files or items that can be sent as part of an e-mail message:

6.10.1 Attach a file to a message

1. On the File menu, click New, and then click Mail Message.
2. On the Message tab, in the Include group, click Attach File.



Figure 6.22: Include group.

3. In the Insert File dialog box, browse to and select the file that you want to attach, and then click Insert.

You do not see the file that you want in the appropriate folder, make sure that **All files (*.*)** is selected in the **Files of type** box, and that Windows Explorer is configured to show file name extensions.

Caution

By default, Outlook blocks potentially unsafe attachments (including .bat, .exe, .vbs, and .js files) that might contain viruses. If you attach such a file to an e-mail message, you will be asked whether you want to send a potentially unsafe attachment. If you answer **Yes**, Outlook will send the attachment. If you answer **No**, you can then remove the potentially unsafe attachment.

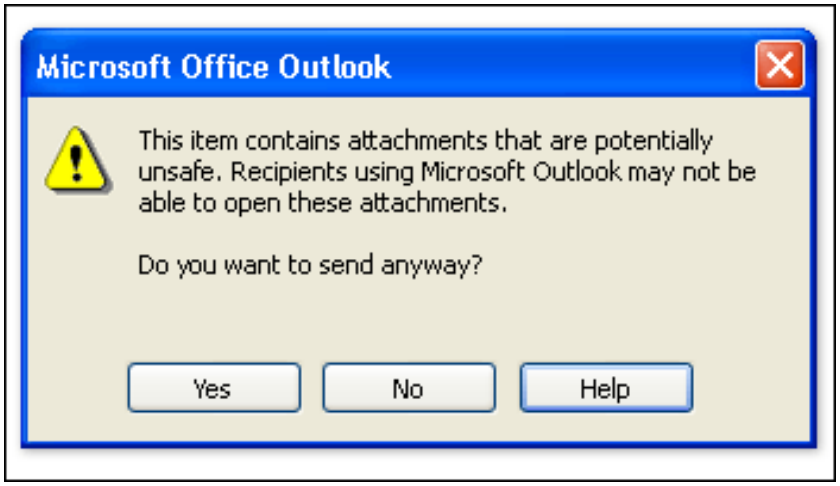


Figure 6.23: Outlook blocks.

You can attach multiple files simultaneously by selecting the files and dragging them from a folder on your computer to an open message in Outlook.

6.10.2 Attach a message or other item to a message

You can attach items and other messages to a new message:

1. On the File menu, click New, and then click Mail Message.
2. On the Message tab, in the Include group, click Attach Item.

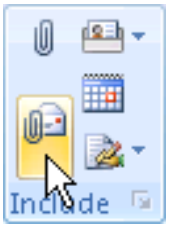


Figure 6.24: Attach Item.

3. In the Look in list, click the folder that contains the item that you want to attach.
4. Click the item that you want, and then click OK.

6.10.3 Attachment size limitations

Outlook does not impose any size limit on attachments. However, most e-mail server administrators and Internet service providers (ISPs) do set limits on the overall message size, which is the sum of all attachments plus the content of the message itself. Typically, premium-level services provide more e-mail file storage space and allow larger messages than standard-level services. For more information on the maximum allowed message size, contact your ISP or the e-mail server administrator in your organization.

If you are attaching a very large file, you should first compress the file by using a zip program. Outlook files that are saved in the default Open XML Formats with the following file extensions are automatically zipped: .docx, .dotx, .xlsx, .xltx, .pptx, .potx, and .ppsx.

6.10.4 Problems with .zip files

If files that you attach are zipped when you do not expect them to be, check to see whether a .zip utility has been installed in Outlook as a COM add-in. You can disable this sort of add-in by doing the following:

1. On the **Tools** menu, click Trust Center, and then click Add-ins.
2. In the Manage list, click COM Add-Ins, and then click Go.
3. In the COM Add-ins dialog box, clear the check box next to the add-in name, and then click **OK**.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: How to use Outlook?

……..……

……..……

……...

6.11 Inserting Hyperlink Using Outlook

1. Open Outlook from the start menu by clicking the Windows Icon in the lower left hand corner.
2. After Outlook is opened. In the upper left hand corner of the screen, click the "New" button.



Figure 6.25: After Outlook is opened.

4. Populate the To, CC, Subject fields as you normally would and type in a sentence of text including the word that you want to change into a link.

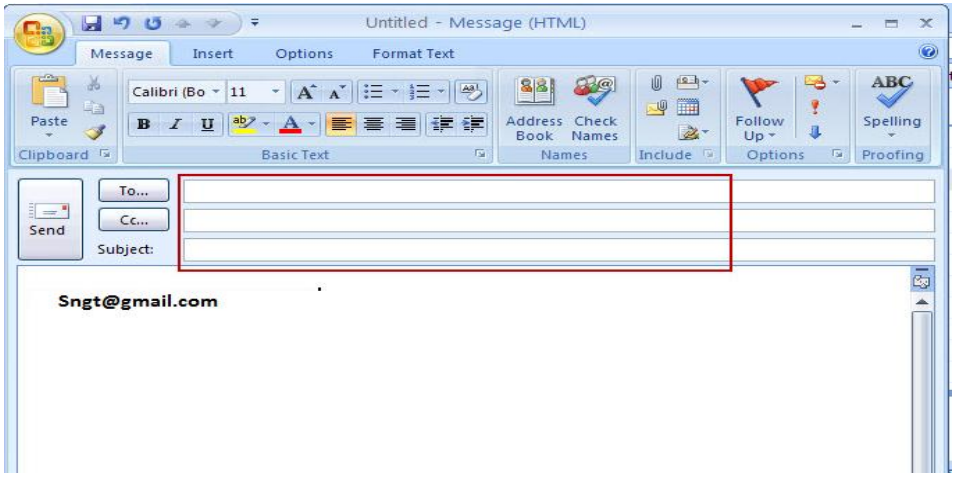


Figure 6.26: Subject fields.

5. Highlight the word or word that you want to change into a link by left clicking at the start and dragging across until you reach the end. Click the “Insert” tab at the top of the page.

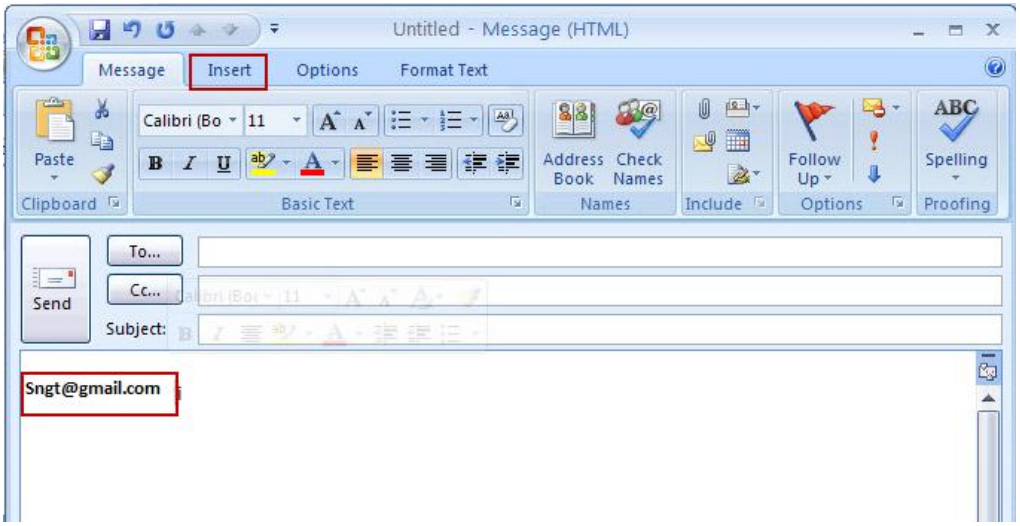


Figure 6.27 Insert

5. Click the “Hyperlink” button.

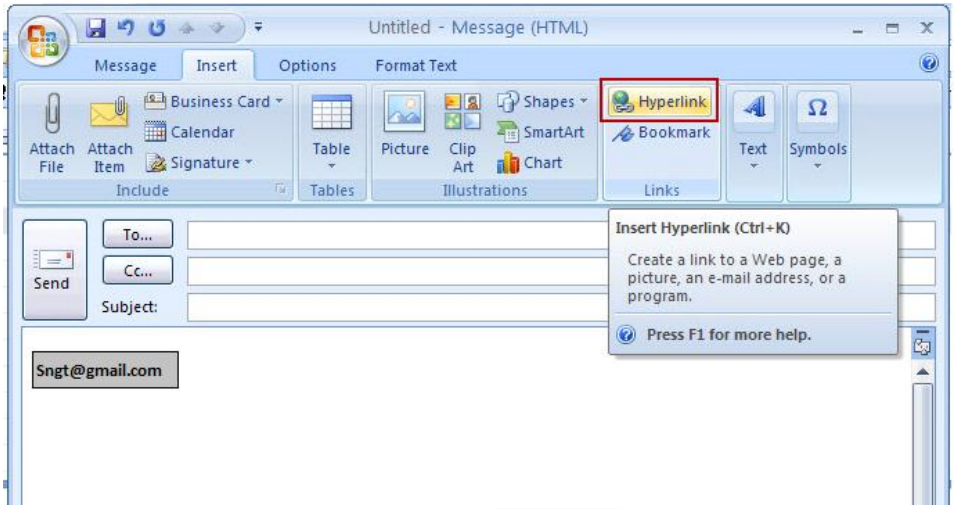


Figure 6.28: Hyperlink.

6. In the "Address" field, type the URL of the website that you want to link to and click the "OK" button.

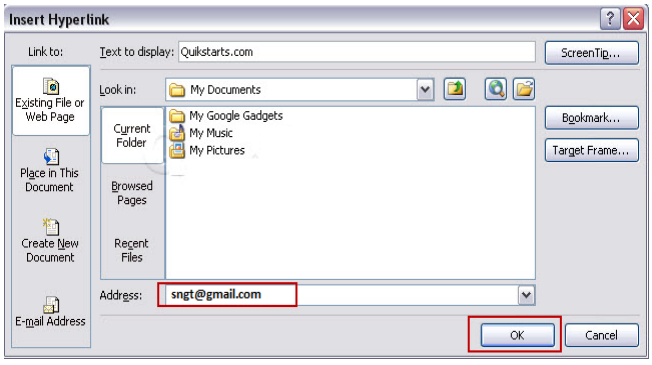


Figure 6.29: Address field.

7. The word or text you selected in step 4 should now be a clickable link. Send the e-mail as you would normally.

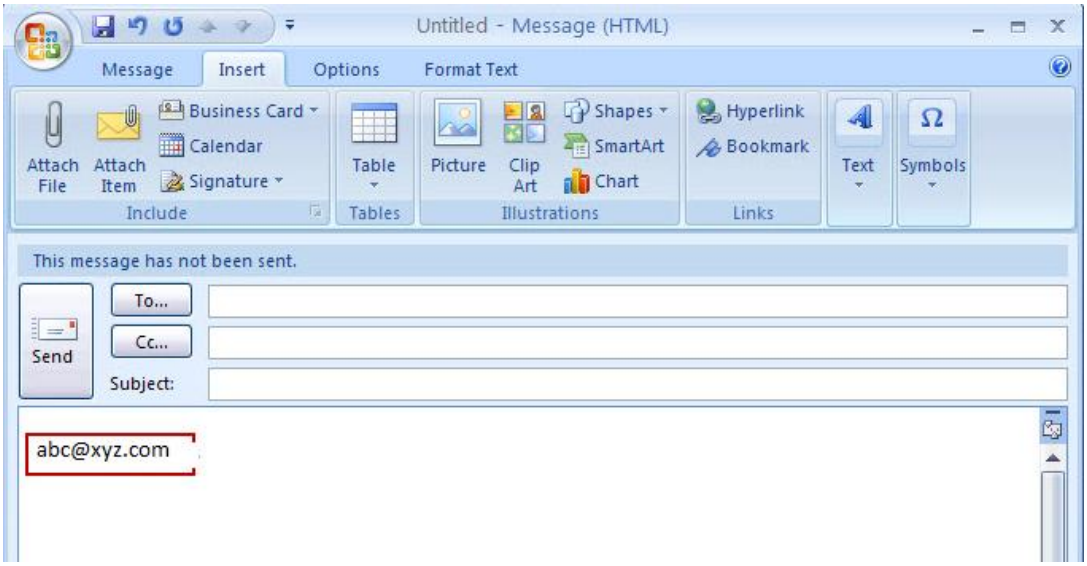


Figure 6.30: clickable link.

8. This task should now be complete. If not, review and repeat the steps as needed. Submit any questions using the section at the bottom of this page.

6.12 Creating and using Send and Receive Group E-mails

6.12.1 Creating Groups

Creating your own Groups of e-mail addresses is a handy function, as it allows you to send e-mail to a pre-defined collection of people easily, without having to remember their names or e-mail addresses. Whether you are on a server at work, trying to define a group of people to receive regular reports, or you want to set up a group at home to forward all your jokes to, this can save you a ton of time in the long run.

Part 1–Creating your own Groups

If you are in the e-mail view, click the Contact**s** button in the left navigation bar on your screen:

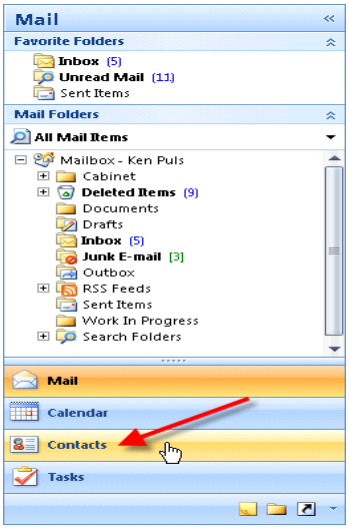


Figure 6.31: Creating your own Groups.

Now, you should be in the Contacts view. Choose Actions then New Distribution List (or press Ctrl + Shift + L)

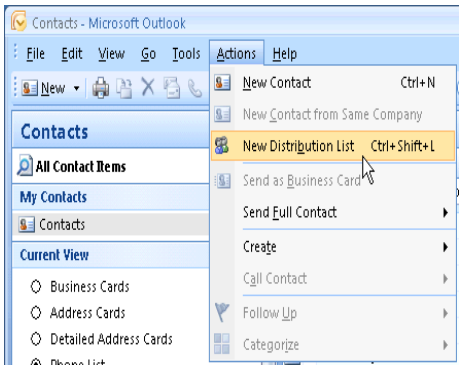


Figure 6.32: Contacts.

Next, you will want to choose Select Members from the Members group on the Ribbon.

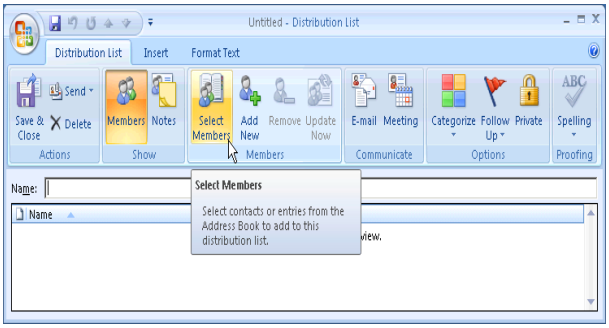


Figure 6.33: Select Members**.**

I have doubled clicked both Administrator and Citrix Resource Manager Accounts, at which point they show in the "Members ->" area at the bottom of the screen. (Pick someone you actually want to e-mail.) Once you have got a nice list of people that you had like to include in your group, click OK.

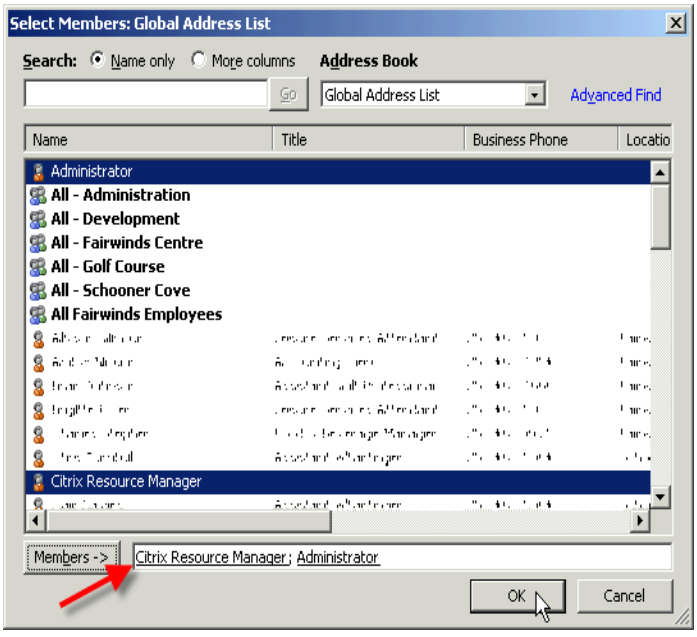


Figure 6.34: Citrix Resource Manager Accounts.

Next, give your group a name that makes sense and the click the Save and Close button on the Actions group.

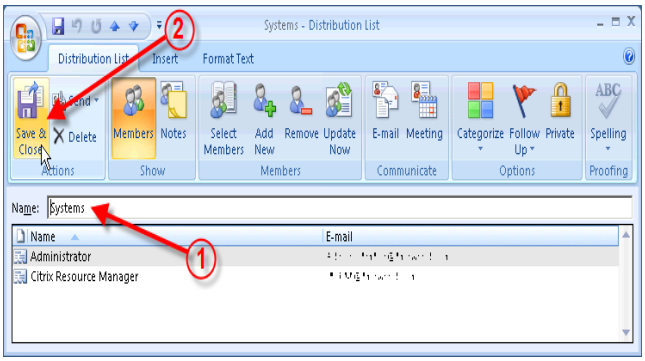


Figure 6.35: Save and Close.

Part 2–E-mailing Your Group

Okay, now the useful par let us send our group an e-mail. From the **New** menu, choose Mail Message (or just press Ctrl + Shift + M to create a new mail message.)

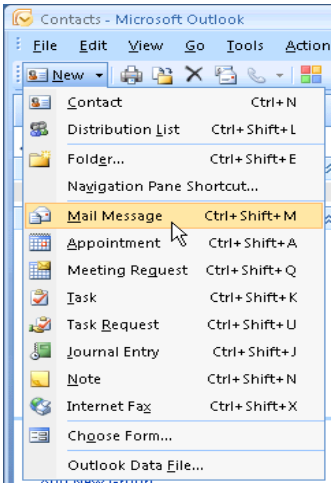


Figure 6.36: choose Mail Message.

In the e-mail message, click the To button:

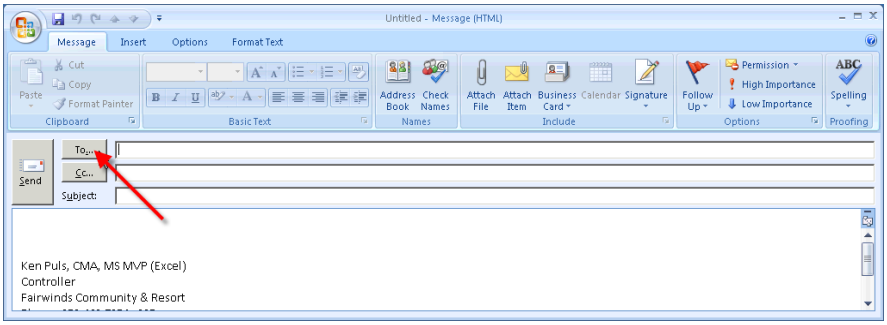


Figure 6.37: To button.

The secret to finding your new group is to click the drop down arrow and choose the address book that holds the group you created. (In this case Outlook Contacts instead of our Global Address List):

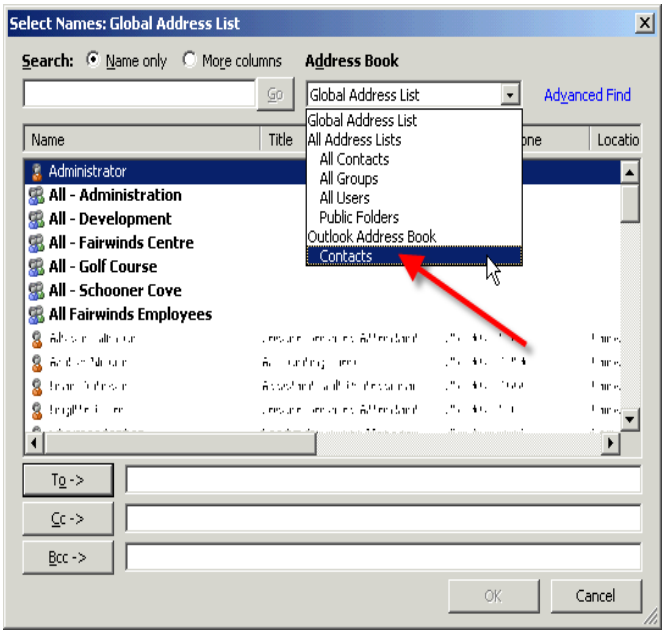


Figure 6.38: Outlook Contacts instead of our Global Address List.

And there it is! Double click it to select it, and then click OK.

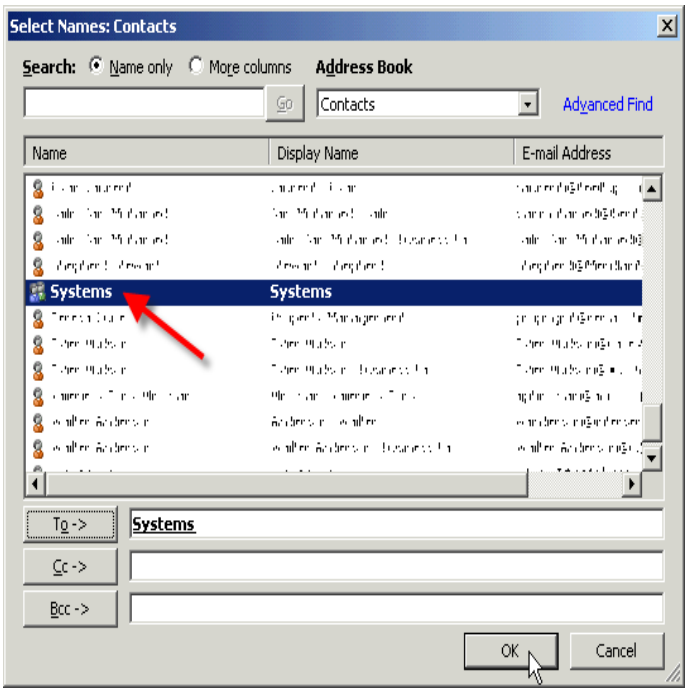


Figure 6.39: Select name Contacts.

You will notice that your e-mail is now addressed to the group. Now do you remember who you put in the group? Ok, you probably do at this stage, but as time goes on, you may forget and want to verify it. (Or maybe it is your joke list, and you want to forward it to everyone *except* the person who just e-mailed the joke to you.) Do you see the little + sign just in front of the group name?

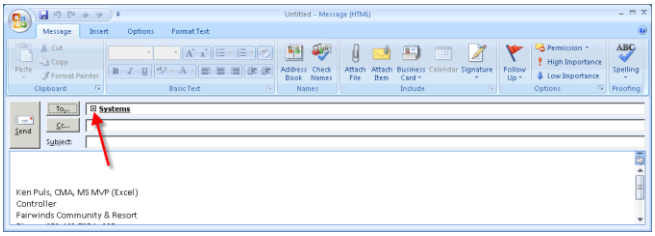


Figure 6.40: Messages Title name.

Go ahead and give it a click.

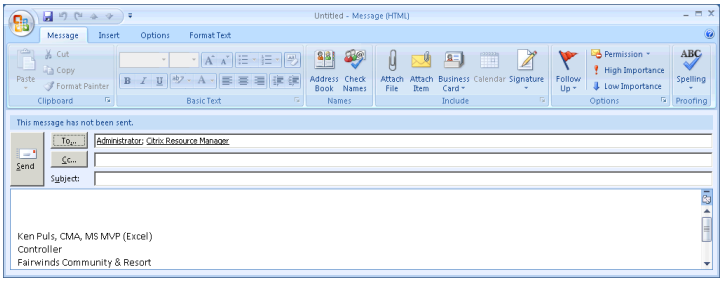


Figure 6.41: Go ahead.

The e-mail group automatically expands to show you the names of all of the members! You will never wonder who you are sending e-mail to again! And if you want to remove one, just highlight the name and press the delete key. It is just that easy!

Part 3–Editing Existing Groups

Assume now that months have gone by, and you want to make a change to your group either adding or removing someone. How do you do it? The first step is to locate the group in your contacts list. Once you have found it, right click it, and choose Open.

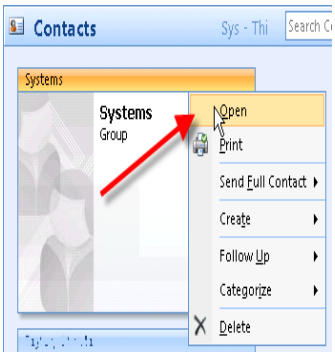


Figure 6.42: The first step is to locate the group in your contacts list. Once you have found it, right click it, and choose Open.

Your "Distribution List" will open, and show who the members are.

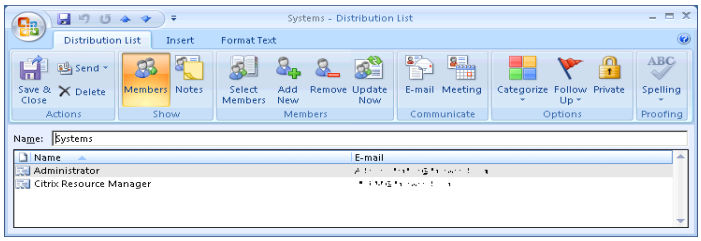


Figure 6.43: Distribution List.

At this point, here are just some of the options you have:

Add Members:

- Click the **Select Members** button
- Select them from the address book(s)
- Click **OK**
- Click **Save and Close**

Remove Members:

- Click on the users name
- Click the **Remove** button on the [/b]Members[/b] group
- Click **Save and Close**

Rename the Group:

- Change the Name field
- Click **Save and Close**

Delete the Entire Group:

- Click the **Delete** button in the **Actions** group

Self Assessment Questions

5. Just press …….to creates a new mail message.

(a) Ctrl + Shift + I (b) Ctrl + Shift + M

(c) Ctrl + Shift + B (d) Ctrl + Shift + C.

6. Select the reason you want to apply a flag from the…….drop-down menu.

(a) Follow up (b) Flag to

(c) Custom (d) Actions.

7. Outlook can be customized to search particular address lists first when you use the…….

(a) Messages (b) Tools

(c) Address Book (d) Actions.

6.13 Opening Received Messages

In order to access and read an e-mail message in your *Inbox*:

From the *Folder List,* select ***Inbox***

Click the desired message

The message will appear on the left hand side of the screen, in the *Reading* pane.

OR

Double click the message you want to read.The message appears on the screen.

NOTE: To split your screen to include both the listing of messages and the full message, refer to Customizing e-mail Settings: Using Reading Pane.

To close an open message window, from the *File* menu, select ***Close***

OR

From the upper right corner, click **CLOSE**

The message window closes.

NOTE: If you read a message in the reading pane, you do not have to close the pane. Simply select the next desired message in the *Inbox*.

6.14 Opening Messages with Attachment

Open the message and then use Shift + Tab, which will select the Message field as shown highlighted below:

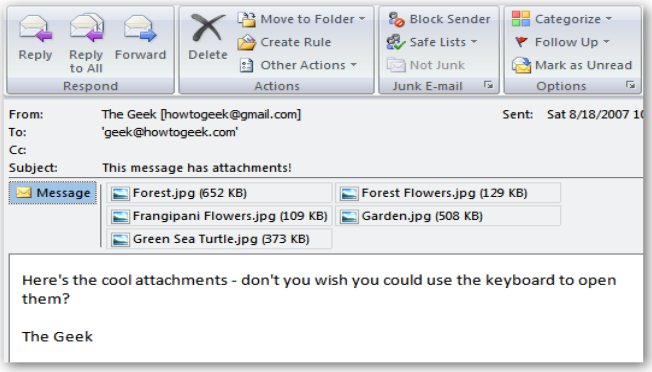


Figure 6.44: Message field as shown highlighted.

From here you can use the arrow keys to select the different attachments. If the attachment is an image, it will automatically preview below, as will certain types of office documents.

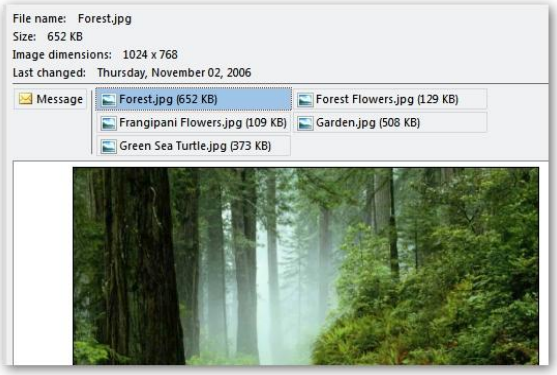


Figure 6.45: attachment to an image.

To open the attachment in the default application, just hit the Enter key. You will probably get a message similar to this one:

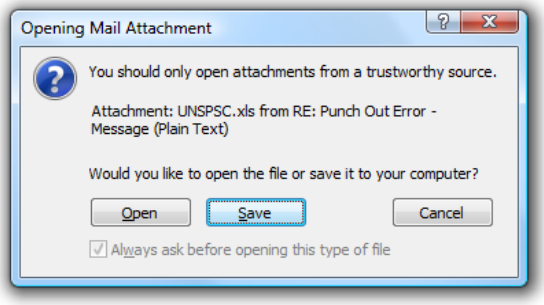


Figure 6.46: To open the attachment.

And finally save the attachment file.

6.15 Forwarding Messages

1. Open Outlook by clicking the icon on your desktop or start menu.

2. Select the e-mail that you wish to forward and click the “Forward” button on the menu bar.

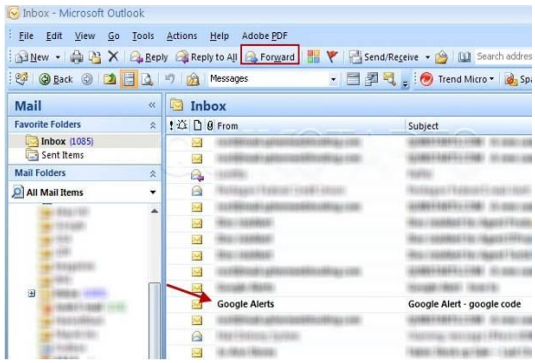


Figure 6.47: Forward button on the menu bar**.**

3. Type the e-mail address of the person you want to forward the e-mail to and press the “Send” button on the menu bar.

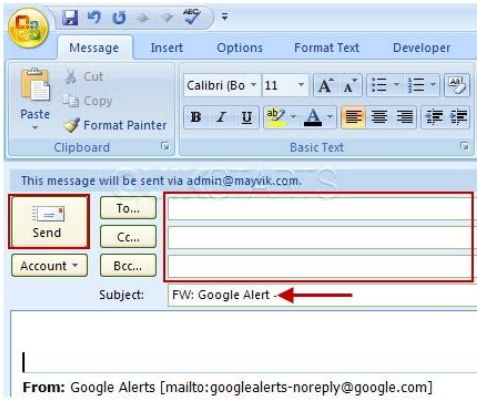


Figure 6.48: Send button on the menu bar.

4. This task should now be complete. If not, review and repeat the steps as needed. Submit any questions using the section at the bottom of this page.

Exercise: Check Your Progress 2
Note: i) Use the space below for your answer.
Ex1: Send e-mail using inserting mail address from your Address Book.

……..……

……..……

……..

6.16 Flagging for Further Action

1. Start Microsoft Outlook 2007. Click to select the message that you want to flag
2. Choose the “Actions” menu from Outlook, point to “Follow Up” and click on “Custom” to open the Custom dialog box.
3. Select the reason you want to apply a flag from the "Flag to" drop-down menu. The follow up options include: call, do not forward, follow up, for your information, forward, no response necessary, read, reply, reply to all or review.

4. Make a selection from the “Start date” drop-down menu for when you want to start the flag. Beneath that, choose the “Due date” for when you want the flag to become due.
5. Add a reminder to the e-mail message flag by clicking to add a check mark in the “Reminder” check box. Once checked, you can then specify the day and time you want to be reminded of the flag you are adding with the drop-down lists below the “Reminder” option.
6. Click the “OK” button to close the Custom dialog box. You will see a flag appear to the right of the e-mail message sender and title. The flag will remain there until you clear it.

6.17 Setting E-mail Options

Outlook 2007 provides additional option for your e-mail service. For example, you can set Outlook to automatically alert you on your desktop when you receive new messages. This LTS Document provides explanations for desktop alerts, address book settings, reading pane settings, and how to save outgoing messages for later review.

6.17.1 New Mail Desktop Alert

You have the option of having the New Mail Desktop Alert displayed on your screen when new mail arrives. If the alert is displayed, you can open the most recent message directly from the dialog box. As some viruses can be spread by the opening of Outlook e-mail messages, this option is discouraged. To turn this feature on or off, follow these steps:

1. From the *Tools* menu, select *Options*

The *Options* dialog box appears.

2. Select the ***Preferences*** tab
3. From the *e-mail* section, click e-mail options

The *e-mail Options* dialog box appears.

4. Click advanced e-mail options

The *Advanced e-mail Options* dialog box appears.

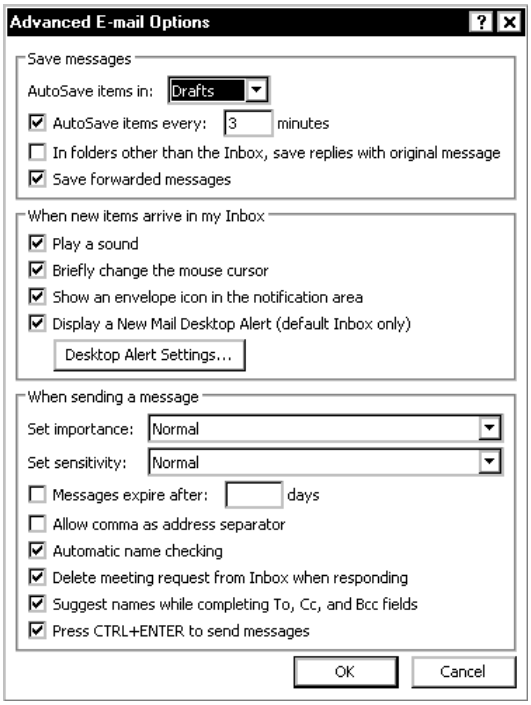


Figure 6.49: Advanced e-mail Options dialog box appears.

5. In the *When new items arrive in my Inbox* section, select *Display a New Mail Desktop Alert (default Inbox only)*

The option is selected if a checkmark appears in the box.

6. OPTIONAL: To change how long the *New Mail Desktop Alert* will appear on your screen or how transparent it is,

a. Click desktop alert settings
b. The *Desktop Alert Settings* dialog box appears.
c. To adjust the duration or transparency, in the *Duration* or *Transparency* section, click and drag the movable arrow
d. To preview the *Desktop Alert*, click PREVIEW
e. Click OK
f. The *Desktop Alert Settings* dialog box closes.

7. Click OK
8. Click OK
8. To return to Outlook, click OK

The *Options* dialog box closes and any changes are made.

6.17.2 Adjusting Address Book Settings

Outlook can be customized to search particular address lists first when you use the *Address Book*. The *Address Book* itself can be set up to show your address lists in a certain order as well.

To modify the order of your address lists and the search order that Outlook will use, follow these steps:

1. From the *Tools* menu, select *Address Book...*

The *Address Book* dialog box appears.

2. In the *Address Book* dialog box, from the *Tools* menu, select *Options...*

The *Addressing* dialog box appears.

3. To set the first listed address list, from the *Show this address list first* pull-down menu, select the address list you want Outlook display first
4. To adjust the search order,

a. From the *When sending mail, check names using these address lists in the following order* scroll box, select an address list
b. To move an address list up or down, click UP or DOWN to the right of the scroll box

5. Click OK

The *Addressing* dialog box closes.

6. In the *Address Book* dialog box, from the *File* menu, select ***Close***
7. The *Address Book* dialog box closes and any changes are made.

6.17.3 Using Reading Pane

The *Reading Pane*, known as the *Preview Pane* in past versions of Outlook, allows you to split your *Inbox* or other folder screens between a listing of messages and a preview of a selected message. To use the *Reading Pane*, simply select a message from your list and the full message appears in the *Reading Pane*. You can also opt to delete or mark messages as read using the *Reading Pane*, without having to open them in their own window.

6.17.4 Showing/Hiding the Reading Pane

The *Reading Pane* can be displayed in two different views.

1. To show the *Reading Pane*, from the *View* menu select *Reading Pane* » select *Right* or *Bottom*
2. To hide the *Reading Pane*, from the *View* menu select *Reading Pane* » select *Off*

6.17.5 Marking Messages as Read

Outlook 2007 allows you to customize the way it reacts to messages as you read them. By default, messages are marked as *read* immediately after switching to a different message. Use these steps to customize this setting.

1. From the *Tools* menu, select *Options*

The *Options* dialog box appears.

2. Select the *Other* tab
3. In the *Outlook Panes* section, click Reading Pane

The *Reading Pane* dialog box appears.

4. To mark your messages as *read* as soon as you click to the next message, select *Mark item as read when selection changes*

The option is selected if a checkmark appears in the box.

5. To specify a length of time that you can view a message in the *Reading Pane* before it is marked *read*,

a. Select *Mark items as read when viewed in the Reading Pane*

The option is selected if a checkmark appears in the box.

b. In the *Wait ... seconds before marking item as read* text box, adjust the number of seconds

6. Click OK
7. Click OK

The *Options* dialog box closes and any changes are made.

6.17.6 Saving the Messages You Send

Outlook automatically saves all messages you receive, but you should make sure that Outlook is configured to save the messages you send as well. Outgoing (sent) e-mail can be stored in the *Sent Items* folder and then deleted at any time.

1. From the Tools menu, select Options

The Options dialog box appears.

2. Select the ***Preferences*** tab
3. In the e-mail section, click E-mail options**.**

The e-mail Options dialog box appears.

4. In the message handling section, select Save copies of messages in Sent Items folder

The option is selected if a checkmark appears in the box.

5. Click **OK**
6. Click **OK**

The Options dialog box closes and your changes are made.

Exercise: Check Your Progress 3

Note: i) Use the space below for your answer.

Ex2: Send e-mail messages using the To, Cc, and Bcc fields.

6.18 Managing Contacts with Outlook

A contact is a person you know on a business or a personal level. You can use the Outlook 2007's contact folder to manage information about your business and personal contacts.

You can store a contact's name, address, telephone number, e-mail address, Web page address, company name, birthday, and anniversary details. The Contacts folder is integrated with the Inbox and the Calendar for sending mail and scheduling meetings.

6.18.1 Add a Contact

1. Click on Contacts in the Folder List
2. Click on New in the Standard Toolbar to open a new Contact window.
3. Fill in the information in the Full Name box. Press Tab

- Observe the File as list box. The contact's name appears automatically in the format "last name, first name. By accepting the “last name, first name” option, Outlook files the contact information alphabetically by last name.

4. Complete the rest of the information.

6.18.2 Edit Contact Information

1. Click on Contacts in the Folder List.
2. Double-click on the contact you want to edit.
3. Make edits as necessary.
4. Click Save and Close in toolbar.

6.18.3 Create Distribution Lists

When you work in a team, you might find yourself frequently addressing messages to the same group of people. To make this quick and easy, you can create a distribution list with the e-mail addresses of a group of people. A distribution list contains multiple e-mail addresses under a single entry.

1. Click on Contacts in the Folder List.
2. From the File menu, choose New...Distribution List to open the distribution List dialog box
3. In the Name box, enter the name you want to use for the Group
4. Click Select Members to open the Select Members dialog box
5. Add members from Name list and click OK.

- If you do not see a person's name you need, refer to How to Use PSU Directory Services to "Find People."

6. When you have added all the people you need for the distribution list, click Save and Close.

6.18.4 Use a Distribution List

From Your Inbox (e-mail)

1. Click on the arrow beside the New button in the Standard Toolbar and choose New Message.
2. Click on the To button.
3. Find the name of the distribution list you need. Select it.
4. Click the To (or Cc) button to move it to the right-hand side.
5. Click OK when finished.

6.18.5 From Your Contacts List

1. Click on the Contacts in the Folder List.
2. Find the Distribution List you created.
3. Right-click on it and choose New Message to Contact.
4. Automatically addresses an e-mail to the distribution list.

6.18.6 Use PSU Directory Services to Find PSU Employees

1. From the Tools menu choose Address Book or press Ctrl + Shift + B.
2. Make sure that LDAP is selected in the "Show Names from the:" drop-down list. (You can force it to be first by going to Tools then Options and moving it up in the list. To do that, highlight the LDAP and use the up arrow on the right side of the dialog box).

3. From the Tools menu choose Find or press Ctrl + Shift + F. The "Display Name" field is the field that appears in the Address Book. The problem with this field is that you must have an exact match! At the more advanced screen you can search on First Name and Last Name.
4. Enter a name. Press OK. You should find the person. Be warned. The only info seems to be in First, Last, Address, Title, Phone and e-mail. And you can only search in First and Last Name.

6.18.7 To Add a Person to your Contacts (Address book) from Directory Services

1. Once you are used the above steps to find a contact, Right Click on their name and choose Add to Contacts or click Add to Contacts on the toolbar.
2. You're Contacts or Address Book will be opened and you will see the information there. You can edit the contact information if necessary.

6.19 Setting up Multiple E-mail Accounts on Single Machine

Outlook 2007 can manage all your e-mail accounts for you, becoming one central place to work with all your messages. It does, however, add some complexity in that you need to know which account any given message came in on or will be sent out on. The rest of this article explains how you do this.

6.19.1 When you Receive Messages

Let us talk about how Outlook receives and manages messages from more than one account at a time. Outlook will automatically check each e-mail account (the settings in your Send/Receive groups determine when). Messages from most types of e-mail accounts end up in the Inbox, while messages from HTTP mail accounts, like Hotmail, end up in their own collection of mail folders.

Wondering how you will know which messages are associated with which accounts? It turns out that is easy. The Hotmail messages end up in their own set of folders. For the others, if you cannot tell simple by which they are from, you can look at the two: field of the message. The e-mail address the sender addressed it to appear here.

Except when it does not sometimes you will see a nickname instead of the e-mail address in the To: field. If there is a nickname, following these steps will show you the actual e-mail address associated with that nickname:

1. Right-click the nickname.
2. In the shortcut menu, click Outlook Properties.
3. In the dialog box, look on the e-mail Addresses tabbed page. This will tell you which address the message were sent to.

6.19.2 Sending Messages from Multiple Accounts

When you are sending messages, you're in control of which account Outlook uses. Outlook always has a default account for sending messages (usually the first account you set up), but you can tell Outlook to use a different account. Here is how it works:

- When you create a new message, Outlook assumes you will want to use the default account to send it (I shall tell you how to change the default account in a few minutes) unless you specific a different account.
- If you are replying to a message, Outlook uses the account that the message came in on. In other words, if you receive a message sent to your account named xyz@mymailserver.com, and you clicked Reply or Reply to All, Outlook would assume you want to send the reply using the xyz@mymailserver.com account. Unless, of course, you tell Outlook to use a different account.
- When you forward a message, Outlook will assume you want to do so using the same account as the original message.

The way Outlook selects the account to use makes great sense, but what if you do not want to use the account Outlook thinks you want to use? How do you tell Outlook to use a different account? That is easy.
You tell Outlook which e-mail account to use when you have the message window open. When you have multiple accounts set up, you will see an Account button below the Send button. Click the Account button, and Outlook will display a menu containing all your e-mail accounts. Select one, and Outlook uses that one to send your message.

6.19.3 Changing the Default Account

You may sometimes decide that you want Outlook to use a different e-mail account as your default. If you want to change the default, follow these steps:

1. Go to the main Outlook menu and click Tools > Account Settings to open the Account Settings dialog box.
2. On the e-mail tabbed page of the Account Settings dialog box, find the box listing all your e-mail accounts. Click the account you want to make your new default account.
3. In the space above the account list, find and click Set as Default. Outlook dims this option until you select a new account to use as the default.
4. The list rearranges itself so the new default option appears at the top of the list.
5. Click Close.

6.20 Summary

- The rich text format is proprietary to Microsoft e-mailing software.
- Outlook is smart enough to transmit messages in HTML, plain text, or rich text format when you reply to a message that was sent to you in that format.
- Microsoft Outlook is the collaboration application in Microsoft's Office productivity suite.
- The contacts folder is integrated with the inbox and the calendar for sending mail and scheduling meetings.
- Outlook Express does not display the Bcc field for e-mails by default.

6.21 Keywords

E-mail address: E-mail address is the POP e-mail address, enter the same e-mail you use for your user name when accessing the Admin Console.
Internet Message Access Protocol (IMAP): Internet message access protocol is one of the two most prevalent Internet standard protocols for e-mail retrieval, the other being the Post Office Protocol.
Internet Service Provider (ISP): An Internet service provider is a company that provides access to the Internet. Access ISPs directly connect customers to the Internet using copper wires, wireless or fiber-optic connections.
Quick Access Toolbar: The Quick Access Toolbar is a customizable toolbar that contains a set of commands that are independent of the tab that is currently displayed
Rich text format: The Rich Text Format is a proprietary document file format with published specification developed by Microsoft Corporation since 1987 for Microsoft products and for cross-platform document interchange.

6.22 Review Questions

1. What is process of the set up your e-mail account in Microsoft outlook 2007?
2. How to send and receive e-mail through outlook?

3. What are concepts of Cc and Bcc?
4. How to attach a file to an e-mail message?
5. What are formats for sending e-mail messages?
6. Describe the concept of MIME.
7. What is the outlook Protocol?
8. Explain the concept of attachment of file into the message.
9. How to create groups for sending a message?
10. What are the e-mail services of outlook 2007?

Answers for Self Assessment Questions

1 (b)	2 (a)	3 (c)	4 (d)	5 (b)	6 (b)	7 (c)

6.23 Further Studies

1 Microsoft Office Outlook 2007 Inside and Out By Jim Boyce

Database Management System

(MCA/PGDCA - 103)



Jaipur National University
Directorate of Distance Education

Established by Government of Rajasthan
Approved by UGC under Sec 2(f) of UGC ACT 1956
(**Recognised by Joint Committee of UGC-AICTE-DEC, Govt. of India**)

1

Introduction to DBMS

STRUCTURE

1.0 Objectives

1.1 Introduction

1.2 Database

1.3 Three-Level Architecture Proposal

1.4 Purpose of Database Systems

1.5 Data Models

1.6 Database Languages

1.7 Data Base Administrator

1.8 Data Base Users

1.9 Data Base Management System Structure

1.10 Advantages and Disadvantages of DBMS

1.11 Summery

1.12 Keywords

1.13 Review Questions

1.14 Further Studies

1.0 Objectives

After studying this chapter, you will be able to:

- Define the database
- Discuss the three-level architecture of proposal
- Explain the purpose of database system
- Discuss the data model and abstraction

1.1 Introduction

A database management system (DBMS) is a set of software programs that allows users to create, edit and update data in database files, and store and retrieve data from those database files. Data in a database can be added, deleted, changed, sorted or searched all using a DBMS. If you were an employee or part of any large organization, the information about you would likely be stored in different files that are linked together. One file about you would pertain to your skills and abilities, another file to your income tax status, another to your home and office address and telephone number, and another to your annual performance ratings. By cross-

referencing these files, someone could change a person's address in one file and it would automatically be reflected in all the other files.

1.2 Database

A database is a collection of related files that are usually integrated, linked or cross-referenced to one another. The advantage of a database is that data and records contained in different files can be easily organized and retrieved using specialized database management software called a database management system (DBMS) or database manager.

1.2.1 Views of Data

DBMS is a collection of interrelated files and a set of programs that allow users access and modify these files. A major purpose of a database system is to provide users with an abstract view of the data. That is, the system hides certain details of how the data are stored and maintained.

1.2.2 Data Abstraction

Major purpose of dbms is to provide users with abstract view of data i.e. the system hides certain details of how the data are stored and maintained.

Since database system users are not computer trained, developers hide the complexity from users through 3 levels of abstraction, to simplify user's interaction with the system.

Physical level: The lowest level of abstraction describes how the data are actually stored. At the physical level, complex low-level data structures are described in detail.

Logical level: The next-higher level of abstraction describes what data are stored in the database, and what relationships exist among those data. The entire database is thus described in terms of a small number of relatively simple structures. Although implementation of the simple structures at the logical level may involve complex physical-level structures, the user of the logical level does not need to be aware of this complexity. The logical level of abstraction is used by database administrators, who must decide what information is to be kept in the database.

View level: The highest level of abstraction describes only part of the entire database. Despite the use of simpler structures at the logical level, some complexity remains, because of the large size of the database. Many users of the database system will not be concerned with all this information. Instead, such users need to access only a part of the database. So that their interaction with the system is simplified, the view level of abstraction is defined. The system may provide many views for the same database.

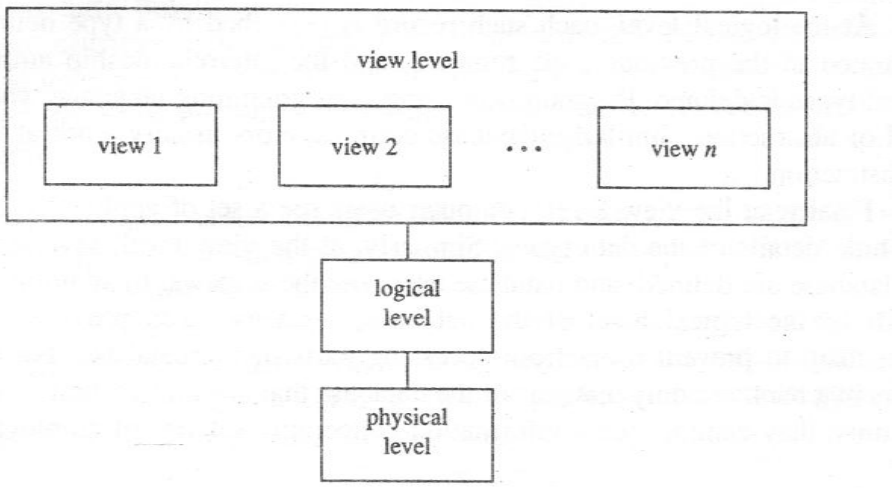


Figure 1.1: The three levels of data abstraction.

The interrelationship among these three levels of abstraction is illustrated in Figure 1.1. An analogy to the concept of data types in programming languages may clarify the distinction among levels of abstraction. Most high-level programning languages support the notion of a record type. For example, in a Pascal-like language, we may declare a record as follows:

Type customer = record

Customer_name: string; *social_security*: string;

Customer_street: string; *customer_city*: string; end;

This code defines a new record called *customer* with three fields. Each field has a name and a type associated with it. A banking enterprise may have several such record types, including

- *account*, with fields *account-number* and *balance*
- *employee*, with fields *employee-name* and *salary*

At the physical level, a customer, account, or employee can be described as a block of consecutive storage locations (for example, words or bytes). The language compiler hides this level of detail from programmers. Similarly, the database system hides many of the lowest-level storage details from database programmers. Database administrators may be aware of certain details of the physical organization of the data.

At the logical level, each such record is described by a type definition, as illustrated in the previous code segment, and the interrelationship among these record types is defined. Programmers using a programming language work at this level of abstraction. Similarly, database administrators usually work at this level of abstraction. Finally, at the view level, computer users see a set of application programs that hide details of the data types. Similarly, at the view level, several views of the database are defined, and database users see these views. In addition to hiding details of the logical level of the database, the views also provide a security mechanism to prevent users from accessing parts of the database. For example, tellers in a bank see only that part of the database that has information on customer accounts; they cannot access information concerning salaries of employees.

1.2.3 Instances and Schemas

Databases change over time as information is *inserted* and *deleted*. The collection of information stored in the database at a particular moment is called an *instance* of the database. The overall design of the database is called the database schema. Schemas are changed infrequently, if at all.

An analogy to the concepts of data types, variables, and values in programming languages is useful. Returning to the customer-record type definition, note that, in declaring the type *customer*. To declare such variables in a Pascal-like language.

var customer: *customer;*

Variable *customer* now corresponds to an area of storage containing a *customer* type record. A database schema corresponds to the programming-language type definition. A variable of a given type has a particular value at a given instant. Thus, the value of a variable in programming languages corresponds to an instance of a database schema. Database systems have several schemas, partitioned according to the levels of abstraction that we discussed. At the lowest level is the physical schema; at the intermediate level is the logical schema; and at the highest level is a sub. In general, database systems support one physical schema, one logical schema, and several subschemas.

1.2.4 Data Independence

The ability to modify a schema definition in one level without affecting a schema definition in the next higher level is called *data independence*. There are two levels of data independence:

1. Physical data independence is the ability to modify the physical schema without causing application programs to be rewritten. Modifications at the physical level are occasionally necessary to improve performance.

2. Logical data independence is the ability to modify the logical schema without causing application programs to be rewritten. Modifications at the logical level are necessary whenever the logical structure of the database is altered (for example, when money-market accounts are added to a banking system). Logical data independence is more difficult to achieve than is physical data independence, since application programs are heavily dependent on the logical structure of the data that they access.

The concept of data independence is similar in many respects to the concept of *abstract data types* in modem programming languages. Both hide implementation details from the users, to allow users to concentrate on the general structure, rather than on low-level implementation details.

1.3 Three-Level Architecture Proposal

The database system provides users with an abstract view of data, by hiding certain details of how data is stored and manipulated. Therefore, before designing a database, the data of an organization is considered on abstract level. Database system architecture means design or construction of database system. The database system architecture provides general concept and structure of database system. The architecture of most commercial database systems is based on the three-level architecture proposed by DBTG (Data Base Task Group). As a result of these and later reports, databases can be viewed at three levels, known as external, conceptual and internal levels. These levels form three-level architecture. These levels are represented by three schemas or three models known as three-level schema. The models refer to the structure of database, not to the data that is stored in any given time or moment. The structure of database is called the intension of the database, or the database schema. The data stored into the database at a given time is called the extension of database, or a database instance. A database model is a collection of tools that may include diagram and specialized vocabulary (or special words) for describing the structure of database. Typically a database model describes the data, the relationships between the data items, constraints on the data, and data meanings etc.

The purpose of the three-level architecture is to separate the user applications and the physical database. The reasons of this separation are:

1. Different users need different views of the same data.
2. Users should not have to deal directly with the physical database storage details.
3. The DBA should be able to change the database storage structure or storage device without affecting other user's views and applications.

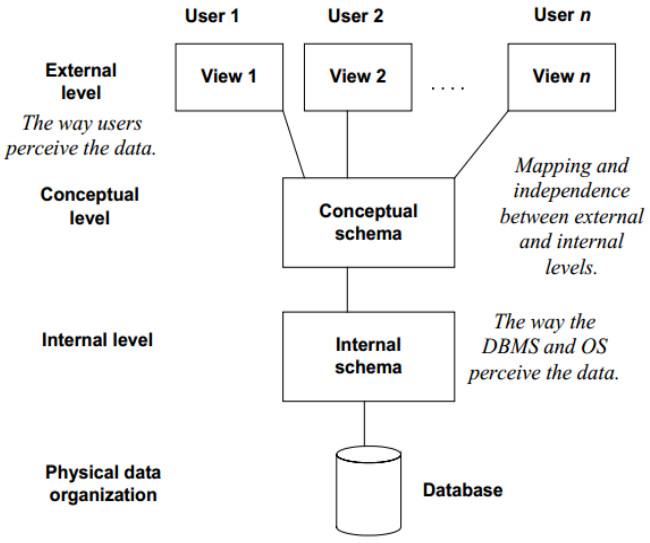


Figure 1.2: The Three-level architecture of the database system

External Level

The external level or view is closest to the users. It is concerned with the way the data is viewed by individual users. You can say that external level is the individual user level. A user can either be an application programmer or an end-user, but (Database Administrator) DBA is an important special case. The external level consists of many different external views or external models of the database. Each external view describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group. In addition, different views may have different representations of the same-data. For example, one user may view date in the form (day, month, year), while another may view date as (year, month, day). Some users may view virtual or calculated data, which is not actually stored in database. The calculated data is created temporarily when needed. For example, marks of subject "C++" of students are stored in the database and average marks of this subject are calculated by system when the user refers to it. Similarly, if the date-of-birth of a student is stored in the database then you can find the age of the student.

A view may include data combined or calculated from several records. An external record is a record as seen by a particular user, which is a part of his/her external view. Actually, an external view is a collection of external records. The external views are defined by means of external schemas (also called subschema's) that are written in the data definition language (DDL). Usually, the DBA writes an external schema to create a user view. Each user's schema gives a complete description of each type of external record that appears in that user's view. The external schemas are compiled by the DBMS and stored in its data dictionary. The DBMS uses the external schema created for a specific user, to create a user interface to access data of the database. Thus user-interface created through external schema accepts and displays information in the format the user expects. It also acts as barrier to hide the information of database from users that is not permitted to him. Typically, it hides the conceptual, internal, and physical details from the user. If any user needs some advance access of database such as to change the existing record of database then the external schema for that user is re-written (modified) by DBA to allow access to him.

Logical or Conceptual Level

The middle level in the three-level architecture is the logical or conceptual level. It describes the entire structure of database such as entities, attributes, data types, relationships, constraints on the data, and user operations. It hides the details of physical storage structures. The conceptual level supports the external level to present the data to end-users as they need. This level is relatively constant and the DBA designs it after determining the present and future information needs of the organization. However, to expand the conceptual level, the DBA adds new objects to fulfil the requirements of the organization, without affecting the external level. The conceptual level is defined by means of the conceptual schema, which includes definitions of each of the various conceptual record types. The conceptual schema is a complete description of information of database structure such as every record type with all its fields. It also includes security and integrity rules. The conceptual schema is written in DDL, compiled by the DBMS and stored in its data dictionary. The DBMS uses the conceptual schema to create the logical record interface, which defines and creates the working environment for the conceptual level to present data to end-users. Actually, conceptual level is a collection of logical records.

Internal or Physical Level

The internal or physical level describes the physical storage structures of database and file organizations used to store data on physical storage devices. The internal level describes how the data is stored in the database. It works with operating system and DBMS for storing and retrieving data to and from the storage devices. Although, both internal level and physical level is considered as a single level, but there is slight difference between them. Actually, physical level in one, that is managed by the operating system under the direction of DBMS, while the internal level is managed by DBMS. An internal record is a single stored record. It is a

storage unit that is passed up to the internal level through stored record interface. The stored record interface is the boundary between the physical level (for which the operating system may be responsible) and the internal level (for which the DBMS is responsible). This interface is provided to the DBMS by operating system. In some cases, the DBMS itself may create this interface. The physical level below this interface consists of data of database, which is stored and managed by operating system. The operating system creates the physical record interface to access the data of database stored on storage devices.

Did You Know?
An early proposal for a standardized terminology (or vocabulary) and architecture for database system were developed and published in 1971 by DBTG (Data Base Task Group) appointed by CODASYL.

1.4 Purpose of Database Systems

Purpose of database systems are to store the organizational information in computer systems such a way that the data can be manipulated easily by it's user.
Consider part of a savings bank enterprise that keeps information about all customers and savings accounts. One way to keep the information on a computer is to store it in permanent system files to allow users to manipulate the information, the system has a number of application programs that manipulate the files including

- Debit or credit an account
- Add a new account
- Find the balance of an account
- Generate monthly statements

The main purpose of database systems is to manipulate information and to provide for data mining tasks. System programmers wrote these application programs to meet the needs of the bank. New application programs are added to the system as the need arises. For example, suppose that a savings bank decides to offer checking accounts. As a result, the bank creates new permanent files that contain information about all the checking accounts maintained in the bank, and it may have to write new application programs to deal with situations that do not arise in savings accounts, such as overdrafts. Thus, as time goes by, the system acquires more files and more application programs.
This typical file-processing system is supported by a conventional operating system. The system stores permanent records in various files, and it needs different application programs to extract records from, and add records to, the appropriate files. Use of database server could be the main task to introduce. Before database management systems (DBMS's) came along, organizations usually stored information in such systems. Some important issues when talk about purpose of database systems is introduction to database, it must be complete to more understand the purpose database systems.
Keeping organizational information in a file-processing system has a number of major disadvantages:

Data redundancy and inconsistency: Since different programmers create the files and application programs over a long period, the various files are likely to have different structures and the programs may be written in several programming languages. Moreover, the same information may be duplicated in several places (files). For example, the address and telephone number of a particular customer may appear in a file that consists of savings-account records and in a file that consists of checking-account records. This redundancy leads to higher storage and access cost. In addition, it may lead to data inconsistency; that is, the various copies of the same data may no longer agree. For example, a changed customer address may be reflected in savings-account records but not elsewhere in the system.

Difficulty in accessing data: Suppose that one of the bank officers needs to find out the names of all customers who live within a particular postal-code area. The officer asks the data-processing department to generate such a list. Because the designers of the original system did not anticipate this request, there is no application program on hand to meet it. There is, however, an application program to generate the list of all customers. The bank officer has now two choices: either obtain the list of all customers and extract the needed information manually or ask a system programmer to write the necessary application program. Both alternatives are obviously unsatisfactory. Suppose that such a program is written, and that, several days later, the same officer needs to trim that list to include only those customers who have an account balance of INR100000 or more. As expected, a program to generate such a list does not exist. Again, the officer has the preceding two options, neither of which is satisfactory. The point here is that conventional file-processing environments do not allow needed data to be retrieved in a convenient and efficient manner. More responsive data-retrieval systems are required for general use.

Data isolation: Because data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult. Database archiving is one of data isolation techniques. Computer tutorial about database and this data isolation is very important to understanding these issues.

Integrity problems: The data values stored in the database must satisfy certain types of consistency constraints. For example, the balance of certain types of bank accounts may never fall below a prescribed amount (say, INR250). Developers enforce these constraints in the system by adding appropriate code in the various application programs. However, when new constraints are added, it is difficult to change the programs to enforce them. The problem is compounded when constraints involve several data items from different files.

Atomicity problems: A computer system, like any other mechanical or electrical device, is subject to failure. In many applications, it is crucial that, if a failure occurs, the data be restored to the consistent state that existed prior to the failure. Consider a program to transfer INR500 from account A to account B. If a system failure occurs during the execution of the Program, it is possible that the INR500 was removed from account but was not credited to account B, resulting in an inconsistent database state. Clearly, it is essential to database consistency that either both the credit and debit occur, or that neither occur. That is, the funds transfer must be atomic-it must happen in its entirety or not at all. It is difficult to ensure atomicity in a conventional file-processing system.

Concurrent-access anomalies: For the sake of overall performance of the system and faster response, many systems allow multiple users to update the data simultaneously. Indeed, today, the largest Internet retailers may have millions of accesses per day to their data by shoppers. In such an environment, interaction of concurrent updates is possible and may result in inconsistent data. Consider bank account A, containing 5000. If two customers withdraw funds (say INR500 and INR1000, respectively) from account A at about the same time, the result of the concurrent executions may leave the account in an incorrect (or inconsistent) state. Suppose that the program executing on behalf of each withdrawal read the old balance, reduce that value by the amount being withdrawn, and write the result back. If the two programs run concurrently, they may both read the value INR5000, and write back INR4500 and 9400, respectively. Depending on which one writes the value last, the account may contain either INR 4500 or 4000, rather than the correct value of INR3500. To guard against this possibility, the system must maintain some form of supervision. But supervision is difficult to provide because may be accessed by many different application programs that have not been coordinated previously.

Security problems: Not every user of the database system should be able to access all the data. For example in a banking system, payroll personnel need to see only that part of the database that has information about the various bank employees. They do not need access to information about customer accounts. But" since application programs are added to the file-processing system in an alcohol manner, enforcing such security constraints is difficult.

Computer development bought us to the development of DBMS and database systems. These difficulties, among others, prompted the development of database systems. In what follows, we shall see the concepts and algorithms that enable database systems to solve the problems with file-processing systems. In most of this book, we use a bank enterprise as a running example of a typical data-processing application found in a corporation.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: What is data independence?

……..……

……..……

……...

Ex2: What is three Level Architecture?

……..……

……..……

……...

1.5 Data Models

Data models are a collection of conceptual tools for describing data, data relationships, data semantics and data constraints. There are three different groups:

1. Object-based Logical Models.
2. Record-based Logical Models.
3. Physical Data Models.

1.5.1 Object-Based Logical Models

Object-based logical models are used in describing data at the logical and *vie* levels. They are characterized by the fact that they provide fairly flexible structuring capabilities and allow data constraints to be specified explicitly. There are many different models, and more are likely to come. Several of the more widely known ones are:

- The entity-relationship model
- The object-oriented model
- The semantic data model
- The functional data model

We examine the entity-relationship model and the object-oriented model as representatives of the class of the object-based logical models. The entity-relationship model has gained acceptance in database design and is widely used in practice. The object-oriented model, includes many of the concepts of the entity-relationship

model, but represents executable code as well as data. It is rapidly gaining acceptance in practice. We shall give brief descriptions of both models next.

1.5.2 The Entity-Relationship Model

The entity-relationship (E-R) data model is based on a perception of a real world that consists of a collection of basic objects, called *entities*, and of *relationships* among these objects. An entity is a "thing" or "object" in the real world that is distinguishable from other' objects. For example, each person-is an entity, and bank accounts can be considered to be entities. Entities are described in a database by a set of *attributes*. For example, the attributes *account-number* and *balance* describe one particular account in a bank. A *relationship* is an association among several entities. For example, a *Depositor* relationship associates a customer with each account that she has. The set of all entities of the same type, and the set of all relationships of the same type, are termed an *entity set* and *relationship set*, respectively.

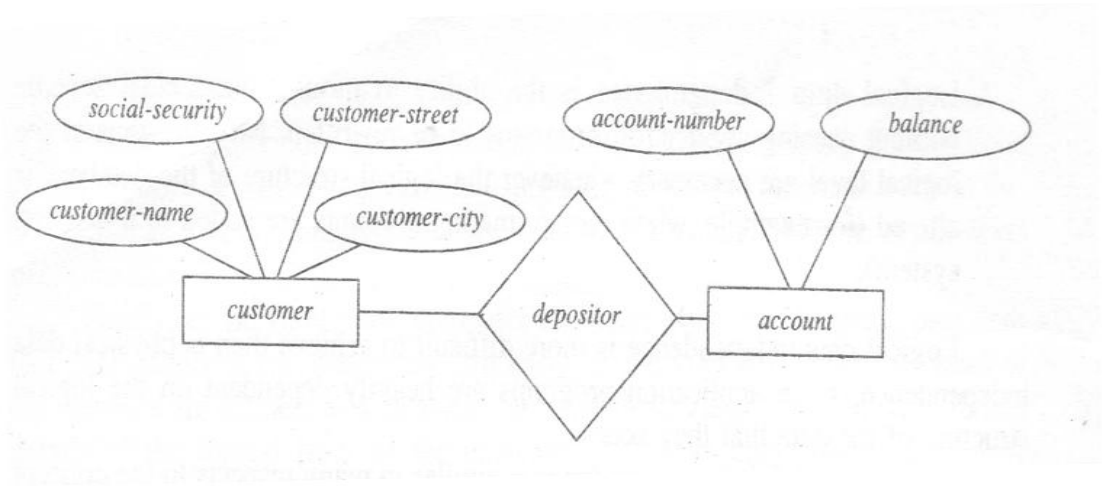


Figure 1.3: A sample of E-R diagram.

In addition to entities and relationships, the E-R model represents certain constraints to which the contents of a database must conform. One important constraint is *mapping cardinalities*, which express the number of entities to which another entity can be associated via relationship set. The overall logical structure of a database can be expressed graphically by an E-R diagram, which is built up from the following components:

- Rectangles, which represent entity sets
- Ellipses, which represent attributes
- Diamonds, which represent relationships among entity sets
- Lines, which link attributes to entity sets and entity sets to relationships

Each component is labelled with the entity or relationship that it represents. An illustration, consider part of a database banking system consisting of customers and of the accounts that these customers have. The corresponding E-R diagram is shown in Figure 1.3.

1.5.3 The Object-oriented Model

The object-oriented model is based on a collection of objects, like the E-R model.

- An object contains values stored in instance variables within the object.
- Unlike the record-oriented models, these values are themselves objects.
- Thus objects contain objects to an arbitrarily deep level of nesting.
- An object also contains bodies of code that operate on the the object.
- These bodies of code are called methods.
- Objects that contain the same types of values and the same methods are grouped into classes.

- A class may be viewed as a type definition for objects.
- Analogy: the programming language concept of an abstract data type.
- The only way in which one object can access the data of another object is by invoking the method of that other object.
- This is called sending a message to the object.
- Internal parts of the object, the instance variables and method code, are not visible externally.
- Result is two levels of data abstraction.

For example, consider an object representing a bank account.
- The object contains instance variables number and balance.
- The object contains a method pay-interest which adds interest to the balance.
- Under most data models, changing the interest rate entails changing code in application programs.
- In the object-oriented model, this only entails a change within the pay-interest method.

2. Unlike entities in the E-R model, each object has its own unique identity, independent of the values it contains:
- Two objects containing the same values are distinct.
- Distinction is created and maintained in physical level by assigning distinct object identifiers.

Exercise: Check Your Progress 2
Note: i) Use the space below for your answer.
Ex1: Prepare the block diagram working process of the object-oriented model.

……..……

……..……

……...

1.5.4 Network Model
Data in the network model are represented by collections of records (in the Pascal sense), and relationships among data are represented by links, which can be viewed as pointers. The records in the database are organized as collections of arbitrary graphs. The hierarchical model is similar to the network model in the sense that data and relationships among data are represented by records and links, respectively.
- It differs from the network model in that the records are organized as collections of trees rather than arbitrary graphs.

Differences among the Models
The relational model differs from the network and hierarchical models in that it does not use pointers or links. Instead, the relational model relates records by the values they contain this freedom from the use of pointers allows a formal mathematical foundation to be defined.

1.5.5 Physical Data Models
Physical data models are used to describe data at the lowest level. In contrast to logical data models, there are few physical data models in use. Two of the widely known ones are the *unifying model* and the *frame memory-model*.

Did You Know?
The network model's original inventor was Charles Bachman, and it was developed into a standard specification published in 1969 by the CODASYL Consortium.

1.6 Database Languages

A database system provides two different types of languages one to specify the database schema, and the other to express database queries and updates.

1.6.1 Data Definition Language

A database schema is specified by a set of definitions expressed by a special language called a data-definition language (DDL). The result of compilation of DDL statements is a set of tables that is stored in a special file called data dictionary or data directory.
A data dictionary is a file that contains *metadata* that is data, about data. This file is consulted before actual data are read or modified in the database system.
The storage structure and access methods used by the database system are specified by a set of definitions in a special type of DDL called a data storage definition language. The result of compilation of these definitions is set of instructions to specify the implementation details of the database schemas-details are usually hidden from the users.

1.6.2 Data-Manipulation Language

The levels of abstraction at we discussed in apply not only to the definition or structuring of data, but also to the manipulation of data. By data manipulation, we mean

- The retrieval of information stored in the database.
- The insertion of new information into the database.
- The deletion of information from the database.
- The modification of information stored in the database.

At higher levels of abstraction, we emphasize ease of use. The goal is to provide efficient human interaction with the system.
A data-manipulation language (DML) is a language that enables users to access or manipulate data as organised by the appropriate data model. There are basically two types:

- ***Procedural DMLs*** require a user to specify what data are needed and how to get these-data.
- ***Nonprocedural DMLs*** require a user to specify what data are needed without specifying how to get 'those data.

Nonprocedural DMLs are usually easier to learn and use than are procedural DMLs. However, since a user does not have to specify how to get the data, these languages may generate code that is not as efficient as that produced by procedural languages. We can remedy this difficulty through various optimization techniques.
A query is a statement request schema the retrieval of information. The portion of a DML that involves information retrieval is called a query language. Although technically incorrect, it is common practice to use the terms query language and data-manipulation language synonymously.

Caution
At the physical level, we must define algorithms that allow efficient access to data.

Self Assessment Questions

1. …….attempts to transform 'a user's request into an equivalent but more efficient form, thus finding a good strategy for executing the query.

(a). DML compiler (b). DDL
(c). MSIL (d). None of these.

2. A ……is partitioned into modules that deal with each of the responsibilities of the overall system.
(a). Database system (c).Data manipulation language
(b). Over all system (d). None of these

3. The portion of a DML that involves information retrieval is called a query language.
(a). True (b). False
4. The retrieval of information stored in the……
(a). Management (b). Database
(c). Relational database (d). None of these

5. A data-manipulation language (DML) is a language that enables users to access or manipulate data as organized by the appropriate data model.
(a). True (b). False

6. Nonprocedural DMLs are usually easier to learn and use than are procedural DMLs.
(a). True (b). False

1.7 Data Base Administrator

One of the main reasons for using DBMSs is to have central control of both the data and the programs that access those data. The person who has such central control over the system is called the database administrator (DBA). The functions of the DBA include the following:

Schema Definition: The DBA creates the original database schema by writing a set of definitions that is translated by the DDL compiler to a set of tables that is stored permanently in the data dictionary.
Storage Structure and Access-method Definition: The DBA creates appropriate storage structures and access methods by writing a set of definitions. This is translated by the data-storage and data-definition-language compiler.

Schema and Physical-Organization Modification: Programmers accomplish the relatively-we modifications either to the database schema or to the-description of the physical storage organization by writing a set of definitions that is used by either the DDL compiler or the data-storage and data-definition-language compiler to generate modifications to the appropriate internal system tables (for example, the data dictionary).

Granting of Authorization for Data Access: The granting of different types of authorization allows the database administrator to regulate which parts of the database various users can access. The authorization information is kept in a special system structure that is consulted by the database system whenever access to data is attempted in the system.

Integrity-constraint Specification: The data values stored in the database must satisfy certain.
Consistency-constraints: For example, perhaps the number of hours an employee may work in 1 week may not exceed a specified limit (say, 80 hours), such a constraint must be specified explicitly by the database administrator. The integrity constraints are kept in a special system structure that is consulted by the database system whenever an update takes place in the system.

1.8 Data Base Users

A primary goal of a database system is to provide an environment for retrieving information from and storing new information into the database. There are four different types of database-system users, differentiated by the way that they expect to interact with the system.

1. Application programmers are computer professionals who interact with the system through DML calls, which are embedded in a program written in a host language (for example, COBOL, PL/1, Pascal, C). These programs are commonly referred to as application programs. Examples in a banking system include programs that generate payroll checks that debit accounts, that credit accounts or that transfer funds between accounts.

   Since the DML syntax is usually markedly different from the host language syntax, DML calls are-usually prefaced by a special character so that the appropriate code can be generated. A special pre-processor, called the DML pre-compiler, converts the DML statements to normal procedure calls in the host language. The resulting program is then run through the host-language compiler, which generates appropriate object code.

   There are special types of programming languages that combine control structures of Pascal-like languages with control structures for the manipulation of a database object (for example, relations). These languages sometimes called fourth-generation languages often include special features to facilitate the generation of forms and the display of data on the screen. Most major commercial database systems include a fourth-generation language.
2. Sophisticated users interact with the system without writing programs. Instead, they form their requests in a database query language. Each such query is submitted to a query processor whose function is to break down DML statement into instructions that the storage manager understands. Analysts who submit queries to explore data in the database fall in this category.
3. Specialized users are sophisticated users who write specialized database applications that do not fit into the traditional data-processing framework. Among these applications are computer-aided design systems, knowledgebase and expert systems, systems that store data with complex data types (for example, graphics data and audio data), and environment-modelling systems.
4. Naive users are unsophisticated users who interact with the system by invoking one of the permanent application programs that have been written previously. For example, a bank teller who needs to transfer 2250 from account A to account B invokes a program called transfer. This program asks the teller for the amount of money to be transferred, the account from which the money is to be transferred, and the account to which the money is to be transferred.

Exercise: Check Your Progress 3

Note: i) Use the space below for your answer.

Ex1: What is Entity-Relationship Model?

……..……

……..……

……...

Ex2: What is Data Base Administrator?

……..……

……..……

……...

1.9 Data Base Management System Structure

A database system is partitioned into modules that deal with each of the responsibilities of the overall system. Some of the functions of the database system may be provided by the computer's operating system. In most cases, the computer's operating system provides only the most basic services, and the database system must build on that base. The functional components of a database system can be broadly divided into query processor components and storage manager components. The query processor components include.

- ***DML compiler,*** which translates DML statements in a query language into low-level instructions that the query evaluation engine understands. In addition on the DML compiler attempts to transform 'a user's request into an equivalent but more efficient form, thus finding a good strategy for executing the query.
  - ***Embedded DML Pre-compiler,*** which converts DML statements, embedded in an application program to normal procedure calls in the host language. The pre-compiler must interact with the DML compiler to generate the appropriate code.
  - ***DDL interpreter,*** which interprets DDL statements and records them in a set of tables containing metadata.
  - ***Query Evaluation engine,*** this executes low-level instructions generated by the DML compiler.

The storage manager components provide the interface between the low level data stored in the database and the application programs and queries submitted to the system. The storage manager components include:

- ***Authorization and integrity manager,*** which tests for the satisfaction of integrity constraints and checks the authority of users to access data.
- ***Transaction manager***, which ensures that the database remains in a consistent (correct) state despite system failures, and that concurrent transaction executions proceed without conflicting.
- ***File manager,*** which manages the allocation of space on disk storage and the data structures used to represent information stored on disk.
- ***Buffer manager***, which is responsible for fetching data from disk storage into main memory, and deciding what data to cache in memory.

In addition, several data structures are required as part of the physical system implementation:

- ***Data files***, which store the database itself.
- ***Data dictionary***, which stores metadata about the structure of the database. The data dictionary is used heavily. Therefore, great emphasis should be placed on developing a good design and efficient implementation of the dictionary.
- ***Indices***, which provide fast access to data items that hold particular values.
- ***Statistical data***, which store statistical information about the data in the database. This information is used by the query processor to select efficient ways to execute a query.

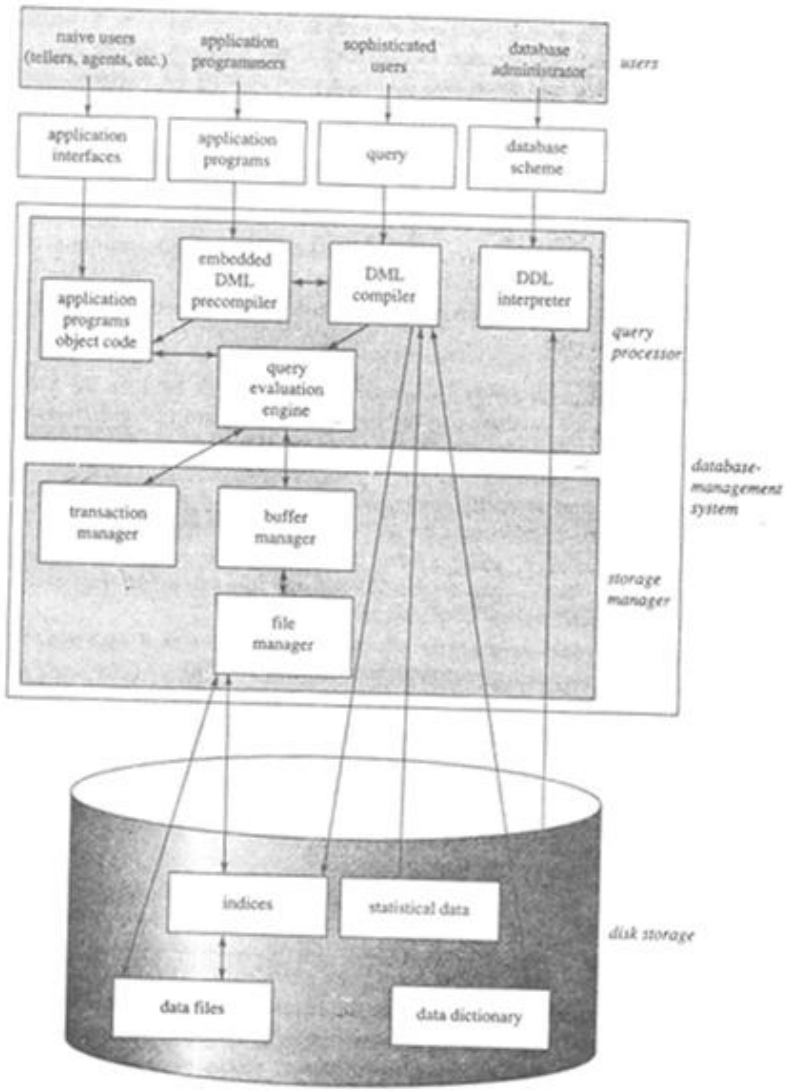


Figure 1.4: System structure.

Exercise: Check Your Progress 4
Note: i) Use the space below for your answer.
Ex1: Prepare the overall structure of DBMS.

……..……

……..……

……...

Caution
The design of a database system must include consideration of the interface between the database system and the operating system.

1.10 Advantages and Disadvantages of DBMS

Advantages of Database Management System:

The DBMS has a number of advantages as compared to traditional computer file processing approach. The DBA must keep in mind these benefits or capabilities during designing databases, coordinating and monitoring the DBMS.

The major advantages of DBMS are described below.

1. Controlling Data Redundancy:

In non-database systems (traditional computer file processing), each application program has its own files. In this case, the duplicated copies of the same data are created at many places. In DBMS, all the data of an organization is integrated into a single database. The data is recorded at only one place in the database and it is not duplicated. For example, the dean's faculty file and the faculty payroll file contain several items that are identical. When they are converted into database, the data is integrated into a single database so that multiple copies of the same data are reduced to-single copy.

In DBMS, the data redundancy can be controlled or reduced but is not removed completely. Sometimes, it is necessary to create duplicate copies of the same data items in order to relate tables with each other.By controlling the data redundancy, you can save storage space. Similarly, it is useful for retrieving data from database using queries.

2. Data Consistency:

By controlling the data redundancy, the data consistency is obtained. If a data item appears only once, any update to its value has to be performed only once and the updated value (new value of item) is immediately available to all users.

If the DBMS has reduced redundancy to a minimum level, the database system enforces consistency. It means that when a data item appears more than once in the database and is updated, the DBMS automatically updates each occurrence of a data item in the database.

3. Data Sharing:

In DBMS, data can be shared by authorized users of the organization. The DBA manages the data and gives rights to users to access the data. Many users can be authorized to access the same set of information simultaneously. The remote users can also share same data. Similarly, the data of same database can be shared between different application programs.

4. Data Integration:

In DBMS, data in database is stored in tables. A single database contains multiple tables and relationships can be created between tables (or associated data entities). This makes easy to retrieve and update data.

5. Integrity Constraints:

Integrity constraints or consistency rules can be applied to database so that the correct data can be entered into database. The constraints may be applied to data item within a single record or they may be applied to relationships between records.

Examples:

The examples of integrity constraints are:

(i) 'Issue Date' in a library system cannot be later than the corresponding 'Return Date' of a book.

(ii) Maximum obtained marks in a subject cannot exceed 100.

(iii) Registration number of BCS and MCS students must start with 'BCS' and 'MCS' respectively etc.

There are also some standard constraints that are intrinsic in most of the DBMSs. These are;

Constraint Name	**Description**
PRIMARY KEY	Designates a column or combination of columns as Primary Key and therefore, values of columns cannot be repeated or left blank.
FOREIGN KEY	Relates one table with another table.
UNIQUE	Specifies that values of a column or combination of columns cannot be repeated.

NOT NULL	Specifies that a column cannot contain empty values.
CHECK	Specifies a condition which each row of a table must satisfy.

6. Data Security:

Data security is the protection of the database from unauthorized users. Only the authorized persons are allowed to access the database. Some of the users may be allowed to access only a part of database i.e., the data that is related to them or related to their department. Mostly, the DBA or head of a department can access all the data in the database. Some users may be permitted only to retrieve data, whereas others are allowed to retrieve as well as to update data. The database access is controlled by the DBA. He creates the accounts of users and gives rights to access the database. Typically, users or group of users are given usernames protected by passwords.

7. Data Atomicity:

A transaction in commercial databases is referred to as atomic unit of work. For example, when you purchase something from a point of sale (POS) terminal, a number of tasks are performed such as;

- Company stock is updated.
- Amount is added in company's account.
- Sales person's commission increases etc.

All these tasks collectively are called an atomic unit of work or transaction. These tasks must be completed in all; otherwise partially completed tasks are rolled back. Thus through DBMS, it is ensured that only consistent data exists within the database.

8. Database Access Language:

Most of the DBMSs provide SQL as standard database access language. It is used to access data from multiple tables of a database.

9. Development of Application:

The cost and time for developing new applications is also reduced. The DBMS provides tools that can be used to develop application programs. For example, some wizards are available to generate Forms and Reports. Stored procedures (stored on server side) also reduce the size of application programs.

10. Creating Forms:

Form is very important object of DBMS. You can create Forms very easily and quickly in DBMS, Once a Form is created, it can be used many times and it can be modified very easily. The created Forms are also saved along with database and behave like a software component.

A Form provides very easy way (user-friendly interface) to enter data into database, edit data, and display data from database. The non-technical users can also perform various operations on databases through Forms without going into the technical details of a database.

11. Report Writers:

Most of the DBMSs provide the report writer tools used to create reports. The users can create reports very easily and quickly. Once a report is created, it can be used many times and it can be modified very easily. The created reports are also saved along with database and behave like a software component.

12. Control over Concurrency:

In a computer file-based system, if two users are allowed to access data simultaneously, it is possible that they will interfere with each other. For example, if both users attempt to perform update operation on the same

record, then one may overwrite the values recorded by the other. Most DBMSs have sub-systems to control the concurrency so that transactions are always recorded" with accuracy.

13. Backup and Recovery Procedures:

In a computer file-based system, the user creates the backup of data regularly to protect the valuable data from damaging due to failures to the computer system or application program. It is a time consuming method, if volume of data is large. Most of the DBMSs provide the 'backup and recovery' sub-systems that automatically create the backup of data and restore data if required. For example, if the computer system fails in the middle (or end) of an update operation of the program, the recovery sub-system is responsible for making sure that the database is restored to the state it was in before the program started executing.

14. Data Independence:

The separation of data structure of database from the application program that is used to access data from database is called data independence. In DBMS, database and application programs are separated from each other. The DBMS sits in between them. You can easily change the structure of database without modifying the application program. For example you can modify the size or data type of a data items (fields of a database table).

http://www.basicsofcomputer.com/figure_processing.JPG

On the other hand, in computer file-based system, the structure of data items are built into the individual application programs. Thus the data is dependent on the data file and vice versa.

15. Advanced Capabilities:

DBMS also provides advance capabilities for online access and reporting of data through Internet. Today, most of the database systems are online. The database technology is used in conjunction with Internet technology to access data on the web servers.

Disadvantages of Database Management System (DBMS):

Although there are many advantages but the DBMS may also have some minor *disadvantages*. These are:

1. Cost of Hardware & Software:

A processor with high speed of data processing and memory of large size is required to run the DBMS software. It means that you have to upgrade the hardware used for file-based system. Similarly, DBMS software is also Very costly.

2. Cost of Data Conversion:

When a computer file-based system is replaced with a database system, the data stored into data file must be converted to database files. It is difficult and time consuming method to convert data of data files into database. You have to hire DBA (or database designer) and system designer along with application programmers; alternatively, you have to take the services of some software houses. So a lot of money has to be paid for developing database and related software.

3. Cost of Staff Training:

Most DBMSs are often complex systems so the training for users to use the DBMS is required. Training is required at all levels, including programming, application development, and database administration. The organization has to pay a lot of amount on the training of staff to run the DBMS.

4. Appointing Technical Staff:

The trained technical persons such as database administrator and application programmers etc are required to handle the DBMS. You have to pay handsome salaries to these persons. Therefore, the system cost increases.

5. Database Failures:

In most of the organizations, all data is integrated into a single database. If database is corrupted due to power failure or it is corrupted on the storage media, then our valuable data may be lost or whole system stops.

Exercise: Check Your Progress 3
Note: i) Use the space below for your answer.
Ex1: Who was the founder of Relational DBMS?

……..……

……..……

……...

Ex2: Write the brief history of DBMS.

……..……

……..……

……...

1.11 Summary

- A database system is partitioned into modules that deal with each of the responsibilities of the overall system.
- The DBMS define as in interface between the program and the operating system to access or manipulate the database.
- The collection of information stored in the database at a particular moment is called an instance of the database.
- The overall design of the database is called the schema. It is of three types: physical schema, conceptual schema and external schema.
- A database administrator (DBA) directs or performs all activities related to maintaining a successful database environment. Responsibilities include designing, implementing, and maintaining the database system.

1.12 Keywords

Abstraction: It is the process of taking away or removing characteristics from something in order to reduce it to a set of essential characteristics.
Database Management System (DBMS): Collection of interrelated data set of programs to access those data DBMS contains information about a particular enterprise.
Data Manipulation Language (DML): Language for accessing and manipulating the data organized by the appropriate data model DML also known as query language.
Data Definition Language: A DDL is a language used to define data structures within a database. It is typically considered to be a subset of SQL, the Structured Query Language, but can also refer to languages that define other types of data.
Data Models: A collection of tools for describing data relationships, data semantics, and data constraints.

1.13 Review Questions

1. Describe the three levels of data abstraction.
2. What are the views of data?
3. What is the three-level architecture proposal?
4. Describe the instances and schemas.
5. Explain the purpose of database system and data abstraction.
6. What is the data independence?
7. Differentiate between DDL and DML.
8. Discuss the role of database administrator and users.
9. Explain the overall structure of DBMS.
10. What are the advantages and disadvantages of DBMS?

Answers for Self Assessment Questions

1. (a) 2. (a) 3. (a) 4. (b) 5. (a) 6. (a)

1.14 Further Studies

1. Database Management System By Seema Kedar

2

Entity-relationship Model

STRUCTURE

2.0 Objectives

2.1 Introduction

2.2 Concept of Entity-relationship Model

2.3 Relationship and Relationship Set

2.4 Mapping Constraints

2.5 Entity-relationship Diagram

2.6 Strong and Weak Entities

2.7 Specialization and Generalization

2.8 Aggregation

2.9 Reduction of an E-R Schema to Tables

2.10 Summary

2.11 Keywords

2.12 Review Questions

2.0 Objectives

After studying this chapter, you will be able to:

- Define the E-R model
- Explain the types of keys
- Discuss the relationship sets
- Explain the mapping constraints

2.1 Introduction

The entity-relationship (E-R) data model is based on a perception of a real world that consists of a set of basic objects called entities, and of relationships among these objects. It was developed to facilitate database design by allowing the specification of an enterprise schema, which represents the overall logical structure of a database. The E-R data model is one of several semantic data models; the semantic aspect of the model lies in the attempt to represent the meaning of the data. The E-R model is extremely useful in mapping the meanings and interactions of real-world enterprises onto a conceptual schema. Because of this utility, many database-design tools draw on concepts from the E-R model.

2.2 Concept of Entity-relationship Model

When a relational database is to be designed, an entity-relationship diagram is drawn at an early stage and developed as the requirements of the database and its processing become better understood. Drawing an entity-relationship diagram aids understanding of an organization's data needs and can serve as a schema diagram for the required system's database. A schema diagram is any diagram that attempts to show the structure of the data in a database. Nearly all systems analysis and design methodologies contain entity-relationship diagramming as an important part of the methodology and nearly all CASE (Computer Aided Software Engineering) tools contain the facility for drawing entity-relationship diagrams. An entity-relationship diagram could serve as the basis for the design of the files in a conventional file-based system as well as for a schema diagram in a database system.

There are three basic notions that the E-R data model employs: entity sets, relationship sets, and attributes.

2.2.1 Entity and Entity Sets

- An entity is an object that exists and is distinguishable from other objects. For instance, John Harris with S.I.N. 890-12-3456 is an entity, as he can be uniquely identified as one particular person in the universe.
- An entity may be concrete (a person or a book, for example) or abstract (like a holiday or a concept).
- An entity set is a set of entities of the same type (e.g., all persons having an account at a bank).
- Entity sets need not be disjoint. For example, the entity set employee (all employees of a bank) and the entity set customer (all customers of the bank) may have members in common.
- An entity is represented by a set of attributes.
  - E.g. name, S.I.N., street, city for ``customer'' entity.
  - The domain of the attribute is the set of permitted values (e.g. the telephone number must be seven positive integers).
- Formally, an attribute is a function which maps an entity set into a domain.
  - Every entity is described by a set of (attribute, data value) pairs.
  - There is one pair for each attribute of the entity set.
  - E.g. a particular customer entity is described by the set {(name, Harris), (S.I.N., 890-123-456), (street, North), (city, Georgetown)}.

An analogy can be made with the programming language notion of type definition.

- The concept of an entity set corresponds to the programming language type definition.
- A variable of a given type has a particular value at a point in time.
- Thus, a programming language variable corresponds to an entity in the E-R model.

We will be dealing with five entity sets in this section:

- *branch,* the set of all branches of a particular bank. Each branch is described by the attributes branch-name, branch-city and assets.
- *customer,* the set of all people having an account at the bank. Attributes are customer-name, S.I.N., street and customer-city.
- *employee,* with attributes employee-name and phone-number.
- *account,* the set of all accounts created and maintained in the bank. Attributes are account-number and balance.
- *transaction,* the set of all account transactions executed in the bank. Attributes are transaction-number, date and amount..

2.2.2 Attributes and Keys

An entity is represented by a set of attributes. Attributes are descriptive properties possessed by each member of an entity set. The designation of an attribute for an entity set expresses that the database stores similar information concerning each entity in the entity set; however, each entity may have its own value for each attribute. Possible attributes of the customer entity set are customer-name, social security, customer-street, and customer-city. Possible attributes of the loan entity set are loan-number and amount. For each attribute, there is a set of permitted values, called the domain, or value set, of that attribute. The domain of attribute customer-name might be the set of all text strings of a certain length. Similarly, the domain of attribute loan-number might be the set of all positive integers.

A database thus includes a collection of entity sets each of which contains any number of entities of the same type. Figure 2.1 shows part of a bank database which consists of two entity sets: customer and loan.

Amit	321-12-3123	Main	Harrison
Soumy	019-28-3746	North	Ray
Hari	677-89-9011	Main	Harrison
Tina	555-55-5555	Dupont	Woodside
Ramlal	244-66-8800	North	Ray
Gita	963-96-3963	Nassau	Princeton
Shailja	335-57-7991	Spring	Princeton

Customer

L-17	1000
L-23	2000
L-15	1500
L-14	1500
L-93	500
L-11	900
L-16	1300

Loan

Figure 2.1: Entity sets customer and loan

Note also that a composite attribute may appear as a hierarchy. Returning to our example composite attribute of customer-address, its component attribute street can be further divided into street-number, street-name, and apt-number. These examples of composite attributes, for the customer entity set, are depicted in Figure 2.2.

Single-valued and multivalve attributes: The attributes that we have specified in our examples all have a single value for a particular entity. For instance, the loan-number attribute for a specific loan entity refers to only one loan number. Such attributes are said to be single valued. There may be instances where an attribute has a set of values for a specific entity. Consider an employee entity set with the attribute dependent-name. Any particular employee may have zero, one, or more dependents; therefore, different employee entities within the entity set will have different numbers of values for the dependent-name attribute. This type of attribute is said to be multivalued. Where appropriate, upper and lower bounds may be placed on the number of values in a multivalued attribute. For example, a bank may limit the number of addresses recorded for a single customer to two. Placing bounds in this case expresses that the customer-address attribute of the customer entity set may have between zero and two values.

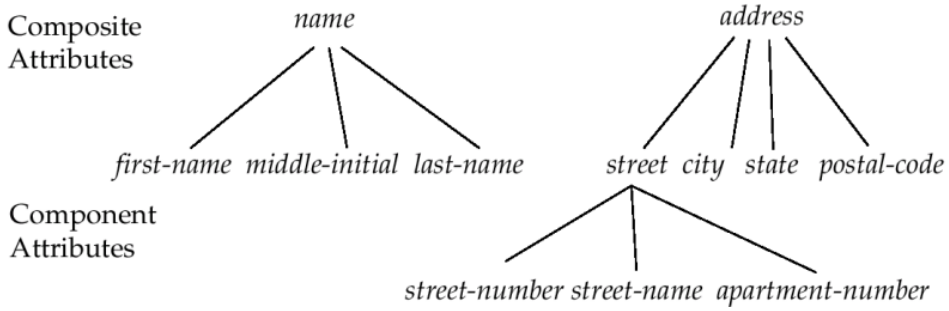


Figure 2.2 Composite attributes customer-name and customer-address.

Null attributes: A null value is used when an entity does not have a value for an attribute. As an illustration, if a particular employee has no dependents, the dependent-name value for that employee will be null, and will have the meaning of "not applicable." Null can also designate that an attribute value is unknown. An unknown

value may be either missing (the value does exist, but we do not have that information) or not known (we do not know whether or not the value actually exists). For instance, if the social-security value for a particular customer is null, we assume that the value is missing, since it is required for tax reporting. A null value for the apt-number attribute could mean that the address does not include an apartment number, that an apartment number exists but we do not know what it is, or that we do not know whether or not an apartment number is part of the customer's address.

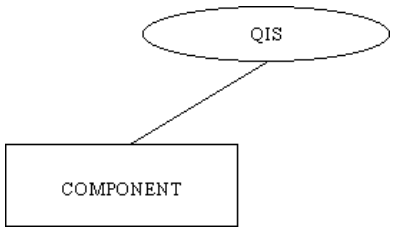


Figure 2.3: An entity type CUSTOMER and one of its attributes Cus_no.

In Figure 2.3, the attribute CUS_NO is shown. Assuming the organization storing the data ensures that each customer is allocated a different cus_no, that attribute could act as the primary key, since it identifies each customer; it distinguishes each customer from all the rest. No two customers have the same value for the attribute cus_no. Some people would say that an attribute is a candidate for being a primary key because it is 'unique'. They mean that no two entities within that entity type can have the same value of that attribute. In practice it is best not to use that word because it has other connotations.

As already mentioned, you may need to have a group of attributes to form a primary key, rather than just one attribute, although the latter is more common. For example if the organization using the CUSTOMER entity type did not allocate a customer number to its customers, then it might be necessary to use a composite key, for example one consisting of the attributes SURNAME and INITIALS together, to distinguish between customers with common surnames such as Smith. Even this may not be sufficient in some cases.

Primary keys are not the only attributes you might want to show on the entity-relationship diagram. For example, in a manufacturing organization you might have an entity type called COMPONENT and you want to make it clear on the entity-relationship diagram that the entities within the type are not single components but a component type such as a BC109 transistor. There are thousands of BC109s in stock and any one will do for any application. It is therefore not necessary to identify each BC109 differently (they all look and work the same). However you might want to distinguish BC109s from another transistor type BC108. To make it clear that you are considering all the BC109s as one entity and all the BC108s as another entity, you might put the attribute QIS (quantity in stock) on the entity-relationship diagram as in Figure. 2.4. This makes it clearer at the entity-relationship model level that each entity in the entity type is in fact a stock item of which there will be several in stock. Any doubts on this point should be resolved by inspecting the entity description, which shows all the attributes of the entity type and (ideally) their meaning. The primary key might be STOCK_NO and one of the attributes QIS, which should remove any doubt on this point.

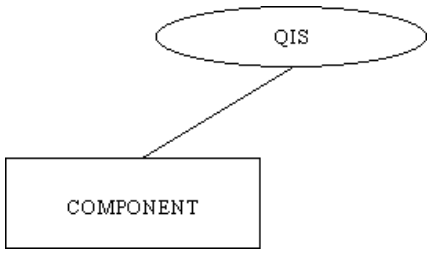


Figure 2.4: A well-placed attribute may clarify the meaning of an entity type.

Did You Know?

The database concept has evolved since the 1960s to ease increasing difficulties in designing, building, and maintaining complex information system.

2.3 Relationship and Relationship Set

A relationship is an association among several entities. For example, we can define a relationship that associates customers Hari with Loan L-15. This relationship specifies that Hari is a customer with loan number L-15.

A relationship set is a set of relationships of the same type. Formally, it is a mathematical relation on $n \geq 2$ (possibly non-distinct) entity sets. If $E_1, E_2, \ldots\ldots, E_n$ are entity sets, then a relationship set R is a subset of

$$\{(e_1, e_2, ..., e_n) \mid e_1 \in E_1, e_2 \in E_2, ..., e_n \in E_n\}$$

where $(e_1, e_2, ... , e_n)$ is a relationship.

Consider the two entity sets customer and loan in Figure 2.1. We define the relationship set borrower to denote the association between customers and the bank loans that the customers have. This association is depicted in Figure 2.5.

The number of entity sets that participate in a relationship set is also the degree of the relationship set. A binary relationship set is of degree 2; a ternary relationship set is of degree 3.

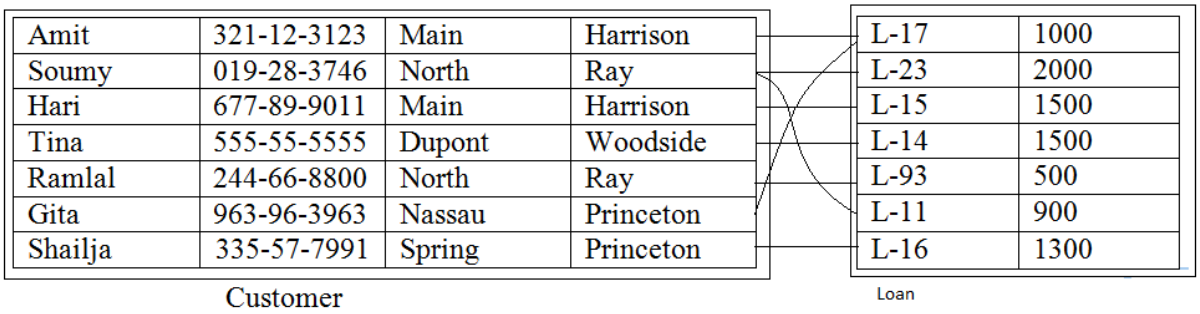


Figure 2.5 Relationship set borrower.

2.3.1 Design Issues

The notions of an entity set and a relationship set are not precise, and it is possible to define a set of entities and the relationships among them in a number of different ways. We examine basic issues in the design of an E-R database schema. The design process is covered in further detail.

Use of Entity Sets or Attributes

Consider the entity set employee with attributes employee-name and telephone-number. It can easily be argued that a telephone is an entity in its own right with attributes telephone-number and location (the office where the telephone is located). If we take this point of view, the employee entity set must be redefined as follows:

- The employee entity set with attribute employee-name
- The telephone entity set with attributes telephone-number and location
- The relationship set emp-telephone, which denotes the association between employees and the telephones that they have

Use of Entity Sets or Relationship Sets

It is not always clear whether an object is best expressed by an entity set or a relationship set. We assumed that a bank loan is modeled as an entity. An alternative is to model a loan not as an entity, but rather as a relationship between customers and branches, with loan-number and amount as descriptive attributes. Each loan is represented by a relationship between a customer and a branch.

If every loan is held by exactly one customer and is associated with exactly one branch, we may find satisfactory the design where a loan is represented as a relationship. However, with this design, we cannot represent conveniently a situation in which several customers hold a loan jointly. We must define a separate relationship for each holder of the joint loan. Then, we must replicate the values for the descriptive attributes loan-number and amount in each such relationship. Each such relationship must, of course, have the same value for the descriptive attributes loan-number and amount.
Two problems arise as a result of the replication:
(1) The data are stored multiple times, wasting storage space; and
(2) Updates potentially leave the data in an inconsistent state, where the values differ in two relationships for attributes that are supposed to have the same value. The issue of how to avoid such replication is treated formally by normalization theory. The problem of replication of the attributes loan-number and amount is absent in the original design. Because there loan is an entity set.
One possible guideline in determining whether to use an entity set or a relationship set is to designate a relationship set to describe an action that occurs between entities. This approach can also be useful in deciding whether certain attributes may be more appropriately expressed as relationships.

2.3.2 Binary versus n-ary Relationship Sets

It is always possible to replace a non-binary (n-ary, for n > 2) relationship set by a number of distinct binary relationship sets. For simplicity, consider the abstract ternary (n = 3) relationship set R, relating entity sets A, B, and C. We replace the relationship set R by an entity set E, and create three relationship sets:

- R_A, relating E and A
- R_B, relating E and B
- R_C, relating E and C

If the relationship set R had any attributes, these are assigned to entity set E; otherwise, a special identifying attribute is created for E (since every entity set must have at least one attribute to distinguish members of the set). For each relationship (a_i, b_i; c_i;) in the relationship set R, we create a new entity e_i; in the entity set E. Then, in each of the three new relationship sets, we insert a relationship as follows:

- (e_i, a_i;) in R_A
- (e_i, b_i;) in R_b
- (e_i, c_i;) in R_C

We can generalize this process in a straightforward manner to n-ary relationship sets. Thus, conceptually, we can restrict the E-R model to include only binary relationship sets. However, this restriction is not always desirable.

- An identifying attribute may have to be created for the entity set created to represent the relationship set. This attribute, along with the extra relationship sets required, increase the complexity of the design overall storage requirements.
- An n-ary relationship set shows more clearly that several entities participate in a single relationship. In the corresponding design using binary relationships, it is more difficult to enforce this participation constraint.

2.4 Mapping Constraints

An E-R enterprise schema may define certain constraints to which the contents of a database must conform. In this section, we examine mapping cardinalities and existence dependencies-two of the most important types of constraints.

2.4.1 Mapping Cardinalities

Mapping cardinalities, or cardinality ratios, express the number of entities to which another entity can be associated via a relationship set.

Mapping cardinalities are most useful in describing binary relationship sets, although occasionally they contribute to the description of relationship sets that involve more than two entity sets. We shall concentrate on only binary relationship sets.

For a binary relationship set R between entity sets A and *B*, the mapping cardinality must be one of the following:

- **One to one.** An entity in A is associated with at most one entity in *B*, and an entity in *B* is associated with at most one entity in *A.*
- **One to many.** An entity in A is associated with any number of entities in *B.* An entity in *B*, however, can be associated with at most one entity in A.
- **Many to one.** An entity in A is associated with at most one entity in *B.* An entity in *B*, however, can be associated with any number of entities in A.
- **Many to many.** An entity in A is associated with any number of entities in *B*, and an entity in *B* is associated with any number of entities in A.

The appropriate mapping cardinality for a particular relationship set is obviously dependent on the real-world situation that is being modelled by the relationship set.

As an illustration, consider the borrower relationship set. If, in a particular bank, a loan can belong to only one customer, and a customer can have several loans, then the relationship set from customer to loan is one to many. If a loan can belong to several customers (as can loans taken jointly by several business partners), the relationship set is many to many.

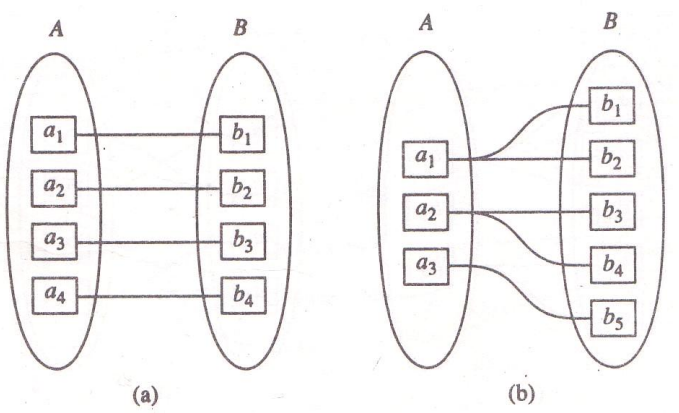


Figure 2.6: Mapping cardinalities. (a) One to one. (b) One to many.

The cardinality ratio of a relationship can affect the placement of relationship attributes. Attributes of one-to-one or one-to-many relationship sets can be associated with one of the participating entity sets, rather than with the relationship set. For instance, let us specify that depositor is a one-to-many relationship set such that one customer may have several accounts, but each account is held by only one customer. In this case, the attribute access-date could be associated with the account entity set, as depicted in Figure 2.8; to keep the figure simple, only some of the attributes of the two entity sets are shown. Since each account entity participates in a relationship with at most one instance of customer, making this attribute designation would have the same meaning as would placing access-date with the depositor relationship set. Attributes of a one-to-many relationship set can be repositioned to only the entity set on the "many" side of the relationship. For one-to-one relationship sets, the relationship attribute can be associated with either one of the participating entities.

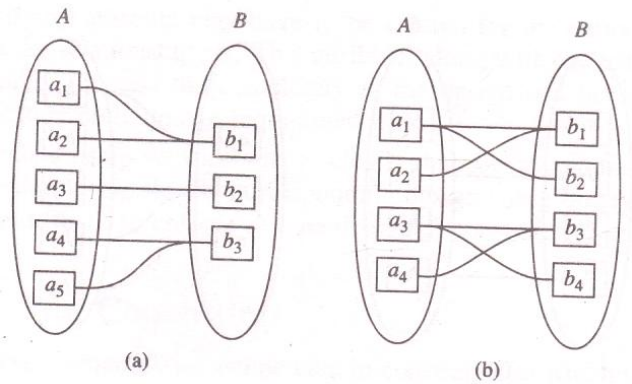


Figure 2.7: Mapping cardinalities. (a) Many to one. (b) Many to many.

The design decision of where to place descriptive attributes in such cases-as a relationship or entity attribute-should reflect the characteristics of the enterprise being modelled. The designer may choose to retain access-date as an attribute of depositor to express explicitly that an access occurs at the point of interaction between the customer and account entity sets.

The choice of attribute placement is more clear-cut for many-to-many relationship sets. Returning to our example, let us specify the perhaps more realistic case that depositor is a many-to-many relationship set expressing that a customer may have one or more accounts, and that an account can be held by one or more customers. If we are to express the date on which a specific customer last accessed a specific account, access-date must be an attribute of the depositor relationship set, rather than either one of the participating entities. If access-date were an attribute of account, for instance, we could not determine which customer made the most recent access to a joint account. When an attribute is determined by the combination of participating entity set, rather than by either entity separately, that attribute must be associated with the many-to-many relationship set. The placement of access-date as a relationship attribute is depicted in Figure 2.8; again, to keep the figure simple, only some of the attributes of the two entity sets are shown.

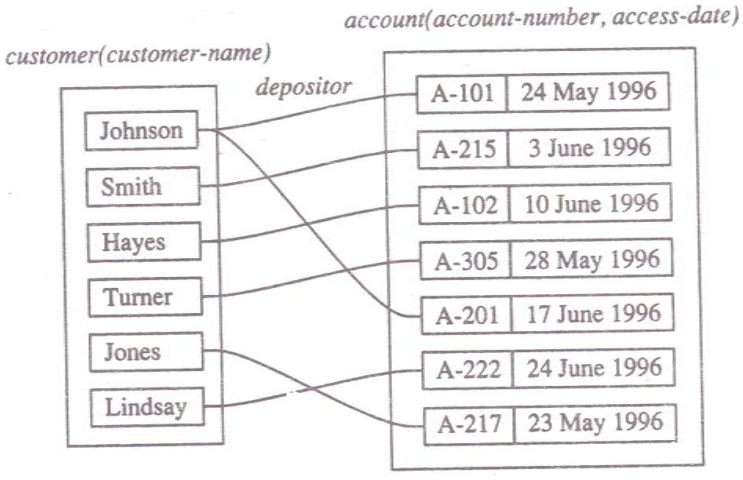


Figure 2.8: Access-date as attribute of the account entity set.

2.4.2 Existence Dependencies

Another important class of constraints is existence dependencies. Specifically, if the existence of entity x depends on the existence of entity y, then x is said to be existence dependent on y. Operationally, if y is deleted, so is x. Entity y is said to be a dominant entity, and x is said to be a subordinate entity.

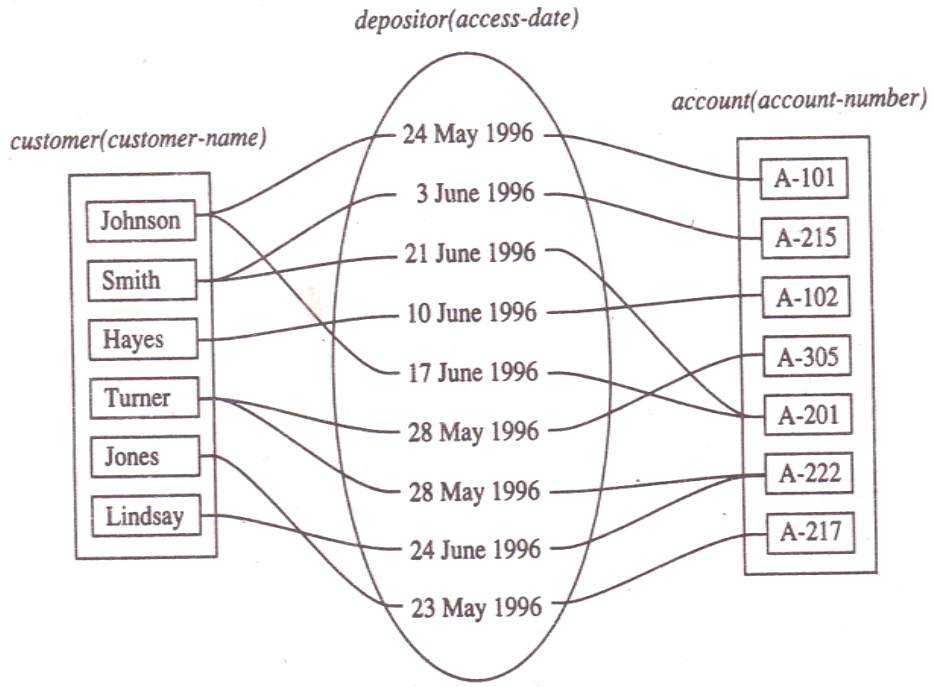


Figure 2.9: Access-date as attribute of the depositor relationship set.

As an illustration, consider the entity set loan and the entity set payment that keeps information about all the payments that were made in connection to a particular loan. The payment entity set is described by the attributes payment-number, payment-date, and payment-amount. We form a relationship set loan-payment between these two entity sets, which is one-to-many from loan to payment. Every payment entity must be associated with a loan entity. If a loan entity is deleted, then all its associated payment entities must be deleted also. In contrast, payment entities can be deleted from the database without affecting any loan. The entity set loan, therefore, is dominant and payment is subordinate, in the loan-payment relationship set.

The participation of an entity set E in a relationship set R is said to be total if every entity in E participates in at least one relationship in R. If only some entities in E participate in relationships in R, the participation of entity set E in relationship R is said to be partial. Total participation is closely related to existence dependency. For example, since every payment entity must be related to some loan entity by the loan-payment relationship, the participation of payment in the relationship set loan-payment is total. In contrast, an individual can be a bank customer whether or not she has a loan with the bank. Hence, it is possible that only a partial set of the customer entities relate to the loan entity set.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: What is the concept of Entity Relationship model?

……..……

……..……

……...

Ex2: What is Mapping Constraints?

……..……

……..……

……...

2.5 Entity-relationship Diagram

The overall logical structure of a database can be expressed graphically by an E-R diagram the relative simplicity and pictorial clarity of this diagramming technique may well account in large part for the widespread use of the E-R model. Such a diagram consists of the following major components:

- **Rectangles,** which represent entity sets r'
- **Ellipses,** which represent attributes f
- **Diamonds,** which represent relationship sets
- **Lines,** which link attributes to entity sets and entity sets to relationship sets
- **Double ellipses,** which represent multivalve attributes
- **Dashed ellipses,** which denote derived attributes
- **Double Lines,** which indicate total participation of an entity in a relationship set

As depicted in Figure 2.10, attributes of an entity set that are members of the primary key are underlined.
Consider the entity-relationship diagram in Figure 2.10, which consists of two entity sets, customer and loan, related through a binary relationship set borrower. The attributes associated with customer are customer-name, social-security, customer-street, and customer-city. The attributes associated with loan are loan-number and amount.

The relationship set borrower may be many-to-many, one-to-many, many-to-one, or one-to-one. To distinguish among these types, we draw either a directed line (->) or an undirected line (-) between the relationship set and the entity set in question.

- A directed line from the relationship set borrower to the entity set loan specifies that borrower is a one-to-one, or many-to-one relationship set, from customer to loan; borrower cannot be a many-to-many or a one-to-many relationship set, from customer to loan.

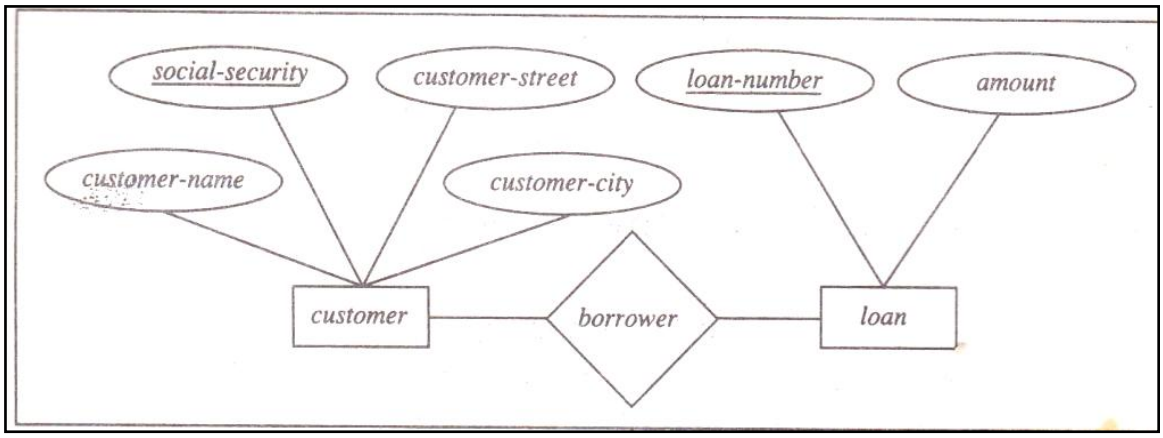


Figure 2.10: E-R diagram corresponding to customers and loans.

- An undirected line from the relationship set borrower to the entity set loan specifies that borrower is either a many-to-many, or one-to-many relationship set, from customer to loan.

Returning to the E-R diagram of Figure 2.10, we see that the relationship set borrower is many to many. If the relationship set borrower were one to many, from customer to loan, then the line from borrower to customer would be directed, with an arrow pointing to the customer entity set (Figure 2.11a). Similarly, if the relationship set borrower were many to one from customer to loan, then the line from borrower to loan would have an arrow pointing to the loan entity set (Figure 2.11b). Finally, if the relationship set borrower were one to one, then both lines from borrower would have arrows: one pointing to the loan entity set, and one pointing to the customer entity set (Figure 2.12).

If a relationship set has also some attributes associated with it, then we link these attributes to that relationship set. For example, in Figure 2.13, we have the access-date descriptive attribute attached to the relationship set depositor to specify the most recent date on which a customer accessed that account.

We indicate roles in E-R diagrams by labelling the lines that connect diamonds to rectangles. Figure 2.14 shows the role indicators manager and worker between the employee entity set and the works for relationship set.

Non binary relationship sets can be specified easily in an E-R diagram. Figure 2.15 consists of the three entity sets customer, loan, and branch, related through the relationship set CLB. This diagram specifies that a customer may have several loans, and that loan may belong to several different customers. Further, the arrow pointing to branch indicates that each customer-loan pair is associated with a specific bank branch. If the diagram had an arrow pointing to customer, in addition to the arrow pointing to branch, the diagram would specify that each loan is associated with a specific customer and a specific bank branch.

Self Assessment Questions

1. An entity is represented by a set of ……

(a). Attributes (b). Object

(b). Both (a) and (b) (d). None of these

2. ……is a set of one or more attributes

(a). Candidate key (b). Primary key

(c). Super key (d). All of these

3. Ellipses, which represent attributes.

(a). True (b). False

4. ……**,** which represent entity sets

(a). Ellipse (b). Rectangles

(c). Circle (d). None of these

2.6 Strong and Weak Entities

2.6.1 Strong and Weak Entity Sets

Strong entity does not depend on some other entity. An entity set may not have sufficient attributes to form a primary key. Such an entity set is termed a weak entity set An entity set that lids a primary key is termed a strong- entity.

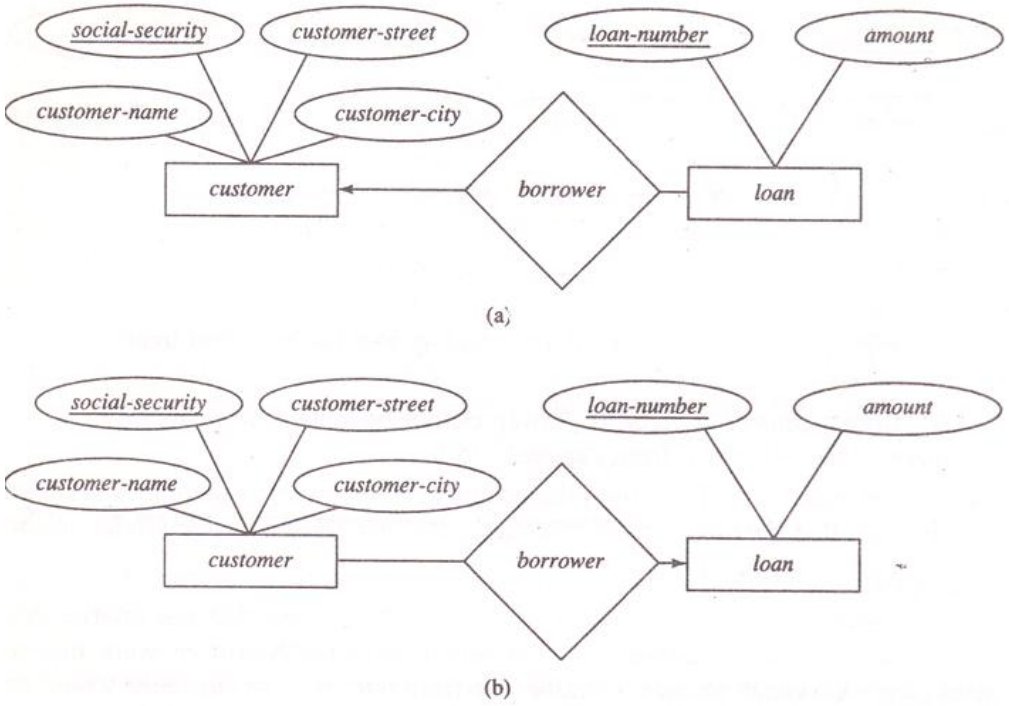


Figure 2.11: Relationships. (a) One to many. (b) Many to one.

As an illustration, consider the entity set payment, which has the three attributes: payment-number, payment-date, and payment-amount. Although each payment entity is distinct, payments for different loans may share the same payment number. Thus, this entity set does not have a primary key; it is a weak entity set. (For a weak entity set to be meaningful, it must be part of a one-to-many relationship set. This relationship set should have no descriptive attributes, since any required attributes can be associated with the weak entity set)

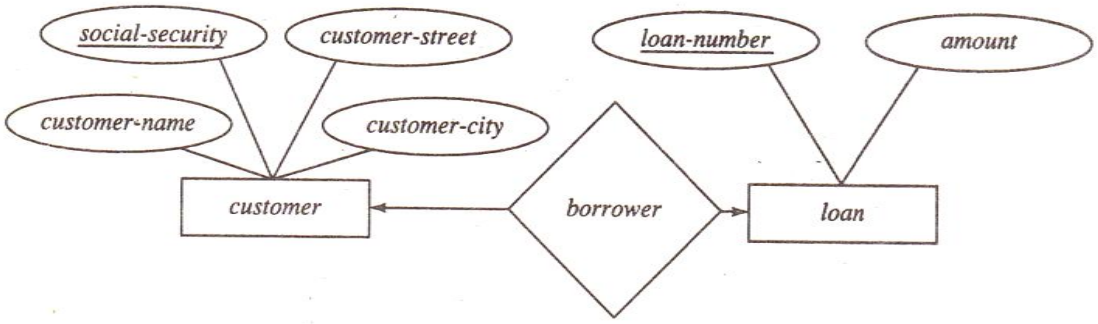


Figure 2.12: one to one relationship

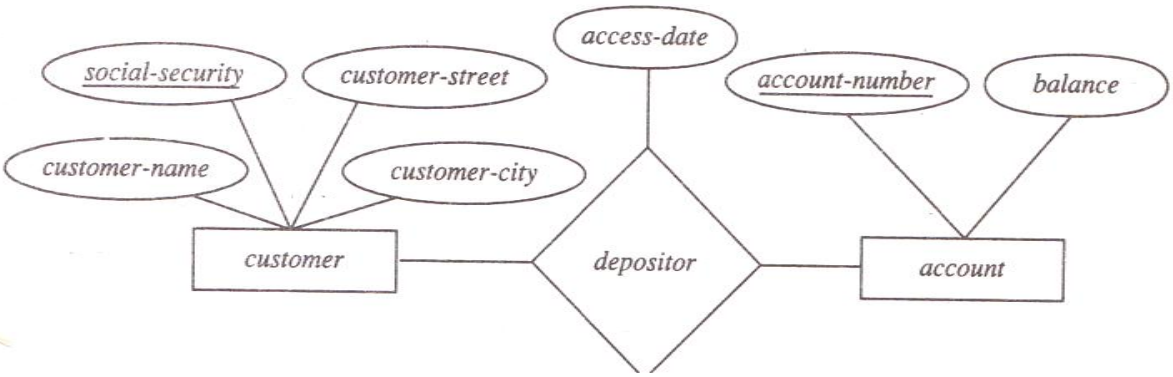


Figure 2.13: E-R diagram with an attribute attached to a relationship set.

The concepts of strong and weak entity sets are related to the existence dependencies introduced. A member of a strong entity set is by definition a dominant entity, whereas a member of a weak entity set is a subordinate entity

Although a weak entity set does not have a primary key, we nevertheless need means of distinguishing among all those entities in the entity set that depend on one particular strong entity. The discriminator of a weak entity set is a set of attributes that allows this distinction to be made) For example, the discriminator of the weak entity set payment is the attribute payment-number, since, for each loan, a payment number uniquely identifies one single payment for that loan. The discriminator of a weak entity set is also called the partial key of the entity set.

Primary key of a weak entity set is formed by the primary key of the strong entity set on which the weak entity set is existence dependent, plus the weak entity set's discriminator. In the case of the entity set payment, its primary key is {loan-number, payment-number}, where loan-number identifies the dominant entity of a payment, and payment-number distinguishes payment entities within the same loan.
Identifying dominant entity set is said to own the weak entity set that it identifies. The relationship that associates the weak entity set with an owner is the identifying relationship. In our example, loan-payment is the identifying relationship for payment.

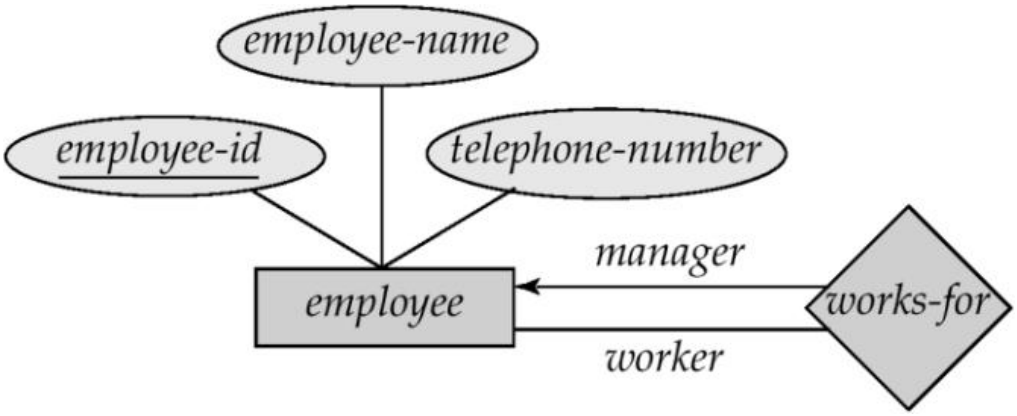


Figure 2.14: E-R diagram with role indicators.

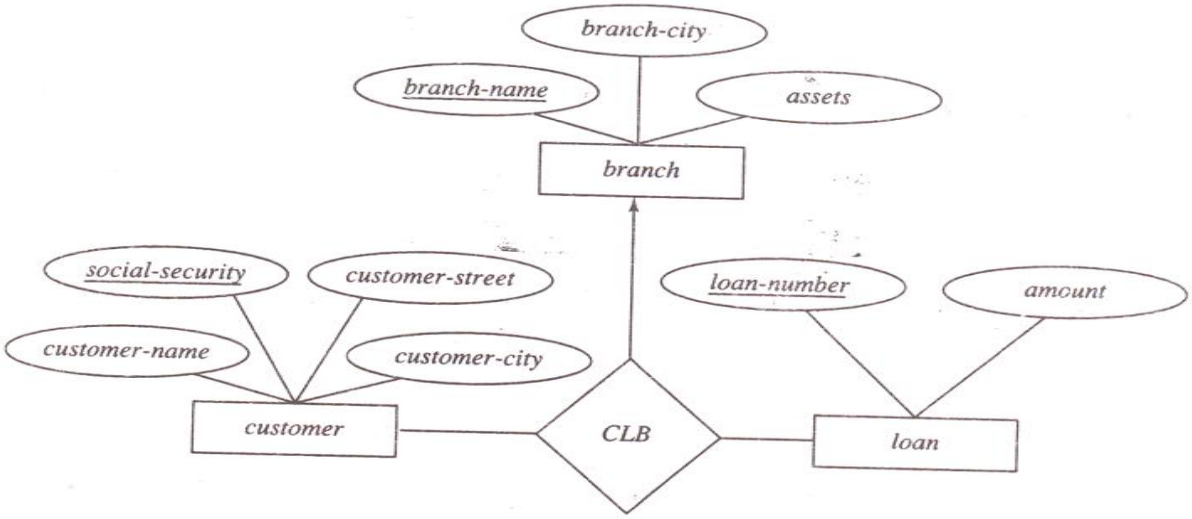


Figure 2.15: E-R diagram with a ternary relationship.

Weak entity set is indicated in E-R diagrams by a doubly outlined box, and the corresponding identifying relationship by a doubly outlined diamond In Figure 2.16, the weak entity set payment is dependent on the strong entity set loan via the relationship set loan-payment. The figure also illustrates the use of double lines to indicate total participation-the participation of the (weak) entity set payment in the relationship loan-payment is total, meaning that every payment must The related via loan-payment to some account. Finally, the arrow from loan-payment to loan indicates that each payment is for a single loan.(The discriminator of a weak entity set also is underlined, but with a dashed, rather than a solid, line.)
A weak entity set may participate as owner in an identifying relationship with another weak entity set. Even though a weak entity set is always existence dependent on a dominant entity, an existence dependency does not necessarily result in a weak entity set; that is, the subordinate entity set may have a primary key.

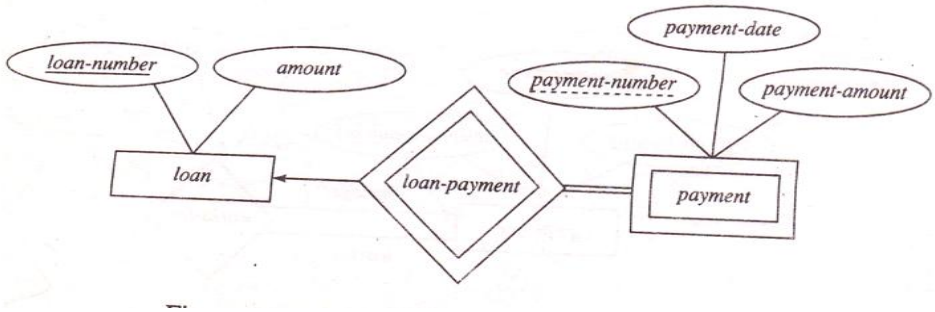


Figure 2.16: E-R diagram with a weak entity set.

In some cases, the database designer may choose to express a weak entity set as a multivalve, composite attribute of the owner entity set. In our example, this alternative would require that the entity set loan have a multivalve, composite attribute payment, consisting of payment-number, payment-date, and payment-amount. A weak entity set may be more appropriately modelled as an attribute if it participates in only the identifying relationship, and if it has few attributes. Conversely, a weak-entity-set representation will more aptly model a situation where the set participates in relationships other than the identifying relationship, and where the weak entity set has several attributes.

2.7 Specialization and Generalization

2.7.1 Specialization: When you see an upside down triangle point downwards with the text 'ISA' it is Specialization. In programming terms you can relate to it as some form of inheritance.
For example, A person entity is the top level entity with some common attributes(age,name etc) of any person. This entity is then specialized and sub groupings can be done for other entities such as 'Employee' or 'Customer'. Because employee and customer are both persons, so they will inherit the common properties of a person but in addition they will have their own attributes.
This can also be multiple levels. For example, the 'Employee' entity can further be sub grouped to 'Officer', 'temp-staff' etc. with their own attributes.

2.7.2 Generalization: Generalization and specialization are actually the same thing. They are the inverse of each other. However, it only differs in the design process. Specialization is a top-down design process, whereas generalization is a bottom-up design process. Which means you will first design the sub groupings like officer; temp-staff etc. and slowly move upwards to 'employer' - 'customer' and then design the 'person' entity. In the ER diagram generalization and specialization are both represented exactly same.

Exercise: Check Your Progress 2
Note: i) Use the space below for your answer.
Ex1: What are Strong and Weak Entities?

……..……

……..……

……...

Ex2: What is Specialization and Generalization?

……..……

……..……

……...

2.8 Aggregation

The E-R model cannot express relationships among relationships. When would we need such a thing?
Consider a DB with information about employees who work on a particular project and use a number of machines doing that work. We get the E-R diagram shown in Figure

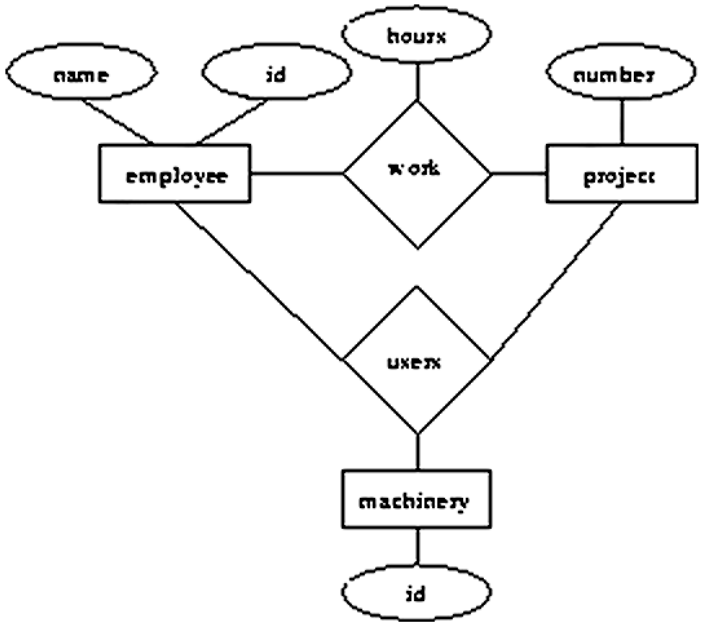
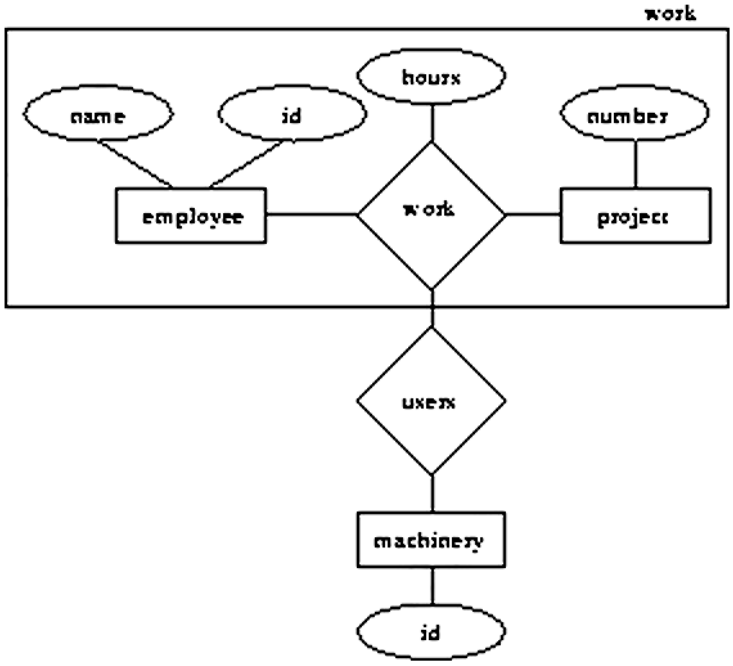


Figure 2.17: E-R diagram with redundant relationships Relationship sets work and uses could be combined into a single set. However, they shouldn't be, as this would obscure the logical structure of this scheme.
The solution is to use aggregation.

Through an abstraction relationships are treated as higher-level entities. For our example, we treat the relationship set work and the entity sets employee and project as a higher-level entity set called work.
Figure 2.18 shows the E-R diagram with aggregation.



Transforming an E-R diagram with aggregation into tabular form is easy. We create a table for each entity and relationship set as before. The table for relationship set uses contains a column for each attribute in the primary key of machinery and work.

2.9 Reduction of an E-R Schema to Tables

A database that conforms to an E-R database schema can be represented by a collection of tables. For each entity set and for each relationship set in the database, there is a unique table that is assigned the name of the corresponding entity set or relationship set. Each table has multiple columns, each of which has a unique name.

Both the E-R model and the relational-database model are abstract, logical representations of real-world enterprises. Because the two models employ similar design principles, we can convert an E-R design into a relational design. Converting a database representation from an E-R diagram to a table format is the basis for deriving a relational-database design from an E-R diagram. Although important differences exist between a relation and a table, informally, a relation can be considered to be a table of values. We describe how an E-R schema can be represented by tables; we show how to generate a relational-database schema from an E-R schema.

2.9.1 Tabular Representation of Strong Entity Sets

Let E be a strong entity set with descriptive attributes a_l, $a_2 . . ., a_n,,$. We represent this entity by a table called *E* with n distinct columns, each of which corresponds to one of the attributes of *E*. Each row in this table corresponds to one entity of the entity set *E*.

As an illustration, consider the entity set loan of the E-R diagram shown in Figure 2.10. This entity set has two attributes: loan-number and amount. We represent this entity set by a table called loan, with two columns, as shown in Figure 2.19. The row

(L-17, 1000)

In the loan table means that loan number L-17 has a loan amount of INR1000. We can add a new entity to the database by inserting a row into a table. We can also delete or modify rows.

Let D_1 denote the set of all loan numbers, and let D_2 denote the set of all balances. Any row of the loan table must consist of a 2-tuple (v_j, v_2), where v_1 is a loan (that is, v is in set D_1) and v_2 is an amount (that is, u_2 is in set D_2). In general, the loan table will contain only a subset of the set of all possible rows. We refer to the set of all possible rows of loan as the Cartesian product of D_1 and D_2, denoted by

D_1 x D_2

In general, if we have a table of *n* columns, we denote the Cartesian product of D_1, D_2, ..., D_n,, by

D_1 x D_2 x ... x D_{n-1} x D_n,,

loan-number	*amount i*
L-17	1000
L-23	2000
L-15	1500
L-14	1500
L-93	500
L-11	900
L-16	1300

Figure 2.19: The *loan* table.

customer-id	*customer-name*	*customer-street*	*customer-city*
019-28-3746	Smith	North	Rye
182-73-6091	Turner	Putnam	Stamford
192-83-7465	Johnson	Alma	Palo Alto
244-66-8800	Curry	North	Rye
321-12-3123	Jones	Main	Harrison
335-57-7991	Adams	Spring	Pittsfield
336-66-9999	Lindsay	Park	Pittsfield
677-89-9011	Hayes	Main	Harrison
963-96-3963	Williams	Nassau	Princeton

Figure 2.20: The *customer* table.

As another example, consider the entity set- customer of the E-R diagram shown in Figure 2.10.This entity set has the attributes customer-name, social-security, customer-street, and customer city. The table corresponding to customer has four columns, as shown in Figure 2.22.

2.9.2 Tabular Representation of Weak Entity Sets

Let A be a weak entity set with attributes a_l, a_z, ..., a_m. Let B be the strong entity set on which A is dependent. Let the primary key of B consist of attributes b_1, b_2, ..., b_n,,. We represent the entity set A by a table called A with one column for each attribute of the set:

$$\{a_1, a_2 + .. ., a_{m\}} U (b_1, b_{2+} \ldots, b_n\}$$

As an illustration, consider the entity set payment shown in the E-R diagram of Figure 2.16. This entity set has three attributes: payment-number, payment-date, and payment-amount. The primary key of the loan entity set, on which payment is dependent, is loan-number. Thus, payment is represented by a table with four columns labeled loan-number, payment-number, payment-date, and payment-amount, as depicted in Figure 2.22

2.9.3 Tabular Representation of Relationship Sets

Let R be a relationship set, let a_1, a_2, ...,a_m be the set of attributes formed by the union of the primary keys of each of the entity sets participating in R, and let the descriptive attributes (if any) of R be b_1, $b_{2,}$ ..., b,,. We represent this relationship set by a table called R with one column for each attribute of the set:

$$\{a_1,a_2,...,am\} \ ^{U} (b_1 \ b_2,...,b_{n,,})$$

loan-number	*payment-number*	*payment-date*	*payment-amount*
L-17	5	10 May 1996	50
L-23	11	17 May 1996	75
L-15	22	23 May 1996	300
L-14	69	28 May 1996	500
L-93	103	3 June 1996	900
L-17	6	7 June 1996	50
L-11	53	7 June 1996	125
L-93	104	13 June 1996	200
L-17	7	17 June 1996	100
L-16	58	18 June 1996	135

Figure 2.21: The payment table.

As an illustration, consider the relationship set borrower in the E-R diagram of Figure 2.10. This relationship set involves the following two entity sets:

- customer, with the primary key social-security
- loan, with the primary key loan-number

Since the relationship set has no attributes, the borrower table has two columns labeled social-security and loan-number, as shown in Figure 2.22.

2.9.4 Redundancy of Tables

The case of a relationship set linking a weak entity set to the corresponding strong entity set is special. As we noted earlier, these relationships are many to one and have no descriptive attributes. Furthermore, the primary key of a weak entity set includes the primary key of the strong entity set. In the E-R diagram of Figure 2.16, the weak entity set payment is dependent on the strong entity set loan via the relationship set loan-payment. The primary key of payment is (loan-number, payment-number), and the primary key of loan is {loan-number}. Since loan-payment has no descriptive attributes, the table for loan-payment would have two columns, loan-number and payment-number. The table for the entity set payment has four columns, loan-number, payment-number, payment-date, and payment-amount. Thus, the loan payment table is redundant. In general, the table for the relationship set linking a weak entity set to its corresponding strong entity set is redundant and does not need to be present in a tabular representation of an E-R diagram.

social-security	*loan-number '*
321-12-3123	L-17
019-28-3746	L-23
677-89-9011	L-15
555-55-5555	L-14
244-66-8800	L-93
019-28-3746	L-11
963-96-3963	L-17
335-57-7991	L-16

Figure 2.22: The borrower table.

Combination of Tables

Consider a many-to-one relationship set AB from entity set A to entity set B. Using our table-construction scheme outlined previously, we get three tables: A, B, and AB. However, if there is an existence dependency

of A on B (that is, for each entity a in A, the existence of a depends on the existence of some entity b in B), then we can combine the tables A and AB to form a single table consisting of the union of columns of both tables.

As an illustration, consider the E-R diagram of Figure 2.23. The relationship set account-branch is many to one from account to branch. Further, the double line in the E-R diagram indicates that the participation of account in the account-branch is total. Hence, an account cannot exist without being associated with a particular branch. Therefore, we require only the following two tables

- account, with- attributes account-number, balance, and branch-name branch, with attributes branch-name, branch-city, and assets

Multivalued Attributes

We have seen that attributes in an E-R diagram generally map directly into columns for the appropriate tables. Multivalued attributes, however, are an exception; new tables are created for these attributes.

For a multivalued attribute M, we create a table T with a column C that corresponds to M and columns corresponding to the primary key of the entity set or relationship set of which M is an attribute. As an illustration, consider the E-R diagram depicted in Figure 2.19. The diagram includes the multivalued attribute dependent-name. For this multivalued attribute, we create a table dependent-name, with columns dname, referring to the dependent-name attribute.

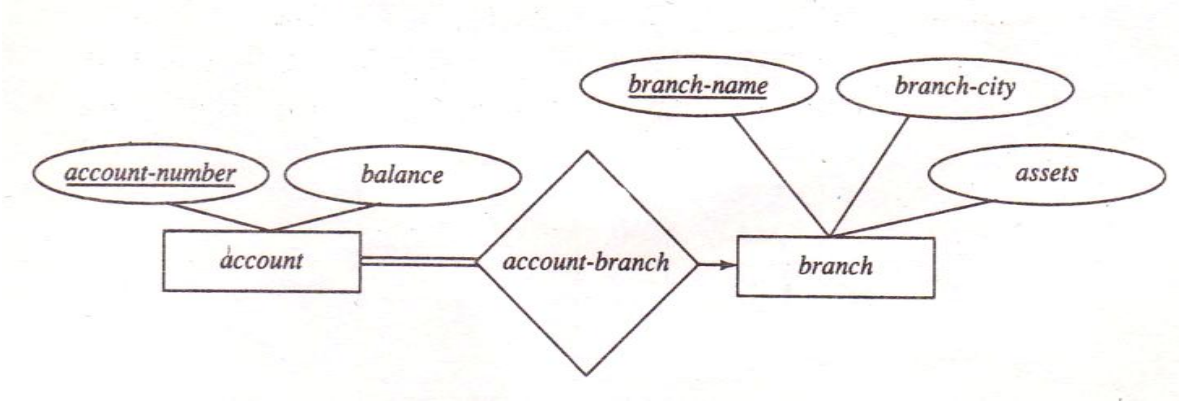


Figure 2.23: E-R diagram.

Employee, and e-social-security, representing the primary key of the entity set employee. Each dependent of an employee is represented as a unique row in the table.

2.9.5 Tabular Representation of Generalization

There are two different methods for transforming to a tabular form an E-R diagram that includes generalization: Although we refer to the generalization depicted in Figure 2.17, we have chosen to simplify this discussion by including only the first tier of lower-level entity sets-that is, savings-account and checking-account.

1. Create a table for the higher-level entity set. For each lower-level entity set, create a table that includes a column for each of the attributes of that entity set plus a column for each attribute of the primary key of the higher-level entity set. Thus, for the E-R diagram of Figure 2.17, we have three tables:

- account, with attributes account-number and balance
- savings-account, with attributes account-number and interest-rate
- checking-account, with attributes account-number and overdraft-amount

2. If the generalization is disjoint and complete-that is, if no entity is a member of two lower-level entity sets directly below a higher-level entity set, and if every entity in the higher level entity set is also a member of

one of the lower-level entity sets-then an alternative representation is possible. Here, create no table for the higher-level entity set. Instead, for each lower-level entity set, create a table that includes a column for each of the attributes of that entity set plus a column for *each* attribute of the higher-level entity set. Then, for the E-R diagram of Figure 2.17, we have two tables.

- savings-account, with attributes account-number, balance, and interest rate
- checking-account, with attributes account-number, balance, and overdraft-amount

The savings-account and checking-account relations corresponding to these tables both have balance as the primary key.

If the second method were used for an overlapping generalization, some values such as balance would be stored twice unnecessarily. Similarly, if the generalization were not complete-that is, if some accounts were neither savings nor checking accounts-then such accounts could not be represented with the second method.

2.9.6 Tabular Representation of Aggregation

Transforming to a tabular form an E-R diagram which includes aggregation is straightforward. Consider the diagram of Figure 2.19. The table for the relationship set loan-officer includes a column for each attribute in the primary key of the entity set employee and the relationship set borrower. It would also include a column for any descriptive attributes, if they existed, of the relationship set loan-officer. Using the same procedure as before for the rest of the diagram, we create, the following tables:

- customer, with attributes customer-name, social-security, customer-street, and customer-city
- loan, with attributes loan-number and amount
- borrower, with attributes social-security and loan-number
- employee, with attributes e-social-security, employee-name, and telephone-number
- Loan-officer, with attributes social-security, loan-number, and e-social-security.

Exercise: Check Your Progress 3

Note: i) Use the space below for your answer.

Ex1: What is Tabular Representation of Generalization?

……..……

……..……

……...

2.10 Summary

- The entity-relationship (E-R) data model is based on a perception of a real world that consists of a set of basic objects called entities, and of relationships among these objects
- A superkey is a set of one or more attributes that, taken collectively, allows us to identify uniquely an entity in the entity set
- Aggregation is an abstraction through which relationships are treated as higher-level entities.
- Mapping cardinalities, or cardinality ratios, express the number of entities to which another entity can be associated via a relationship set.
- A null value is used when an entity does not have a value for an attribute called null attribute.

2.11. Keywords

Attributes: It is descriptive properties possessed by each member of an entity set. The designation of an attribute for an entity

Database: A database thus includes a collection of entity sets each of which contains any number of entities of the same type.

Entity: It is a "thing" or "object" in the real world that is distinguishable from all other objects

Key: A key (primary, candidate, and super) is a property of the entity set.

Null attributes: A null value is used when an entity does not have a value for an attribute.

2.12. Review Questions

1. What is key concept?
2 Define the E-R model with example.
3. What is mapping constraints?
4. Describe the aggregation.
5 What is difference between null attributes and derived attribute?
6. Define the entity-relationship diagram?
7. Discuss the strong and weak entities?
8. How many types of attributes?
9. Explain the reducing E-R diagram to tables?
10. What is difference between specialization and aggregation?

Answers for Self Assessment Questions

1. (a) 2. (c) 3. (a) 4. (b)

2.13. Further Studies

1. Entity-Relationship Modeling: Foundations of Database Technology By Bernhard Thalheim

3

RDBMS Concept and Terminology

STRUCTURE

3.0 Objectives

3.1 Introduction

3.2 Concept of RDBMS and its Terminologies

3.3 Set Theory Concepts

3.4 Extension and Intention

3.5 Relationships

3.6 Integrity Rules

3.7 Domains and Relations

3.8 Concept of Keys

3.9 Summary

3.10 Keywords

3.11 Review Questions

3.12 Further Studies

3.0 Objectives

After studying this chapter, you will be able to:

- Discuss the set theory concepts in RDBMS
- Explain the extension and intention
- Understand database relationships
- Discuss the integrity rules

3.1 Introduction

The RDBMS stands for Relational Database Management System. RDBMS is the basis for SQL, and for all modern database systems like MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.

The model is based on branches of mathematics called set theory and predicate logic. The basic idea behind the relational model is that a database consists of a series of unordered tables (or relations) that can be manipulated using non-procedural operations that return tables. This model was in vast contrast to the more

traditional database theories of the time that were much more complicated, less flexible and dependent on the physical storage methods of the data.

It is commonly thought that the word relational in the relational model comes from the fact that you relate together tables in a relational database. Although this is a convenient way to think of the term, it is not accurate. Instead, the word relational has its roots in the terminology that Codd used to define the relational model. The table in Codd's writings was actually referred to as a relation (a related set of information). In fact, Codd (and other relational database theorists) use the terms relations, attributes and tuples where most of us use the more common terms tables, columns and rows, respectively (or the more physical—and thus less preferable for discussions of database design theory—files, fields and records).

The relational model can be applied to both databases and database management systems (DBMS) themselves. The relational fidelity of database programs can be compared using Codd's 12 rules (since Codd's seminal paper on the relational model, the number of rules has been expanded to 300) for determining how DBMS products conform to the relational model.

3.2 Concept of RDBMS and its Terminologies

When designing a database, you have to make decisions regarding how best to take some system in the real world and model it in a database. This consists of deciding which tables to create, what columns they will contain, as well as the relationships between the tables. While it would be nice if this process was totally intuitive and obvious, or even better automated, this is simply not the case. A well-designed database takes time and effort to conceive, build and refine.

The benefits of a database that has been designed according to the relational model are numerous.

Some of them are:

- Data entry, updates and deletions will be efficient.
- Data retrieval, summarization and reporting will also be efficient.
- Since the database follows a well-formulated model, it behaves predictably.
- Since much of the information is stored in the database rather than in the application, the database is somewhat self-documenting.
- Changes to the database schema are easy to make.

3.2.1 Tables, Uniqueness and Keys

Tables in the relational model are used to represent "things" in the real world. Each table should represent only one thing. These things (or entities) can be real-world objects or events. For example, a real-world object might be a customer, an inventory item, or an invoice. Examples of events include patient visits, orders, and telephone calls. Tables are made up of rows and columns.

The relational model dictates that each row in a table be unique. If you allow duplicate rows in a table, then there is no way to uniquely address a given row via programming. This creates all sorts of ambiguities and problems that are best avoided. You guarantee uniqueness for a table by designating a primary key—a column that contains unique values for a table. Each table can have only one primary key, even though several columns or combination of columns may contain unique values. All columns (or combination of columns) in a table with unique values are referred to as candidate keys, from which the primary key must be drawn. All other candidate key columns are referred to as alternate keys. Keys can be simple or composite. A simple key is a key made up of one column, whereas a composite key is made up of two or more columns.

The decision as to which candidate key is the primary one rests in your hands—there is no absolute rule as to which candidate key is best. Let us illustrate with an example. Say that a company has a table of customers called tblCustomer, which looks like the table shown in Figure 3.1.

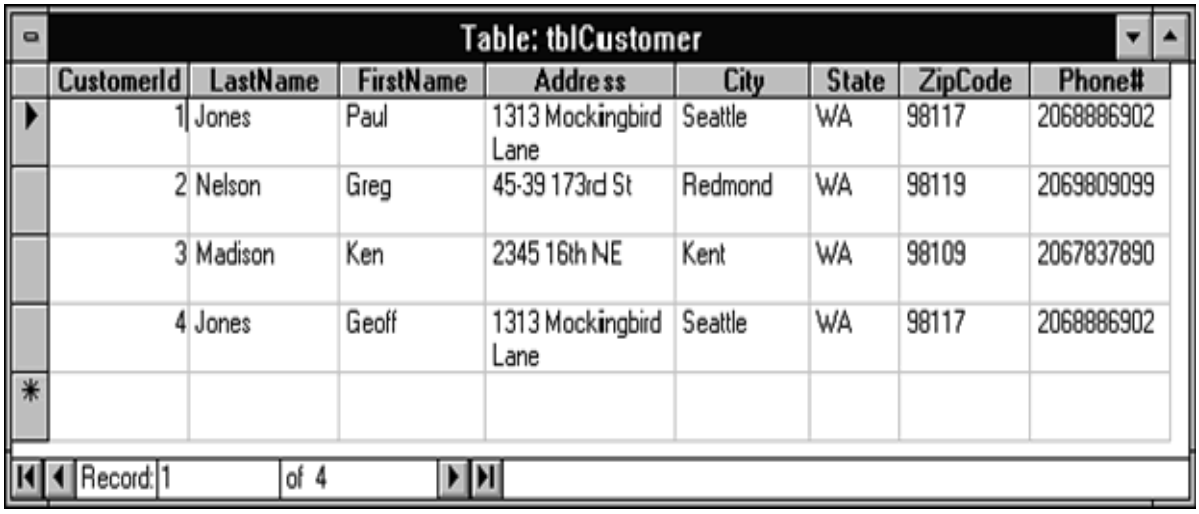


Table: tblCustomer

CustomerId	LastName	FirstName	Address	City	State	ZipCode	Phone#
1	Jones	Paul	1313 Mockingbird Lane	Seattle	WA	98117	2068886902
2	Nelson	Greg	45-39 173rd St	Redmond	WA	98119	2069809099
3	Madison	Ken	2345 16th NE	Kent	WA	98109	2067837890
4	Jones	Geoff	1313 Mockingbird Lane	Seattle	WA	98117	2068886902

Record: 1 of 4

Figure 3.1: The best choice for primary key for tblCustomer would be CustomerId.

Candidate keys for tblCustomer might include CustomerId, (LastName + FirstName), Phone#, (Address, City, and State), and (Address + ZipCode). Following Pascal's guidelines, you would rule out the last three candidates because addresses and phone numbers can change fairly frequently. The choice among CustomerId and the name composite key is less obvious and would involve tradeoffs. How likely would a customer's name change (e.g., marriages cause names to change)? Will misspelling of names be common? How likely will two customers have the same first and last names? How familiar will CustomerId be to users? There is no right answer, but most developers favor numeric primary keys because names do sometimes change and because searches and sorts of numeric columns are more efficient than of text columns in most of the databases.

Note: In many situations, it is best to use some sort of arbitrary static whole number (e.g., employee ID, order ID, etc.) as a primary key rather than a descriptive text column. This avoids the problem of misspellings and name changes. Also, do not use real numbers as primary keys since they are inexact.

Caution

In Relational Database Management we cannot allows for duplicate row, if we do it may be cause of data ambiguity.

Did You Know?

Fabian Pascal stated that the decision of selection primary key, should be based upon the principles of minimality (choose the fewest columns necessary), stability (choose a key that seldom changes), and simplicity/familiarity (choose a key that is both simple and familiar to users).

3.2.2 Foreign Keys and Domains

Although primary keys are a function of individual tables, if you created databases that consisted of only independent and unrelated tables, you would have little need for them. Primary keys become essential, however, when you start to create relationships that join together multiple tables in a database. A foreign key is a column in a table used to reference a primary key in another table.

Continuing the example presented in the last section, let us say that you choose CustomerId as the primary key for tblCustomer. Now define a second table, tblOrder, as shown in Figure 3.2.

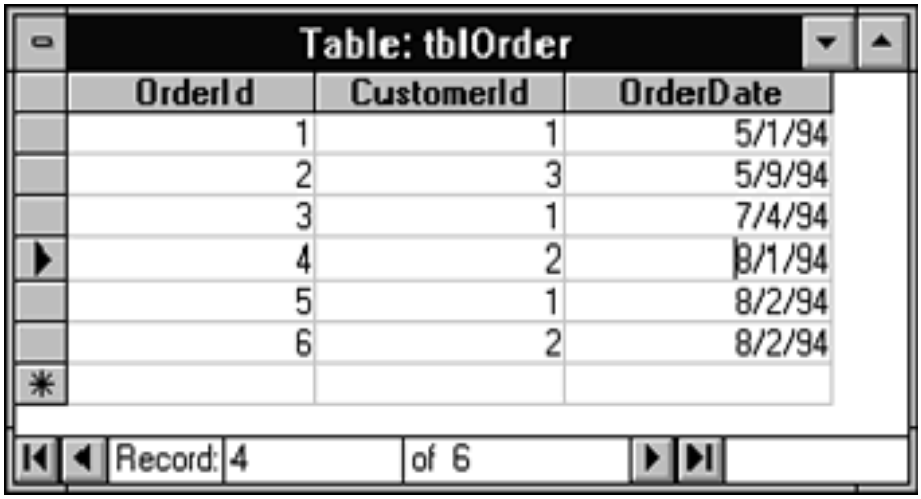


Figure 3.2: Customer Id is a foreign key in tb l Order which can be used to reference a customer stored in the tblCustomer table.

CustomerId is considered a foreign key in tblOrder since it can be used to refer to given customer (i.e., a row in the tblCustomer table). It is important that both foreign keys and the primary keys that are used to reference share a common meaning and draw their values from the same domain. Domains are simply pools of values from which columns are drawn. For example, CustomerId is of the domain of valid customer ID #'s, which in this case might be Long Integers, ranging between 1 and 50,000. Similarly, a column named Sex might be based on a one-letter domain equalling 'M' or 'F'. Domains can be thought of as user-defined column types whose definition implies certain rules that the columns must follow and certain operations that you can perform on those columns.

3.2.3 Names and Objects

Databases are useful when they tell us something about the real world. But things in the real world are dealt with in a fuzzy way that can trip us up when we translate them into a database model.

Names are a good example. When we need to refer to something, we will give it a name. People do not often think about it, but the name is not the same thing as the object. "Mark Twain" is a string of characters that we usually use to refer to the author Mark Twain, but the two things are not the same: "Mark Twain" consists of ten characters, but it is not correct to say that Mark Twain consists of ten characters.

It is also possible for the same object to have several names. "Mark Twain" and "Samuel Clemens" are both strings that can be understood to refer to the author of *The Adventures of Tom Sawyer*.

This becomes important because you cannot put real-world objects into a database, only pieces of information that identify them. You would like to choose identifying information that is unique for each object and cannot be confused, but often this is more difficult than it first appears. If your means of identifying objects is fuzzy, then however rigorous your database is the conclusions you draw will still be fuzzy.

3.2.4 Entities and Classes

Databases refer to objects in the real world, but not always in the same way. A database recording sales for a car dealership may track the individual cars that are sold, so that we know exactly which car has been sold to which customer. The database that runs the manufacturer's web site, on the other hand, probably deals with facts about the model lines as a whole, i.e. about the whole class of such cars.

This is important to be aware of because language does not always make it plain. "Joe Smith bought a Ford Mondeo yesterday" uses the same language as "a Ford Mondeo has four wheels", but the former refers to an individual car, while the latter is a general statement about every Ford Mondeo. The latter statement may even

be true if there are no Ford Mondeos in existence (e.g. after the model had been designed, but before one had been manufactured). Do not let the language confuse you, these are entirely different from the point of view of database design.

Did You Know?
The relational database model was conceived by E. F. Codd in 1969, then a researcher at IBM.

3.2.5 Terminologies
There are two sets of relational database terminology in use. The original developers of the relational theory approached it from a theoretical perspective and used terminology that came from set theory and formal logic. These terms never caught on among practitioners, who preferred to use more intuitive and practical terms, which were eventually enshrined in the SQL standard.

Relational term	**SQL equivalent**
relation, base relvar	table
derived relvar	view, query result, result set
Tuple	row
attribute	column

Most modern publications about databases use the SQL terms as described below, but you should be aware of the different terminology. The pairs of terms are not entirely synonymous, so some writers on relational theory prefer to use the strict relational terminology.

Domains
Domains are the set of allowable data values for a Column. For example, the FiveDigitZipCode Column on the customer entity can be in the integer domain. As such, the database would not allow you to place values like 123.45 (floating point) or ABC (character) into that Column.
Some authors draw a distinction between a domain and a type in the fact that a type is a fundamental concept built into the DBMS (e.g. string, integer, floating point) while a domain can have additional business rules about what values are acceptable. For example, if you have a database storing scores in ten-pin bowling, the score for a game will be of integer type, but the rules of the game (it is impossible to score more than 300) mean that the domain of the score would be integers between 0 and 300. The additional constraints on the domain make it harder for bad data to be inserted into the database.
***Columns*-**Columns are the attributes that describe an entity in the database model. For example, the *customer* entity may have attributes for First Name, Last Name, Address, City, State, and FiveDigitZipCode.

Row, Record or Tuple
A Row is a complete set of Columns that describe the entity that you are trying to model.

Tables
Tables are collections of Rows that act as logical groupings of entities.

Databases
A collection of related Tables and any supporting objects (e.g. stored procedures) is often referred to as a Database (or schema). Multiple Databases are usually logically separate from one another.

The term ‘database’ is sometimes used loosely to refer to the software that manages the database. To avoid ambiguity it is standard in more formal contexts to refer to the software as a Database Management System or DBMS—or more specifically a Relational Database Management System or RDBMS.

3.3 Set Theory Concepts

A relational database is a collection of data organized in two-dimensional tables consisting of named columns and rows. Each table represents the mathematical concept of a relation as defined in set theory. In set theory, columns are known as attributes and rows are known as tuples. The operations that may be performed on tables are similarly based on manipulation of relations to produce new relations, usually referred to as queries or views.

Relational databases differ from non-relational databases in that the database user is not aware of system dependencies that may be stored within the data. No knowledge of the underlying database is required; data can be queried and updated using standard languages (these languages together make up SQL) that produce a consistent result. SQL Server databases are relational.

An RDBMS is responsible for:

- Storing and making data available in tables.
- Maintaining the relationships between tables in the database.
- Ensuring the integrity of data, by making sure that rules governing the data values and defining the relationships between tables are not violated.
- Recovering all data to a point of known consistency in case of a system failure.

The theory of relational databases is built upon the mathematical theory of *sets*. In mathematics, a set is one of the most general ways of considering more than one object at once, and a substantial amount of theory has been built up to describe this. You would not need to know too much set theory to understand relational databases, but it helps to have the basic concepts straight.

A set is simply a collection of zero or more objects, all of which are different, without any further structure.

The following are all sets:

$\{1, 2, 3\}$ $\{a, b\}$ $\{-371, r, 4.5\}$

Note that the definition says zero or more: a set with zero members is still a set, even though it is empty. The set with zero elements is written as a pair of empty braces $\{\}$, and is often represented by the symbol $\emptyset$.

If two sets have the exact same elements, then they are the same set. There is nothing special about one set that can distinguish it from others, apart from the elements it contains. The order of the elements is not important, so the sets are the same no matter what order we choose to write the elements in:

$\{1, 2, 3\} = \{3, 2, 1\}$

Anything can be put into set, not just mathematical concepts such as numbers. You could have the set of all weekdays, and deal with it in set theory the same as any other set:

S= {Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday}

Even sets can be members of sets:

$S = \{1, \{red, orange, yellow\}, 42\}$

Sets can be infinite, for example the set of all positive whole numbers.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Create a data table and operate the different operations on it.

……..……

……..……

……...

3.4 Extension and Intention

A relational in a relational database has two components, an extension and an intension.

3.4.1 Extension

The extension of a given relation is the set of tuples appearing in that relation at any given instance. The extension thus varies with time. It changes as tuples are created, destroyed, and updated.

Relation: Employee at time= t1

EmpNo	EmpName	Age	Dept
1001	Janmendra	23	SD
1002	Vicky	24	HR
1003	Jonathan	28	Fin
1004	Harry	20	Fin

Relation: Employee at time= t2 after adding more records

EmpNo	EmpName	Age	Dept
1001	Janmendra	23	SD
1002	Vicky	24	HR
1003	Jonathan	28	Fin
1004	Harry	20	Fin
1005	Smith	22	HR
1006	Mary	19	HR
1007	Sarah	23	SD

Relation: Employee at time= t2 after adding more records

EmpNo	EmpName	Age	Dept
1001	Janmendra	23	SD
1002	Vicky	24	HR

3.4.2 Intension

The intension of a given relation is independent of time. It is the permanent part of the relation. It corresponds to what is specified in the relational schema. The intension thus defines all permissible extensions. The intension is a combination of two things: a structure and a set of integrity constraints.

The naming structure consists of the relation name plus the names of the attributes (each with its associated domain name).

The integrity constraints can be subdivided into key constraints, referential constraints, and other constraints.

For example,
Employee (EmpNo Number (4) Not NULL, EName Char(20), Age Number(2), Dept Char(4))
This is the intension of Employee relation.

3.5 Relationships

You define foreign keys in a database to model relationships in the real world. Relationships between real-world entities can be quite complex, involving numerous entities each having multiple relationships with each other. For example, a family has multiple relationships between multiple people—all at the same time. These tables can be related in one of three different ways: one-to-one, one-to-many or many-to-many.

3.5.1 One-to-Many Relationships

In Figure 3.3 the procedure for deriving the degree of a relationship type and putting it on the entity relationship diagram is shown. The example concerns part of a sales ledger system. Customers may have received zero or more invoices from us. The relationship type is thus called 'received' and is from CUSTOMER to INVOICE. The arrow shows the direction. The minimum number of invoices the customer has received is zero and thus the 'received' relationship type is optional. This is shown by the zero on the line. The maximum number of invoices the customer may have received is 'many'. This is shown by the crow's foot. This is summarized in Figure 3.3(a). To complete the definition of the relationship type the next step is to name the inverse relationship type. Clearly if a customer received an invoice, the invoice *was sent to* the customer and this is an appropriate name for this inverse relationship type. Now consider the degree of the inverse relationship type. The minimum number of customers you would send an invoice to is one; you would not send it to no-one. The optionality is thus one. The inverse relationship type is mandatory. The maximum number of customers you would send an invoice to is also one so the cardinality is also one. This is summarized in Figure 3.3(b). Figure 3.3(b) shows the completed relationship.

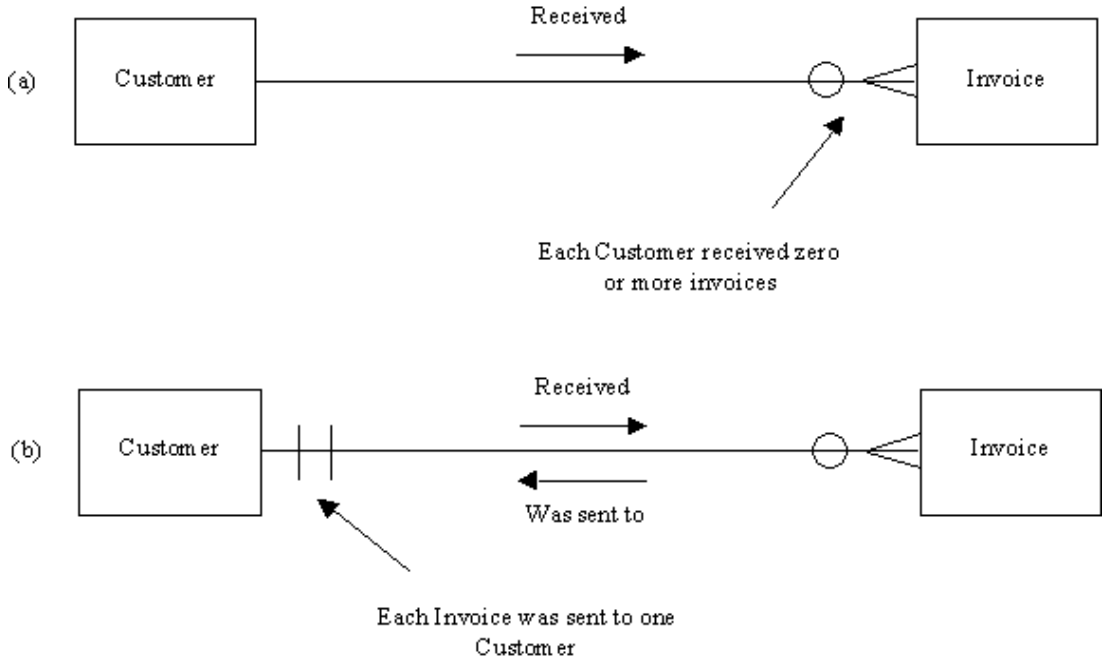


Figure 3.3: Deriving a 1: N (one: many) relationship.

A word of warning is useful here. In order to obtain the correct degree for a relationship type (one-one or one-many or many-many) you must ask two questions. Both questions must begin with the word 'one'. In the present case (Figure 3.3), the two questions you would ask when drawing in the relationship line and deciding on its degree would be:

Question 1: One customer received how many invoices?

Answer: Zero or more.
Question 2: One invoice was sent to how many customers?
Answer: One.
This warning is based on observations of many student database designers getting the degree of relationship types wrong. The usual cause of error is only asking one question and not starting with the word 'one'. For example a student might say (incorrectly): 'Many customers receive many invoices' (which is true) and wrongly conclude that the relationship type is many-many. The second most common source of error is either to fail to name the relationship type and say something like 'Customer to Invoice is one-to-many' (which is meaningless) or give the relationship type an inappropriate name.

3.5.2 Many-to-Many Relationship

Figure 3.4 gives an example of a many-many relationship type being derived.

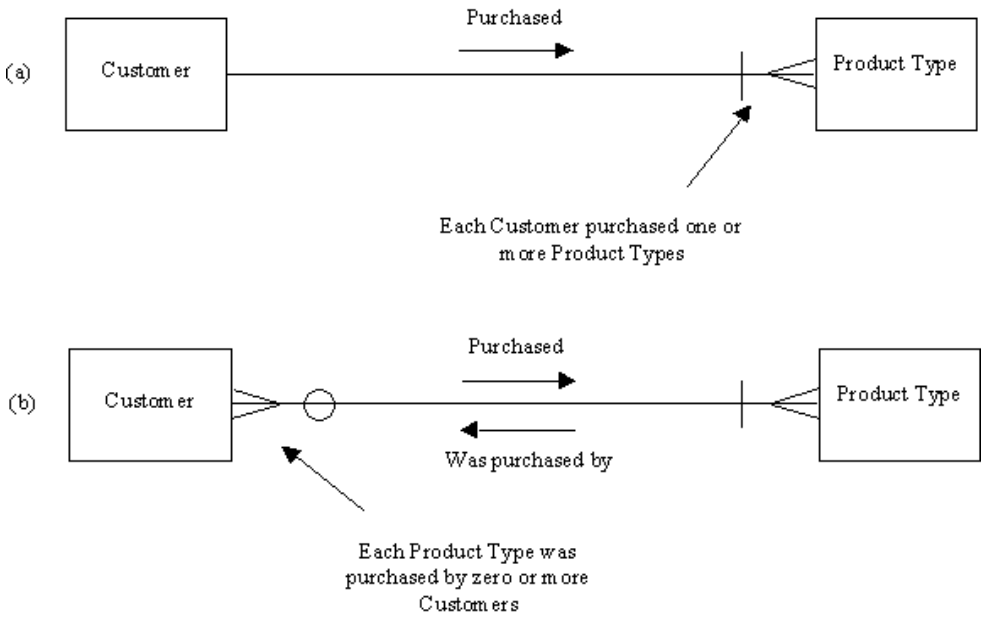


Figure 3.4: Deriving an M: N (many-many) relationship.

The two questions you have to ask to correctly derive the degree of this relationship (and the answers) are:
Question 1: One customer purchased how many product types?
Answer: One or more.
Question 2: One product type was purchased by how many customers?
Answer: Zero or more.
Note that the entity type has been called PRODUCT TYPE rather than PRODUCT which might mean an individual piece that the customer has bought. In that case the cardinality of 'was_purchased_by' would be one not many because an individual piece can of course only go to one customer. This point is another common source of error: the tendency to call one item (e.g. an individual 4" paintbrush) a product and the whole product type (or 'line') (e.g. the 4" paintbrush product type) a product. You should make the meaning clear from the name you give the entity type.
We have assumed here that every customer on the database has purchased at least one product; hence the mandatory optionality of 'purchased'. If this were not true in the situation under study then a zero would appear instead. The zero optionality of 'was_purchased_by' is due to our assumption that a product type might as yet have had no purchases at all.

In practice it is wise to replace many-many relationship types such as this with a set (often two) of one-many relationship types and a set (often one) of new, previously hidden entity types.

3.5.3 One-One Relationship

Figure 3.5 gives an example of a one-one relationship type being derived. It concerns a person and his or her birth certificate. We assume that everyone has one and that a certificate registers the birth of one person only.

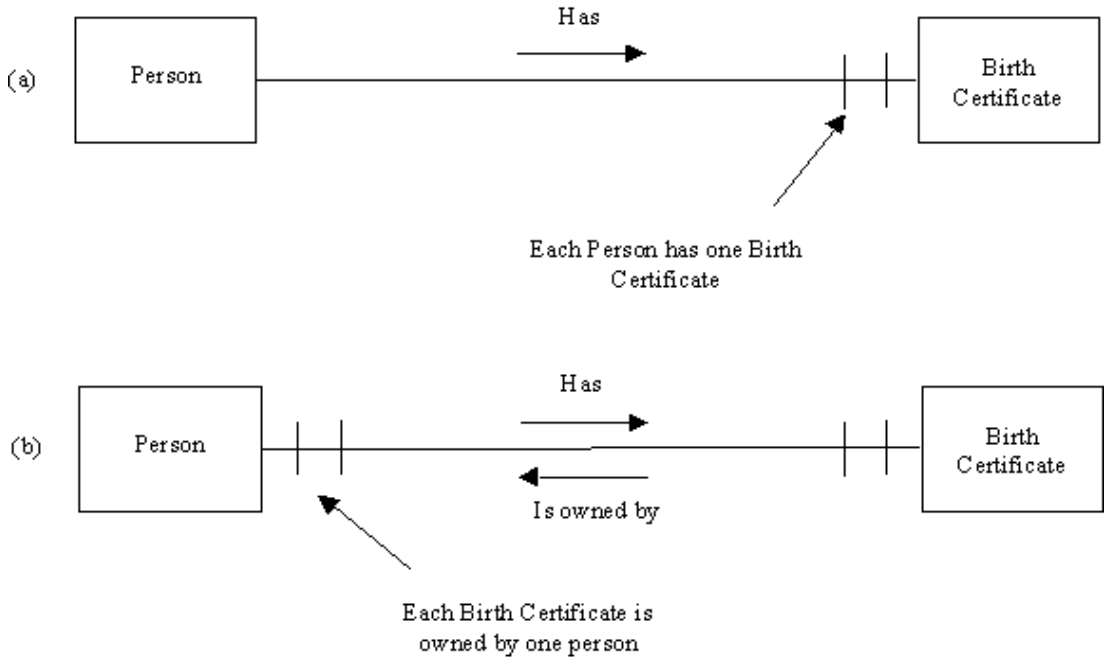


Figure 3.5 Deriving a 1:1 (one: one) relationship.

Question 1: How many birth certificates have a person?
Answer: One.
Question 2: How many persons is a birth certificate owned by?
Answer: One.
Where there is a one-one relationship type we have the option of merging the two entity types. The birth certificate attributes may be considered as attributes of the person and placed in the person entity type. The birth certificate entity type would then be removed. There are two reasons for not doing this. Firstly, the majority of processing involving PERSON records might not involve any or many of the BIRTH_CERTIFICATE attributes. The BIRTH CERTIFICATE attributes might only be subject to very specific processes which are rarely executed. The second reason for not merging might be that the BIRTH CERTIFICATE entity type has relationship types to other entity types that the PERSON entity type does not have. The two entity types have different relationship types to other entity types.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Create a database structure using one-one relations.

……..……

……..……

……...

3.6 Integrity Rules

The relational model defines several integrity rules that, while not part of the definition of the Normal Forms are nonetheless a necessary part of any relational database. There are two types of integrity rules: entity and referential integrity.

3.6.1 Integrity Rule 1: Entity integrity

It says that no component of a primary key may be null. All entities must be distinguishable. That is, they must have a unique identification of some kind. Primary keys perform unique identification function in a relational database. An identifier that was wholly null would be a contradiction in terms. It would be like there was some entity that did not have any unique identification. That is, it was not distinguishable from other entities.

3.6.2 Integrity Rule 2: Referential integrity

The referential integrity constraint is specified between two relations and is used to maintain the consistency among tuples of the two relations. Suppose we wish to ensure that value that appears in one relation for a given set of attributes also appears for a certain set of attributes in another. This is referential integrity. The referential integrity constraint states that, a tuple in one relation that refers to another relation must refer to the existing tuple in that relation. This means that the referential integrity is a constraint specified on more than one relation. This ensures that the consistency is maintained across the relations.

Caution

If two entities are not distinguishable from each other, then by definition there are not two entities but only one. It may be cause of data redundancy.

Self Assessment Questions

1. RDBMS stands for:

(a) Repeat Database Management System (b) Real Database Management System

(c) Relational Database Management System (d) None of these

2. The relational model is based on branches of mathematics called set theory and predicate logic.

(a) True (b) False

3. The relational model can be applied to both databases and database management systems.

(a) True (b) False

3.7 Domains and Relations

The relational model can be regarded as having three principal parts, having to do with data structure, data integrity, and data manipulation, respectively. Each part has its own special terminology. The most important structural terms are (which as you can see is based on the sample suppliers relation from the suppliers and parts database of expanded to show the applicable data types or domains), The terms in question are relation itself (of course), tuple, cardinality, attribute, degree, domain, and primary key.

We explain the other terms informally here, and then go on to give more formal definitions in subsequent sections. Briefly, if we think of a relation as a table, then a tuple corresponds to a row of such a table and an attribute to a column; the number of tuples is called the cardinality and the number of attributes is called the degree; and a domain is a pool of values, from which the values of specific attributes of specific relations are

taken. The domain labelled S# in for example is the set of all possible supplier numbers, and every S# value appearing in the supplier's relation is some value from that set presents a summary of the foregoing. Please understand, however, that the "equivalences" are all only approximate (the formal relational terms have precise definitions, while the informal "equivalents" have only rough and ready definitions).

A domain is nothing more nor less than a data type (type for short)-possibly a simple system-defined type like INTEGER or CHAR, more generally a user-defined type like S# or P# or WEIGHT or QTY in the suppliers and parts database. Indeed, we can use the terms type and domain interchangeably. (Though we prefer the term type; when we use the term domain, we do so mainly for historical reasons).
Among other things, it is a set of values-all possible values of the type in question. The type INTEGER for example, is the set of all possible integers; the type S# is the set of all possible supplier numbers; and so on. Also, along with the notion of a given type is the associated notion of the valid operator's that can legally be applied to values of that type; i.e., values of that type can be operated upon solely by means of the operators defined for that type. For example, type INTEGER (which we assume for simplicity is system-defined).

- The system provides operators "=", "<", and so on, for comparing: integers;
- It also provides operators "+", "*", and so on, performing arithmetic on integers;
- It does not provide operators '||' (concatenate), SUBSTR (substring), and so on, for performing string operations on integers. In other words, string operations on integers are not supported.

3.7.1 Kinds of Relations

Several kinds of relationships can be defined in a database. Consider the possible relationships between employees and departments.

One-to-many and many-to-one relationships

An employee can work in only one department; this relationship is single-valued for employees. On the other hand, one department can have many employees; this relationship is multi-valued for departments. The relationship between employees (single-valued) and departments (multi-valued) is a one-to-many relationship.
To define tables for each one-to-many and each many-to-one relationship:

1. Group all the relationships for which the "many" side of the relationship is the same entity.
2. Define a single table for all the relationships in the group.

Table 3.6: Many-to-One Relationships

Entity	Relationship	Entity
Employees	are assigned to	Departments
Employees	work at	Jobs
Departments	report to	(administrative) departments

In the third relationship, "departments" is on the "many" side, so a department table, DEPARTMENT, is defined.

Table 3.7: The employee

EMPNO	WORKDEPT	JOB
000010	A00	President
000020	B01	Manager
000120	A00	Clerk

EMPNO	WORKDEPT	JOB
000130	C01	Analyst
000030	C01	Manager
000140	C01	Analyst
000170	D11	Designer

Table 3.8: The Department

DEPTNO	ADMRDEPT
C01	A00
D01	A00
D11	D01

Many-to-many relationships

A relationship that is multi-valued in both directions is a many-to-many relationship. An employee can work on more than one project, and a project can have more than one employee. For example the questions "What does Dolores Quintana work on?", and "Who works on project IF1000?" both yield multiple answers. A many-to-many relationship can be expressed in a table with a column for each entity ("employees" and "projects"), as shown in the following example.

Table 3.9 shows how a many-to-many relationship (an employee can work on many projects, and a project can have 7 many employees working on it) is represented.

Table 3.9: The employee activity (EMP_ACT)

EMPNO	PROJNO
000030	IF1000
000030	IF2000
000130	IF1000
000140	IF2000
000250	AD3112

One-to-one relationships

One-to-one relationships are single-valued in both directions. A manager manages one department; a department has only one manager. The questions, "Who is the manager of Department C01?", and "What department does Sally Kwan manage?" both have single answers. The relationship can be assigned to either the DEPARTMENT table or the EMPLOYEE table. Because all departments have managers, but not all employees are managers, it is most logical to add the manager to the DEPARTMENT table, as shown in the following example.

The Table 3.10 shows the representation of a one-to-one relationship.

Table 3.10: The Department

DEPTNO	MGRNO
A00	000010
B01	000020
D11	000060

You can have more than one table describing the attributes of the same set of entities. For example, the EMPLOYEE table shows the number of the department to which an employee is assigned, and the DEPARTMENT table shows which manager is assigned to each department number. To retrieve both sets of attributes simultaneously, you can join the two tables on the matching columns, as shown in the following example. The values in WORKDEPT and DEPTNO represent the same entity, and represent a join path between the DEPARTMENT and EMPLOYEE tables.

Table 3.11: The Department

DEPTNO	DEPTNAME	MGRNO	ADMRDEPT
D21	Administration Support	000070	D01

Table 3.12: The Employee

EMPNO	FIRSTNAME	LASTNAME	WORKDEPT	JOB
000250	Ram	Mohan	D21	Clerk

When you retrieve information about an entity from more than one table, ensure that equal values represent the same entity. The connecting columns can have different names (like WORKDEPT and DEPTNO in the previous example), or they can have the same name (like the columns called DEPTNO in the department and project tables).

Did You Know?

As computers grew in speed and capability, a number of general-purpose database systems emerged; by the mid-1960s there were a number of such systems in commercial use.

3.8 Concept of Keys

A Key is a single attribute or combination of two or more attributes of an entity set that is used to identify one or more instances of the set.

Primary Key: A primary key is a field that uniquely identifies each record in a table. As it uniquely identifies each entity, it cannot contain null value and duplicate value.eg:-Consider the customer table, which has field customer_number, customer_socialsecurity_number, and customer_address.here customer_number of each entity in customer table is distinct so customer-number can be a primary key of customer-table.

Super Key: If we add additional attributes to a primary key, the resulting combination would still uniquely identify an instance of the entity set. Such augmented keys are called super key. A primary key is therefore a minimum super key.

Candidate Key: A nominee's for primary key field are known as candidate key.eg:-From above example of customer table, customer_socialsecurity_number is candidate key as it has all characteristics of primary key.

Alternate Key: A candidate key that is not the primary key is called an Alternate key. eg:- In above example, customer_socialsecurity_number is a candidate key but not a primary key so it can be considered as alternate key.

Composite Key: Creating more than one primary key is jointly known as composite key.eg:-In if customer_number and customer_social security_number are made primary key than they will be jointly known as composite key table, so it is known as foreign key in the current table. A foreign key is one or more columns which value must exist in the primary key of another table.eg:-Consider two tables' employees

Foreign Key: Foreign key is a primary key of master table, which is reference in the current (contains employee's description) and emp_edu (contains details of employee's education), so emp_id which is primary key in employee table will be referred as foreign key in emp_edu table.

Caution

If your data files contain integrity constraints, do not use your operating environment commands to copy, move, or delete your data files. This can cause of data loss.

3.8.1 Candidate Keys

A candidate key is a combination of attributes that can be uniquely used to identify a database record without any extraneous data. Each table may have one or more candidate keys. One of these candidate keys is selected as the table primary key.

In the relational model of databases, a candidate the relation does not have two distinct tuples (i.e. rows or records in common database language) with the same values for these attributes (which means that the set of attributes is a super key).

1. There is no proper subset of these attributes for which (1) holds (which means that the set is minimal). Key of a relation is a minimal super key for that relation; that is, a set of attributes such that.
2. The constituent attributes are called prime attributes. Conversely, an attribute that does not occur in ANY candidate key is called a non-prime attribute.

Since a relation contains no duplicate tuples, the set of all its attributes is a super key if NULL values are not used. It follows that every relation will have at least one candidate key.

The candidate keys of a relation tell us all the possible ways we can identify its tuples. As such they are an important concept for the design database schema.

For practical reasons RDBMSs usually require that for each relation one of its candidate keys is declared as the primary key, which means that it is considered as the preferred way to identify individual tuples. Foreign keys, for example, are usually required to reference such a primary key and not any of the other candidate keys.

Example:

The definition of candidate keys can be illustrated with the following (abstract) example. Consider a relation variable (relvar) R with attributes (A, B, C, D) that has only the following two legal values r1 and r2.

Table 3.13: r1

A	B	C	D
a1	b1	c1	d1
a1	b2	c2	d1
a2	b1	c2	d1

Table 3.14: r2

A	**B**	**C**	**D**
a1	b1	c1	d1
a1	b2	c2	d1

a1	b1	c2	d2

Here r2 differs from r1 only in the A and D values of the last tuple.
For r1 the following sets have the uniqueness property, i.e., there are no two distinct tuples in the instance with the same values for the attributes in the set.
{A,B}, {A,C}, {B,C}, {A,B,C}, {A,B,D}, {A,C,D}, {B,C,D}, {A,B,C,D}
For r2 the uniqueness property holds for the following sets.
{B,C}, {B,D}, {C,D}, {A,B,C}, {A,B,D}, {A,C,D}, {B,C,D}, {A,B,C,D}
Since super keys of a relvar are those sets of attributes that have the uniqueness property for all legal values of that relvar and because we assume that r1 and r2 are all the legal values that R can take, we can determine the set of super keys of R by taking the intersection of the two lists.
{B,C}, {A,B,C}, {A,B,D}, {A,C,D}, {B,C,D}, {A,B,C,D}
Finally we need to select those a set for which there is no proper subset in the list, which are in this case.
{B,C}, {A,B,D}, {A,C,D}
These are indeed the candidate keys of relvar R.
We have to consider all the relations that might be assigned to a relvar to determine whether a certain set of attributes is a candidate key. For example, if we had considered only r1 then we would have concluded that {A, B} is a candidate key, which is incorrect. However, we might be able to conclude from such a relation that a certain set is not a candidate key, because that set does not have the uniqueness property (example {A, D} for r1). Note that the existence of a proper subset of a set that has the uniqueness property cannot in general be used as evidence that the superset is not a candidate key. In particular, note that in the case of an empty relation, every subset of the heading has the uniqueness property, including the empty set.

Did You Know?
Telephone books and dictionaries cannot use names, words, or Dewey Decimal system numbers as candidate keys because they do not uniquely identify telephone numbers or words.

3.8.2 Candidate Keys and Nulls

Three absolute demands on the candidate key, if it is to be regarded as a possible primary key. There are three fundamental demands on the candidate keys that we should never deviate from, if it is to become the subject for a primary key.

- The candidate key must be unique within its domain.
- The candidate key cannot hold NULLs (NULL is not zero. Zero is a number. NULL is ‘unknown value’).
- The candidate key should never change. It must hold the same value for a given occurrence of an entity for the lifetime of that entity.
- We say that the main purpose of the candidate key is to help us to identify one single row in a table, regardless of whether there exist billions of rows. This sets high demands on the flexibility in terms of delivering uniqueness.

3.8.3 Primary and Alternate Keys

3.8.3.1 Primary Key

The first type of key we will discuss is the primary key. Every database table should have one or more columns designated as the primary key. The value this key holds should be unique for each record in the database. For example, assume we have a table called Employees that contains personnel information for

every employee in our firm. We would need to select an appropriate primary key that would uniquely identify each employee. Your first thought might be to use the employee's name.

Once you decide upon a primary key and set it up in the database, the database management system will enforce the uniqueness of the key. If you try to insert a record into a table with a primary key that duplicates an existing record, the insert will fail.

Most databases are also capable of generating their own primary keys. Microsoft Access, for example, may be configured to use the auto number data type to assign a unique ID to each record in the table. While effective, this is a bad design practice because it leaves you with a meaningless value in each record in the table.

3.8.3.2 Alternate Key

The alternate keys of any table are simply those candidate keys which are not currently selected as the primary key. Exactly one of those candidate keys is chosen as the primary key and the remainder, if any, are then called alternate keys. "An alternate key is a function of all candidate keys minus the primary key".

Caution

If you want to redefine the primary key, any relationships to the existing primary key must be deleted before the new primary key can be created or it will be automatically deleted as part of this process.

Exercise: Check Your Progress 3

Note: i) Use the space below for your answer.

Ex1: Create a table and show how we assign an alternate key

……..……

……..……

……...

3.8.4 Foreign Key

A foreign key is a field in a relational table that matches the primary key column of another table. The foreign key can be used to cross-reference tables.

3.8.4.1 Foreign Key Rules (Referential)

The purpose of a referential constraint is to guarantee that table relationships are maintained and that data entry rules are followed. This means that as long as a referential constraint is in effect, the database manager guarantees that for each row in a child table that has a non-null value in its foreign key columns, a row exists in a corresponding parent table that has a matching value in its parent key. When an SQL operation attempts to change data in such a way that referential integrity will be compromised, a foreign key (or referential) constraint could be violated. The database manager handles these types of situations by enforcing a set of rules that are associated with each referential constraint. This set of rules consists of:

- An insert rule
- An update rule
- A delete rule

When an SQL operation attempts to change data in such a way that referential integrity will be compromised, a referential constraint could be violated. For example,

- An insert operation could attempt to add a row of data to a child table that has a value in its foreign key columns that does not match a value in the corresponding parent table's parent key.
- An update operation could attempt to change the value in a child table's foreign key columns to a value that has no matching value in the corresponding parent table's parent key.

- An update operation could attempt to change the value in a parent table's parent key to a value that does not have a matching value in a child table's foreign key columns.
- A delete operation could attempt to remove a record from a parent table that has a matching value in a child table's foreign key columns.
- The database manager handles these types of situations by enforcing a set of rules that are associated with each referential constraint.

Insert Rule

The insert rule of a referential constraint is that a non-null insert value of the foreign key must match some value of the parent key of the parent table. The value of a composite foreign key is null if any component of the value is null. This rule is implicit when a foreign key is specified.

Update Rule

The update rule of a referential constraint is specified when the referential constraint is defined. The choices are NO ACTION and RESTRICT. The update rule applies when a row of the parent or a row of the dependent table is updated. In the case of a parent row, when a value in a column of the parent key is updated, the following rules apply:

- If any row in the dependent table matches the original value of the key, the update is rejected when the update rule is RESTRICT.
- If any row in the dependent table does not have a corresponding parent key when the update statement is completed (excluding AFTER triggers), the update is rejected when the update rule is NO ACTION.

The value of a composite foreign key is null if any component of the value is null.

Delete Rule

The delete rule of a referential constraint is specified when the referential constraint is defined. The choices are NO ACTION, RESTRICT, CASCADE, or SET NULL. SET NULL can be specified only if some column of the foreign key allows null values. If the identified table or the base table of the identified view is a parent, the rows selected for delete must not have any dependents in a relationship with a delete rule of RESTRICT, and the DELETE must not cascade to descendent rows that have dependents in a relationship with a delete rule of RESTRICT.

If the delete operation is not prevented by a RESTRICT delete rule, the selected rows are deleted. Any rows that are dependents of the selected rows are also affected:

- The null able columns of the foreign keys of any rows that are their dependents in a relationship with a delete rule of SET NULL are set to the null value.
- Any rows that are their dependents in a relationship with a delete rule of CASCADE are also deleted, and the above rules apply, in turn, to those rows.

The delete rule of NO ACTION is checked to enforce that any non-null foreign key refers to an existing parent row after the other referential constraints have been enforced.

The delete rule of a referential constraint applies only when a row of the parent table is deleted. More precisely, the rule applies only when a row of the parent table is the object of a delete or propagated delete operation and that row has dependents in the dependent table of the referential constraint. Consider an example where P is the parent table, D is the dependent table, and p is a parent row that is the object of a delete or propagated delete operation. The delete rule works as follows.

- With RESTRICT or NO ACTION, an error occurs and no rows are deleted.
- With CASCADE, the delete operation is propagated to the dependents of p in table D.
- With SET NULL, each null able column of the foreign key of each dependent of p in table D is set to null.

Any table that can be involved in a delete operation on P is said to be delete-connected to P. Thus, a table is delete-connected to table P if it is a dependent of P, or a dependent of a table to which delete operations from P cascade.

The following restrictions apply to delete-connected relationships.

- When a table is delete-connected to itself in a referential cycle of more than one table, the cycle must not contain a delete rule of either RESTRICT or SET NULL.
- A table must not both be a dependent table in a CASCADE relationship (self-referencing or referencing another table) and have a self-referencing relationship with a delete rule of either RESTRICT or SET NULL.
- When a table is delete-connected to another table through multiple relationships where such relationships have overlapping foreign keys, these relationships must have the same delete rule and none of these can be SET NULL.
- When a table is delete-connected to another table through multiple relationships where one of the relationships is specified with delete rule SET NULL, the foreign key definition of this relationship must not contain any distribution key.
- When two tables are delete-connected to the same table through CASCADE relationships, the two tables must not be delete-connected to each other where the delete connected paths end with delete rule RESTRICT or SET NULL.

Did You Know?

A foreign key is started to call as a self-referencing or recursive foreign key in SQL: 2003.

Self Assessment Questions

4. ..... are used to represent "things" in the real world.

(a) Keys (b) Tables

(c) Uniqueness (d) All of these

5. The relational model dictates that each row in a table be unique.

(a) True (b) False

6. A ……..is made up of two or more columns.

(a) Candidate key (b) Primary key

(c) Composite key (d) None of these.

3.8.5 Foreign Key and Nulls

A foreign key (FK) is a column or combination of columns that is used to establish and enforce a link between the data in two tables. You can create a foreign key by defining a FOREIGN KEY constraint when you create or modify a table.

In a foreign key reference, a link is created between two tables when the column or columns that hold the primary key value for one table are referenced by the column or columns in another table. This column becomes a foreign key in the second table.

For example, the Sale.SalesOrderHeader table in the AdventureWorks2008R2 database has a link to the Sales.SalesPerson table because there is a logical relationship between sales orders and salespeople. The SalesPersonID column in the SalesOrderHeader table matches the primary key column of the SalesPerson table. The SalesPersonID column in the SalesOrderHeader table is the foreign key to the SalesPerson table. Figure 3.15 shows primary key and null combinations.

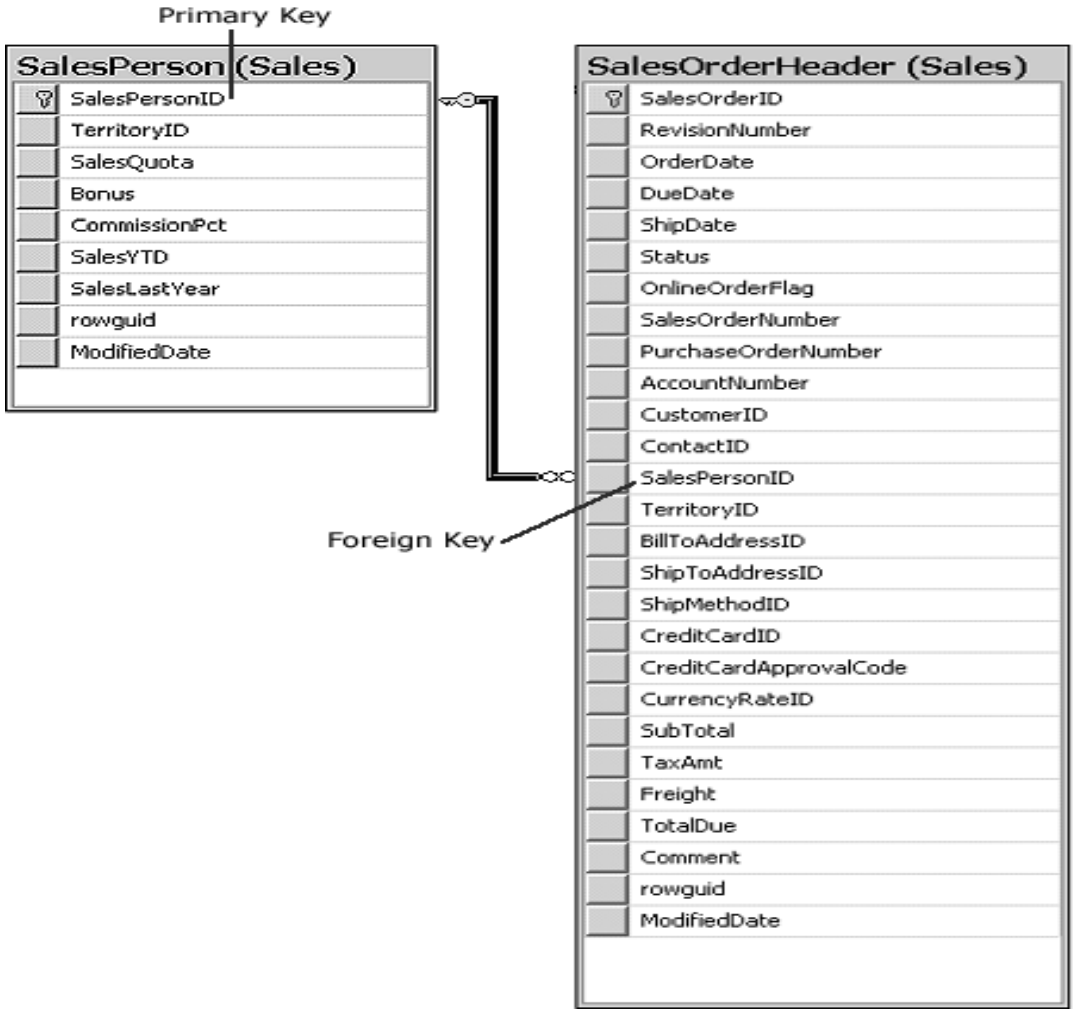


Figure 3.15: Foreign key and nulls.

A FOREIGN KEY constraint does not have to be linked only to a PRIMARY KEY constraint in another table; it can also be defined to reference the columns of a UNIQUE constraint in another table. A FOREIGN KEY constraint can contain null values; however, if any column of a composite FOREIGN KEY constraint contains null values, verification of all values that make up the FOREIGN KEY constraint is skipped. To make sure that all values of a composite FOREIGN KEY constraint are verified, specify NOT NULL on all the participating columns.

Exercise: Check Your Progress 5

Case Study-RDBMS Use on the EUVE Mission

The Extreme Ultraviolet Explorer satellite (EUVE) mission has employed a Relational Database Management System (RDBMS) in several ways. Shortly after the launch of EUVE in 1992, an indexing scheme for the EUVE telemetry was implemented to facilitate access to the calibration, in-flight, and supplemental telemetry data. The RDBMS has been used for tracking EUVE observations and other administrative information. Over time, it has evolved into the central data repository of the distributed EUVE data reduction system. A number of experimental systems also used the RDBMS. In 1992 EUVE began participating in the Astrophysics Data System, using the RDBMS as the repository for the EUVE bright source list and additional, ancillary information. With the advent of the World Wide Web, EUVE used its RDBMS to publish dynamic information for use within the project and across the Internet. More recently, EUVE has created prototype client programs to facilitate use of relational databases for scientific applications.

Telemetry Archive

A terabyte optical disk jukebox serves as the primary storage facility for the Telemetry Archive. The jukebox also supports other hardware devices for various operational reasons beyond the scope of this case. The jukebox supports the Network File System (NFS) protocol, which greatly simplifies access and manipulation of the archive. The Telemetry Archive database itself is maintained in the Sybase RDBMS. The Telemetry Archive comprises two components: a general purpose archive database and a telemetry-specific archive database.

The general purpose archive is used for a wide range of purposes. Its primary purpose is to provide a mechanism for the archiving of telemetry. However, a number of supplementary files related to the telemetry and to the general operations of *EUVE* are also archived. This archive behaves like a simple file system that supports several different media. It maintains indexes, based on file names, tracking the location, size, and type of the file, including its media and media-specific information (such as file position and ID on the tape carousel device). This database resolves queries based on a unique ID, returning a specific file. The general-purpose archive also synchronizes processing. Handling telemetry is essentially an event-oriented processing problem. At any time the system must be prepared to receive telemetry. After the initial reception, the telemetry is processed in a variety of ways depending on its type. The features of RDBMS used are transaction locking, consistency checks, reliability, and robustness (particularly in light of machine crashes). Interestingly, the archive does not use the RDBMS relational features, and the system logic is coded in the application software. Essentially, the RDBMS is used as a reliable data store. Routinely, autonomous application software interacts with this database with little direct user interaction. The application software anticipates most routine exceptions and processes them accordingly. Highly unusual exceptions require the manual intervention of the system administrator. Since the system uses unique hardware, it is subject to single point failures. Additionally, the unique hardware and the 24 hour per day operation make testing of the database system difficult.

Proposal Database

The Proposal Database of *EUVE* pointed observations is designed to provide not only information about the target, the observation, and the principal investigator for the data, but also information about the observation schedule, the location of the acquired data, and various types of historical data such as the software version used for processing. Over time, the design expanded to include the state of the data reduction processing. In general, information in the Proposal Database handles higher level information than the information found in the Archive Database. The Proposal database has two components: an observation database and a proposal database.

The proposal component of the Proposal Database tracks high-level information about investigators, their proposals, the related observation requests, and completed observations. This database has the greatest amount of operator data entry. Several mechanisms have been used to check the consistency of the data in this database, since errors can directly effect the productivity of the *EUVE* mission. Moreover, the information changes frequently as adjustments are made to observing proposals. Runtime exceptions occur frequently and are handled by operators using a WWW interface. This interface is closely related to the schema structure of the database. Operators must be familiar with the schema structure in order to appropriately correct data in the database. However, this cost is small compared to the cost of identifying and anticipating the multitude of exception conditions that may occur in this database.

Discussion

The *EUVE* mission has used RDBMS technology in mission-critical processing. The features used are transactions and synchronization (particularly suited to distributed systems). In some cases the relational features of the RDBMS were also used. However, in many cases complex logical expressions were handled in

the application software instead of the RDBMS, and the RDBMS was used as a reliable data store. Overall design, development, and testing of the databases were not difficult.

Providing operators with the capability to access and manipulate data has been an ongoing problem in working with databases on the *EUVE* mission. We experimented with several commercial products, as well as the Astronomical Data System, in an attempt to provide an operator front-end to the RDBMS. None of these systems provided an appropriate solution. Typically such products are designed to provide complete solutions that require extensive development of interface specifications intimately connected to the schema of the database. This type of extensive development was never justified, nor did it appear maintainable since the database schemas have continuously evolved. The evolution of the database schema is a natural result of the evolution of the scientific goals of the mission. Therefore, instead of providing a complete solution only applicable at a given moment, the *EUVE* mission has concentrated on providing general-purpose, partial solutions. Specifically, the WWW interface used in the Proposal Database is derived from the database schema. This interface requires operators be trained in the structure of the database, but dramatically reduces development and maintenance costs. More recently, the *EUVE* mission has developed a WWW server prototype (named xdb) that provides structured access to databases based solely on their Meta data information.

Questions

1. Write a brief history of EUVE mission.
2. Why the RDBMS technology was used for the mission?

3.9 Summary

- RDBMS is the basis for SQL, and for all modern database systems like MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.
- The relational model can be applied to both databases and database management systems (DBMS) themselves.
- A well-designed database takes time and effort to conceive, build and refine.
- Primary keys become essential, however, when you start to create relationships that join together multiple tables in a database.
- A relational database is a collection of data organized in two-dimensional tables consisting of named columns and rows.
- In set theory, columns are known as attributes and rows are known as tuples.
- A composite key is a key that contains more than one attribute.
- A foreign key is a field in a relational table that matches the primary key column of another table. The foreign key is used to cross-reference tables.
- The insert rule of a referential constraint is that a non-null insert value of the foreign key must match some value of the parent key of the parent table.
- The primary key is usually the key selected to identify a row when the database is physically implemented. For example, a part number is selected instead of a part description.
- Many relational database management systems include mechanisms that enforce a database's referential integrity.
- Referential integrity is another measure of the consistency of the data in databases.

3.10 Keywords

Alternate Key: All candidate keys excluding the primary key are known as alternate keys.

Artificial Key: If no obvious key either stands alone or compound is available, then the last resort is to simply create a key, by assigning a unique number to each record or occurrence. Then this is known as developing an artificial key.

Compound Key: If no single data element uniquely identifies occurrences within a construct, then combining multiple elements to create a unique identifier for the construct is known as creating a compound key.

Foreign key*:* It is an attribute (or set of attributes) that appears (usually) as a non key attribute in one relation and as a primary key attribute in another relation.

Partial Key: It is a set of attributes that can uniquely identify weak entities and that are related to same owner entity. It is sometime called as discriminator.

Arity: Arity refers to the number of columns in a table.

Cardinality: Cardinality refers to the number of elements in a set.

Columns: Columns are the attributes that describe an entity in the database model.

Domains: Domains are the set of allowable data values for a Column.

Entity Integrity: It says that no component of a primary key may be null. All entities must be distinguishable. That is, they must have a unique identification of some kind.

Foreign Key: A foreign key is a column in a table used to reference a primary key in another table.

Referential Integrity: The referential integrity constraint is specified between two relations and is used to maintain the consistency among tuples of the two relations.

Tables: Tables are collections of Rows that act as logical groupings of entities.

Tuple: A row or tuple is a complete set of Columns that describe the entity that you are trying to model.

3.11 Review Questions

1. What is the Concept of RDBMS?
2. What are the terminologies of RDBMS?
3. Explain the entities and classes in RDBMS.
4. What is the set theory concepts used in RDBMS?
5. Explain extension and intention in database systems.
6. What is the one-to-one relationship in database? Explain with example.
7. Explain the many-to-many relationship. Give an example.
8. Discuss the one to many relationships in RDBMS. Explain with example.
9. What is the integrity rule?
10. Differentiate between entity and referential integrity.

Answers for Self Assessment Questions

1. (c) 2. (a) 3. (a) 4. (b) 5. (a) 6. (c)

3.12 Further Studies

1. http://books.google.co.in/books?id=t1b9hc4Q2W0C&pg=PA35&dq=RDBMS+Concept+and+Terminology&hl=en&sa=X&ei=gYLCUt7lCuayiQf0xoHgDA&ved=0CC8Q6AEwAA#v=onepage&q=RDBMS%20Concept%20and%20Terminology&f=false

4

Normalization

STRUCTURE

4.0 Objectives

4.1 Functional Dependency

4.2 Universal Relation

4.3 Database Anomalies

4.4 Decomposition

4.5 Normalization

4.6 Concept of Normalization

4.7 Summary

4.8 Keywords

4.9 Review Questions

4.10 Further Studies

4.0 Objectives

After studying this chapter, you will be able to:

- Define the concept of normalization
- Explain the database anomalies
- Discuss the decomposition

4.1 Functional Dependency

Functional Dependency is the starting point for the process of normalization. Functional dependency exists when a relationship between two attributes allows you to uniquely determine the corresponding attribute's value. If 'X' is known, and as a result you are able to uniquely identify 'Y', there is functional dependency. Combined with keys, normal forms are defined for relations.

Examples

Bear Number determines Student Name:

BearNum ---> StuName

Department Number and Job Rank determine Security Clearance:

(DeptNum, JRank) --->SecClear

Social Security Number determines Employee Name and Salary:

SSN ---> (EmpName, Salary)

Additionally, the above can be read as:

SSN --->EmpName and SSN Salary

4.2 Universal Relation

NotesSQL recognizes Notes forms, views, and folders as tables. Upon connection to a Notes database, NotesSQL generates a special table that has the same name as the "title" of the Notes database. This table is called the Universal Relation. The Universal Relation contains all fields defined in all forms in the Notes database. From the standpoint of NotesSQL, the Universal Relation is the only true table associated with a Notes database. We allow users to access this table for their convenience; however, this access is limited. To learn more about accessing the Universal Relation, see "Using the Universal Relation." The Universal Relation table lasts for the duration of the connection.

In contrast to the Universal Relation, ordinary SQL tables created by NotesSQL behave more like SQL views than like traditional relational database tables. For example, with NotesSQL you can create a Notes form with the CREATE TABLE statement. However, the DROP TABLE statement deletes the Notes form but does not delete any data from the database. Using DROP TABLE with NotesSQL is like deleting a SQL view, in that the data remains in the database. You can view the data through other views that use the same field names, or by referencing the Universal Relation table.

Continuing this example, if you create a new table with the same name as a previously deleted table, and use some of the field names from the deleted table, you could find data in the table before you insert any data. This is because the table is actually a view of existing data in the Universal Relation table. Documents (records) in the Notes database contain the name of the form used to create them. NotesSQL uses the form name stored in the document to identify the document when selecting from a form.

4.3 Database Anomalies

The goal of designing a database schema is to minimize the storage space which is occupied by the data stored on the hard drive. Database anomalies are the errors in data, contained in the database, which reduces the performance of database Management System (DBMS). The database anomalies also affect the performance of the DBMS by increasing the size of data files. The following type of database anomalies can increase the size of data files:

Insertion Anomalies

These occur when it becomes difficult to insert data in the database. You cannot insert data having null values in a table, which has a primary key constraint. So when you have a record that contains values for all the columns apart from the primary key column, you cannot insert that record into the table. This restricts the ability of inserting the records into the database.

Deletion Anomalies

These occur when deletion of a particular record affects the existence of a particular relation in the database. For example, in a database, a table contains records of students. The subject column of the table contains the

information about the subjects, which the student have opted. Now, if you delete all the records for the multimedia subject, then you may lose the information about the students who are studying only multimedia.

Modification Anomalies

These occur when a database user changes the value of a data item and the value of that data item do not change in other tables.

4.4 Decomposition

The relational database design algorithm start with a single universal relation schema, R = {A_1, A_2, A_3,……A_n}, which includes all the attributes of a database. The database designers specify the set, F of functional dependencies, which holds true for all the attributes of R. This set, F of functional dependencies is also provided to the design algorithms. With the help of functional dependencies, these algorithms decompose the universal relation schema, R into a set of relation schemas, D= {R_l, R_2..., R_m}, which becomes the relational database schema. In this case, D is referred as a decomposition of R. The properties of decomposition are as follows:

- ***Attribute preservation:*** It involves preserving all the attributes of the relation, which is being decomposed by the design algorithms. While decomposing a relation, you need to make sure that each attribute in R exists in at least one relation schema, R_i while decomposing the relation.
- ***Lossless-join decomposition:*** It ensures that the join remains in the same relation, as it was before the decomposition of the relation. The decomposition of the relation R into several relations, R_1, R_2, ..., R_n is called a lossless join decomposition, if the relation R is the natural join of the relations R_1, R_2, ..., R_n,,. To test whether a given decomposition is a lossless join for a given set F of functional dependencies, you need to decompose the relation, R into R_1 and R_2. If the decomposition of the relation R is lossless join, then one of the following conditions has to be true:

○ (R_l intersection R_1) → (R_1 - R_2) and

○ (R_1 intersection R_1) → (R_1 – R_2)

- ***Dependency preservation:*** It states that if each functional dependency X-> Y, specified in F, either directly appears in one of the relation schemas R_i in the decomposition D or is inferred from the dependencies that appear in the relation, R_i. The need of dependency preservation arises because each dependency in F represents a constraint on the database. When decomposition does not preserve the dependency, then some dependency can be lost in the decomposition. You can check for a lost dependency by creating a join of two or more relations in decomposition to get a relation, which includes all the left and right-hand side attributes of the lost dependency. Then, check whether or not the dependency is preserved on the result of join.

4.4.1 Multi-Valued Dependency

An entity in E-R model can have multi-valued attributes. A multi-valued attribute is one that does not have a single value but a collection of values. If you have to store such an entity in one relation, you will have to repeat all the information other than the multi-valued attribute value. In this way, the same instance of the entity will have many tuples. The situation becomes much inferior if any entity has more than one multi-valued attributes. The multi-valued dependency gives a solution to the problem of more than one multi-valued attributes.

MVD: Let R (X, Y, Z) be a relation. The multi-valued dependency X→→Y can hold for relation R if for the given set of zero or more value for attribute X there is a set of associated values for the set of attribute Y. The values of Y only depend on X values and these have no dependence on the set of attributes Z.

Suppose a Students table, which has Stud _name, Qualifications and Languages as attributes. In the relation a student can have more than one qualification (Stud_ id→ Qualifications) and know more than one language

(Stud _id→ Languages). This relation shows duplication of data and inc You can decompose the Students relation into two relations having attributes Stud_id, Qualifications and Stud_id, Languages.
In this example, if there were dependency between Qualifications and Languages attributes then Student relation would not have MVD and cannot be decomposed into two relations.

4.4.2 Fourth Normal Form

The fourth and fifth normal forms depend on multi-valued facts and composite keys. The multi-valued facts correspond to many-to-many relationship and many to one relationship. In the fourth and fifth normal forms, you need to minimize the number of columns in a composite key.
In the fourth normal form, you must ensure that a record satisfies the third normal form and does not contain two or more independent multi-valued facts about an entity. For example, consider a table, R, with attributes employees, skills and languages. Since in this table, an employee may have various skills and may know it various languages, therefore, the table has too many-to-many relationships. Under the fourth normal form, the two many-to-many relationships are not represented in a single row and you need to split R into two tables. Therefore, the table, R, is split into a table with attributes, employees and skill and another table with attributes employees and language. A relation is considered to be as the fourth normal form if each defined table contains not more than one multi-valued dependency per key attribute.
For example, consider an Instructor table shown in Table 4.1 This tables lists the various attributes of instructor table.

Table 4.1: The Instructor Table

MID	Database	Instructor
1	Access	Anurag
8	Access	Samar
1	Access	Roshan
1	DB2	Gita
1	DB2	John
8	Oracle	Ram

The-redundancy of data is easily perceived. For each MID there are defined multiple of Instructors and Database. This is a perfect example of a multi-valued dependency. Figure 4.1 shows the fourth normal form of instructor table.

MID DATA Table

MID	Database
1	Access
8	Access
1	DB2
8	Oracle

MID Instructor Table

MID	Database
1	Miller
1	John
8	Smith

Figure 4.2: The Fourth Normal Form of Instructor Table.

4.4.3 Join Dependency and the Fifth Normal Form

The fifth normal form (5NF) is based on join dependency. Join dependency implies that after a table is decomposed into three or more tables, the tables can be joined to from the original table. A table is in 5NF if the table cannot have lossless decomposition into smaller tables. A lossless decomposition implies that after a relational table is decomposed into smaller tables, the joining of the table's results in exactly the same relation table, which was decomposed. If an instance is added to a table which is not in 5NF, it gives spurious results when the tables are decomposed and then rejoined. Table 4.24 lists the various attributes of Instructor-MID-Location table.

Table 4.3: The Instructor-MID-Location Table

Instructor	MID	Location
Sumit	1	Delhi
Nina	2	Kanpur
Karan	1	Kanpur

If you were to add the MID 2 to Kanpur, you would have to add a line to the table for each instructor located in Kanpur. If Jones were certified for MID 2 and could travel to Kanpur, you would have to add two lines to reflect this.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Which normal form of below database?

The database contains data about cars, their owners and when the car was registered for that owner.

PersonID	FirstName	LastName	LicensePlate	RegistrationDate	Birthdate
1000	Radha	Gupta	ABC123	2004-10-12	1981-04-04
1010	Amit	Nigam	DEF234	2003-02-12	1945-12-12
1000	Anil	Singh	ABC123	2001-04-23	1981-04-04

……..……

……..……

……...

4.5 Normalization

Data normalization is a process in which data attributes within a data model are organized to increase the cohesion of entity types. In other words, the goal of data normalization is to reduce and even eliminate data redundancy, an important consideration for application developers because it is incredibly difficult to stores objects in a relational database that maintains the same information in several places. Table 4.1 summarizes the three most common forms of normalization (First normal form (1NF), Second normal form (2NF), and Third normal form (3NF)) describing how to put entity types into a series of increasing levels of normalization. With respect to terminology, a data schema is considered to be at the level of normalization of its least normalized entity type. For example, if all of your entity types are at second normal form (2NF) or higher then we say that your data schema is at 2NF.

Table 4.4: Data Normalization Rules

Level	**Rule**
First normal form (1NF)	An entity type is in 1NF when it contains no repeating groups of data.
Second normal form (2NF)	An entity type is in 2NF when it is in 1NF and when all of its non-key attributes are fully dependent on its primary key.
Third normal form (3NF)	An entity type is in 3NF when it is in 2NF and when all of its attributes are directly dependent on the primary key.

In this chapter we will also discuss about the database anomalies and database decomposition.

Self Assessment Questions

1. The relational database design algorithm start with a …… universal relation schema.
(a) Double (b) Single
(c) Single and double (d) None of these

2. The primary key of a relational table uniquely identifies each …….. in a table.
(a) Row (b) Column
(c) Both (a) and (b) (c) None of these

3. Normalization is a process of eliminating the redundancy of data in a database.
(a) True (b) False

4. BCNF is also present in ……
(a) 1NF (b) 2NF
(c) 3NF (d) 4NF

4.6 Concept of Normalization

Normalization is a process of eliminating the redundancy of data in a database. A relational table in a database is said to be in a normal form if it satisfies certain constraints. The normalization process involves various levels of normal forms that allow you to separate the data into multiple related tables. The various normal

forms are first normal form (1NF), second normal form (2NF), third normal form (3NF), fourth normal form (4NF) and fifth normal form (5NF).
The goals of normalization are:
- Removing the redundant data
- Ensuring that only related data is stored in a table

Therefore, normalization helps you to remove data redundancy and update inconsistencies when data is inserted, deleted or modified in a database. The benefits of normalization are as follows:
- It provides better overall database organization and data consistency within a database.
- It allows you to create tables that can be easily joined with other tables with related information.
- It helps to reduce redundant data across the tables.
- It prevents data loss by assigning primary and foreign keys in a table.
- It helps to reduce modification anomalies such as deletion, insertion and update anomalies.
- It defines relation constraints that are a logical consequence of keys.

4.6.1 Normalization Terminology

Normalization terminology consists of various concepts that are frequently used in normalization, for example primary key and functional dependency.

Primary Key

The primary key of a relational table uniquely identifies each row in a table. A primary key is either a column in a table that is unique such as identification number and social security number or it is generated by the DBMS such as a Globally Unique Identifier (GUID). Primary key is a set of single column or multiple columns from a table. For example, consider a student records database that contains tables related to student's information. The first table, STUDENTS, contains a record for each student at the university. The table, STUDENTS, consists of various attributes such as student_id, first _name, last _name and student_ stream.
Table 4.2 lists the various attributes in the STUDENTS table.

Table 4.5: The STUDENTS table

Student_id	First_ name	Last_ name	Student_ stream
Sol	Eva	Gupta	Computers
SO1	Denesh	Sharma	Electronics
S01	Dipa	Bajpai	Electronics

A unique Student __id number of a student is a primary key in the STUDENTS table. You cannot make the first or last _name of a student a primary key because more than one student can have the same first name and can have same stream.

Functional Dependency

A functional dependency is termed as a constraint between two sets of attributes of the database. Functional dependency is represented by X→Y between two attributes, X and Y, in a table. The functional dependency X→Y implies that Y is functionally dependent on X. Table 4.3 lists the various attributes in the EMPLOYEE table.

Table 4.6: EMPLOYEE table

Employee _id	Employee_ name	Employee _dept
K067263	Bindu	Sales
K067264	Sariks	Accounts
K067265	Kavita	Sales

In Table 4.6, the various attributes of the EMPLOYEE are Employee_id Employee_ name and Employee _dept. You can state that:

Employee_id→Employee_name

In the above representation the Employee name attribute is functionally dependent on the Employee_ id. This implies that the name of an employee can be uniquely identified from the id of the employee. However, you cannot uniquely identify the Employee_id from the Employee_name column because more than one employee can have the same name. However, each employee has a different value in: Employee_ id column.

Functional dependencies are a type of constraints based on keys such as primary Key or foreign key. For a relation table R, a column Y is said to be functionally dependent on a column X of the same table if each value of the-column X is associated with only one value of the column Y at a given time. All the columns in the relational table R should be functionally dependent on X if the column X is a primary key.

If the columns X and Y are functionally dependent, the functional dependency can be represented as:

R. X→R. Y

For example, consider the following functional dependency in a table.

Employee_ id→Salary, the column Employee id functionally determines the Salary column because the salary of each employee is unique and remains same for an employee, each time the name of the employee appears in the table.

A functional dependency represented by X→Y between two sets of attributes X and Y are the subsets of R and is termed as trivial functional dependency if Y is a subset of X. For example, Employee id→ Project is a trivial functional dependency.

A functional dependency represented by X→Y between two sets of attributes X and Y are subsets of R and is termed as non-trivial functional dependency if at least one of the attributes of Y is not among the attributes of X. For example, Employee_id→Salary is a non-trivial functional dependency.

4.6.2 Inference Rules and Attribute Closure

Inference Rules

Inference rules for functional dependencies define the new dependencies, which can exist between two relations. The inference rules help to deduce these new dependencies from a given set of functional dependencies, F. The set dependencies, which cannot be specified, is called the closure of F and is, denoted by F^+. Following are the six inference rules of functional dependencies F:

- *IRl (reflexive rule):* If X⊇Y, then X→Y: This rule states that if XÊY and two tuples t^1 and t^2exist in a relation instance t of relation R such that t^1 [X] = t^2 [X]. Now, t^2 [X] = t^2 [X] because XÊY. This implies that X→Y holds true in relation instance r of relation R.
- *IR2 (augmentation rule):* { X→Y} J= XZ→YZ: This rule states that if X→Y holds true in a relation instance r of R but XZ →YZ does not exist, then tuple t^1 and t^2 must exist in relation R.
- *IR3(transitive rule):* { X→Y, Y → Z} |= X → Y: This rule states that if both, X→Y and Y → Z hold true in a relation r, then for any two tuples $t^1$and t^2 in r, you must have t^1 [Y] = t^2 [Y].
- *IR4 (decomposition or projective rule):* This rule states that if X →YZ holds true, then X→ Y and X→ Z also hold true.

- *IR5 (union or additive rule):* This rule states that if X→ Y and X→ Z hold true, then in the relation R, X→ YZ also holds true.
- *IR6 (pseudo transitive rule):* This rule states that if X→Y and WY→Z hold true, then WX →Y also holds true.

Attribute Closure

To compute the closure J^+ of a given set J of functional dependencies, you can apply the inference rules until they stop producing new functional dependencies. You can test whether a set of attributes, J is a super key or not, by finding the set of attributes, which are functionally determined by J. You can use the following algorithm to compute the closure J^+:

result := J
while (changes to result) do
for each functional dependency B→ Y
in F do
begin
if B $\subseteq$ result;
then result:= result UY
end

The above code assumes that J is a set of attributes and you can call the set of attributes determined by a set F of functional dependencies. The closure of J under F is denoted by J^+.

4.6.3 First Normal Form

A table is said to be in 1NF if the data in the table has an identifying key and does not include repeating groups of data. To reduce data redundancy by using first normal form, you need to:

- Remove the duplicate columns from a table
- Create a separate table for related data and identify the primary key in the table

According to the first normal form, a table should be atomic, which implies that no duplicate data exists within the same row of a table. For example, consider the items table shown in Table 4.7.

Table 4.7: The items table

Order No	Item l	Iteml_Qty	Iteml Price	Item2	Item2_Qty	Item2 Price
001	IT90	322	700	IT91	564	800

In Table 4.7, the information provided is redundant. The multiple values of the same type, such as quantity and price of two items, are stored in different columns.

The requirements of the first normal form are:

- Eliminate the multi-valued fields from the table
- Each column in the table must be atomic
- Each column in the table must have a key such as primary or foreign key
- Remove the repeated information from the table

Eliminate the Multi-valued Columns from the Table

The first requirement to apply INF to a table is to ensure that the table does not contain multi-valued columns. For example, consider a Books table with attributes book-name, book-author, book_ISBNno, book_price, book_publisher and book-category. Table 4.8 lists the various attributes in the Books table.

Table 4.8: Books table

Book _author	Book _ISBNno	Book_price	Book_publisher	Book_ category
Namita	8790478	35	ABC	Sales

Anu	8790388	25	PQR	Accounts
Amit	8790689	77	ABC	Sales

In Table 4.8, since a book can have more than one author and also a book can be included in different categories, therefore, columns that consist of multi-valued elements should be removed from the table. Therefore, the Books table should contain Book_ISBNno, Book_price and Book_publisher columns.
Table 4.9 lists the various attributes of the Books table after the multi-valued elements are removed.

Table 4.9: The Books Table after the Multi-valued Elements is removed

Book_ ISBN no	Book_price	Book_publisher
8790478	35	ABC
8790388	25	PQR
8790689	77	ABC

Table 4.10 lists the Books_category table.

Table 4.10 : Books_category table.

Book_ ISBNno	**Book _category**
8790478	Sales
8790388	Accounts
8790689	Sales

Each Column in the Table is Atomic
You need to ensure that each column in a table that is to be normalized is atomic. For example, the author table can be divided into two columns, the first name and the last name to make the table atomic. Table 4.11 lists the various attributes in the author table.

Table 4.11: Author table

Book _ISBNno	First_ name	Last_ name
8790478	Ram	Singh
8790388	Tina	Gupta
8790689	Mahesh	Sharma

Each Column Must Have a Key
You can determine if each column in a table contains a unique value by verifying the keys in the table. The various keys that can exist in a table are:
Super key: It refers to one or more than one column that identifies a unique row with in a table.

Candidate key: It refers to the super key consisting of minimum number of columns required to identify a unique row in a table.
Primary key: It refers to the candidate key required to uniquely identify a row in a table.
Alternate key: It refers to the candidate key, which is not selected as a primary key.
Foreign key: It refers to one or more than one column in a table that matches a candidate key in the same or different table. You can link a row in a table to a row in another table by using a foreign key.
In the Books table, the super keys are Book __author and book__ISBNno. The super keys for the author table are the combination of first __name and last _ name. Similarly, for the categories table, the super key is category.
The primary key for the Books table is Book __ISBNno and the primary keys for the author table are first _name and last _name. The primary key for the categories table is CategoryII. To ensure that each row in the author table is unique, you can add the Author city and Author zip code columns in the primary key field. Table 4.12 lists the various attributes in the author table.

Table 4.12: Keys in Author table

Book_ author	Book_ ISBNno	Author_ address	Author_phoneno
John Wilkins	8790478	Houston	0098392978
Chris Burton	8790388	New York	008273827
Ken Wilkins	8790689	Denver	002632878

Eliminate the Repeating Values
To make a table compliant with 1NF, you need to eliminate the repeated values from the table For example; in the Books table the publisher column can contain same values for different books. Therefore, to remove the repeating values you can make a separate table, publisher, with attributes Publisher _ id and Publisher _ name. The Publisher_ id can be identified as the primary key for the publisher table.Table 4.13 lists the various attributes in the Publisher table.

Table 4.13: The Publisher.

Publisher id	Publisher name
P0240	ABC
P0240	PQR
P0240	ABC

Caution
Each column has a unique name and the content within it must be of the same type. The different type of content will assumed as invalid and not accepted by the database.

4.6.4 Partial Dependency
In a table, a primary key consists of one or more than one column to uniquely identify each row in the table. Partial dependency occurs when a row of a table is uniquely identified by one column that constitutes a primary key without requiring the entire primary key to uniquely identify the row. For example, consider a table. ‘Stocks’ with attributes cust_id, stock and stock_price.

Table 4.14: lists the various attributes in the Stocks table.

cust _id	Stock	stock_price
C012	Stkl	15
C013	Stk2	10
C014	Stk3	20

In Table 4.14, suppose cust_id and stock are identified as the primary key for the Stocks table. However, the column stock_price is partially dependent on the primary key because only the stock column determines the stock_price. Also, the values in the stock_price column do not need the cust_id column to uniquely identify the price of the stocks. Therefore, you need to make a separate table for the stock_price where the stock column is the primary key. In the new table, partial dependency is eliminated because the stock_price column is entirely dependent on the primary key.

Partial dependencies can only occur when more than one field constitutes the primary key. If there is only one field in the primary identifier, then partial dependencies cannot occur.

4.6.5 Second Normal Form

A table is in 2NF if the table satisfies all the conditions of the first normal form and does not consists of any column that depends on only one part of the identified primary key. The 2NF is based on the concept of full dependency

To apply 2NF to a table you need to:

- Ensure that the table conforms to 1NF
- Create a separate table for the sets of values which can be applied to multiple records
- Relate the table with a foreign key

For example, consider the following table employee_project. Table 4.15 lists the various attributes in the employee_project table.

Table 4.15: The Employee_project Table.

Emp_ id	Proj_ no	Proj _hrs	Emp_name	Proj_ name	Proj _loc
H76320	W36	08	Abc	Payroll	Houston
H76321	W37	02	Xyz	Billing	Denver

Table conforms to 1NF since it does not contain repeated values and Emp_id and Proj_ id are identified as the primary keys for the table. However, the table is not in 2NF because all the columns of the table depend on only a part of the primary key, which comprises of Emp_id and Proj _no, identified for the table. For example, the column Emp_ name is dependent on only the Emp_ id and does not depend on the proj_no part of the primary key. Similarly, the Proj_ name column is dependent only the Proj_ no column and not on the Emp_ id primary key.

Therefore, to apply 2NF to the employee_project table, you need to make a separate table for columns that depend on only a part of the primary key. The new table should contain columns that are dependent on the entire primary key identified for the table. The tables formed after applying 2NF to the employee_project table are emp_proj table and emp table and proj table.

Table 4.16 lists the various attributes in the emp_proj table.

Table 4.16: The Emp_project table

Emp_ id	Proj_ no	Proj_ hrs
H76320	W36	08
H76321	W37	02

Table 4.16 lists the various attributes in the emp _table.

Table 4.17: The Emp table

Emp_ id	Emp_ name
H76320	W36
H76321	W37

Table 4.18 lists the various attributes in the Proj table.

Table 4.18: The Proj Table

Proj _no	Proj_ name	Proj_ loc
H76320	Payroll system	Houston
H76321	Billing system	Denver

Similarly, consider an ORDERS table that you need to normalize to 2NF. Table 4.19 lists the various attributes in the ORDERS table.

Table 4.19: The ORDERS Table

Order_ no	Item _no	Customer	Item	Qty	Price
H76320	01	ABC Corp	IT90	322	3600
H76320	02	ABC Corp	IT91	564	4500
H76321	01	XYZ Corp	IT92	736	1200

In Table 4.19, Order_no and Item_no are identified as the primary keys for the table. Also, the table conforms to 1NF since it does not contain repeated value. However, to apply 2NF to the ORDERS table, you need to create a separate table for the columns that do not depend on either Order no or Item no primary key.
The tables, which are created after 2NF is applied to the ORDERS table, are order_cust table and orders table. Table 4.20 lists the various attributes in the Order_cust table.

Table 4.20: Order_cust table

Order_ no	Customer
H76320	ABC Corp
H76321	XYZ Co

In the above Order_cust table, the customer column is dependent on the primary key order_no. Similarly, another table is created in which all the columns, Order_no and Item_no, Item, Qty and Price are dependent on the primary keys, Order _no, Item, Qty and Price are dependent on the primary keys Order_no and Item_no. Table 4.21 lists the various attributes in the orders table.

Table 4.21: Orders table

Order no	Item no	Item	Qty	Price
H76320	01	IT90	322	700
H76320	02	IT91	564	1000
H76321	01	IT92	736	480

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Create a table in second normal form.

……..……

……..……

……...

Did You Know?

The principle of normalization was developed in Scandinavia during the 1960s and was first developed and articulated by Bengt Nirje.

4.6.6 Transitive Dependency

Transitive dependency occurs when a non-key column is uniquely identified by values in another non-key column of a table. A non-key column of a table refers to the column that is not identified as a key such as candidate or primary key. For example, consider a SUPPLIER table with attributes supplier_id, supplier_status and supplier_address. The functional dependencies that exist in the SUPPLIER table help to understand the concept of transitive dependency. Table 4.22 lists the various attributes in the SUPPLIER table.

Table 4.22: SUPPLIER table

Supplier_id	Supplier_status	supplier_address
SO1	10	Delhi
S02	20	Noida
S03	30	Kanpur

In the above table, the following functional dependencies hold:

Supplier_id →supplier_status

Supplier_id →Supplier_address

Supplier_address → Supplier_status

In the SUPPLIER table, the non-key column supplier status is identified by both the primary key supplier id and non-key column supplier address. Therefore, transitive dependency exists in the above table. To eliminate transitive dependency, you need to apply 3NF to the table.

Caution

A database to be in 2NF, it must first fulfill all the criteria of a 1NF database.

4.6.7 Third Normal Form

A table is said to be in third normal form or 3NF if the table satisfies the requirements of 2NF and the non-key columns should be only functionally dependent on the primary key. The third normal form is based on the concept of transitive dependency. A functional dependency, A→B, in a relation, R is a transitive dependency if the following conditions are satisfied:

- A column or set of columns, C, exists in the table that is neither the candidate key of R nor the subset of any key of R.
- The functional dependencies A→*C and C→B hold in the table.

For example, consider a Subject table with attributes such as Subject_no and Chapter_name. Table 4.23 lists the various attributes in the Subject table.

Table 4.23: Subject table

Subject _no	Chapter_ name	Instructor	Department
H76320	Data structure	ABC	Computer
H76320	Communication	XYZ	Electronics

In the above table, Subject_no is the only candidate key. Therefore, the following functional dependency exists for the Subject table.

Subject _no	Chapter_ name	Instructor	Department
H76320	Data structure	ABC	Computer
H76320	Communication	XYZ	Electronics

Subject_ no →Chapter_ name
Subject_ no → Instructor
Instructor →Department

From the above functional dependencies, you can say that Subject _no Department and therefore the above table are in 2NF. However, the table is not in 3NF since Department is not directly dependent on Subject no. In the Subject table the Department column is determined by another non-key column, Instructor. Therefore to apply 3NF to the Subject table, you need to decompose the table in two tables, subject-inst table and instructor table. Table 4.24 lists the various attributes in the Subject_inst table.

Table 4.24: The Subject_inst table

Subject _no	Subject _name	Instructor
H76320	Data structure	ABC
H76320	Advanced OS	XYZ

Exercise: Check Your Progress 3

Note: i) Use the space below for your answer.

Ex1: Create a relational table to third normal form.

……..……

……..……

……...

4.6.8 Boyce-Codd Normal Form

Boyce-Codd Normal Form (BCNF) is stricter than the third normal form. In BCNF the relation, who is in BCNF is also present in Third Normal Form (3NF), but the relation present in 3NF form is not necessarily be present in BCNF. In 3NF if a relation has more than one candidate 1ey then anomalies can occur. In case of overlapping of candidate keys, 3NF is unable to stop the occurrence of anomalies. This provides a base for BCNF and is based on the determinant concept. A determinant is an attribute on which some other attribute is fully functionally dependent. The following code shows the relation and determinants:

R(a,b,c,d)

a, c → b, d

In the above code the first determinant states that you can change the primary key of relation R from a,b to a,c. After applying this change, you can still determine the non key attributes present in relation R. The second determinant indicates that a, d determine b, but as ad do not determine all the non-key attributes of R, it cannot be considered as the primary key of R. This implies that the first determinant is a candidate key, but the second determinant is not a candidate key, hence this relation t in BCNF but is in 3NF.

To be in BCNF, every determinant of the relation has to be a candidate key. The definition of BCNF specifies that a relation schema R is in BCNF if a non-trivial functional dependency X→A holds in R, then X is a super-key of R.

Caution

The form of a string, security mechanisms or character validation algorithms should usually be implemented after normalization, because the normalization can result unexpected change.

Did You Know?

BCNF was developed in 1974 by Raymond F. Boyce and Edgar F. Codd to address certain types of anomaly not dealt with by 3NF as originally defined.

Exercise: Check Your Progress 4

Case Study-An Example of a Hospital

In order to explain the first three normal forms, let us take an example of a hospital. Patients visit the hospital and their visit history is maintained by the hospital staff. Different physicians may be available on different dates. They diagnose and treat the patients of all categories. Some of treatments are free while others are to be paid by the patients. Sample data of the case is shown in the following table.

PatientID	Name	Address	Visit Date	Physician	Diagnosis	Treatment
P-100809	A	City: X	12-02-2007	Dr. Z	Chest	Free
			20-02-2007	Dr. F	Infection	Free
			29-02-2007	Dr. R	Cold	Paid
			15-03-2007	Dr. L	Hepatitis-A	Paid
					Eyes Infection	
P-200145	N	City: Y	10-01-2007	Dr. L	Bone Fracture	Paid
			15-02-2007	Dr. K	Cough	Free
			25-03-2007	Dr. A	Flu	Free

First Normal Form

In First Normal Form, we eliminate repeating groups so that intersections of rows and columns have atomic or single values. So after applying the First Normal Form, the above data will take the following shape.

PatientID	Name	Address	Visit Date	Physic	Diagnosis	Treatment

				ian		
P-100809	A	City: X	12-02-2007	Dr. Z	Chest Infection	Free
P-100809	A	City: X	20-02-2007	Dr. F	Cold	Free
P-100809	A	City: X	29-02-2007	Dr. R	Hepatitis-A	Paid
P-100809	A	City: X	15-03-2007	Dr. L	Eyes Infection	Free
P-200145	N	City: Y	10-01-2007	Dr. L	Bone Fracture	Paid
P-200145	N	City: Y	15-02-2007	Dr. K	Cough	Free
P-200145	N	City: Y	25-03-2007	Dr. A	Flu	Free

After applying First Normal Form, Repeating Groups are eliminated. Suppose the above relation is named as PATIENT and the primary key of this relation is selected as combination of PatientID and VisitDate attributes.
PATIENT(PatientID, Name, Address, VisitDate, Physician, Diagnosis Treatment)

Second Normal Form

A relation is said to be in Second Normal Form if it is in INF and all non-key attributes are fully functionally dependent on key attribute.

In the above PATIENT relation, we see that 'Name' and 'Address' are functionally dependent on 'PatientID' but the remaining attributes depend upon 'PatientID', 'VisitDate'. So the functional dependencies in the above relation are:

PatientID → Name, Address

PatientID, VisitDate → Physician, Diagnosis, Treatment

The primary key for this relation is the composite key (PatientID, VisitDate). Therefore, the non-key attributes 'Name' and 'Address' are functionally dependent on part of the primary key (PatientID) but not on 'VisitDate'. So the partial functional dependency exists in this relation.

The partial functional dependency must be removed to convert a relation into Second Normal Form. Therefore, we split the relation into the following two relations such as PATIEN2 and PATIENT HISTORY. The relations with sample data are given below.

PATIENT HISTORY (PatientID, Name, Address)

PatientID	**Name**	**Address**
P-100809	A	City: X
P-200145	N	City: Y

PATIENT HISTORY (PatientID, VisitDate, Physician, Diagnosis, Treatment)

PatientID	**VisitDate**	**Physician**	**Diagnosis**	**Treatment**
P-100809	12-02-2007	Dr. Z	Chest Infection	Free
P-100809	20-02-2007	Dr. F	Cold	Free
P-100809	29-02-2007	Dr. R	Hepatitis-A	Paid
P-100809	15-03-2007	Dr. L	Eyes Infection	Free
P-200145	10-01-2007	Dr. L	Bone Fracture	Paid
P-200145	15-02-2007	Dr. K	Cough	Free
P-200145	25-03-2007	Dr. A	Flu	Free

In the relation PATIENT2, the primary key is PatientID and all other attributes are functionally dependent on this primary key. Similarly, in the relation PATIENT HISTORY, the primary key is PatientID, VisitDate and all other attributes are functionally dependent on this primary key. Therefore, the relations PATIENT2 and PATIENT HISTORY are in 2NF.

Third Normal Form

A relation is in Third Normal Form (3NF) if it is in Second Normal Form and no transitive dependency exists. If we see the relation PATIENT HISTORY, it is in 2NF. It is obvious that 'Physician' and 'Diagnosis' attributes directly depend on primary key but 'Treatment' is indirectly dependent on the primary key. It means that 'Treatment' is transitively dependent on 'Diagnosis'. Therefore, we split the relation into two relations to get the relations in 3NF. Suppose these relations are named as PAT-HISTORY and DIAGNOSIS. The relations with sample data are given below.

PAT-HISTORY (PatientID, VisitDate, Physician, Diagnosis)

PatientID	VisitDate	Physician	Diagnosis
P-100809	12-02-2007	Dr. Z	Chest Infection
P-100809	20-02-2007	Dr. F	Cold
P-100809	29-02-2007	Dr. R	Hepatitis-A
P-100809	15-03-2007	Dr. L	Eyes Infection
P-200145	10-01-2007	Dr. L	Bone Fracture
P-200145	15-02-2007	Dr. K	Cough
P-200145	25-03-2007	Dr. A	Flu

Diagnosis (diagnosis, Treatment)

Diagnosis	Treatment
Chest Infection	Free
Cold	Free
Hepatitis-A	Paid
Eyes Infection	Free
Bone Fracture	Paid
Cough	Free
Flu	Free

Questions

1. Explain the database arrangement of the hospital.
2. Which normal forms are used to arrange the hospital's database?

4.7 Summary

- The goal of designing a database schema is to minimize the storage space which is occupied by the data stored on the hard drive.
- Database anomalies are the errors in data, contained in the database, which reduces the performance of database Management System (DBMS).
- Normalization is a process of eliminating the redundancy of data in a database relational table in a database is said to be in a normal form if it satisfies constraints.
- The normalization process involves various levels of normal forms that allow you to separate the data into multiple related tables. The various normal forms are first normal form (1NF), second normal form (2NF), third normal form (3NF) fourth normal form (4NF) and fifth normal form (5NF).
- The primary key of a relational table uniquely identifies each row in a table.

4.8 Keywords

Candidate key: If there is more than one key in a relation, the keys are called candidate keys.
Functional dependency: It is termed as a constraint between two sets of attributes of the database.
Key: A set of attributes that uniquely and minimally identifies a tuple of a relation.
1NF: A table is said to be in 1NF if the data in the table has an identifying key and does not include repeating groups of data.
Super key: It refers to one or more than one column that identifies a unique row with in a table.

4.9 Review Questions

1. What is normalization?
2. List the various normal forms involved in the normalization process.
3. What is functional dependency?
4. What is a super key?
5. How do you apply 1NF in a relational table?
6. How do you apply 2NF to a relation table?
7. Explain the 3NF and BCNF.
8. Define the term transitive dependency.
9. Explain join dependency and fifth normal form.
10. What is fully functional dependency?

Answers for Self Assessment Questions

1. (b) 2. (a) 3. (a) 4. (c)

4.10 Further Studies

1. http://books.google.co.in/books?id=JWoKCHJheSUC&pg=PT202&dq=normalization+in+dbms&hl=en&sa=X&ei=64LCUtnmHtGXiAeC3YDwDA&ved=0CC0Q6AEwAA#v=onepage&q=normalization%20in%20dbms&f=false

5
Relational Algebra

STRUCTURE

5.0 Objectives

5.1 Introduction

5.2 The Relational Algebra

5.3 The Select Operation

5.4 Project Operation

5.5 Join Operation

5.6 Division Operation

5.7 Cross Product Operation

5.8 Set Operators

5.9 Summary

5.10 Keywords

5.11 Review Questions

5.12 Further Studies

5.0 Objectives

After studying this chapter, you will be able to:

- Define the relational algebra
- Explain the select operation
- Discuss the project operation
- Explain the join operation

5.1 Introduction

This chapter begins a study of database programming, that is, how the user can ask queries of the database and can modify the contents of the database. Our focus is on the relational model and in particular on a notation for describing queries about the content of relations called "relational algebra".

While ODL uses methods that, in principle, can perform any operation on data, and the E/R model does not embrace a specific way of manipulating data, the relational model has a concrete set of "standard" operations on data. Surprisingly, these operations are not "Turing complete" the way ordinary programming languages are. Thus, there are operations we cannot express in relational algebra that could be expressed. This situation is

not a defect of the relational model or relational algebra, because the advantage of limiting the scope of operations is that it becomes possible to optimize queries written in a very high level language such as SQL.

5.2 The Relational Algebra

The relational algebra is a procedural query language. It consists of a set of operations that take one or two relations as input and produce a new relation as their result. The fundamental operations in the relational algebra are select, project, union, set difference, Cartesian product, and rename. In addition to the fundamental operations, there are several other operations-namely, set intersection; natural join, division, and assignment. These operations will be defined in terms of the fundamental operations.

To begin our study of operations on relations, we shall learn about a special algebra, called *relational* algebra, that consists of some simple but powerful nays to construct new relations from given relations. When DBMS's that used the relational model were first developed, their query languages largely implemented the relational algebra. However, for efficiency purposes, these systems regarded relations as bags, not sets. That is unless the user asked explicitly that duplicate tuples be condensed into one (i.e., that "duplicates be eliminated"), relations were allowed to contain duplicates.

Another change to the algebra that was necessitated by commercial implementations of the relational model is that several other operations are needed. Most important is a way of performing aggregation, e.g., finding the average value of some column of a relation.

Algebra, in general, consists of operators and atomic operands. For instance, in the algebra of arithmetic, the atomic operands are variables like r and constants like 15. The operators are the usual arithmetic ones: addition. Subtraction, multiplication, and division. Any algebra allows us to build expressions by applying operators to atomic operands and/or other expressions of the algebra. Usually, parentheses are needed to group operators and their operands.

For instance, in arithmetic we have expressions such as $(x + y) * z$ or $((x + 7)/(y - 3)) + x$.

Relational algebra is another example of algebra. Its atomic operands are:

1. Variables that stand for relations.
2. Constants, which are finite relations.

As we mentioned, in the classical relational algebra, all operands and the results of expressions are sets. The operations of the traditional relational algebra fall into four broad classes:

a) The usual set operations - union, intersection, and difference – applied to relations.

b) Operations that remove parts of a relation: "selection" eliminates some rows (tuples), and "projection" eliminates some columns.

c) Operations that combine the tuples of two relations, including "Cartesian product," which pairs the tuples of two relations in all possible ways, and various kinds of "join" operations, which selectively pair tuples from two relations.

d) An operation called renaming that does not affect the tuples of a relation, but changes the relation schema, i.e., the names of the attributes and/or the name of the relation itself.

Did You Know?

The educational form of relation algebra treated here was developed by Alfred Tarski and his students, starting in the 1940s.

5.3 The Select Operation

The Selection Operation is used to select a subset of tuples (or horizontal subset or rows) from a single relation that satisfy the given selection condition. The result is stored into a new table. The Select Operator is used for

this purpose. The Selection operation is also considered a filter that retrieves only those tuples that satisfy the given condition.

The general form of Selection operation is:

SELECT<table-name> WHERE <condition>

[GIVING <new-table-name>]

Where

table-name	Specifies the name of existing table (relation) you want to select tuples from.
condition	Specifies the selection condition. It is a relational condition or Boolean expression. You can use the relational operators; . >, < , > = < = , = and logical operators AND, OR, NOT to form the selection condition.
new-table-name	Specifies the new table in which you want to store the result. The new table has the same attributes as existing table. The use of new table is optional. If it is omitted, then only the result is displayed, not stored in a table.

Symbolically, the Selection operation is written as:

"δ condition $^{(R)}$ Where δ (sigma) symbol is used to denote the Select Operator and R is the name of relation.

Example:

To retrieve the rows from "Student" table that has value 'L' in the attribute City, the Selection operation is written as;

SELECT student WHERE city = 'L' GIVING temp

The above Selection operation on 'Student' table can be written as;

δ city = 'L' (student)

The left-hand-side table below shows the actual "Student" table, while the right-hand-side table shows the result of the statement.

Table 5.1: "Student" table

Roll_No	Name	City
1	S	L
2	K	P
3	F	Q
4	H	L

Table 5.2: Result

Roll_No	Name	City
1	S	L
4	H	L

Similarly, to retrieve those rows from 'Marks' table that have value greater than 60 in 'Phy' attribute and have value greater than 70 in attribute 'Comp', the Selection operation is written as:

δphy >60 AND comp >70 (Marks)

The Table 5.3 shows the actual 'Marks' table, while the Table 5.4 shows the result of the statement.

Table 5.3: Marks Table

Roll_No	Phy	Math	Comp
1	86	58	86
2	78	75	78
3	96	74	54
4	54	76	78

Table 5.4: Result

Roll_No	Phy	Math	Comp
1	86	58	86
2	78	75	78

Did You Know?

Relational algebra received little attention outside of pure mathematics until the publication of E.F. Codd's relational model of data in 1970.

5.4 Project Operation

The projection operation is used to select data of particular attributes (columns) from a single relation and discards the other columns. The project operator is used for this purpose. The general form of the projection operation is:

PROJECT <table-name> OVER <(col_1, col_3, ... col_n)>

[GIVING <new-table-name>]

Where

table-name	Specifies the name of existing table (relation) you want to select tuples from.
col_1, col_3, ...col_n	Specifies the name of attributes or columns from which you want to extract data.
new-table-name	Specifies the new table in which you want to store the result. The new table has the selected attributes only. The use of new table is optional. If it is omitted, then only the result is displayed, not stored in a table.

Symbolically, the project operator is denoted by Π, so the general form of projection operation is written as:
Π <attribute list>(R)

For example, to extract 'Name' and 'City' attributes from "Student" table, the projection operation is:

PROJECT student OVER (Name, City) GIVING temp

Or symbolically

Π Name, City (Student)

The Table5.5 shows the actual "Student table", while the Table 5.6 shows the result of the statement.

Table 5.5: "Student table"

Roll_No	Name	City
1	S	L
2	K	P
3	F	Q
4	H	L

Table 5.6: Result

Name	City
S	L
K	P
F	Q

H	L

Projection operation can also be used to change the order of attributes in a relation. The resulting relation has the attributes in the same order as specified in the projection operation.

5.4.1 Combining Selection and Projection

You can combine the selection and projection operations together. For example, to extract the columns Roll-No & Name, and retrieve those tuples that have value 'L' in the attribute "City" of 'Student' relation, the relational algebra expression is written as:

Π Roll-No, Name (δ City = 'L' (Student)))

Table 5.7: "Student table"

Roll_No	Name	City
1	S	L
2	K	P
3	F	Q

Table 5.8: Result

Roll_No	Name
1	S
4	H

Self Assessment Questions

1. Which of the following relational algebra operations do not require the participating tables to be union-compatible?

(a) Union (b) Intersection
(c) Difference (d) Join

2. Which of the following is not a property of transactions?

(a) Atomicity (b) Concurrency
(c) Isolation (d) Durability

3. Relational Algebra does not have

(a) Selection operator (b) Projection operator.
(c) Aggregation operators (d) Division operator

5.5 Join Operation

Join operations are probably the most common relational operations. Certainly they are fundamental to the model it would not be feasible to decompose data into multiple relations were it not possible to recombine it as necessary. This is precisely what a join operator does; it combines record sets based on the comparison of one or more common fields.

The join operation is used to combine related tuples from two relations. This operation is very important in relational database because it allows to process relationship among relations.

The join operation is the combination of the product, selection and projection operations. The join operation on two relations is performed as follows:

- Product operation is performed on two relations.

- Selection operation is performed to eliminate duplicate tuples by the join criteria or condition.
- Projection operation is performed to remove some attributes.

It must be noted that in a join operation, only combination of tuples are included in the result that satisfy the join condition, whereas in the product operation all combination of tuples is included in the result. The join condition is specified on attributes of the two relations and this condition is tested for each combination of tuples. If R and S are two relations then the general form of the join condition is:

<condition 1> AND <condition2> AND <condition3> .

Where each condition is of form A_i □ Bj, Ai is an attribute of R, while B_j is an attribute of S. A_i and B_j have the same domain. The q is one of the relational operators such as <, < =, >,< =, =, ≠

The join operation is denoted by S. The general form of a join operation on two relations R and S is:

R <join condition> S

There are various forms of join operations. These are:

(i) Theta join

(ii) Equi-join

(iii) Natural join

5.5.1 Theta Join (or □- Join)

The most general join operation is Theta Join (or □ join). The Theta Join is defined as the result of performing a selection operation using comparison operator Theta (q), on the product. In other words, a join operation with a general join condition using Theta (q) operator is called a Theta Join.

For example, if you want only those tuples of the product of two relations 'Student1' and 'Student2' whose value is 'L' in the attribute 'City', Theta join operation is written as:

Student1 TIMES student2 WHERE City ='L'

This is equivalent to:

Student1 TIMES student2 GIVING temp

SELECT temp WHERE City ='L'

5.5.2 Equi - Join

The join operation may contain the join condition, which uses the relational operators such as; <, <=, >, >=, =, ≠ . Equi-Join is the join in which the joining, condition is based on the equality (i.e., only the relational operator '='is used). It means that if Theta (q) join uses only '='relational operator, then q-join becomes the equi-join. In the above example of Theta Join, only the '='sign is used in the join condition, which also represents the Equi-Join. Therefore, the main difference between Theta Join and Equi-Join is that a Theta Join may use one of the <, <=, >,>=, =, ≠ relational operators, while Equi-Join can only use the '='relational operator.

5.5.3 Natural Join

The natural join is similar to Equi-Join, except that one of the duplicated columns is eliminated in the resultant relation. This is the most common form of the join operation. The standard definition of Natural Join requires that the two join attributes have the same name in both relations.

Natural join is a binary operator that is written as (R* S) where R and S are relations. The result of the natural join is the set of all combinations of tuples in R and S that are equal on their common attribute names. In this only one column along attributes having same name is retained

If the tuples r and s are successfully paired in the join R ? S, then the result of the pairing is a tuple, called the joined tuple, with one component for each of the attributes in the union of the schemas of R and S. The joined tuple agrees with tuple r in each attribute in the schema of R, and it agrees with s in each attribute in the

schema of S. Since r and s are successfully paired, the joined tuple is able to agree with both these tuples on the attributes they have in common.

Table 5.9: Employee

E#	Name	D#
101	Vineet	D1
103	Rahul	D1
104	Amit	D2

Table 5.10: Department

D#	Dname
D1	Sales
D2	Accounts

Table 5.11: The natural join of employee and department

E#	Name	D#	DName
101	Vineet	D1	Sales
103	Rahul	D1	Sales
104	Amit	D2	Accunts

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Retrieve the name and address of all the employees who work for CSE dept CSE-Dept (Dname='CSE'(DEPARTMENT) CSE-Emp (CSE-Dept Dno=Dno EMPLOYEE) Result FN, Add(CSE-Emp).

……..……

……..……

……...

5.6 Division Operation

The Division opertion, denoted÷, is suited to queries that include the phrase "for all''. Suppose we want to find all the customers who have an account at all the branches located in Brooklyn.

We can obtain the names of all branches located in Brooklyn by

$$r_1 = \Pi_{bname}(\sigma_{bcity="Brooklyn"}(branch))$$

The result relation for this expression appears in Figure

We can find all (customer-name, branch-name) pairs for which the customer has an account at a branch by writing

$$r_2 = \Pi_{cname,bname}(deposit)$$

Table 5.12 shows the result relation for this expression.

Now, we need to find customers who appear in r_2 with every branch name in r_1. The operation that provides exactly those customers is the divide operation.

Table 5.12: Result table

Customer-name	Branch-name
Johnson	Downtown
Smith	Mianus
Hayes	Perryridge
Samar	Round Hill
Williams	Perryridge
Lindsay	Redwood
Samar	Brighton
	Brighton

Result of $\Pi_{\text{Customer-name, branch-name}}$ *(depositor? account).*

We formulate the query by writing

$\Pi_{\text{Customer-name, branch-name}}$ *(depositor? account)* $\div \Pi_{\text{branch-name}}(\sigma_{\text{branch-city=Booklyn}}(\text{branch}))$

The result of this expression is a relation that has the schema (Customer-name) and that contains the tuple (Samar).

Formally, let r(R) and s(S) be relations, and let S $\subseteq$R; that is every attribute of schema S is also in schema R. The relation r÷ s is a relation on schema R – S that is, on the schema containing all attributes of schema R that are not in schema S. A tuple t is in r÷ s if and only if both of two conditions hold:

1. t is in $\Pi_{R-s}(r)$
2. For every tuple t_s in s, there is a tuple t_r in r satisfying both of the following:

a. $t_r[s] = t_s[s]$

b. $t_r [R - S] = t_s$

It may surprise you to discover that, given a division operation and the schemas of the relations, we can, in fact, define the division operation in terms of the fundamental operations. Let r (R) and s(S) be given, with S Í R:

$r \div s= \Pi_{R-S} (r) – \Pi_{R-S} ((\Pi_{R-S} (r) \times s) – \Pi_{R-S,S}(r))$

To see that this expression Π is true, we observe that $\Pi_{R-S} (r)$ gives us all tuples t that satisfy the first condition of the definition of division. The expression on the right side of the set difference operator,

$\Pi_{R-S} ((\Pi_{R-S}(r) \times s) – \Pi_{R-S,S}(r))$,

serves to eliminate those tuples that fail to satisfy the second condition of the definition of division. Let us see how it does so. Consider $\Pi_{R-S} (r) \times s$. This relation is on schema R, and pairs every,tuple in $\Pi_{R-S} (r)$ with every tuple in s. The expression $\Pi_{R-S,S} (r)$ merely reorders the attributes of r.

Thus, $(\Pi_{R-S} (r) \times s) – \Pi_{R-S,S} (r)$ gives us those pairs of tuples from $\Pi_{R-S} (r)$ and s that do not appear in r. If a tuple t_j is in

ΠR-S $((\Pi_{R-S} (r) \times s) – \Pi_{R-S,S} (r))$,

then there is some tuple t_s in s that does not combine with tuple t_j to form a tuple in r. Thus, t_j holds a value for attributes R - S that does not appear in r ÷ s. It is these values that we eliminate from $\Pi_{R-S} (r)$.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Find the name of the employees who work on all projects controlled By dept no5 Dep5-Proj Pno(Dno=5(PROJECT)) //project number Emp-Proj ESSN, Pno(WORKS_ON) Result_Emp_SSNS Emp_Proj /Dept5_Proj Result FN, LN(Resul_Emp_SSNS).

……..……

……..……

……...

5.7 Cross Product Operation

The Cartesian product (or product as shortcut or cross product or cross join) is denoted by '×'. The product of two relations A and B is denoted by A × B (A TIMES B). The product of two relations would be a set of ordered pairs of tuples. In other words, a product of two relations is the concatenation of every tuple of first relation with every tuple of a second relation:

Suppose the relation A has 'm' tuples and relation B has 'n' tuples. The product of relation A and relation B has m × n tuples and whose columns are the columns of A followed by the columns of B,

For example, the product of two relations "Student1" and "Student2" is given below. The "Student1" has 2 tuples and 3 columns, while "Student1" has 3 tuples and two columns. The resulting relation has 6 tuples (2 × 3 = 6) and 5 columns (3 + 2 = 5) as shown in figure below.

Table 5.13: "Student1 Table"

Roll_No	Name	City
1	S	L
2	K	P

Table 5.14: "Student2 Table"

Roll_No	Marks
1	789
3	589
4	687

Table 5.15: Result of Product Operation

Roll_No	Name	City	Roll_No	Marks
1	S	L	1	789
1	S	L	3	589
1	S	L	4	687
2	k	P	1	789
2	K	P	3	589
2	k	P	4	687

Self Assessment Questions

4. The operation which is not considered a basic operation of relational algebra is

(a) Join. (b) Selection.

(c) Union. (d) Cross product

5. Relational Algebra is

(a)Data Definition Language (b) Meta Language

(c)Procedural query Language (d) None of these

5.8 Set Operators

The three most common operations on sets are union intersection; and difference. We assume the reader is familiar with these operations. Which are defined as follows on arbitrary sets R and S:

R ∪ S, the union of R and S; is the set of elements that are in R or S or both. An element appears only once in the union even if it is present in both R and S.

R ∩S, the intersection of R and S. is the set of elelilents that are in both R and S.

R - S, the difference of R and S, is the set of elements that are in R but not in S. Note that R - S is different from S - R; the latter is the set of elements that are in S but not in R.

When we apply these operations to relations, we need to put some conditions on R and S:

1. R and S must have schemas with identical sets of attributes, and the types (domains) for each attribute must be the same in R and S.

2. Before me compute the set-theoretic union, intersection, or difference of sets of tuples, the columns of R and S must be ordered so that the order of attributes is the same for both relations.

Sometimes we would like to take the union, intersection, or difference of relations that have the same number of attributes, with corresponding domains, but that use different names for their attributes. If so, we may use the renaming operator to change the schema of one or both relations and give them the same set of attributes.

name	*address*	*gender*	*birthdate*
Carrie Fisher	123 Maple St., Hollywood	F	9/9/99
Mark Hamill	456 Oak Rd., Brentwood	M	8/8/88

Relation R

name	*address*	*gender*	*birthdate*
Carrie Fisher	123 Maple St., Hollywood	F	9/9/99
Harrison Ford	789 Palm Dr., Beverly Hills	M	7/7/77

Relation S

Figure 5.16: TIYO relations.

5.8.1 The Union Operation

The union of two relations A and B is denoted by A È B (or A + B). It is a relation that includes all tuples that are either in relation A or in relation B or both in relations A and B, Duplicate tuples are eliminated. The UNION operator is used for this purpose.

For example, if A = {1, 5, 10,15} and B - {5, 15, 20, 25} then A ∪ B will be {1, 5, 10, 15, 20, 25}

Actually, the union of two relations is formed by adding the tuples from first relation to those of a second relation and a third resultant relation is produced. The two relations must be type-compatible. This means that they must have the same number of attributes, and the attributes in corresponding columns must have the same domain. For example, the fourth column in the first table must have the same domain as the fourth column in the second table, although the column names may be different. The resulting new table will be produced with the same structure as the two original tables.

Suppose two tables “Student1” and “Student2” are type-compatible (or union compatible) as shown below. The result of union operation is also shown in table 5.17. The union is a commutative operation, i.e.

A ÈB = B È A

Table 5.17: "Student1 Table"

Roll_No	Name	City
1	S	L
2	K	P
3	F	Q
4	H	L

Table 5.18: Table: "Student2 Table"

Roll_No	Name	City
1	S	L
5	A	B
4	H	L
6	M	M

Table 5.18: Result of Union Operation

Roll_No	Name	City
1	S	L
2	K	P
3	F	Q
4	H	L
5	A	B
6	M	M

5.8.2 The Intersection Operation

The intersection of two relations A and B is denoted by A ∩ B. The intersection of two relations returns a relation that includes all those common tuples. That are both in the first relation and second relation, the two relations must be type-compatible. The Intersection operator is used for this purpose.

Suppose two tables "Student1" and "Student2" are type-compatible as shown in Table 5.20. The result of intersection operation is also shown. Like union, intersection is also commutative, i.e. A ∩ B = B ∩ A

Table 5.20: "Student1 Table"

Roll_No	Name	City
1	S	L
2	K	P
3	F	Q
4	H	L

Table 5.21: "Student1 Table"

Roll_No	Name	City
1	S	L
5	A	B
4	H	L
6	M	M

Table 5.22: Result of Intersection Operation

Roll_No	Name	City
1	S	L
4	H	L

5.8.3 The Difference Operation

The difference of two relations A and B is denoted by A – B. The difference of two relations returns a relation that includes all those tuples that occur in the first relation but not in the second.

Suppose two tables, "Student1" and "Student2" are type-compatible as shown in Table 5.23. The result of difference operation is also shown. It must be noted that the result of difference operation includes all those tuples that are in the "Student1" table but not in the "Student2" table. The difference operation is not commutative, i.e.

A–B B-A

Table 5.23: "Student1 Table"

Roll_No	Name	City
1	S	L
2	K	P
3	F	Q
4	H	L

Table 5.24: "Student1 Table"

Roll_No	Name	City
1	S	L
5	A	B
4	H	L
6	M	M

Table 5.25: Result of Difference Operation

Roll_No	Name	City
1	S	L
4	H	L

Exercise: Check Your Progress 3

Note: i) Use the space below for your answer.

Ex1: Consider the following tables :

customer (c_id, c_name, c_address)

branch (br_name, br_city, assets)

account (c_id, act_no, br_name, balance)

(i) Customers who have accounts in all branches of Bhopal.

(ii) Customers who have accounts in branches with assets more than 50 crores.

……..……

……..……

……...

5.9 Summary

- The union of two relations is formed by adding the tuples from first relation to those of a second relation and a third resultant relation is produced.
- The intersection of two relations returns a relation that includes all those common tuples.
- The difference of two relations returns a relation that includes all those tuples that occur in the first relation but not in the second.
- The projection operation is used to select data of particular attributes (columns) from a single relation and discards the other columns.
- The Selection Operation is used to select a subset of tuples (or horizontal subset or rows) from a single relation that satisfy the given selection condition.

5.10 Keywords

Cartesian product or Cross Product (×): The Cartesian product of two relations is the concatenation of tuples belonging to the two relations and consisting of all possible combination of the tuples. $R = P \times Q$

Join (?): Allows the combining of two relations to form a single new relation.

Selection (σ): Selects only some of the tuples, those satisfy given criteria, from the relation. It yields a horizontal subset of a given relation. $R = \sigma_B(P)$

Natural join: It is that one of the duplicated columns is eliminated in the resultant relation.

Union (∪): Selects tuples that are in either P or Q or in both of them. The duplicate tuples are eliminated. $R = P \cup Q$

5.11 Review Questions

1. Describe the division and the join operation of the relational algebra. Give an example for each. Express each of them in terms of the basic operations.
2. List the operations of relational algebra and purpose of each.
3. What is the difference between select and project operation? Explain with example.
4. What is relational algebra in database System?
5. Define the set operators.
6. Discuss the cross product operation.
7. What is the division operation?
8. What is the difference between natural join and equi join?
9. Discuss the combining of selection and projection operations.
10. Discuss the select operation.

Answers for Self Assessment Questions

1. (d) 2. (b) 3. (c) 4. (a) 5. (c)

5.12 Further Studies

1. Theory and Practice of Relational Databases By Stefan Stanczyk, Bob Champion, Richard Leyton

6
Relational Calculus

STRUCTURE

6.0 Objectives

6.1 Introduction

6.2 Tuple Relational Calculus

6.3 Domain Relational Calculus

6.4 Comparison of TRC, DRC and RA

6.5 Summary

6.6 Keywords

6.7 Review Questions

6.8 Further Studies

6.0 Objectives

After studying this chapter, you will be able to:

- Explain the tuple relational calculus.
- Discuss the domain relational calculus.
- Comparison of TRC, DRC and RA.

6.1 Introduction

The relational calculus is a non-procedural query language. R-C uses both languages are logically differentiation. "The relational calculus is a non-procedural a different approach than relational algebra, but query language whereas relational algebra is a procedural query language."

In non-procedural query language, the user is concerned with the details of how to obtain the end results. Whereas in procedural query language, we define each step in order to obtain the end result. In relational calculus, a query is expressed as a formula consisting variables. There is no mechanism to specify a how formula should be evaluated. Relational calculus is of two types

(i) Tuple Relational Calculus

(ii) Domain Relational Calculus

In this chapter we are going to discuss about both the relational tuple and domain calculus. We will also discuss about the example queries of this calculus.

6.2 Tuple Relational Calculus

A tuple variable is a variable that takes on tuples of a particular relation schema as values. That is, every value assigned to a given tuple variable has the same number and type of field. A tuple relational calculus query has

the form {T | p(T) }, where T is a tuple variable and p(T) denotes a formula that describes T; we will shortly Define formulas and queries rigorously. The result of this query is the set of all tuples t for which the formula p (T) evaluates to true with T = t. The language for writing formulas p (T) is thus at the heart of TRC and is essentially a simple subset of first-order logic. As a simple example, consider the following query.

(Q) Find all sailors with a rating above 7.

{S | S ∈ Sailors S.rating > 7}

When this query is evaluated on an instance of the Sailors relation, the tuple variable S is instantiated successively with each tuple, and the test S.rating>7 are applied. The answer contains those instances of S that pass this test. On instance S3 of Sailors, the answer contains Sailors tuples with sid 31, 32, 58, 71, and 74.

Did You Know?

Tuple calculus is a calculus that was introduced by Edgar F. Codd as part of the relational model, in order to provide a declarative database-query language for this data model.

6.2.1 Operators used in TRC

We now define these concepts formally, beginning with the notion of a formula. Let Rel be a relation name, R and S be tuple variables, an attribute of R, and b an attribute of S. Let op denote an operator in the set {<, >,=,£³, ≠}. An atomic formula is one of the following:

- R ∈ Rel
- R.a op S dene.b
- R.a op constant, or constant op R.a

A formula is recursively defined to be one of the following, where p and q are themselves formulas, and p(R) denotes a formula in which the variable R appears:

- any atomic formula
- ¬p, p q, p q, or pÞq
- ∃ R(p(R)), where R is a tuple variable
- ∀ R(p(R)), where R is a tuple variable

In the last two clauses above, the quantifiers ∃ and ∀ are said to bind the variable R. A variable is said to be free in a formula or sub-formula (a formula contained in a larger formula) if the (sub) formula does not contain an occurrence of a quantifier that binds it.

We observe that every variable in a TRC formula appears in a sub-formula that is atomic, and every relation schema species a domain for each field; this observation ensures that each variable in a TRC formula has a well-defined domain from which values for the variable are drawn. That is, each variable has a well-defined type, in the programming language sense. Informally, an atomic formula R Rel gives R the type of tuples in Rel, and comparisons such as R.a op S.b and R.a op constant induce type restrictions on the field R.a. If a variable R does not appear in an atomic formula of the form R ϵ Rel (i.e., it appears only in atomic formulas that are comparisons), we will follow the convention that the type of R is a tuple whose fields include all (and only) fields of R that appear in the formula.

We will not define types of variables formally, but the type of a variable should be clear in most cases, and the important point to note is that comparisons of values having different types should always fail. (In discussions of relational calculus, the simplifying assumption is often made that there is a single domain of constants and that this is the domain associated with each field of each relation.)

A TRC query is defined to be expression of the form {T | p (T)}, where T is the only free variable in the formula p.

Formal definition

A tuple relational calculus expression is of the form {t/P(t)} where P is a formula, several tuple may appear in a

formula a tuple variable is said to be a free variable unless it is quantified by a ∃ or ∀. Thus, in t ∈ loan ∧ ∃ s ∈ cust (t [branch_name] = s[branch_name])

t is a free variable. Tuple variable s is said to be bound variable.

A tuple relational calculus formula is built up out of atoms. An atom has one of the following forms.

- S ∈ r, where s is a tuple variable and r is a relation.
- S[x] Θ u[y] where s, u are tuple variable
  - x is an attribute on which s in defined
  - *y* is an attribute an which *u* is defined
  - Θ is a comparison operator (<, ≤, =, ≠, >, ≥)
- S[x] Θ c, where s, x, e are as above, and c is a constant in the domain of attribute x.

We build up formulae from atoms by using the following rules:

- An atom is a formula
- If P_1 is a formula, then so are, ¬ P_1 and (P_1).
- If P_1 and P_2 are formulae, then so are P_1 v P_2, P_1 ∧ P_2 and P_1 Þ P_2.
- If $P_1(s)$ is a formula containing a free tuple variable s, and r is a relation, then ∃ s ∈ r *($P_1(s)$)* and ∀ s ∈ r *($P_1(s)$)* are also formulae.

We can write equivalent expression that is not identical in appearance. In the tuple relational calculus these equivalences include the following three rules.

1. P_1 ∧ P_2 is equivalent to ¬ (¬ (P_1) v ¬ (P_2)).
2. *t E r* (P_1(t)) is equivalent to ¬ ∃ *t* E r (¬ P_1(t))
3. P_1 Þ P_2 is equivalent to ¬ (P_1) v P_2.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Write a TRC query for find, the branch-name, loan number, and amount for loans of over 1700.

……..……

……..……

……...

6.2.2 Example queries using TRC

What does a TRC query mean? More precisely, what is the set of answer tuples for a given TRC query? The answer to a TRC query {T | p(T)}, as we noted earlier, is the set of all tuples t for which the formula p(T) evaluates to true with variable T assigned the tuple value t. To complete this definition, we must state which assignments of tuple values to the free variables in a formula make the formula evaluate to true.

A query is evaluated on a given instance of the database. Let each free variable in a formula F be bound to a tuple value. For the given assignment of tuples to variables, with respect to the given database instance, F evaluates to (or simply 'is') true if one of the following holds:

- F is an atomic formula R ∈ Rel, and R is assigned a tuple in the instance of relation Rel.
- F is a comparison r.a op s.b, r.a op constant, or constant op r.a and the tuples assigned to r and s have field values r.a and s.b that make the comparison true.
- F is of the form ¬p, and p is not true; or of the form p ∧ q, and both p and q are true; or of the form p Úq, and one of them is true, or of the form p Þ q and q is true whenever p is true.
- F is of the form ∃r (p(r)), and there is some assignment of tuples to the free variables in p(r), including the variable r, that makes the formula p(r) true.

- F is of the form ? r(p(r)), and there is some assignment of tuples to the free variables in p(r) that makes the formula p(r) true no matter what tuple is assigned to r.

We now illustrate the calculus through several examples, using the instances B1 of Boats, R2 of Reserves, and S3 of Sailors showed in Figures 6.2, 6.3, and 6.4. We will use parentheses as needed to make our formulas unambiguous. Often, a formula p(R) includes a condition R Rel and the meaning of the phrases some tuple R and for all tuples R is intuitive. We will use the notation R Î Rel(p(R)) for R(R Rel p(R)).
Similarly, we use the notation ∀ R Î Rel(p(R)) for ∀ R(R ∈ Rel ⇒ p(R)).
(Q) Find the names and ages of sailors with a rating above 7.
{P | ∃ S ∈ Sailors (S.rating > 7 P.name= S.sname P.age= S.age)}
This query illustrates a useful convention: P is considered to be a tuple variable with exactly two fields, which are called name and age, because these are the only fields of
P that are mentioned and P do not range over any of the relations in the query; that is, there is no sub formula of the form P Î Relname. The result of this query is a relation with two fields, name and age. The atomic formulas P.name = S.sname and P.age = S.age give values to the fields of an answer tuple P. On instances B1, R2, and S3, the answer is the set of tuples.
(Q) Find the sailor name, boat id, and reservation date for each reservation.
{P | 'R Î Reserves 'S Î Sailors (R.sid = S.sid P.bid= R.bid P.day= R.day P.sname= S.sname)}
For each Reserves tuple, we look for a tuple in Sailors with the same sid. Given a pair of such tuples, we construct an answer tuple P with fields name, bid, and day by copying the corresponding fields from these two tuples. This query illustrates how we can combine values from different relations in each answer tuple. The answer to this query on instances B1, RÎ, and S3 is shown in Figure 6.1.

sname	*bid*	*day*
Dustin	101	10/10/98
Dustin	102	10/10/98
Dustin	103	10/8/98
Dustin	104	10/7/98
Lubber	102	11/10/98
Lubber	103	11/6/98
Lubber	104	11/12/98
Horatio	101	9/5/98
Horatio	102	9/8/98
Horatio	103	9/8/98

Figure 6.1: Answers to Query.

(Q) Find the names of sailors who have reserved boat 103.

{P | 'S Î Sailors ∃ R ∈ Reserves (R.sid = S.sid R.bid = 103 P.sname = S.sname)}
This query can be read as follows: "Retrieve all sailor tuples for which there exists a
tuple in Reserves, having the same value in the sideld, and with bid = 103." That is, for each sailor tuple, we look for a tuple in Reserves that shows that this sailor has reserved boat 103. The answer tuple P contains just one field, sname.
(Q) Find the names of sailors who have reserved a red boat.
{P | ∃ S ∈ Sailors ∃ R ∈ Reserves (R.sid = S.sid P.sname = S.sname ∃ B ∈ Boats (B.bid = R.bid B.color = 'red'))}

This query can be read as follows: “Retrieve all sailor tuples S for which there exist tuples R in Reserves and B in Boats such that S.sid = R.sid, R.bid = B.bid and B.color = ‘red’.” Another way to write this query, which corresponds more closely to this reading, is as follows:
{P | ∃ S ∈ Sailors ∃ R ∈ Reserves ∃ B ∈ Boats
(R.sid = S.sid □ B.bid = R.bid □ B.color =‘red’ □ P.sname = S.sname)}

(Q) Find the names of sailors who have reserved at least two boats.
{P | ∃ S ∈ Sailors ∃ R1 ∈ Reserves ∃ R∈ ∈ Reserves
(S.sid = R1.sid □ R1.sid = R∈ .sid □ R1.bid 6= R∈ .bid □ P.sname = S.sname)}
Contrast this query with the algebra version and see how much simpler the calculus version is. In ipart, this di erence is due to the cumbersome renaming of felds in the algebra version, but the calculus version really is simpler.
(Q) Find the names of sailors who have reserved all boats.
{P | ∃ S ∈ Sailors ∀ B ∈ Boats (∃ R ∈ Reserves (S.sid = R.sid □ R.bid = B.bid □ P.sname = S.sname))}
This query was expressed using the division operator in relational algebra. Notice how easily it is expressed in the calculus. The calculus query directly reflects how we might express the query in English. “Find sailors S such that for all boats B there is Reserves tuple showing that sailor S has reserved boat B.”
(Q) Find sailors who have reserved all red boats.
{S | S ∈ Sailors □ ∀ B ∈ Boats
(B.color = ‘red’Þ (∃ R ∈ Reserves (S.sid = R.sid □ R.bid = B.bid)))}
This query can be read as follows: For each candidate (sailor), if a boat is red, the sailor must have reserved it. That is, for a candidate sailor, a boat being red must imply the sailor having reserved it. Observe that since we can return an entire sailor
tuple as the answer instead of just the sailor’s name, we have avoided introducing a new free variable (e.g., the variable P in the previous example) to hold the answer values. On instances B1, R2, and S3, the answer contains the Sailors tuples with sids 22 and 31.
We can write this query without using implication, by observing that an expression of the form p Þq is logically equivalent to: ¬p □ q: {S | S∈ Sailors □ ∀ B ∈ Boats (B.color ¹‘red’ Ú(∃ R ∈ Reserves(S.sid = R.sid □ R.bid= B.bid)))}
This query should be read as follows: “Find sailors S such that for all boats B, either the boat is not red or Reserves tuple shows that sailor S has reserved boat B.”

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 6.2: An Instance S3 of Sailors.

sid	*bid*	*day*
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Figure 6.3: An Instance R2 of Reserves.

bid	*bname*	*color*
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Figure 6.4: An Instance B1 of Boats.

Caution

Take care while declaring TRC queries do not use Aggregations, Groupings, and Orderings function, it may create problem running query.

6.3 Domain Relational Calculus

A domain variable is a variable that ranges over the values in the domain of some attribute (e.g., the variable can be assigned an integer if it appears in an attribute whose domain is the set of integers). A DRC query has the form {(x_1; x_2; : : : ; x_n) p((x_1; x_2; : : : ; x_n)) }, where each x_i is either a domain variable or a constant and p((x_1; x_2; : : : ; x_n)) denotes a DRC formula whose only free variables are the variables among the xi; 1 £ i £n. The result of this query is the set of all tuples (x_1; x_2; : : : ; x_n) for which the formula evaluates to true.

A DRC formula is defined in a manner that is very similar to the definition of a TRC formula. The main difference is that the variables are now domain variables. Let op denote an operator in the set {<; >;=;£;³; ≠} and let X and Y be domain variables. An atomic formula in DRC is one of the following: (x_1; x_2; : : : ; x_n) Î Rel, where Rel is a relation with n attributes; each x_i, 1 £ i £ n is either a variable or a constant.

X op Y

X op constant or constant op X

A formula is recursively defined to be one of the following, where p and q are themselves formulas, and p(X) denotes a formula in which the variable X appears:

- any atomic formula
- ¬p, p □ q, p Ú q, or p Þ q
- 'X (p(X)), where X is a domain variable
- ?X (p(X)), where X is a domain variable

The reader is invited to compare this definition with the definition of TRC formulas and see how closely these two definitions correspond. We will not define the semantics of DRC formulas formally; this is left as an exercise for the reader.

Domain relational calculus formulae are also built from atom. A T.R.C. is expressed as

{t/P(t)} where (F) is an expression / formula consisting of t {t} is set of tuple variables.

Did You Know?

In computer science, domain relational calculus (DRC) is a calculus that was introduced by Michel Lacroix and Alain Pirotte as a declarative database query language for the relational data model.

6.3.1 Operators used in DRC

Ú (OR)

 (AND)

¬ (NOT)

→ IMPLICANT

A second form of relational calculus, called domain; relational calculus, uses domain variable that take values form an attributes domain rather than values for an entire tuple, the domain relational calculus however is closely related to the tuple relational calculus.

An expression in the domain relational calculus is of the form

{<x_1, x_2,…… ,x_n> | P (x_l, x_2, ……., x_n)}

where x_l, x_2,……., x_n represents domain variable. P represents a formula composed of atoms: An atom in the domain relational calculus has one of the following forms.

- <x_1, x_2, x_3, ……x_n> r where *r* is a relation on n attributes and x_l, x_2,……., x_n are domain variables or domain constant.
- x Θ *y*, where x and *y* are domain variables and Θ is a compassion operator.
- x Θ c, where c is a constant in the domain of the attribute for which *x* is a domain variable.

We build up formulae from atoms by using the following rules.

- An atom is a formula.
- If P_l is a formula then so is ¬ P_l and (P_l).
- If P_l and P_2 are formulae, the so are P_l v P_2, P_1 P_2 and P_1 Þ P_2.
- If P_l(x) is a formula in *x*, where *x* is a free domain variable, then

  x(P_l(x)) and x *(P_1(x))* are also formulae.

As rotational shorthand, we write:

 a, b, c (P(a, b, c)) for a(b(c(P(a, b, c))))

Example: Find the loan_no, branch name and amount for loans of over 1200.

{< 1, b, a> | < l, b, a> E loan a> 1200}

Find all loan_no for loans with an amount > 1200

{<l>/3 b,a(<l,b,a>Eloan n a>1200)}

Find emp_name, emp_no. designation, salary, Dept_n for employees salary is over 10000.

{<e,n,d,s,d n>(<e,n,d,s,d n) Emp. s> 10000}.

Example:

Show that every expression of the domain calculus has a tuple calculus equivalent.

Solution:

The Domain Relational Calculus: An expression of the domain calculus is of the form

{X_1, X_2, ……, X_n/COND(X_1, X_2,……..,X_n, X_{n+1}, X_{n+2},……., X_{n+m})}

where X_1, X_2, ……., X_n, X_{n+1}, X_{n+2} ……..X_{n+m} are domain variables that range over domains and COND is a condition or formula of the domain relational calculus.

Retrieve the birthdates and address of the employee whose name is ‘John B. Smith’.

{uv | (q) (r) (s) (t) (w) (x) (y) (z) (EMPLOYEE (qrstuvwxyz) and q = 'John' and r = 'B' and s='smith')}

Tuple Variable and Range Relations

A simple tuple relational calculus query is of the form

$$\{t_l \text{ COND } (t)\}$$

where t is a tuple variable and COND(t) is conditional expression involving t. The result of such a query is the set of all tuples t that satisfy COND (t).

A general expression of the tuple relational calculus is of the form

$$\{t_1.A_1,t_2.A_2,\ldots\ldots\ldots t_n.A_n| \text{COND}(t_1, t_2\ldots\ldots t_n.t_{n+1},t_{n+2}) \ldots,t_{n+m})\}$$

where t_l, t_2 ... t_n, t_n+l ..., t_{n+m} are tuple variables, each A_i is an attribute on the relation of which t_i ranges and COND is a condition of formula of the tuple relational calculus.

Retrieve the birthdate and address of the employee whose name is 'John B. Smith' {t.BDATAE, t.ADDREE| EMPLOYEE(t) and t. FNAME='John' and t.MINIT='B' and t, *L*. NAME ='SMITH']

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Prepare a DRC query for find emp_name, emp_no. designation, salary, Dept_n for employees salary is over 20000.

……..……

……..……

……...

6.3.2 Examples of DRC Queries

We now illustrate DRC through several examples. The reader is invited to compare these with the TRC versions.

Q Find all sailors with a rating above 7.

$$\{\langle I, N, T, A\rangle \mid \langle I, N, T, A\rangle \in Sailors \wedge T > 7\}$$

This differs from the TRC version in giving each attribute a (variable) name. The condition (I N; T, A) ∈ Sailors ensures that the domain variables I, N, T, and A are restricted to be field of the same tuple. In comparison with the TRC query, we can say T > 7 instead of S.rating > 7, but we must specify the tuple (I, N, T, A) in the result, rather than just S.

Q Find the names of sailors who have reserved boat 103.

$$\{\langle N\rangle \mid \exists I, T, A(\langle I, N, T, A\rangle \in Sailors$$
$$\wedge \exists Ir, Br, D(\langle Ir, Br, D\rangle \in Reserves \wedge Ir = I \wedge Br = 103))\}$$

Notice that only the sname field is retained in the answer and that only N is a free variable. We use the notation ' Ir,Br,D(….) as shorthand for 'Ir('Br('D(….))). Very often; all the quantified variables appear in a single relation, as in this example. An even more compact notation in this case is ' (Ir,Br,Br,D) Î Reserves. With this notation, which we will use henceforth, the above query would be as follows:

$$\{\langle N\rangle \mid \exists I, T, A(\langle I, N, T, A\rangle \in Sailors$$
$$\wedge \exists \langle Ir, Br, D\rangle \in Reserves(Ir = I \wedge Br = 103))\}$$

The comparison with the corresponding TRC formula should now be straightforward. This query can also be written as follows; notice the repetition of variable *I* and the use of the constant 103:

$$\{\langle N\rangle \mid \exists I,T,A(\langle I,N,T,A\rangle \in Sailors \wedge \exists D(\langle I,103,D\rangle \in Reserves))\}$$

Find the names of sailors who have reserved a red boat.

$$\{\langle N\rangle \mid \exists I,T,A(\langle I,N,T,A\rangle \in Sailors \wedge \exists\langle I,Br,D\rangle \in Reserves \wedge \exists\langle Br,BN,'red'\rangle \in Boats)\}$$

(Q) Find the names of sailors who have reserved at least two boats.

$$\{\langle N\rangle \mid \exists I,T,A(\langle I,N,T,A\rangle \in Sailors \wedge \exists Br1,Br2,D1,D2(\langle I,Br1,D1\rangle \in Reserves \wedge \langle I,Br2,D2\rangle \in Reserves \wedge Br1 \neq Br2))\}$$

Notice how the repeated use of variable *I* ensure that the same sailor has reserved both the boats in question

*(Q)*Find the names of sailors who have reserved all boats.

$$\{\langle N\rangle \mid \exists I,T,A(\langle I,N,T,A\rangle \in Sailors \wedge \forall B,BN,C(\neg(\langle B,BN,C\rangle \in Boats) \vee (\exists\langle Ir,Br,D\rangle \in Reserves(I=Ir \wedge Br=B))))\}$$

This query can be read as follows: Find all values of *N* such that there is some tuple (I, N, T, A) in Sailors satisfying the following condition: for every (B, BN, C) , either this is not a tuple in Boats or there is some tuple (Ir, Br, D) in Reserves that proves that Sailor *I* has reserved boat *B*.

TheÚ quantifier allows the domain variables *B*, *BN*, and *C* to range over all values in their respective attribute domains, and the pattern '¬ ((B, BN, C)ÎBoats)Ú' is necessary to restrict attention to those values that appear in tuples of Boats. This pattern is common in DRC formulas, and the notation ?(B, BN, C) Î Boats can be used as shorthand instead. This is similar to the notation introduced earlier for *9*. With this notation the query would be written as follows:

$$\{\langle N\rangle \mid \exists I,T,A(\langle I,N,T,A\rangle \in Sailors \wedge \forall\langle B,BN,C\rangle \in Boats (\exists\langle Ir,Br,D\rangle \in Reserves(I=Ir \wedge Br=B)))\}$$

(Q) Find sailors who have reserved all red boats.

$$\{\langle I,N,T,A\rangle \mid \langle I,N,T,A\rangle \in Sailors \wedge \forall\langle B,BN,C\rangle \in Boats (C='red' \Rightarrow \exists\langle Ir,Br,D\rangle \in Reserves(I=Ir \wedge Br=B))\}$$

Here, we find all sailors such that for every red boat there is a tuple in Reserves that shows the sailor has reserved it.

Self Assessment Questions

1. Which of the following is true for relational calculus?

(a) x(P(x))º¬('x)(¬P(x)) (b) ?x(P(x))º¬('x)(P(x))

(c) ? x(P(x))º('x)(¬P(x)) (d) ?x(P(x))º('x)(P(x))

2. In tuple relational calculus P1 AND P2 is equivalent to

(a)(¬P1OR¬P2). (b) ¬(P1OR¬P2).

(c)¬(¬P1OR P2). (d) ¬(¬P1OR ¬P2).

3. When R ∩ S = f , then the cost of computing R ? S is

(a) The same as R × S (b) Greater the R × S

(c) Less than R × S (d) Cannot say anything

4. The FD A → B , DB →C implies

(a) DA → C (b) A →C

(c) B → A (d) DB →A

5. The expression $sq_1(E1?q_2E2)$ is the same as

(a) E1? $q_{1\square}$ $_{q2}$E2 (b) sq_1 E1□ s ɋ E2

(c)E1? $q_{1\acute{U}}$ q_2E2 (d) none of the these

6.4 Comparison of TRC, DRC and RA

We have presented two formal query languages for the relational model. Are they equivalent in power? Can every query that can be expressed in relational algebra also be expressed in relational calculus? The answer is yes, it can. Can every query that can be expressed in relational calculus also be expressed in relational algebra? Before we answer this question, we consider a major problem with the calculus as we have presented it.

Consider the query {S | ¬(S ε Sailors)}. This query is syntactically correct. However, it asks for all tuples S such that S is not in (the given instance of) Sailors. The set of such S tuples is obviously infinite, in the context of infinite domains such as the set of all integers. This simple example illustrates an unsafe query. It is desirable to restrict relational calculus to disallow unsafe queries.

We now sketch how calculus queries are restricted to be safe. Consider a set I of relation instances, with one instance per relation that appears in the query Q. Let Dom (Q,I) be the set of all constants that appear in these relation instances I, or in the formulation of the query Q itself. Since we only allow finite instances I, Dom (Q, I) is also finite.

For a calculus formula Q to be considered safe, at a minimum we want to ensure that for any given I, the set of answers for Q contains only values that are in Dom(Q, I).

While this restriction is obviously required, it is not enough. Not only do we want the set of answers to be composed of constants in Dom (Q,I), we wish to compute the set of answers by only examining tuples that contain constants in Dom(Q, I)! This wish leads to a subtle point associated with the use of quantifiers For all and For any: Given a TRC formula of the form For any R(p(R)), we want to find all values for variable R, that make this formula true by checking only tuples that contain constants in Dom(Q, I). Similarly, given a TRC formula of the form For all R(p(R)), we want to find any values for variable R, that make this formula false, by checking only tuples that contain constants in Dom(Q, I).

We therefore define a safe TRC formula Q to be a formula such that:

1. For any given I, the set of answers for Q contains only values that are in Dom (Q,I).
2. For each sub expression of the form For any R(p(R)) in Q, if a tuple r (assigned to variable R) makes the formula true, then r contains only constants in Dom(Q, I).
3. For each sub expression of the form for all R (p(R)) in Q, if a tuple r (assigned to variable R) contains a constant that is not in Dom (Q, I), then r must make the formula true.

6.5 Summary

- The relational calculus is a non-procedural a different approach than relational algebra, but query language whereas relational algebra is a procedural query language.
- A tuple variable is a variable that takes on tuples of a particular relation schema as values.
- The domain relational calculus or domain variables that on values from an attribute's domain rather than values for an entire tuple.
- Non-procedural query language, the user is concerned with the details of how to obtain the end results.

- Each variable in a TRC formula has a well-defined domain from which values for the variable are drawn.

6.6 Keywords

Domain variable: A domain variable is a variable that ranges over the values in the domain of some attribute
Query: A query is way to solve the problems using some SQL commands.
Relational Calculus: The relational calculus is a non-procedural a different approach than relational algebra, but query language whereas relational algebra is a procedural query language.
Schema: The overall design of the database is called the database schema.
Tuple variable: A tuple variable is a variable that takes on tuples of a particular relation schema as values.

6.7 Review Questions

1. Describe the structure of well formed formulae in relational calculus.
2. What is relational completeness? If a query language is relationally complete, can you write any desired query in that language?
3. What is an unsafe query? Give an example and explain why it is important to disallow such queries.
4. What is the tuple relational calculus?
5. Explain the operators used in TRC.
6. Define the example queries using TRC.
7. What is the domain relational calculus?
8. Explain the operators used in DRC
9. Define the examples of DRC Queries
10. What is the basic difference between TRC, DRC and RA?

Answers for Self Assessment Questions

1. (a) 2. (d) 3. (a) 4. (a) 5. (a)

6.8 Further Studies

1. Database Systems: A Practical Approach To Design, Implementation And By Connolly

7

Database Language and SQL

STRUCTURE

7.0 Objectives

7.1 Introduction

7.2 Integrity Constraints

7.3 Structured Query Language (SQL)

7.4 Characteristics of SQL

7.5 Advantage of SQL

7.6 Data Types SQL and Literals

7.7 Type of SQL Commands

7.8 SQL Operators and their Procedure

7.9 Embedded SQL

7.10 Table

7.11 Queries and Sub Queries

7.12 Aggregate Functions

7.13 Cursors in SQL

7.14 Summary

7.15 Keywords

7.16 Review Questions

7.17 Further Studies

7.0 Objectives

After studying this chapter, you will be able to:

- Describes the data definition language
- Discuss data manipulation language
- Discuss characteristics of SQL
- Understand advantage of SQL
- Define data types SQL and literals
- Understand type of SQL commands
- Explain SQL operators and their procedure

- Discuss embedded SQL

7.1 Introduction

A SQL VIEW can be thought of as a saved query that returns a virtual table. This virtual table can be treated like a real or regular database table. In other words, the VIEWs results can be presented to an end user as is, or they can be re-queried to further limit the rows returned or apply grouping and ordering clauses. So we can create a TSQL statement such as "SELECT * FROM myView ORDER BY col". In addition, data can be added to the database through a VIEW. This chapter will examine the syntax and options used in creating SQL VIEWs.

A VIEW is a convenient way to give a user only partial access to a table. The VIEW can restrict the rows being returned as well as the available columns. So granting the user access to the VIEW rather than the table will effectively restrict their access. VIEWs are also a handy method for hiding a complex statement and only presenting the end user with a simple one-table result set.

Defining views can be very simple, but managing and using them can become quite complex. A lot of rules govern view creation and usage. This is focuses on view creation, modification, and usage, starting with the definition and advantages of views.

7.2 Integrity Constraints

Before one can start to implement the database tables, one must define the integrity constraints. Intergrity means something like 'be right' and consistent. The data in a database must be right and in good condition.

There are the domain integrity, the entity integrity, the referential integrity and the foreign key integrity constraints.

7.2.1 Domain Integrity

Domain integrity means the definition of a valid set of values for an attribute. You define

- data type,
- lenght or size
- is null value allowed
- is the value unique or not

for an attribute.

You may also define the default value, the range (values in between) and/or specific values for the attribute. Some DBMS allow you to define the output format and/or input mask for the attribute.

These definitions ensure that a specific attribute will have a right and proper value in the database.

Entity Integrity Constraint

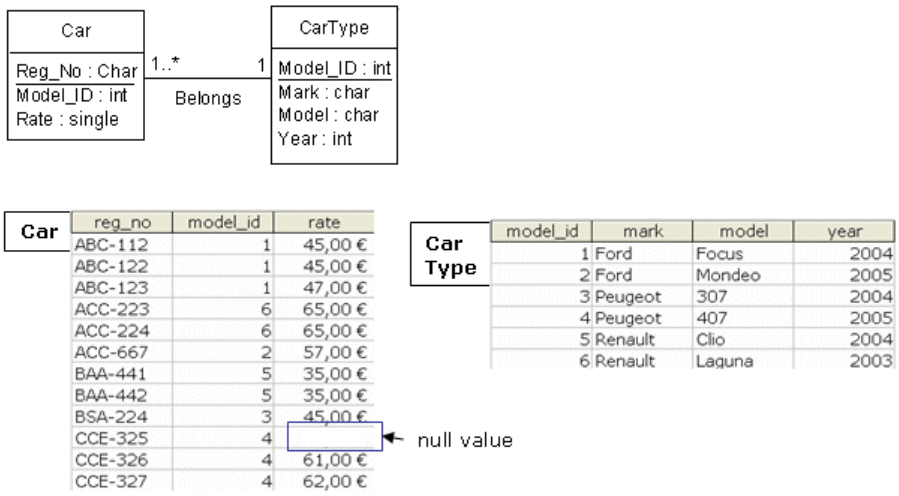
The entity integrity constraint states that primary keys can't be null. There must be a proper value in the primary key field.

This is because the primary key value is used to identify individual rows in a table. If there were null values for primary keys, it would mean that we could not indentify those rows.

On the other hand, there can be null values other than primary key fields. Null value means that one doesn't know the value for that field. Null value is different from zero value or space.

In the Car Rental database in the Car table each car must have a proper and unique Reg_No. There might be a car whose rate is unknown - maybe the car is broken or it is brand new - i.e. the Rate field has a null value. See the picture below.

The entity integrity constraints assure that a spesific row in a table can be identified.



Picture. Car and CarType tables in the Rent database

Referential Integrity Constraint

The referential integrity constraint is specified between two tables and it is used to maintain the consistency among rows between the two tables.

The rules are:

1. You can't delete a record from a primary table if matching records exist in a related table.
2. You can't change a primary key value in the primary table if that record has related records.
3. You can't enter a value in the foreign key field of the related table that doesn't exist in the primary key of the primary table.
4. However, you can enter a Null value in the foreign key, specifying that the records are unrelated.

Examples

Rule 1. You can't delete any of the rows in the CarType table that are visible in the picture since all the car types are in use in the Car table.

Rule 2. You can't change any of the model_ids in the CarType table since all the car types are in use in the Car table.

Rule 3. The values that you can enter in the model_id field in the Car table must be in the model_id field in the CarType table.

Rule 4. The model_id field in the Car table can have a null value which means that the car type of that car in not known

Foreign Key Integrity Constraint

There are two foreign key integrity constraints: cascade update related fields and cascade delete related rows. These constraints affect the referential integrity constraint.

Cascade Update Related Fields

Any time you change the primary key of a row in the primary table, the foreign key values are updated in the matching rows in the related table. This constraint overrules rule 2 in the referential integrity constraints.

If this contraint is defined in the relationship between the tables Car and CarType, it is possible to change the model_id in the CarType table. If one should change the model_id 1 (Ford Focus) to model_id 100 in the

CarType table, the model_ids in the Car table would change from 1 to 100 (cars ABC-112, ABC-122, ABC-123).

7.3 Structured Query Language (SQL)

Structured Query Language (SQL) is a computer language aimed to store, manipulate, and query data stored in relational databases. The first incarnation of SQL appeared in 1974, when a group in IBM developed the first prototype of a relational database. The first commercial relational database was released by Relational Software.

Structured Query Language or SQL is a standard computer language for accessing and manipulating database systems. SQL comprises one of the fundamental building blocks of modern database architecture. SQL defines methods using which user can create and manipulate databases on all major platforms. SQL is a set based declarative programming language and not an imperative programming language like C or BASIC.

Standards for SQL exist. However, the SQL that can be used on each one of the major RDBMS today is in different flavours. This is due to two reasons:

(1) The SQL command standard is fairly complex, and it is not practical to implement the entire standard, and

(2) Each database vendor needs a way to differentiate its product from others. In this tutorial, such differences are noted where appropriate.

SQL has clearly established itself as the standard relational-database language. There are numerous versions of SQL. The original version was developed at IBM's San Jose Research Laboratory (now the Alma den Research Centre). This language, originally called Sequel, was implemented as part of the System R project in the early 1970s. The Sequel language has evolved since then, and its name has changed to SQL (Structured Query Language). Numerous products now support the SQL language.

The SQL language has several parts:

- ***Data-definition language*** (DDL). The SQL DDL provides commands for defining relation schemas, deleting relations, creating indices, and modifying relation schemas.
- ***Interactive data-manipulation language*** (DML). The SQL DML includes a query language based on both the relational algebra and the tuple relational calculus. It includes also commands to insert tuples into, delete tuples from, and to modify tuples in the database.
- ***Embedded DML*** The embedded form of SQL is designed for use within general-purpose programming languages, such as PL/I, Cobol, Pascal, Fortran, and C.
- ***View definition*** The SQL DDL includes commands for defining views.
- ***Authorization*** The SQL DDL includes commands for specifying access rights to relations and views.
- ***Integrity*** The SQL DDL includes commands for specifying integrity constraints that the data stored in the database must satisfy. Updates that violate integrity constraints are disallowed.
- ***Transaction control*** SQL includes commands for specifying the beginning and ending of transactions. Several implementations also allow explicit locking of data for concurrency control.

Did You Know?

The original version of SQL called SEQUEL (structured English query language) was designed by an IBM research centre in 1974 and 1975. SQL was first introduced as a commercial database system in 1979 by Oracle Corporation.

7.4 Characteristics of SQL

SQL enables end user and system persons to deal with a number of database management systems where it is available. Applications written in SQL can be easily ported across systems. Such porting could be required

when the underlying DBMS needs to upgraded because of change in transaction volumes or when a system developed in one environment is to be used on another. SQL as a language is independent of the way it is implanted internally. A query returns the same result regardless of whether optimizing has been done with indexes or not. This is because SQL specifies what is required and not how it is to be done. The language while being simple and easy to learn can cope with complex situations.

- SQL is an ANSI and ISO standard computer language for creating and manipulating databases.
- SQL allows the user to create, update, delete, and retrieve data from a database.
- SQL is very simple and easy to learn.
- SQL works with database programs like DB2, Oracle, MS Access, Sybase, MS SQL Sever etc.

Embedded SQL -allows a host-language program to access the database, but it provides no assistance in presenting results to the user or in generating reports. Most commercial database products include a special language to assist application programmers in creating templates on the screen for a user interface, and. in formatting data for report generation. These special languages are called fourth generation languages.

Some fourth-generation languages also include high-level constructs to allow iteration over relations to be expressed directly, without forcing programmers to deal with the details of cursor management. However, unlike for SQL and embedded SQL, no single accepted standard currently exists for fourth-generation languages. Rather, each product provides its own proprietary language.

An SQL environment contains several components, including a user identifier, and a schema. The presence of multiple schemas allows different applications and different users to work independently. All the usual SQL statements, including the DDL and DML statements, operate in the context of a schema. Schemas can be created and dropped by means of create schema and drop schema statements. If a schema is not set explicitly, a default schema associated with the user identifier is used, so different users see their own schemas. Another use of schemas is to allow multiple versions of an application-one a production version and others versions being tested-to run on the same database system.

Did You Know?

In October 2005, SQL Server 2005 code named Yukon was released where Microsoft rewrote Database Engine and pushed SQL Server product at another level into Enterprise space. This version of SQL Server introduced many new features and improvements.

7.5 Advantage of SQL

The following advantages of SQL:

Vendor Independence

SQL is offered by all of the leading DBMS vendors, and no new database product over the last decade has been highly successful without SQL support. A SQL-based database and the programs that use it can be moved from one DBMS to another vendor's DBMS with minimal conversion effort and little retraining of personnel. Database tools, such as query tools, report writers, and application generators, work with many different brands of SQL databases. The vendor independence thus provided by SQL was one of the most important reasons for its early popularity and remains an important feature today.

Portability Across Computer Systems

SQL-based database products run on computer systems ranging from mainframes and midrange systems to personal computers, workstations, a wide range of specialized server computers, and even handheld devices. They operate on stand-alone computer systems, in departmental local area networks, and in enterprise wide or Internet wide networks. SQL-based applications that begin on single-user or departmental server systems can be moved to larger server systems as they grow. Data from corporate SQL-based databases can be extracted

and downloaded into departmental or personal databases. Finally, economical personal computers can be used to prototype a SQL-based database application before moving it to an expensive multiuser system.

SQL Standards

An official standard for SQL was initially published by the American National Standards Institute (ANSI) and the International Standards Organization (ISO) in 1986, and was expanded in 1989 and again in 1992 and 1999. SQL is also a U.S. Federal Information Processing Standard (FIPS), making it a key requirement for large government computer contracts. Over the years, other international, government, and vendor groups have pioneered the standardization of new SQL capabilities, such as call-level interfaces or object-based extensions. Many of these new initiatives have been incorporated into the ANSI/ISO standard over time. The evolving standards serve as an official stamp of approval for SQL and have speeded its market acceptance.

IBM Endorsement and Commitment (DB2)

SQL was originally invented by IBM researchers and has since become a strategic product for IBM based on its flagship DB2 database. SQL support is available on all major IBM product families, from personal computers through midrange systems (AS/400 and UNIX-based servers) to IBM mainframes. IBM's initial work provided a clear signal of IBM's direction for other database and system vendors to follow early in the development of SQL and relational databases. Later, IBM's commitment and broad support speeded the market acceptance of SQL. IBM's SQL reach today extends well beyond its own computer systems business. SQL-based products that IBM has developed or acquired now run across a broad range of hardware, in many cases from competing computer vendors such as Sun or Hewlett-Packard.

Microsoft Commitment (SQL Server, ODBC, and ADO)

Microsoft has long considered database access a key part of its Windows personal computer software architecture. Both desktop and server versions of Windows provide standardized relational database access through Open Database Connectivity (ODBC), a SQL-based call-level API. Leading Windows software applications (spreadsheets, word processors, databases, etc.) from Microsoft and other vendors support ODBC, and all leading SQL databases provide ODBC access. Microsoft has enhanced ODBC support with higher-level, more object-oriented database access layers as part of its Object Linking and Embedding technology (OLE DB), and more recently as part of Active/X (Active/X Data Objects, or ADO). When Microsoft began its effort in the late 1980s to make Windows a viable server operating system, it introduced SQL Server as its own SQL-based offering. SQL Server continues today as a flagship Microsoft product, and a key component of its .NET architecture for web services.

Relational Foundation

SQL is a language for relational databases, and it has become popular along with the relational database model. The tabular, row/column structure of a relational database is intuitive to users, keeping the SQL language simple and easy to understand. The relational model also has a strong theoretical foundation that has guided the evolution and implementation of relational databases. Riding a wave of acceptance brought about by the success of the relational model, SQL has become the database language for relational databases.

High-Level, English-Like Structure

SQL statements look like simple English sentences, making SQL easy to learn and understand. This is in part because SQL statements describe the data to be retrieved, rather than specifying how to find the data. Tables and columns in a SQL database can have long, descriptive names. As a result, most SQL statements "say what they mean" and can be read as clear, natural sentences.

Interactive, Ad Hoc Queries

SQL is an interactive query language that gives users ad hoc access to stored data. Using SQL interactively, a user can get answers even to complex questions in minutes or seconds, in sharp contrast to the days or weeks it would take for a programmer to write a custom report program. Because of SQL's ad hoc query power, data is more accessible and can be used to help an organization make better, more informed decisions. SQL's ad hoc query capability was an important advantage over no relational databases early in its evolution and more recently has continued as a key advantage over pure object-based databases.

Programmatic Database Access

SQL is also a database language used by programmers to write applications that access a database. The same SQL statements are used for both interactive and programmatic access, so the database access parts of a program can be tested first with interactive SQL and then embedded into the program. In contrast, traditional databases provided one set of tools for programmatic access and a separate query facility for ad hoc requests, without any synergy between the two modes of access.

Multiple Views of Data

Using SQL, the creator of a database can give different users of the database different views of its structure and contents. For example, the database can be constructed so that each user sees data only for his or her department or sales region. In addition, data from several different parts of the database can be combined and presented to the user as a simple row/column table. SQL views can thus be used to enhance the security of a database and tailor it to the particular needs of individual users.

Complete Database Language

SQL was first developed as an ad hoc query language, but its powers now go far beyond data retrieval. SQL provides a complete, consistent language for creating a database, managing its security, updating its contents, retrieving data, and sharing data among many concurrent users. SQL concepts that are learned in one part of the language can be applied to other SQL commands, making users more productive.

Dynamic Data Definition

Using SQL, the structure of a database can be changed and expanded dynamically, even while users are accessing database contents. This is a major advance over static data definition languages, which prevented access to the database while its structure was being changed. SQL thus provides maximum flexibility, allowing a database to adapt to changing requirements while online applications continue uninterrupted.

Client/Server Architecture

SQL is a natural vehicle for implementing applications using a distributed, client/ server architecture. In this role, SQL serves as the link between "front-end" computer systems optimized for user interaction and "back-end" systems specialized for database management, allowing each system to do what it does best. SQL also allows personal computers to function as front-ends to network servers or to larger minicomputer and mainframe databases, providing access to corporate data from personal computer applications.

Extensibility and Object Technology

The major challenge to SQL's continued dominance as a database standard has come from the emergence of object-based programming, and the introduction of object-based databases as an extension of the broad market trend toward object-based technology. SQL-based database vendors have responded to this challenge by slowly expanding and enhancing SQL to include object features. These "object/relational" databases, which continue to be based on SQL, have emerged as a more popular alternative to "pure object" databases and have

perpetuated SQL's dominance through the last decade. The newest wave of object technology, embodied in the XML standard and web services architectures, has once again created a crop of "XML databases" and alternative query languages to challenge SQL. Previous history tends to suggest that XML-based extensions to SQL and the relational model will once again meet this challenge and insure SQL's continuing importance.

Internet Database Access

With the exploding popularity of the Internet and the World Wide Web, and their standards-based foundation, SQL found a new role in the late 1990s as an Internet data access standard. Early in the development of the Web, developers needed a way to retrieve and present database information on web pages and used SQL as a common language for database gateways. More recently, the emergence of three-tiered Internet architectures with distinct thin client, application server and database server layers, has established SQL as the standard link between the application and database tiers. In the future, the role of SQL in the Internet will extend beyond web site architectures to include data management for collaborating applications and distributed objects in a web services architecture.

Java Integration (JDBC)

A major area of SQL development over the last five to ten years has been the integration of SQL with Java. Seeing the need to link the Java language to existing relational databases, Sun Microsystems (the creator of Java) introduced Java Database Connectivity (JDBC), a standard API that allows Java programs to use SQL for database access. JDBC received a further boost when it was adopted as the data access standard within the Java2 Enterprise Edition (J2EE) specification, which defines the operating environment provided by all of the leading Internet application servers. In addition to its role as a programming language from which databases are used, many of the leading database vendors have also announced or implemented Java support within their database systems, allowing Java to be used as a language for stored procedures and business logic within the database itself. This trend toward integration between Java and SQL will insure the continued importance of SQL in the new era of Java-based programming.

7.6 Data Types SQL and Literals

7.6.1 Data Types

Like all other computer languages, SQL deals with data. So let's first look at how SQL defines data.

Data Type: A group of data that shares some common characteristics and operations.

SQL defines the following data types:

Character String - A sequence of characters from a predefined character set.

Bit String - A sequence of bit values: 0 or 1.

Exact Number - A numeric value who's precision and scale need to be preserved. Precision and scale can be counted at decimal level or binary level. The decimal precision of a numerical value is the total number of significant digits in decimal form. The decimal scale of a numerical value is the number of fractional digits in decimal form. For example, the number 123.45 has a precision of 5 and a scale of 2. The number 0.012345 has a precision of 6 and a scale of 6.

Approximate Number - A numeric value who's precision needs to be preserved, and scale floated to its exponent. An approximate number is always expressed in scientific notation of "mantissa"E"exponent". Note that an approximate number has two precisions: mantissa precision and exponent precision. For example, the number 0.12345e1 has a mantissa precision of 5 and exponent precision of 1.

Date and Time - A value to represent an instance of time. A date and time value can be divided into many portions and related them to a predefined calendar system as year, month, day, hour, minute, second, second

fraction, and time zone. A date and time value also has a precision, which controls the number of digits of the second fraction portion. For example: 1999-1-1 1:1:1.001 has precision of 3 on the second fraction portion.

7.6.2 Data Binary Representations

Now we know what types of data SQL must work with. The next step is to understand how different types of data are represented in binary forms. Since computers can only work with binary digits, we have to represent all data in computer memory in binary forms.

1. Character String - A character string is usually represented in memory as an array of characters. Each character is represented in 8 bits (one byte) or 16 bits (two bytes) based on the character set and the character encoding schema. For example, with ASCII character set and its encoding schema, character "A" will be represented as "01000001". Character "1" will be represented as "00110001". Character string "ABC" will be represented as "010000010100001001000011".

2. Bit String - The binary representation of a bit string should be easy. A bit string should be represented in memory as it is. Bit string "01000001" should be represented as "01000001". There might an issue with memory allocation, because computer allocates memory in units of bytes (8 bits per byte). If the length of a bit string is not multiples of 8 bits, the last allocated byte is not full. How to handle the empty space in the last byte? The different SQL implementation will have different rules.

3. Exact Number - Exact numbers can be divided into two groups: integers and non-integers. An integer is an exact number with scale of 0. An integer is represented in either 4 bytes or 8 bytes based on the signed binary value system. For example, with 4 bytes, integer "1" will be represented as "00000000000000000000000000000001". Integer "-1" will be represented as "11111111111111111111111111111111".

4. Approximate Number - An approximate number is normally represented in binary form according to the IEEE 754 single-precision or double-precision standards in either 4 bytes or 8 bytes. The binary representation is divided into 3 components with different number of bits assigned to each component:

Code:

Sign	Exponent	Fraction	Total
Single-Precision	1	8	23	32
Double-Precision	1	11	52	64

With the double precision standard, the mantissa precision can go up to 52 binary digits, about 15 decimal digits.

5. Data and Time - A date and time value is usually stored in memory as an exact integer number with 8 bytes representing an instance by measuring the time period between this instance and a reference time point in millisecond precision, second fraction precision of 3. How MySQL is store date and time values? We will try to find out later.

7.6.3 Data Literals

Now we know the types of data, and how they are stored in memory. Next we need know how data can get in to the computer. One way is to enter it through the program source code as a data literal.

Data Literal: An program source element that represents a data value. Data literals can be divided into multiple groups depending the type of the data it is representing and how it is representing.

1. Character String Literals are used to construct character strings, exact numbers, approximate numbers and data and time values. The syntax rules of character string literals are pretty simple:

A *character string literal* is a sequence of characters enclosed by quote characters.

The quote character is the single quote character “'“.
If “'“ is part of the sequence, it needs to be doubled it as “''“.
Examples of character string literals:
Quote:
'Hello world!'
'Loews L''Enfant Plaza'
'123'
'0.123e-1'
'1999-01-01'

2. *Hex String Literals* are used to construct character strings and exact numbers. The syntax rules for hex string literals are also very simple:
A hex string literal is a sequence of hex digits enclosed by quote characters and prefixed with “x”.
The quote character is the single quote character “'“.
Examples of hex string literals:
Code:
x'41424344'
x'31323334'
x'31323334'
x'01'
x'0001'
x'ff'
x'ffffffff'
x'ffffffffffffffff'

3. Numeric Literals are used to construct exact numbers and approximate numbers. Syntax rules of numeric literals are:
A numeric literal can be written in signed integer form, signed real numbers without exponents, or real numbers with exponents.
Examples of numeric literals:
Quote:
1
-22
33.3
-44.44
55.555e5
-666.666e-6

4. *Date and Time Literals* are used to construct date and time values. The syntax of date and time literals are:
A date literal is written in the form of “DATE 'yyyy-mm-dd'“.
A time literal is written in the form of “TIMESTAMP 'yyyy-mm-dd hh:mm:ss'“.
Examples of data and time literals:
Quote:
DATE '1999-01-01'
TIMESTAMP '1999-01-01 01:02:03'

Self Assessment Question

1. The original version was developed at …… San Jose Research Laboratory (now the Almaden Research Center).

(a) IBM's (b) Microsoft
(c) HCL (d) None of these.

2. The SQL ……..includes a query language based on both the relational algebra and the tuple relational calculus.

(a) DDL (b) DML
(c) Embedded DML (d) None of these.

3. The SQL DDL includes commands for specifying integrity constraints that the data stored in the database must satisfy. Updates that violate integrity constraints are disallowed.

(a) True (b) False

4. The SQL is a standard interactive and programming language for querying and modifying data and managing……..

(a) data warehouse (b) table
(c) databases (d) None of these

7.7 Type of SQL Commands

The SQL commands are lines of SQL code that ask the SQL application to perform simple tasks against with data inside of a database. Often we refer to commands as query statements or scripts; all of these terms are synonymous. Most of the commands and statements following sections provide a functional summary of SQL statements and are divided into these categories:

- Data Definition Language (DDL) Statements
- Data Manipulation Language (DML) Statements
- Transaction Control Statements
- Session Control Statements
- System Control Statement
- Embedded SQL Statements

Data Definition Language (DDL) Statements

Data definition language (DDL) statements let you to perform these tasks:

- Create, alter, and drop schema objects
- Grant and revoke privileges and roles
- Analyze information on a table, index, or cluster
- Establish auditing options
- Add comments to the data dictionary

The CREATE, ALTER, and DROP commands require exclusive access to the specified object. For example, an ALTER TABLE statement fails if another user has an open transaction on the specified table.

The GRANT, REVOKE, ANALYZE, AUDIT, and COMMENT commands do not require exclusive access to the specified object. For example, you can analyze a table while other users are updating the table. Oracle Database implicitly commits the current transaction before and after every DDL statement.

Many DDL statements may cause Oracle Database to recompile or reauthorize schema objects. For information on how Oracle Database recompiles and reauthorizes schema objects and the circumstances under which a DDL statement.

Data Manipulation Language (DML) Statements

Data manipulation language (DML) statements access and manipulate data in existing schema objects. These statements do not implicitly commit the current transaction. The data manipulation language statements are:

CALL
DELETE
EXPLAIN PLAN
INSERT
LOCK TABLE
MERGE
SELECT
UPDATE

The SELECT statement is a limited form of DML statement in that it can only access data in the database. It cannot manipulate data in the database, although it can operate on the accessed data before returning the results of the query.

Transaction Control Statements

Transaction control statements manage changes made by DML statements. The transaction control statements are:

COMMIT Saves database transactions
ROLLBACK Undoes database transactions
SAVEPOINT Creates points within groups of transactions in which to ROLLBACK
SET TRANSACTION Places a name on a transaction all transaction control statements, except certain forms of the COMMIT and ROLLBACK commands, are supported in PL/SQL. For information on the restrictions, see COMMIT and ROLLBACK.

Session Control Statements

Session control statements dynamically manage the properties of a user session. These statements do not implicitly commit the current transaction. PL/SQL does not support session control statements. The session control statements are:

ALTER SESSION
SET ROLE

System Control Statement

The single system control statement, ALTER SYSTEM, dynamically manages the properties of an Oracle Database instance. This statement does not implicitly commit the current transaction and is not supported in PL/SQL.

Embedded SQL Statements

Embedded SQL statements place DDL, DML, and transaction control statements within a procedural language program. Embedded SQL is supported by the Oracle precompilers and is documented.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Create the database of your college including some important fields.

……..……

……..……

……...

7.8 SQL Operators and their Procedure

Operations are evaluated is the same as in the DATA step with this one exception: NOT is grouped with the logical operators AND and OR in PROC SQL; in the DATA step, NOT is grouped with the unary plus and minus signs.

- Use equal (=) to select rows for which a specified column contains a specified value.
  COL = value
- Use the IN predicate to select each row that has a column value equal to one of several listed values.
  COL IN ('B01', 'C01', 'D01')
- Use LIKE to specify a character string that is similar to the column value of rows you want to select:
  - Use a percent sign (%) to indicate any string of zero or more characters.

COL LIKE '%char'

  - Use an underscore (_) to indicate any single character.

COL LIKE '_char'

- Use BETWEEN to select rows in which a column has a value within two limits.
  COL BETWEEN value1 AND value2.
- Use IS NULL to retrieve rows that contain a null value in some column. A null value indicates the absence of a column value in a row. A null value is not the same as zero or all blanks.
  COL IS NULL
- Use the NOT keyword to select all rows except the rows identified with the search condition.
  - Not equal (<>)
    COL <> value
  - not in
    COL NOT IN (list)
  - not like
    COL NOT LIKE ' char'
  - not between
    COL NOT BETWEEN value1 AND value2
  - is not null
    COL IS NOT NULL
  - You can also use the symbols:
    - <=
    - <
    - >
    - >=

Higher priority operations are performed first: that is, group 0 operators are evaluated before group 5 operators. The following Table 7.1 shows the operators and their order of evaluation, including their priority groups.

Table 7.1: Operators and Order of Evaluation

Group	Operator	Description
0	( )	forces the expression enclosed to be evaluated first

1	case-expression	selects result values that satisfy specified conditions
2	**	raises to a power
	unary +, unary -	indicates a positive or negative number
3	*	Multiplies
	/	Divides
4	+	Adds
	-	Subtracts
5	||	Concatenates
6	<NOT> BETWEEN condition	BETWEEN condition.
	<NOT> CONTAINS condition	CONTAINS condition.
	<NOT> EXISTS condition	EXISTS condition.
	<NOT> IN condition	IN condition.
	IS <NOT> condition	IS condition.
	<NOT> LIKE condition	LIKE condition.
7	=, eq	Equals
	¬=, ^=, < >, ne	does not equal
	>, gt	is greater than
	<, lt	is less than
	>=, ge	is greater than or equal to
	<=, le	is less than or equal to
	=*	Sounds like (use with character operands only). Retrieving Values with the SOUNDS-LIKE Operator.
	Eqt	Equal to truncated strings (use with character operands only). Truncated String Comparison Operators.
	Gtt	greater than truncated strings
	Ltt	less than truncated strings
	Get	greater than or equal to truncated strings
	Let	less than or equal to truncated strings
	Net	not equal to truncated strings
8	¬, ^, NOT	indicates logical NOT
9	&, AND	indicates logical AND
10	|, OR	indicates logical OR

7.9 Embedded SQL

SQL provides a powerful declarative query language. Writing queries in SQL is, typically much easier than is coding the same queries in a general-purpose programming language. However, access to a database from a general-purpose programming language is required for at least two reasons:

1. Not all queries can be expressed in SQL, since SQL does not provide the full .expressive power of a general-purpose language. That is, there exist queries that can be expressed in a language such as Pascal, C,

Cobol, or Fortran that cannot be expressed in SQL. To write such queries, we can embed SQL within a more powerful language.

SQL is designed such that queries written in it can be optimized auto matically and executed efficiently--and providing the full power of a programming language makes automatic optimization exceedingly difficult.

2. Non declarative actions-such as printing a report, interacting with a user, or sending the results of a query to a graphical user interface-cannot be done from within SQL. Applications typically have several components, and querying or updating data is only one component; other components are written in general-purpose programming languages. For an integrated application, the programs written in the programming language must be able to access the database.

The SQL standard defines embeddings of SQL in a variety of programming languages, such as Pascal, PL/I, Fortran, C, and Cobol. A language in which SQL queries are embedded is referred to as a host language, and the SQL structures permitted in the host language constitute embedded SQL.

Programs written in the host language can use the embedded SQL syntax to access and update data stored in a database. This embedded form of SQL extends the programmer's ability to manipulate the database even further. In embedded SQL, all query Kocessing is performed by the database system. The result of the query is then made available to the program one tuple (record) at a time.

An embedded SQL program must be processed by a special preprocessor._ prior to compilation. L<nbedded SQL requests are replaced with host-language declarations and procedure calls that allow run-time execution of the database accesses. Then, the resulting program is compiled by the host-language compiler. To identify embedded SQL requests to the preprocessor, we use the EXEC SQL statement; it has the form:

EXEC SQL <embedded SQL statement > END-EXEC

The exact syntax for embedded SQL requests depends on the language in which SQL is embedded. For instance, a semi-colon is used instead of END-EXEC when SQL is embedded in C or Pascal.

We place the statement SQL INCLUDE in the program to identify the place where the preprocessor should insert the special variables used for communication between the program and the database system. Variables of the host language can be used within embedded SQL statements, but they must be preceded by a colon (:) to distinguish them from SQL variables."

Embedded SQL statements are of a form similar to the SQL statements. There are, however, several important differences, as we note here.

To write a relational query, we use the declare cursor statement. The result of the query is not yet computed. Rather, the program must use the open and fetch to obtain the result tuples.

Consider the banking schema, Assume that we have a host-language variable amount, and that we wish to find the names and cities of residence of customers who have more than amount dollars in any account. We can write this query as follows:

```
EXEC SQL
declare c cursor for
select customer-name, customer-city from deposit, customer
where deposit.customer-name = customer.customer-name and
deposit.balance > :amount
END-EXEC
```

The variable c in the preceding expression is called a cursor for the query. We use this variable to identify the query in the open statement, which causes the query to be evaluated, and in the fetch statement, which causes the values of one tuple to be placed in host-language variables.

The open statement for our sample query is as follows:

```
EXEC SQL open c END-EXEC
```

This statement causes the database system to execute the query and to save the re-sults within a temporary relation. If the SQL query results in an error, the database system stores an error diagnostic in the SQL communication-area (SQLCA) vari-ables, whose declarations are inserted by the SQL INCLUDE statement.
A series of fetch statements is executed to make tuples of the result available to the program. The fetch statement requires one host-language variable for each attribute of the result relation. For our example query, we need one variable to hold the customer-name value, and another to hold the customer-city value. Suppose that those variables are cn and cc, respectively. A tuple of the result relation is obtained by the statement:
EXEC SQL fetch c into :cn, :cc END-EXEC
The program can then manipulate the variables cn and cc using the features of the host programming language.

7.10 Table

Tables are the basic structure where data is stored in the database. Given that in most cases, there is no way for the database vendor to know ahead of time what your data storage needs are, chances are that you will need to create tables in the database yourself. Many database tools allow you to create tables without writing SQL, but given that tables are the container of all the data.
The foundation of every Relational Database Management System is a database object called table. Every database consists of one or more tables, which store the database's data/information. Each table has its own unique name and consists of columns and rows.

The database table columns (called also table fields) have their own unique names and have a pre-defined data types. Table columns can have various attributes defining the column functionality (the column is a primary key, there is an index defined on the column, the column has certain default value, etc.). While table columns describe the data types, the table rows contain the actual data for the columns. It is important to include the CREATE TABLE syntax.

The SQL syntax for CREATE TABLE is
CREATE TABLE "table_name"
("column 1" "data_type_for_column_1", "column 2" "data_type_for_column_2",...)
So, if we are to create the customer table specified as above, we would type in
CREATE TABLE customer (First_Name char(50), Last_Name char(50), Address char(50), City char(50), Country char(25), Birth_Date date).

Sometimes, we want to provide a default value for each column. A default value is used when you do not specify a column's value when inserting data into the table. To specify a default value, add "Default [value]" after the data type declaration. In the example, if we want to default column "Address" to "Unknown" and City to "Mumbai", we would type in

```
CREATE TABLE customer
 (First_Name char(50),
 Last_Name char(50),
 Address char(50) default 'Unknown',
 City char(50) default 'Mumbai',
 Country char(25),
 Birth_Date date)
```

Here is an example of a simple database table, containing customer's data. The first row, listed in bold, contains the names of the table columns:
Customers table

FirstName	LastName	Email	DOB	Phone
Kamal	Kumar	kamal.Kumar@yahoo.com	2/4/1968	626 222-2222
Satish	Sharma	s.satish@gmail.com	4/4/1974	323 455-4545
Paula	Bhayel	pb@hotmail.com	5/24/1978	416 323-3232
Rinku	Kumar	rrk@nic.co.in	20/10/1980	416 323-8888

7.10.1 Insert Statement

The SQL tables store data in rows, one row after another. The INSERT command is the command used to insert new data (a new row) into a table by specifying a list of values to be inserted into each table column. The arrangement of values is important, and how they are arranged in the code corresponds to how the data values will be arranged in the SQL table.

The syntax for the INSERT statement is:

INSERT INTO table (column-1, column-2, ... column-n) VALUES (value-1, value-2, ... value-n);

Example

Let's take a look at a very simple example.

INSERT INTO suppliers (supplier_id, supplier_name) VALUES (24553, 'Ram');

This would result in one record being inserted into the suppliers table. This new record would have a supplier_id of 24553 and a supplier_name of Ram.

For example:

INSERT INTO suppliers
(supplier_id, supplier_name)
SELECT account_no, name
FROM customers
WHERE city = 'Allahabad';

By placing a "select" in the insert statement, you can perform multiples inserts quickly.

With this type of insert, you may wish to check for the number of rows being inserted. You can determine the number of rows that will be inserted by running the following SQL statement before performing the insert.

SELECT count(*)
FROM customers
WHERE city = 'Allahabad';

The second type of INSERT INTO allows us to insert multiple rows into a table. Unlike the previous example, where we insert a single row by specifying its values for all columns, we now use a SELECT statement to specify the data that we want to insert into the table. If you are thinking whether this means that you are using information from another table, you are correct. The syntax is as follows:

INSERT INTO "table1" ("column1", "column2", ...)
SELECT "column3", "column4", FROM "table2"

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Create a table and use insert query to insert the data**.**

……..……

……..……

……...

7.10.2 Update Statement

The SQL Update is used to update one table columns with specified values. You can update one or multiple columns at the same time. For instance you might want to update customer address and this case you would update several columns like AddressLine1, AddressLine2, City, Post code with usually either hard coded or provided values from end user application. You could also update values using calculated fields for instance you might want to update Employee holiday entitlement (once a year) and you could use Employee start date to calculate number of years that employee has worked for the company and use some "IF" (case in SQL) logic to specify the correct holiday entitlement. Very often you might want to perform updates using another related table and for instance in our Holiday entitlement case we could use a HolidayEntitlement table and match it against our Years Worked that could be calculated using EmployeeStartDate and provide matching row (HolidayEntitlement) from related HolidayEntitlement table.

The syntax for the UPDATE statement is:
UPDATE table
SET column = expression
WHERE predicates;
Example
Let's take a look at a very simple example.
UPDATE suppliers
SET name = 'HP'
WHERE name = 'IBM';

This statement would update all supplier names in the suppliers table from IBM to HP.
You may wish to update records in one table based on values in another table. Since you cannot list more than one table in the UPDATE statement, you can use the EXISTS clause.
Example:

UPDATE suppliers (SELECT customers.name SET supplier_name = FROM customers WHERE customers.customer_id = suppliers.supplier_id) WHERE EXISTS (SELECT customers.name FROM customers WHERE customers.customer_id = suppliers.supplier_id);
Whenever a supplier_id matched a customer_id value, the supplier_name would be overwritten to the customer name from the customers table.

7.10.3 Delete Statement

The DELETE Statement is used to delete rows from a table.
The Syntax of a SQL DELETE statement is:
DELETE FROM table_name [WHERE condition];
table_name - the table name which has to be updated.
Note: The WHERE clause in the SQL delete command is optional and it identifies the rows in the column that gets deleted. If you do not include the WHERE clause all the rows in the table is deleted, so be careful while writing a DELETE query without WHERE clause.

Example: To delete an employee with id 100 from the employee table, the SQL delete query would be like,
DELETE FROM employee WHERE id = 100;
To delete all the rows from the employee table, the query would be like,
DELETE FROM employee;

SQL DELETE Example

The "Persons"

P_Id	LastName	FirstName	Address	City
1	Kumar	Rahul	Sector 10	Kota
2	Singh	Satyendra	Borgvn 23	Kota
3	Sharma	Pankaj	Sector 20	Kanpur
4	Verma	Johan	Bakken 2	Kanpur
5	Tjessem	Jakob	Sector 67	Kota

Now we want to delete the person "Tjessem, Jakob" in the "Persons".
We use the following SQL statement:
DELETE FROM Persons WHERE LastName='Tjessem' AND FirstName='Jakob'

The "Persons" table will now look like this:

P_Id	LastName	FirstName	Address	City
1	Kumar	Rahul	Sector 10	Kota
2	Singh	Satyendra	Borgvn 23	Kota
3	Sharma	Pankaj	Sector 20	Kanpur
4	Verma	Johan	Bakken 2	Kanpur

Delete All Rows

It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact:
DELETE FROM table_name
or
DELETE * FROM table_name

SQL TRUNCATE Statement

The SQL TRUNCATE command is used to delete all the rows from the table and free the space containing the table.
Syntax to TRUNCATE a table:

```
TRUNCATE TABLE table_name;
```

For Example: To delete all the rows from employee table, the query would be like,

```
TRUNCATE TABLE employee;
```

Difference between DELETE and TRUNCATE Statements:

DELETE Statement: This command deletes only the rows from the table based on the condition given in the where clause or deletes all the rows from the table if no condition is specified. But it does not free the space containing the table.
TRUNCATE statement: This command is used to delete all the rows from the table and free the space containing the table.

SQL DROP Statement:

The SQL DROP command is used to remove an object from the database. If you drop a table, all the rows in the table is deleted and the table structure is removed from the database. Once a table is dropped we cannot get it back, so be careful while using RENAME command. When a table is dropped all the references to the table will not be valid.

Syntax to drop a SQL table structure:

```
DROP TABLE table_name;
```

Example: To drop the table employee, the query would be like

```
DROP TABLE employee;
```

Caution

Changing any part of an object name can break scripts and stored procedures. We recommend you do not use this statement to rename stored procedures, triggers, user-defined functions, or views; instead, drop the object and re-create it with the new name.

7.10.4 Table Expressions

Table expressions are sub-queries where a table is expected. There are two types of table expressions. This post explains how to use the Table expressions and their advantages.

- Derived tables
- Common Table Expressions

Derived Tables

A Derived table is a table expression that appears in the FROM clause of a query. Derived tables can be used when the use of column aliases is not possible because another clause is processed before the alias name.

Example:

```
1: USE AdventureWorks
2: SELECT MONTH(HireDate) as Hire_Month
3: FROM HumanResources.Employee
4: GROUP BY Hire_Month;
```

When we execute above query, we will get the following result

Msg 207, Level 16, State 1, Line 4

Invalid column name 'Hire_Month'.

The reason for getting the above error message is GROUP BY clause is processed before the Select Clause. Here Alias name is not known when group by is processed.

We can solve the above error by re-writing the above query using Derived Tables.

```
1: USE AdventureWorks
2: SELECT Hire_Month
3: FROM (SELECT MONTH(HireDate) as Hire_Month
4: FROM HumanResources.Employee) AS m
5: GROUP BY Hire_Month;
```

The result of a table expression is always a table or expression.

Example:

```
1: SELECT d.StartDate, (SELECT EmployeeID
2: FROM HumanResources.Employee e WHERE e.EmployeeID = d.EmployeeID)
```

```
3:  AS EmployeeID
4: FROM HumanResources.EmployeeDepartmentHistory d
5: WHERE d.StartDate IN ('1998-01-11 00:00:00.000',
6: '1997-02-26 00:00:00.000');
```

Common Table Expressions

A common table expression (CTE) is a named table expression supported by Transact-SQL. It is similar to Derived table but it is not stored as an object and lasts for the duration of the query.

Common Table Expressions can be used in two types of queries

1. Non-Recursive
2. Recursive

CTEs and Non-Recursive Queries

Non-Recursive form of CTEs can be used as an alternative to Derived Tables and Views. Generally CTE's are defined using the WITH statement.

Example:

To find the total dues whose values are greater than the average of all dues and whose freights are greater than 40% of the avg of all dues.

```
1: USE AdventureWorks;
2: SELECT SalesOrderID
3: FROM Sales.SalesOrderHeader
4: WHERE TotalDue > (SELECT AVG(TotalDue)
5: FROM Sales.SalesOrderHeader
6: WHERE YEAR(OrderDate) = '2002')
7: AND Freight > (SELECT AVG(TotalDue)
8: FROM Sales.SalesOrderHeader
9: WHERE YEAR(OrderDate) = '2002')/2.5;
```

The main problem with above query is space consuming and inner query has written twice. We can shorten the syntax of the query and better use CTE to re-write the above query.

Syntax:

WITH cte_name(column_list) AS (inner_query)
outer_query.

```
1: USE AdventureWorks;
2: WITH price_calc(year_2002) AS
3: (SELECT AVG(TotalDue)
4: FROM Sales.SalesOrderHeader
5: WHERE YEAR(OrderDate) = '2002')
6: SELECT SalesOrderID
7: FROM Sales.SalesOrderHeader
8: WHERE TotalDue > (SELECT year_2002 FROM price_calc)
9: AND Freight > (SELECT year_2002 FROM price_calc)/2.5;
```

The CTE name in above query is price_calc and has column named year_2002 The defined CTE name in inner query can be used in outer query.

CTEs and Recursive Queries

We can use CTEs to implement recursion, because CTEs contains reference to themselves.

Syntax:

WITH cte_name (column_list) AS

(anchor_member
UNION ALL
recursive_member)
outer_query

With clause contains the two queries that are connected with UNION ALL operator. The first query will be invoked only once, and it holds the result of the recursion. The recursive_member uses the result of first query. After that, system repeatedly invokes the recursive part. The invocation of the recursive member ends when the result of the previous invocation is an empty set.

Example:

```
1: WITH list_of_parts(assembly1, quantity, cost) AS
2: (SELECT containing_assembly, quantity_contained, unit_cost
3: FROM airplane
4: WHERE contained_assembly IS NULL
5: UNION ALL
6: SELECT a.containing_assembly, a.quantity_contained,
7: CAST(l.quantity*l.cost AS DECIMAL(6,2))
8: FROM list_of_parts l,airplane a
9: WHERE l.assembly1 = a.contained_assembly)
10: SELECT * FROM list_of_parts;
```

In the above query WITH clause defines the CTE called lsit_of_parts which contains three columns.The first select statement in the query will be invoked only once and accumulate the results of recursion process.

Restrictions for a CTE in a recursive query

- The CTE definition must contain at least two select statements combined by UNION ALL operator.
- The number of columns in the anchor and recursive members must be the same.
- The data-type of a column in the recursive member must be the same as the data-type in the anchor member.
- The From clause of the recursive member refers only once to the name of the CTE.

7.10.5 Conditional Expressions

If your needs go beyond the capabilities of these conditional expressions you might want to consider writing a stored procedure in a more expressive programming language.

CASE

CASE WHEN *condition* THEN *result*
[WHEN ...]
[ELSE *result*]
END

The SQL CASE expression is a generic conditional expression, similar to if/else statements in other languages. CASE clauses can be used wherever an expression is valid. *Condition* is an expression that returns a Boolean result. If the result is true then the value of the CASE expression is *result*. If the result is false any subsequent WHEN clauses are searched in the same manner. If no WHEN *condition* is true then the value of the case expression is the *result* in the ELSE clause. If the ELSE clause is omitted and no condition matches, the result is null.

CASE *expression*
WHEN *value* THEN *result*
[WHEN ...]
[ELSE *result*]

END

This "simple" CASE expression is a specialized variant of the general form above. The *expression* is computed and compared to all the *value*s in the WHEN clauses until one is found that is equal. If no match is found, the *result* in the ELSE clause (or a null value) is returned. This is similar to the switch statement in C.

The example above can be written using the simple CASE syntax:

```
=> SELECT a,
       CASE a WHEN 1 THEN 'one'
            WHEN 2 THEN 'two'
            ELSE 'other'
       END
   FROM test;
 a | case
---+-------
 1 | one
 2 | two
 3 | other
```

COALESCE

COALESCE(*value* [, ...])

The COALESCE function returns the first of its arguments that is not null. This is often useful to substitute a default value for null values when data is retrieved for display, for example:

SELECT COALESCE(description, short_description, '(none)') ...

NULLIF

NULLIF(*value1*, *value2*)

The NULLIF function returns a null value if and only if *value1* and *value2* are equal. Otherwise it returns *value1*. This can be used to perform the inverse operation of the COALESCE example given above:

SELECT NULLIF(value, '(none)') ...

COALESCE and NULLIF are just shorthand for CASE expressions. They are actually converted into CASE expressions at a very early stage of processing, and subsequent processing thinks it is dealing with CASE. Thus an incorrect COALESCE or NULLIF usage may draw an error message that refers to CASE.

7.10.6 Join

The join keyword is used in an SQL statement to query data from two or more tables, based on a relationship between certain columns in these tables. Tables in a database are often related to each other with keys.

A primary key is a column (or a combination of columns) with a unique value for each row. Each primary key value must be unique within the table. The purpose is to bind data together, across tables, without repeating all of the data in every table.

With the help of joins, you can retrieve data from two or more table on the basic of the relationship the table. The various types of JOIN you can use, and the differences between them.

- Inner join
- Left join
- Right join
- Full join

Table 7.2: the "Persons"

P_Id	LastName	FirstName	Address	City
1	Kumar	Rahul	Sector 10	Kota
2	Singh	Satyendra	Borgvn 23	Kota
3	Sharma	Pankaj	Sector 20	Kanpur

Note that the "P_Id" column is the primary key in the "Persons" Table 7.2. This means that no two rows can have the same P_Id. The P_Id distinguishes two persons even if they have the same name.

Table 7.3: We have the "Orders" table:

O_Id	OrderNo	P_Id
1	77895	3
2	44678	3
3	22456	1
4	24562	1
5	34764	15

That the "O_Id" column is the primary key in the "Orders" Table 7.3 and that the "P_Id" column refers to the persons in the "Persons" table without using their names.

Inner Join

The inner join keyword return rows when there is at least one match in both tables. Inner join is the same as join.

SQL INNER JOIN Syntax

```
SELECT column_name(s)
 FROM table_name1
INNER JOIN table_name2
ON table_name1.column_name=table_name2.column_name
```

Example

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Kumar	Rahul	Sector 10	Kota
2	Singh	Satyendra	Borgvn 23	Kota
3	Sharma	Pankaj	Sector 20	Kanpur

The "Orders" table:

O_Id	OrderNo	P_Id
1	77895	3
2	44678	3
3	22456	1
4	24562	1
5	34764	15

Now we want to list all the persons with any orders.
We use the following SELECT statement:

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
Inner Join Orders
ON Persons.P_Id=Orders.P_Id
ORDER BY Persons.LastName
```

The result-set will look like this:

LastName	FirstName	OrderNo
Kumar	Rahul	22456
Kumar	Rahul	24562
Sharma	Pankaj	77895
Sharma	Pankaj	44678

The inner join keyword return rows when there is at least one match in both tables. If there are rows in "Persons" that do not have matches in "Orders", those rows will NOT be listed.

Left join

The left join keyword returns all rows from the left table (table_name1), even if there are no matches in the right table (table_name2). In some databases left join is called left outer join.

SQL Left Join Syntax

```
SELECT column_name(s)
 FROM table_name1
left join table_name2
ON table_name1.column_name=table_name2.column_name
```

SQL left join Example

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Kumar	Rahul	Sector 10	Kota
2	Singh	Satyendra	Borgvn 23	Kota
3	Sharma	Pankaj	Sector 20	Kanpur

The "Orders" table:

O_Id	OrderNo	P_Id
1	77895	3
2	44678	3
3	22456	1
4	24562	1
5	34764	15

Now we want to list all the persons and their orders - if any, from the tables above.
We use the following SELECT statement:

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
Left Join Orders
ON Persons.P_Id=Orders.P_Id
ORDER BY Persons.LastName
```

The result-set will look like this:

LastName	FirstName	OrderNo
Kumar	Rahul	22456
Kumar	Rahul	24562
Sharma	Pankaj	77895
Sharma	Pankaj	44678
Singh	Satyendra	

The LEFT JOIN keyword returns all the rows from the left table (Persons), even if there are no matches in the right table (Orders).

Right Join

The right join keyword returns all the rows from the right table (table_name2), even if there are no matches in the left table (table_name1). In some databases right join is called right outer join.

SQL Right Join Syntax

```
SELECT column_name(s)
 FROM table_name1
RIGHT JOIN table_name2
ON table_name1.column_name=table_name2.column_name.
```

Example

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Kumar	Rahul	Sector 10	Kota
2	Singh	Satyendra	Borgvn 23	Kota
3	Sharma	Pankaj	Sector 20	Kanpur

The "Orders" table:

O_Id	OrderNo	P_Id
1	77895	3
2	44678	3
3	22456	1
4	24562	1
5	34764	15

Now we want to list all the orders with containing persons - if any, from the tables above.

We use the following SELECT statement:

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
```

FROM Persons
right join Orders
ON Persons.P_Id=Orders.P_Id
ORDER BY Persons.LastName
The result-set will look like this:

LastName	FirstName	OrderNo
Kumar	Rahul	22456
Kumar	Rahul	24562
Sharma	Pankaj	77895
Sharma	Pankaj	44678
		34764

The right join keyword returns all the rows from the right table (Orders), even if there are no matches in the left table (Persons).

Full Join

The full join keyword return rows when there is a match in one of the tables.
SQL FULL JOIN Syntax
SELECT column_name(s)
FROM table_name1
FULL JOIN table_name2
ON table_name1.column_name=table_name2.column_name
Example
The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Kumar	Rahul	Sector 10	Kota
2	Singh	Satyendra	Borgvn 23	Kota
3	Sharma	Pankaj	Sector 20	Kanpur

The "Orders" table:

O_Id	OrderNo	P_Id
1	77895	3
2	44678	3
3	22456	1
4	24562	1
5	34764	15

Now we want to list all the persons and their orders, and all the orders with their persons.
We use the following SELECT statement:
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo FROM Persons
FULL JOIN Orders
ON Persons.P_Id=Orders.P_Id

ORDER BY Persons.LastName
The result-set will look like this:

LastName	FirstName	OrderNo
Kumar	Rahul	22456
Kumar	Rahul	24562
Sharma	Pankaj	77895
Sharma	Pankaj	44678
Singh	Satyendra	
		34764

The full join keyword returns all the rows from the left table (Persons), and all the rows from the right table (Orders). If there are rows in "Persons" that do not have matches in "Orders", or if there are rows in "Orders" that do not have matches in "Persons", those rows will be listed as well.

7.10.7 Union

The UNION operator is used to combine the result-set of two or more SELECT statements. That each SELECT statement within the UNION must have the same number of columns. The columns must also have similar data types. Also, the columns in each SELECT statement must be in the same order.

The purpose of the SQL UNION query is to combine the results of two queries together. In this respect, UNION is somewhat similar to JOIN in that they are both used to related information from multiple tables. One restriction of UNION is that all corresponding columns need to be of the same data type. Also, when using UNION, only distinct values are selected (similar to SELECT DISTINCT).

SQL UNION Syntax

SELECT column_name(s) FROM table_name1
UNION
SELECT column_name(s) FROM table_name2

Example

Look at the following tables:

"Employees_Norway":

E_ID	E_Name
01	Kumar, Rahul
02	Singh, Satyendra
03	Singh, Stephen
04	Sharma, Pankaj

"Employees_USA":

E_ID	E_Name
01	Turner, Sally
02	Kent, Clark

03	Singh, Stephen
04	Scott, Stephen

Now we want to list all the different employees in Norway and USA.
We use the following SELECT statement:
SELECT E_Name FROM Employees_Norway
UNION
SELECT E_Name FROM Employees_USA
The result-set will look like this:

E_Name
Kumar, Rahul
Singh, Satyendra
Singh, Stephen
Sharma, Pankaj
Turner, Sally
Kent, Clark
Scott, Stephen

Note: This command cannot be used to list all employees in Norway and USA. In the example above we have two employees with equal names, and only one of them will be listed. The UNION command selects only distinct values.
SQL UNION ALL Example
Now we want to list all employees in Norway and USA:
SELECT E_Name FROM Employees_Norway UNION ALL
SELECT E_Name FROM Employees_USA
Result

E_Name
Kumar, Rahul
Singh, Satyendra
Singh, Stephen
Sharma, Pankaj
Turner, Sally
Kent, Clark
Singh, Stephen
Scott, Stephen

7.10.8 Intersections
Similar to the UNION command, INTERSECT also operates on two SQL statements. The difference is that, while UNION essentially acts as an OR operator (value is selected if it appears in either the first or the second statement), the INTERSECT command acts as an AND operator (value is selected only if it appears in both statements). The INTERSECT query allows you to return the results of 2 or more “select” queries. However, it

only returns the rows selected by all queries. If a record exists in one query and not in the other, it will be omitted from the INTERSECT results. Each SQL statement within the INTERSECT query must have the same number of fields in the result sets with similar data types.
The syntax is as follows:
[SQL Statement 1]
INTERSECT
[SQL Statement 2]
Let's assume that we have the following two tables,
Table *Store Information*

store_name	Sales	Date
Los Angeles	$1500	Jan-05-1999
San Diego	$250	Jan-07-1999
Los Angeles	$300	Jan-08-1999
Boston	$700	Jan-08-1999

Table *Internet Sales*

Date	Sales
Jan-07-1999	$250
Jan-10-1999	$535
Jan-11-1999	$320
Jan-12-1999	$750

and we want to find out all the dates where there are both store sales and internet sales. To do so, we use the following SQL statement:
SELECT Date FROM Store_Information
INTERSECT
SELECT Date FROM Internet_Sales
Result:
Date
Jan-07-1999

7.10.9 Minus
SQL MINUS or EXCEPT operator work on two table expressions. The result set takes records from the first table expression, and then subtract out the ones that appear in the second table expression. If the second table expression includes the records which are not appear in the first table expression, these records will be ignored.
The MINUS operates on two SQL statements. It takes all the results from the first SQL statement, and then subtract out the ones that are present in the second SQL statement to get the final answer. If the second SQL statement includes results not present in the first SQL statement, such results are ignored.
The syntax is as follows:
[SQL Statement 1]
MINUS

[SQL Statement 2]
Let's continue with the same example:
Table *Store_Information*

store_name	Sales	Date
Los Angeles	$1500	Jan-05-1999
San Diego	$250	Jan-07-1999
Los Angeles	$300	Jan-08-1999
Boston	$700	Jan-08-1999

Table *Internet_Sales*

Date	Sales
Jan-07-1999	$250
Jan-10-1999	$535
Jan-11-1999	$320
Jan-12-1999	$750

and we want to find out all the dates where there are store sales, but no internet sales. To do so, we use the following SQL statement:
SELECT Date FROM Store_Information
MINUS
SELECT Date FROM Internet_Sales
Result:
Date
Jan-05-1999
Jan-08-1999

7.10.10 Views
A SQL View is a virtual table, which is based on SQL SELECT query. Essentially a view is very close to a real database table (it has columns and rows just like a regular table), except for the fact that the real tables store data, while the views do not. The view's data is generated dynamically when the view is referenced. A view references one or more existing database tables or other views. In effect every view is a filter of the table data referenced in it and this filter can restrict both the columns and the rows of the referenced tables.
A view is a consists of columns from one or more tables. Though it is similar to a table, it is stored in the database. It is a query stored as an object. Hence, a view is an object that derives its data from one or more tables. These tables are referred to as base or underlying tables. A view serves as a security mechanism. This ensures that users are able to retrieve and modify only the data seen by them. Users cannot see or access the remaining data in the underlying tables. A view also serves as a mechanism to simplify query execution. Complex queries can be stored in the form as a view, and data from the view can be extracted using simple queries.

7.10.11 Indexes
Indexes are created on columns in tables or views. The index provides a fast way to look up data based on the values within those columns. For example, if you create an index on the primary key and then search for a row

of data based on one of the primary key values, SQL Server first finds that value in the index, and then uses the index to quickly locate the entire row of data. Without the index, a table scan would have to be performed in order to locate the row, which can have a significant effect on performance.

You can create indexes on most columns in a table or a view. The exceptions are primarily those columns configured with large object (LOB) data types, such as image, text, and varchar (max). You can also create indexes on XML columns, but those indexes are slightly different from the basic index and are beyond the scope of this section. Instead, we will focus on those indexes that are implemented most commonly in a SQL Server database.

Index in SQL is created on existing tables to retrieve the rows quickly. When there are thousands of records in a table, retrieving information will take a long time. Therefore indexes are created on columns which are accessed frequently, so that the information can be retrieved quickly. Indexes can be created on a single column or a group of columns. When a index is created, it first sorts the data and then it assigns a ROWID for each row.

7.11 Queries and Sub Queries

A query is a request for data that is stored in SQL Server. A query can be issued by using several forms:

- An MS Query or Microsoft Access user can use a graphical user interface (GUI) to pick the data the user wants to see from one or more SQL Server tables.
- A user of SQL Server Management Studio or the osql utility can issue a SELECT statement.
- A client or middle tier-based application, such as a Microsoft Visual Basic application, can map the data from a SQL Server table into a bound control, such as a grid.

Although queries have various ways of interacting with a user, they all accomplish the same task: They present the result set of a SELECT statement to the user. Even if the user never specifies a SELECT statement, as is usually the case with graphical tools such as Visual Studio Query Designer, the client software transforms each user query into a SELECT statement that is sent to SQL Server.

The SELECT statement retrieves data from SQL Server and returns it to the user in one or more result sets. A result set is a tabular arrangement of the data from the SELECT. Like an SQL table, the result set is made up of columns and rows.

The full syntax of the SELECT statement is complex, but most SELECT statements describe four primary properties of a result set:

- The number and attributes of the columns in the result set. The following attributes must be defined for each result set column:
- The data type of the column.
- The size of the column, and for numeric columns, the precision and scale.
- The source of the data values returned in the column.
- The tables from which the result set data is retrieved, and any logical relationships between the tables.
- The conditions that the rows in the source tables must meet to qualify for the SELECT. Rows that do not meet the conditions are ignored.
- The sequence in which the rows of the result set are ordered.

The following SELECT statement finds the product ID, name, and list price of any products whose unit price exceeds 40:

```
SELECT ProductID, Name, ListPrice
FROM Production.Product
WHERE ListPrice > 40
ORDER BY ListPrice ASC
```

The column names listed after the SELECT keyword (ProductID, Name, and ListPrice) form the select list. This list specifies that the result set has three columns, and each column has the name, data type, and size of the associated column in the Product table. Because the FROM clause specifies only one base table, all column names in the SELECT statement refer to columns in that table.
The FROM clause lists the Product table as the one table from which the data is to be retrieved.
The WHERE clause specifies the condition that the only rows in the Product table that qualify for this SELECT statement are those rows in which the value of the ListPrice column is more than 40.
The ORDER BY clause specifies that the result set is to be sorted in ascending sequence (ASC) based on the value in the ListPrice column.

Sub Query

A subquery is a query that is nested inside a SELECT, INSERT, UPDATE, or DELETE statement, or inside another subquery. A subquery can be used anywhere an expression is allowed. In this example a subquery is used as a column expression named MaxUnitPrice in a SELECT statement.

```
USE Anuragi;
GO
SELECT Ord.SalesOrderID, Ord.OrderDate,
 (SELECT MAX(OrdDet.UnitPrice)
 FROM Anurag.Sales.SalesOrderDetail AS OrdDet
 WHERE Ord.SalesOrderID = OrdDet.SalesOrderID) AS MaxUnitPrice
FROM Anuragi.Sales.SalesOrderHeader AS Ord
```

A subquery is also called an inner query or inner select, while the statement containing a subquery is also called an outer query or outer select. Many Transact-SQL statements that include subqueries can be alternatively formulated as joins. Other questions can be posed only with subqueries. In Transact-SQL, there is usually no performance difference between a statement that includes a subquery and a semantically equivalent version that does not. However, in some cases where existence must be checked, a join yields better performance. Otherwise, the nested query must be processed for each result of the outer query to ensure elimination of duplicates. In such cases, a join approach would yield better results. The following is an example showing both a subquery SELECT and a join SELECT that return the same result set:

```
/* SELECT statement built using a subquery. */
SELECT Name
FROM Anuragi.Production.Product
WHERE ListPrice =
   (SELECT ListPrice
    FROM Anuragi.Production.Product
    WHERE Name = 'Chainring Bolts' );

/* SELECT statement built using a join that returns
  the same result set. */
SELECT Prd1. Name
FROM Anuragi.Production.Product AS Prd1
    JOIN Anuragi.Production.Product AS Prd2
     ON (Prd1.ListPrice = Prd2.ListPrice)
WHERE Prd2. Name = 'Chainring Bolts';
```

A subquery nested in the outer SELECT statement has the following components:

- A regular SELECT query including the regular select list components.
- A regular FROM clause including one or more table or view names.

- An optional WHERE clause.
- An optional GROUP BY clause.
- An optional HAVING clause.

The SELECT query of a subquery is always enclosed in parentheses. It cannot include a COMPUTE or FOR BROWSE clause, and may only include an ORDER BY clause when a TOP clause is also specified.

A subquery can be nested inside the WHERE or HAVING clause of an outer SELECT, INSERT, UPDATE, or DELETE statement, or inside another subquery. Up to 32 levels of nesting is possible, although the limit varies based on available memory and the complexity of other expressions in the query. Individual queries may not support nesting up to 32 levels. A subquery can appear anywhere an expression can be used, if it returns a single value.

If a table appears only in a subquery and not in the outer query, then columns from that table cannot be included in the output (the select list of the outer query).

Statements that include a subquery usually take one of these formats:

- WHERE *expression* [NOT] IN **(***subquery***)**
- WHERE *expression comparison_operator* [ANY | ALL] **(***subquery***)**
- WHERE [NOT] EXISTS **(***subquery***)**

Self Assessment Question

5. The transaction control statements are……..

(a) COMMIT (b) SELECT

(c) UPDATE (d) None of these

6. The ……..is the command used to insert new data (a new row) into a table by specifying a list of values to be inserted into each table column.

(a) Update command (b) Insert command

(c) Delete command (d) None of these

7. Update is used to update one table columns with specified values. You can update one or multiple columns at the same time.

(a) Delete command (b) Insert command

(c) Update command (d) None of these

8. …….is used to remove an object from the database.

(a) DROP command (b) TRUNCATE statement:

(c) Update command (d) None of these

9. A Derived table is a table expression that appears in the FROM clause of a query.

(a) True (b) False

7.12 Aggregate Functions

Aggregate functions perform a calculation on a set of values and return a single value. Except for COUNT, aggregate functions ignore null values. Aggregate functions are frequently used with the GROUP BY clause of the SELECT statement. All aggregate functions are deterministic. This means aggregate functions return the same value any time that they are called by using a specific set of input values.

- Average: avg
- Minimum: min

- Maximum: max
- Total: sum
- Count: count

The input to sum and avg must be collection of numbers but the other operation can operate on collections of nonnumeric data types, such as strings, as well.

As an illustration, consider the query "Find the average account balance at the Perryridge branch." We write this query as follows:

select avg *(balance)*
from *account*
where *branch-name* = "Perryridge"

The result of this query is a relation with a single attribute, containing a single row with a numerical value corresponding to the average balance at the Perryridge branch. We can optionally give a name to the attribute of the result relation using the as clause.

There is circumstance where we would like to apply the aggregate function not only to a single set of tuple, but also to a group of sets of tuples; we specify this wish in SQL using the group by clause.

As an illustration, consider the query "Find the average account balance at each branch." We write this query as follows:

select *branch-name,*
avg *(balance)* from *account*
group by *branch-name*

Retaining of duplicates is important in computing an average. Suppose that the account balances at the (small) Brighton branch are 1000, 3000, 2000, and 1000. The average balance is 7000/4 = 1750.00. If duplicates were eliminated, we would obtain the wrong answer (6000/3 =2000).

There are cases where we must eliminate duplicates prior to computing an aggregate function. If we do want to eliminate duplicates, we use the keyword distinct in the aggregate expression. An example arises in the query "Find the number of depositors for each branch." In this case, a depositor counts only once, regardless of the number of accounts that depositor may have. We write this query as follows:

select *branch-name,* count (distinct *customer- . name)* from *depositor, account* where *depositor.account-number =, account.account-number* group *by branch-name*

At times, it is useful to state a condition that applies to groups rather than to tuples. For example, we might be interested in only those branches where the average account balance is more than $1200. This condition does not apply to a single tuple; rather, it applies to each group constructed by the group by clause. To express such a query, we use the having clause of SQL. Predicates in the having clause are applied after the formation of groups, so aggregate functions may be used. We express this query in SQL as follows:

select *branch-name, avg (balance)* from *account* group *by branch-name* having avg *(balance)* > 1200

At times, we wish to treat the entire relation as a single group. In such cases, we do not use a group by clause. Consider the query "Find the average balance for all accounts." We write this query as follows:

select avg *(balance)* from *account*

We use the aggregate function count frequently to count the number of tuples in a relation. The notation for this function in SQL is count **(*).** Thus, to find the number of tuples in the *customer* relation, we write

select count (*) from *customer*

SQL does not allow the use of distinct with count(*). It is legal to use distinct with max and min, even though the result does not change. We can use the keyword all in place of distinct to specify duplicate retention, but, since all is the default, there is no need to do so.

If a where clause and a having clause appear in the same query, the predicate in the where clause is applied first. Tuples satisfying the where predicate are then placed into groups by the group by clause. The having clause, if it is present, is then applied to each group; the groups that do not satisfy the having clause predicate are removed. The remaining groups are used by the select clause to generate tuples of the result of the query.

Aggregate functions can be used Table 7.4 and expressions only in the following:

- The select list of a SELECT statement (either a subquery or an outer query).
- A COMPUTE or COMPUTE BY clause.
- A HAVING clause.

Table 7.4: Transact-SQL provides the following aggregate functions:

AVG	MIN
CHECKSUM_AGG	OVER Clause
COUNT	ROWCOUNT_BIG
COUNT_BIG	STDEV
GROUPING	STDEVP
GROUPING_ID	SUM
MAX	VAR
	VARP

7.13 Cursors in SQL

A cursor is a temporary work area created in the system memory when a SQL statement is executed. A cursor contains information on a select statement and the rows of data accessed by it. This temporary work area is used to store the data retrieved from the database, and manipulate this data. A cursor can hold more than one row, but can process only one row at a time. The set of rows the cursor holds is called the active set.

In SQL procedures, a cursor make it possible to define a result set (a set of data rows) and performs complex logic on a row by row basis. By using the same mechanics, an SQL procedure can also define a result set and return it directly to the caller of the SQL procedure or to a client application. A cursor can be viewed as a pointer to one row in a set of rows. The cursor can only reference one row at a time, but can move to other rows of the result set as needed.

To use cursors in SQL procedures, you need to do the following:

1. Declare a cursor that defines a result set.
2. Open the cursor to establish the result set.
3. Fetch the data into local variables as needed from the cursor, one row at a time.
4. Close the cursor when done

To work with cursors you must use the following SQL statements:

- DECLARE CURSOR
- OPEN
- FETCH
- CLOSE

DECLARE CURSOR defines the attributes of a Transact-SQL server cursor, such as its scrolling behaviour and the query used to build the result set on which the cursor operates. The OPEN statement populates the result set, and FETCH returns a row from the result set. The CLOSE statement releases the current result set associated with the cursor. The DEALLOCATE statement releases the resources used by the cursor.
The first form of the DECLARE CURSOR statement uses the ISO syntax for declaring cursor behaviors. The second form of DECLARE CURSOR uses Transact-SQL extensions that allow you to define cursors using the same cursor types used in the database API cursor functions of ODBC or ADO.
You cannot mix the two forms. If you specify the SCROLL or INSENSITIVE keywords before the CURSOR keyword, you cannot use any keywords between the CURSOR and FOR *select_statement* keywords. If you specify any keywords between the CURSOR and FOR *select_statement* keywords, you cannot specify SCROLL or INSENSITIVE before the CURSOR keyword.
If a DECLARE CURSOR using Transact-SQL syntax does not specify READ_ONLY, OPTIMISTIC, or SCROLL_LOCKS, the default is as follows:

- If the SELECT statement does not support updates (insufficient permissions, accessing remote tables that do not support updates, and so on), the cursor is READ_ONLY.
- STATIC and FAST_FORWARD cursors default to READ_ONLY.
- DYNAMIC and KEYSET cursors default to OPTIMISTIC.

Cursor names can be referenced only by other Transact-SQL statements. They cannot be referenced by database API functions. For example, after declaring a cursor, the cursor name cannot be referenced from OLE DB, ODBC or ADO functions or methods. The cursor rows cannot be fetched using the fetch functions or methods of the APIs; the rows can be fetched only by Transact-SQL FETCH statements.

Declaring PL/SQL Cursor

To use PL/SQL cursor, you must first declare it in the declaration section in a package as follows:
PL/SQL Declaration Syntax

```
CURSOR cursor_name [ ( [ parameter_1 [, parameter_2 ...] ) ]
    [ RETURN return_specification ]
IS sql_select_statements
[FOR UPDATE [OF [column_list]];
```

- First you declare the name of cursor *cursor_name* after the keyword CURSOR. The name of cursor can have up to 30 characters in length and follows the rules of identifiers in PL/SQL. It is important to note that cursor's name is not a variable so you cannot use it as a variable such as assign it to other cursor or use it in an expression.
- *parameter1, parameter2…* are optional section in cursor declaration. These parameter allows you to pass arguments into the cursor.
- RETURN return_specification is an optional part
- Next you specify the valid SQL statement which returns a result set where the cursor points to.
- Finally you can indicate a list of columns you want to update after the FOR UPDATE OF. This part is optional so you can omit it in the CURSOR declaration.

PL/SQL Cursor Declaration Example

```
CURSOR cur_chief IS
    SELECT first_name,
        last_name,
        department_name
   FROM employees e
    INNER JOIN departments d ON d.manager_id = e.employee_id;
```

Opening a PL/SQL Cursor

After declaring a cursor you can use open it by following the below syntax:

Opening PL/SQL Cursor Syntax

OPEN cursor_name [(argument_1 [, argument_2 ...])];

You have to specify the cursor's name cursor_name after the keyword OPEN. If the cursor was defined with a parameter list, you need to pass corresponding arguments to the cursor also. When you OPEN the cursor, PL/SQL executes the SQL SELECT statement and identifies the active result set. Note that the OPEN action does not actually retrieve records from database. It happens in the FETCH step. If the cursor was declared with FOR UPDATE clause, PL/SQL locks all the records in the result set.

We can open our cursor *cur_chief* above as follows:

Open PL/SQL Cursor Example

```
OPEN cur_chief;
```

Fetching Records from PL/SQL Cursor

Once cursor is open, you can fetch data from the cursor into a record that has structure corresponding to the cursor. You can also fetch data from cursor to a list of variables. The fetch action retrieves data and fill the record or variable. You then can manipulate this data in memory. You can fetch the data until there is no record found in active result set. The syntax of FETCH is as follows:

PL/SQL Cursor FETCH Syntax

FETCH cursor_name INTO RECORD OR variables

You can test the cursor's attribute %FOUND or %NOTFOUND to check if the fetch against the cursor succeeded. There are more cursor's attributes which will cover in the next section.

We can use PL/SQL LOOP statement together with the FETCH to loop through all records in active result set as follows:

PL/SQL Cursor Fetch Example

```
LOOP
   -- fetch information from cursor into record
   FETCH cur_chief INTO r_chief;

   EXIT WHEN cur_chief%NOTFOUND;

   -- print department - chief
   DBMS_OUTPUT.PUT_LINE(r_chief.department_name || ' - ' ||
                  r_chief.first_name || ',' ||
                  r_chief.last_name);
 END LOOP;
```

Closing PL/SQL Cursor

You should always close the cursor when you are done with it. Otherwise you will have memory leak in your program which is not good. The close cursor syntax is very simple as follows:

PL/SQL code

```
CLOSE cursor_name;
```

Let's now explore the different type of cursors in SQL server.

- Static cursor
- Dynamic cursor
- Forward-only cursor
- Keyset cursor

7.14 Summary

- Structured Query Language, commonly abbreviated to SQL and pronounced as sequel, is not a conventional computer programming language in the normal sense of the phrase.
- Subtypes increase reliability, provide compatibility with ANSI/ISO types, and improve readability by indicating the intended use of constants and variables.
- The SQL commands are lines of SQL code that ask the SQL application to perform simple tasks against with data inside of a database.
- Embedded SQL statements place DDL, DML, and transaction control statements within a procedural language program. Embedded SQL is supported by the Oracle precompilers and is documented.
- The SQL Update is used to update one table columns with specified values. You can update one or multiple columns at the same time.

7.15 Keywords

Cursor: It is a temporary work area created in the system memory when a SQL statement is executed.
DDL: It is provides commands for defining relation schemas, deleting relations, creating indices, and modifying relation schemas.
Join: It is used in an SQL statement to query data from two or more tables, based on a relationship between certain columns in these tables.
SQL (Structured Query Language): It is a computer language aimed to store, manipulate, and query data stored in relational databases.
UNION operator: is used to combine the result-set of two or more SELECT statements. That each SELECT statement within the UNION must have the same number of columns. UNION operator is used to combine the result-set of two or more SELECT statements. That each SELECT statement within the UNION must have the same number of columns.
View: It is a virtual table, which is based on SQL SELECT query.

7.16 Review Questions

1. Define SQL.
2. Explain the characteristics of SQL.
3. What is the advantage of SQL? Explain briefly.
4. Define data types in SQL.
5. Describes the table and its statement.
6. Explain the join in SQL? Explain with explain.
7. What is the main purpose of embedded SQL?
8. Write the short notes on queries and sub queries.
9. Describes the aggregate functions.
10. Explain the cursor in SQL.

Answers for Self Assessment Questions

1. (a) 2. (b) 3. (a) 4. (c) 5(a) 6. (b) 7. (c) 8. (a) 9. (a)

7.17 Review Questions

1. The Language of SQL By Larry Rockoff.

8

Database Administration

STRUCTURE

8.0 Objectives

8.1 Introduction

8.2 Database Administration

8.3 Failure Classification

8.4 Storage Hierarchy

8.5 RAID

8.6 Transaction Model

8.7 Data Dictionary Storage (DDS)

8.8 Storage and File Structures

8.9 File Organization

8.10 Summary

8.11 Keywords

8.12 Review Questions

8.13 Further Studies

8.0 Objectives

After studying this chapter, you will be able to:

- Define database administration
- Discuss about the failure classification in database administration
- Describes the RAID
- Discuss about the transaction model
- Explain the data dictionary storage

8.1 Introduction

Many database administrators and programmers are faced with tables and structures designed by others, perhaps created many years ago. In order to conceptualize database objects and structures, programmers need to understand the capabilities of modern database systems and how to retrieve database metadata.

In this chapter, students will learn how to manipulate the data stored in tables and to return meaningful results to help analyze the data stored. From beginning to end, participants will learn by doing SQL-based projects in their own MySQL shell, and then handing them in for instructor feedback. These projects, as well as the final

project (developing tables for a blog), will add to the student's portfolio and will contribute to certificate completion.

8.2 Database Administration

The term database administration state as how the databases are handled properly, how can manipulate and how can be keep as secure. The person who cares for the entire database is called as database administrator.
A database administrator (short form DBA) is a person responsible for the installation, configuration, upgrade, administration, monitoring and maintenance of physical databases.
The role includes the development and design of database strategies, monitoring and improving database performance and capacity, and planning for future expansion requirements. They may also plan, co-ordinate and implement security measures to safeguard the database.
Each database requires at least one database administrator (DBA). A Database system can be large and can have many users. Therefore, database administration is sometimes not a one-person job, but a job for a group of DBAs who share responsibility.
The database administrator (DBA) is the person (or group of people) responsible for overall control of the database system. The DBA's responsibilities include the following:

- Deciding the information content of the database, i.e. identifying the entities of interest to the enterprise and the information to be recorded about those entities. This is defined by writing the conceptual schema using the DDL.
- Deciding the storage structure and access strategy, i.e. how the data is to be represented by writing the storage structure definition. The associated internal/conceptual schema must also be specified using the DDL.
- Liaising with users, i.e. to ensure that the data they require is available and to write the necessary external schemas and conceptual/external mapping (again using DDL).
- Defining authorization checks and validation procedures. Authorization checks and validation procedures are extensions to the conceptual schema and can be specified using the DDL.
- Defining a strategy for backup and recovery. For example periodic dumping of the database to a backup tape and procedures for reloading the database for backup. Use of a log file where each log record contains the values for database items before and after a change and can be used for recovery purposes.
- Monitoring performance and responding to changes in requirements, i.e. changing details of storage and access thereby organizing the system so as to get the performance that is best for the enterprise.

8.2.1 Role of Database Administrator

A database administrator (DBA) directs or performs all activities related to maintaining a successful database environment. Responsibilities include designing, database design and modeling implementing, and maintaining the database system; establishing policies and procedures pertaining to the management, security, maintenance, and use of the database management system; and training employees in database management and use. A DBA is expected to stay of emerging technologies and new design approaches. Typically, a DBA has either a degree in computer science or some on-the-job training with a particular database product or more extensive experience with a range of database products. A DBA is usually expected to have experience with one or more of the major database management products, such as Structured Query Language, SAP, and Oracle-based database management software.

- Installation, configuration and upgrading of Microsoft SQL Server/MySQL/Oracle server software and related products.
- Evaluate MSSQL/MySQL/Oracle features and MSSQL/MySQL/Oracle related products.
- Establish and maintain sound backup and recovery policies and procedures.

- Take care of the database design and implementation.
- Implement and maintain database security
- Database tuning and performance monitoring.
- Application tuning and performance monitoring.
- Setup and maintain documentation and standards.
- Plan growth and changes (capacity planning).
- Do general technical trouble shooting and give consultation to development teams.
- Interface with MSSQL/MySQL/Oracle for technical support.

8.2.2 Functions of a Database Administrator

One of the main reasons for using DBMS is to have a central control of both data and the programs accessing those data. A person who has such control over the system is called a Database Administrator (DBA). The following are the functions of a database administrator:

Schema Definition

- Storage structure and access method definition.
- Schema and physical organization modification.
- Granting authorization for data access.
- Routine Maintenance.
- Schema Definition.

The Database administrator creates the database schema by executing DDL statements. Schema includes the logical structure of database table (Relation) like data types of attributes, length of attributes, integrity constraints etc. Database tables or indexes are stored in the following ways: Flat files, Heaps, B+ Tree etc.

- Schema and physical organization modification.
- The DBA carries out changes to the existing schema and physical organization.
- Granting authorization for data access.

The DBA provides different access rights to the users according to their level. Ordinary users might have highly restricted access to data, while you go up in the hierarchy to the administrator, you will get more access rights.

Routine Maintenance

Some of the routine maintenance activities of a DBA are given below.

- Taking backup of database periodically.
- Ensuring enough disk space is available all the time.
- Monitoring jobs running on the database.
- Ensure that performance is not degraded by some expensive task submitted by some users.
- Performance tuning.

8.2.3 Importance of DBA

DBA has very important role in the company but it is often not defined completely. There is quite a difference in DBA role for small companies to large enterprises. With the SQL Server availability at affordable price and growing information needs, many small companies are hiring database administrators in the company. It is better to keep a diary or record of tasks, and their results. It will allow you to start a baseline and gather metrics. It can be from a simple spreadsheet or extensive web based application that keeps track of tasks, results and other gathered data. Each day new records or tab in excel is created and populated. As you complete tasks, the results are filled in. Those tasks that do not get completed will have no results associated with them. These way administrators can look back on specific days and see the results on any particular time or point of time or which items were not done. Every server that we are responsible for has a data column,

where individual data can be collected, on a per server basis. Some of the information can be automated as daily job to report server and jobs health. These are the common tasks that DBAs need to check on daily or continuous basis.

Review Os Error Events.

It is good practice to review errors that appear in the application, security, and system sections in computer management, in event viewer. Perform filter for each of these sections by errors and look for anything that has occurred since the last time this process was completed. Anything that appears critical or unusual in these sections, detail it in record keeping and may even have to dig in and find out the reasons for it, and mediate it. This task can be quick, or occur over a few days, depending on the events that encountered.

Review SQL Server Logs

Something that is often missed is simply looking into the SQL Server Logs. Reviewing these logs sometimes gives good idea where the actual problem is and you can solve the issues before user complain about them. The good knowledge of server logs is very helpful in identifying and solving complex issues. Checking and documenting SQL logs regularly is good practice, kind of pro-active maintenance for SQL Servers and lead to solve complex nature of problems.

Connectivity

Check connectivity of server by logging into the custom application and verify that it can connect to the database and pull up data. Verify that it is performing at an acceptable speed. In the event of a failure, check with Application Support Group, DBA group, and the DBA manager, before proceeding to resolve the issue.

Disk Usage and Free Space on Drives

It is very important to monitor free space to ensure that database do not run out of space. There may be other processes that take up space on servers. This often occurs when data or log file grow out of bounds, and the results are often wild and unpredictable. This task may be more relevant if you are running replication and link break between distribution server and subscriber server. It is worth noting and keeping track. Checking this once/twice a day is good practice or set up alerts when goes below threshold level or daily reports for the server.

Check Backups

There are many different solutions used by companies for backing up nightly or periodic transaction logs. Typically companies use maintenance plans, 3rd party solutions, or custom scripts to ensure backups of the systems. Whatever the solution used, make it sure to check it as often as you can while on call. Ensure that backups are being processed properly. It is also important to check backup by occasionally restoring on test DB servers. Unfortunately, most people that implement this task usually do so after finding no exhausting backups for a period. Do not let this happen.

System Specific Output Files

There may be some jobs and tasks where we can set up output, log files, reports and SQL system dumps. Checking the output files, reports, and SQL system dumps etc., is important to check periodically. These will be for a variety of reasons. Identify them, document them, and then monitor them.

Log and Data File Sizes

To keep a handle on the growth of your systems, you should devise a way to monitor and keep tabs on the sizes of your database files. A simple solution is to run a query that gathers all this info, and paste it into a spreadsheet. More complex solutions could be implemented. The end result needs to be that you know the sizes of these files, and be able to have metrics over time to help you plan and monitor those systems. Doing this task on a daily basis, while on-call, will help keep tabs on growth and expected results.

Replication health check

If you have replication setup on your systems, it is important to monitor and document. Check if the replication is functioning properly, check latency, peak latency, compared to non-peak times. Replication can

be checked using native replication monitor or third party replication monitoring software. Gathering the data associated with it and creating a baseline will help solve future issues as well.

Other things of Interest

During DBA's on-call rotation, you may encounter odd things that need to be noted. Make sure to comment or report on these and document them. If find any resolution of them, document this as well. This will be very helpful to other individuals and will greatly appreciate notes and observations of these odd occurrences. If persistent, may need to put in check list and work on finding the resolution.

Performance

Check Performance statistics for all servers using the monitoring tool, research and resolve any issues. Check performance monitor counters on production servers and verify that all are within the normal range. Check for any abnormal behaviors by comparing collected statistics information. Documents and record counters for analysis.

Specific Needs

Every company has their own specific needs and may need to come up with more of these tasks. There may have more tasks that do not appear on this list, that need to be added. Share if you come up with other kind of tasks.

If we can take the time, when we are not head-long into problems of the day, we can better gather our wits and devise solutions to make jobs easier, more automated, and successful. This is an important, so that you are not troubleshooting all the time, but have an action plan to resolve issues as they arise. Keeping a record and documentation of these tasks, and results is a sure way to see patterns and way to fix those pesky problems that always seem to get placed on the back burner. By creating metrics to measure, DBA's can spend more time on the important tasks, and not just fix things as they appear.

8.3 Failure Classification

The major types of failures involving data integrity (as opposed to data security) are:

8.3.1 Transaction Failure

Logical Error: The transaction cannot continue with its normal execution because of such things as bad input, data not found, or resource limit exceeded.

System Error: The system has entered an undesirable state (for example, deadlock), as a result of which a transaction cannot continue with its normal execution. The transaction, however, can be re-executed at a later time.

System Crash: There is a hardware malfunction, or a bug in the database software or the operating system, that causes the loss of the content of volatile storage, and brings transaction processing to a halt. The content of the nonvolatile storage remains intact, and is not corrupted.

Disk Failure: A disk block loses its contents as a result of either a head crash or failure during a data transfer. Copies of data on other disks, or archival backups on tertiary media, such as tapes, are used to recover from the failure.

8.4 Storage Hierarchy

A data stored in a hierarchical format. This time the data is arranged logically in a top-down format. In a hierarchical database, data is grouped in records, which are subdivided into a series of segments.

Primary storage: Fastest media but volatile (cache, main memory).

Secondary storage : next level in hierarchy, non-volatile, moderately fast access time also called on-line storage e.g. flash memory, magnetic disks tertiary storage lowest level in hierarchy, non-volatile, slow access time also called off-line storage E.g. magnetic tape, optical storage. Magnetic Hard Disk Mechanism note Diagram is schematic, and simplifies the structure of actual disk drives

Magnetic Disks: Read-write head Positioned very close to the platter surface (almost touching it) reads or writes magnetically encoded information. Surface of platter divided into circular tracks Over 16,000 tracks per platter on typical hard disks. Each track is divided into sectors. A sector is the smallest unit of data that can be read or written. Sector size typically 512 bytes Typical sectors per track: 200 (on inner tracks) to 400 (on outer tracks) To read/write a sector disk arm swings to position head on right track platter spins continually; data is read/written as sector passes under head Head-disk assemblies multiple disk platters on a single spindle (typically 2 to 4) one head per platter, mounted on a common arm. Cylinder it consists of ith track of all the platters.
Earlier generation disks were susceptible to head-crashes surface of earlier generation disks had metal-oxide coatings which would disintegrate on head crash and damage all data on disk.
Current generation disks are less susceptible to such disastrous failures, although individual sectors may get corrupted Disk controller - interfaces between the computer system and the disk drive hardware. accepts high-level commands to read or write a sector initiates actions such as moving the disk arm to the right track and actually reading or writing the data Computes and attaches checksums to each sector to verify that data is read back correctly If data is corrupted, with very high probability stored checksum would not match recomputed checksum Ensures successful writing by reading back sector after writing it Performs remapping of bad sectors.

Optimization of disk-block access block: A contiguous sequence of sectors from a single track data is transferred between disk and main memory in blocks sizes range from 512 bytes to several kilobytes Smaller blocks: more transfers from disk larger blocks: more space wasted due to partially filled blocks Typical block sizes today range from 4 to 16 kilobytes Disk-arm-scheduling algorithms order pending accesses to tracks so that disk arm movement is minimized elevator algorithm move disk arm in one direction (from outer to inner tracks or vice versa), processing next request in that direction, till no more requests in that direction, then reverse direction and repeat.

Optimization of disk block access file organization: Optimize block access time by organizing the blocks to correspond to how data will be accessed e.g. Store related information on the same or nearby cylinders. Files may get fragmented over time E.g. if data is inserted to/deleted from the file or free blocks on disk are scattered, and newly created file has its blocks scattered over the disk Sequential access to a fragmented file results in increased disk arm movement Some systems have utilities to defragment the file system, in order to speed up file access.

8.5 RAID

The RAID stands for (Redundant Arrays of Independent Disks). RAID is the use of multiple disks and data distribution techniques to get better Resilience and/or Performance. RAID can be implemented in Software or Hardware or any combination of both. This presentation is a simple introduction to the RAID levels with some information on Caching and different I/O profiles.

Redundant Arrays of Independent Disks: Disk organization techniques that manage a large numbers of disks, providing a view of a single disk of high capacity and high speed by using multiple disks in parallel, and high

reliability by storing data redundantly, so that data can be recovered even if a disk fails The chance that some disk out of a set of N disks will fail is much higher than the chance that a specific single disk will fail. e.g., a system with 100 disks, each with MTTF of 100,000 hours (approx. 11 years), will have a system MTTF of 1000 hours (approx. 41 days) Techniques for using redundancy to avoid data loss are critical with large numbers of disks Originally a cost-effective alternative to large, expensive disks I in RAID originally stood for inexpensive'' Today RAIDs are used for their higher reliability and bandwidth. The "I" is interpreted as independent.

Improvement of Reliability via Redundancy: Redundancy store extra information that can be used to rebuild information lost in a disk failure e.g., Mirroring (or shadowing) Duplicate every disk. Logical disk consists of two physical disks. Every write is carried out on both disks reads can take place from either disk If one disk in a pair fails, data still available in the other data loss would occur only if a disk fails, and its mirror disk also fails before the system is repaired Probability of combined event is very small Except for dependent failure modes such as fire or building collapse or electrical power surges Mean time to data loss depends on mean time to failure, and mean time to repair E.g. MTTF of 100,000 hours, mean time to repair of 10 hours gives mean time to data loss of 500*10 6 hours (or 57,000 years) for a mirrored pair of disks (ignoring dependent failure modes).

Improvement in Performance via Parallelism: Two main goals of parallelism in a disk system:
1. Load balance multiple small accesses to increase throughput
2. Parallelize large accesses to reduce response time.

Improve transfer rate by striping data across multiple disks. Bit-level striping – split the bits of each byte across multiple disks In an array of eight disks, write bit i of each byte to disk i. Each access can read data at eight times the rate of a single disk. But seek/access time worse than for a single disk Bit level striping is not used much anymore Block-level striping – with n disks, block i of a file goes to disk (i mod n) + 1 Requests for different blocks can run in parallel if the blocks reside on different disks A request for a long sequence of blocks can utilize all disks in parallel.

RAIDs are Redundant Arrays of Inexpensive Disks. There are six levels of organizing these disks:

- 0 -Non-redundant Striping
- 1 - Mirrored Disks
- 2 - Memory Style Error Correcting Codes
- 3 - Bit Interleaved Parity
- 4 - Block Interleaved Parity
- 5 - Block Interleaved Distributed Parity
- 6 - P + Q Redundancy

RAID Levels Schemes to provide redundancy at lower cost by using disk striping combined with parity bits Different RAID organizations, or RAID levels, have differing cost, performance and reliability characteristics:
RAID Level 1: Mirrored disks with block striping Offers best write performance. Popular for applications such as storing log files in a database system. RAID Level 0 Block striping; non-redundant. Used in high-performance applications where data lost is not critical.

RAID Level 2: Memory-Style Error-Correcting-Codes (ECC) with bit striping. RAID Level 3: Bit-Interleaved Parity a single parity bit is enough for error correction, not just detection, since we know which disk has failed when writing data, corresponding parity bits must also be computed and written to a parity bit disk. To recover data in a damaged disk, compute XOR of bits from other disks (including parity bit disk).

RAID Level 3: Faster data transfer than with a single disk, but fewer I/Os per second since every disk has to

participate in every I/O. Subsumes Level 2 (provides all its benefits, at lower cost). ***RAID Level 4:*** Block-Interleaved Parity; uses block-level striping, and keeps a parity block on a separate disk for corresponding blocks from N other disks. When writing data block, corresponding block of parity bits must also be computed and written to parity disk.
It provides higher I/O rates for independent block reads than Level 3 block read goes to a single disk, so blocks stored on different disks can be read in parallel Provides high transfer rates for reads of multiple blocks than no-striping Before writing a block, parity data must be computed Can be done by using old parity block, old value of current block and new value of current block (2 block reads + 2 block writes) Or by recomputing the parity value using the new values of blocks corresponding to the parity block More efficient for writing large amounts of data sequentially Parity block becomes a bottleneck for independent block writes since every block write also writes to parity disk.

***RAID Level 5* :** Block-Interleaved Distributed Parity ; partitions data and parity among all N + 1 disks, rather than storing data in N disks and parity in 1 disk e.g., with 5 disks, parity block for nth set of blocks is stored on disk (n mod 5) + 1, with the data blocks stored on the other 4 disks. Higher I/O rates than Level 4. Block writes occur in parallel if the blocks and their parity blocks are on different disks. Subsumes Level 4: provides same benefits, but avoids bottleneck of parity disk.

RAID Level 6: P+Q Redundancy scheme; similar to Level 5, but stores extra redundant information to guard against multiple disk failures. Better reliability than Level 5 at a higher cost; not used as widely.
Choice of RAID Level Factors in choosing RAID level Monetary cost Performance, Number of I/O operations per second, and bandwidth during normal operation Performance during failure Performance during rebuild of failed disk Including time taken to rebuild failed disk RAID 0 is used only when data safety is not important e.g. data can be recovered quickly from other sources Level 2 and 4 never used since they are subsumed by 3 and 5 Level 3 is not used anymore since bit-striping forces single block reads to access all disks, wasting disk arm movement, which block striping (level 5) avoids Level 6 is rarely used since levels 1 and 5 offer adequate safety for almost all applications So competition is between 1 and 5 only.
Level 1 provides much better write performance than level 5 Level 5 requires at least 2 block reads and 2 block writes to write a single block, whereas Level 1 only requires 2 block writes Level 1 preferred for high update environments such as log disks Level 1 had higher storage cost than level 5 disk drive capacities increasing rapidly (50%/year) whereas disk access times have decreased much less (x 3 in 10 years) I/O requirements have increased greatly, e.g. for Web servers When enough disks have been bought to satisfy required rate of I/O, they often have spare storage capacity so there is often no extra monetary cost for Level 1! Level 5 is preferred for applications with low update rate, and large amounts of data Level 1 is preferred for all other applications.

Hardware Issues Software RAID: RAID implementations done entirely in software, with no special hardware support. RAID implementations with special hardware use non-volatile RAM to record writes that are being executed.
Beware: power failure during write can result in corrupted disk e.g. failure after writing one block but before writing the second in a mirrored system Such corrupted data must be detected when power is restored, recovery from corruption is similar to recovery from failed disk NV-RAM helps to efficiently detected potentially corrupted blocks otherwise all blocks of disk must be read and compared with mirror/parity block.

Did You Know?
The RAID has its origin starting from the year of 1980. At that time it was referred as the Redundant Array of Inexpensive Disks.

Caution
Before configuring RAID, be sure to back up all data. Once a RAID configuration is executed and saved, data on the selected disks will no longer be retrievable.

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: Search latest RAID model which available in market.

……..……

……..……

……...

8.6 Transaction Model

A transaction is a set of related and interdependent steps that form a logical unit of work. Grouping the steps into a transaction ensures that the set of steps are treated as an atomic unit: either the entire set of steps succeeds, or the entire set of steps fails.
Figure 8.1 illustrates a database transaction that contains two updates, update 1 and update 2.

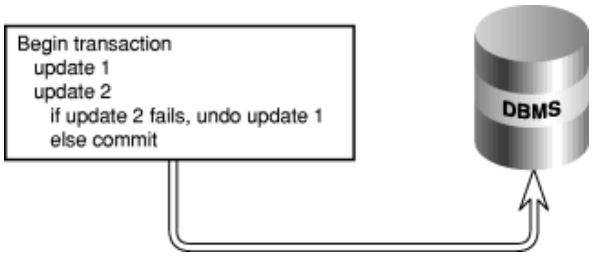


Figure 8.1: A database transaction.

As a transaction is executing, it lock the part of the database that it is modifying, making the data unavailable for modifications by other transactions. Isolation of data from outside interference is one of the defining qualities of transactions.
If the transaction's operations succeed, the transaction completes by writing changes to disk in a **commit** operation. A commit operation releases the transaction's locks, making the updated data available to other transactions.
If an error occurs partway through execution, making it impossible for the entire transaction to succeed, the entire transaction fails. Rather than leave partial results in the database, the transaction backs out the changes that it has already made, leaving the database with the values that it had before the transaction started. The back-out process is called **rollback.**
Another type of database transaction, two-phase commit, is adapted for distributed use across multiple databases. Two-phase commit uses a transaction monitor to coordinate concurrent updates. The transaction monitor first checks that all databases can make the desired change. Even if all can make the change, they must wait for the transaction monitor's signal before doing so. If all cannot make the change, none do.
Every transaction must maintain data consistency in its database, but the two-phase commit protocol extends the scope of this requirement beyond individual databases. A common example is a funds transfer, in which the transaction monitor ensures that the funds debited from one account are credited in the other account or that neither account is modified.

The ACID model is one of the oldest and most important concepts of database theory. It sets forward four goals that every database management system must strive to achieve: atomicity, consistency, isolation and durability. No database that fails to meet any of these four goals can be considered reliable.

Atomicity

Atomicity states that database modifications must follow an "all or nothing" rule. Each transaction is said to be "atomic." If one part of the transaction fails, the entire transaction fails. It is critical that the database management system maintain the atomic nature of transactions in spite of any DBMS, operating system or hardware failure.

Consistency

Consistency states that only valid data will be written to the database. If, for some reason, a transaction is executed that violates the database's consistency rules, the entire transaction will be rolled back and the database will be restored to a state consistent with those rules. On the other hand, if a transaction successfully executes, it will take the database from one state that is consistent with the rules to another state that is also consistent with the rules.

Isolation

Isolation requires that multiple transactions occurring at the same time not impact each other's execution. For example, if Joe issues a transaction against a database at the same time that Mary issues a different transaction; both transactions should operate on the database in an isolated manner. The database should either perform Joe's entire transaction before executing Mary's or vice-versa. This prevents Joe's transaction from reading intermediate data produced as a side effect of part of Mary's transaction that will not eventually be committed to the database. Note that the isolation property does not ensure which transaction will execute first, merely that they will not interfere with each other.

Durability

Durability ensures that any transaction committed to the database will not be lost. Durability is ensured through the use of database backups and transaction logs that facilitate the restoration of committed transactions in spite of any subsequent software or hardware failures.

Did You Know?

The Transactional Analysis theories were started to develop by Eric Berne in 1950's.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Prepare a structure of ATM transaction model.

……..……

……..……

……...

8.7 Data Dictionary Storage (DDS)

The database also needs to store information about the relations, known as the data dictionary. This includes:

- Names of relations.
- Names of attributes of relations.
- Domains and lengths of attributes.
- Names and definitions of views.

- Integrity constraints (e.g., key constraints).

Plus data on the system users:

- Names of authorized users.
- Accounting information about users.

Plus (possibly) statistical and descriptive data:

- Number of tuples in each relation.
- Method of storage used for each relation (e.g., clustered or non-clustered).

When we will also see a need to store information about each index on each relation:

- Name of the index.
- Name of the relation being indexed.
- Attributes the index is on.
- Type of index.

This information is, in itself, a miniature database. We can use the database to store data about itself, simplifying the overall structure of the system, and allowing the full power of the database to be used to permit fast access to system data.

RDBMS needs to maintain data about the relations, such as the schema. This is stored in a data dictionary (sometimes called a system catalog):

- Names of the relations
- Names of the attributes of each relation
- Domains and lengths of attributes
- Names of views, defined on the database, and definitions of those views
- Integrity constraints
- Names of authorized users
- Accounting information about users
- Number of tuples in each relation
- Method of storage for each relation (clustered/non-clustered)
- Name of the index
- Name of the relation being indexed
- Attributes on which the index in defined
- Type of index formed

8.7.1 DDS Benefits

Benefits of a DDS are mainly due to the fact that it is a central store of information about the database.

Benefits include:

- Improved documentation and control.
- Consistency in data use.
- Easier data analysis.
- Reduced data redundancy.
- Simpler programming.
- The enforcement of standards.
- Better means of estimating the effect of change.

8.7.2 DDS Facilities

- A DDS should provide two sets of facilities:
- To record and analyze data requirements independently of how they are going to be met - conceptual data models (entities, attributes, relationships).

- To record and design decisions in terms of database or file structures implemented and the programs which access them internal schema.
- One of the main functions of a DDS is to show the relationship between the conceptual and implementation views.

Caution
Any unsecured data in any data dictionary table can be easily altered or deleted by any user.

8.8 Storage and File Structures

Goals
Understand the basic concepts underlying different storage media, buffer management, les structures, and organization of records in files.
Contents

- Overview of Physical Storage Media
- Magnetic Disks, Tertiary Storage
- Buffer Management
- Storage Access
- File Organization

Classification of Physical Storage Media

- Main criteria: Speed with which data can be accessed and cost per unit of data to buy medium

- Reliability
  - data loss on power failure or system crash
  - physical failure of the storage device

- Classification:
  1. volatile storage: loses contents when power is turned off
  2. non-volatile storage: contents persist when power is switched off

Physical Storage Media

- Cache: fastest and most costly form of storage; volatile; managed by the hardware and/or operating system
- Main Memory:
  - general purpose instructions operate on data in main memory
  - fast access but in general too small to store the entire database (or even an entire relation)
  - volatile { content of main memory is usually lost if a power failure or system crash occurs
- Magnetic-Disk Storage: primary medium for the long term storage of data; typically stores the entire database (i.e., all relations and associated access structures)
  - data must be moved from disk to main memory for access and written back for storage (insert, update, delete, select)
  - direct-access, i.e., it is possible to read data on disk in any order
  - usually survives power failures and system crashes; disk failure, however, can destroy data, but is much less frequent than system crashes

- Optical Storage: non volatile; CD-ROM/DVD most popular form; Write-Once-Read-Many (WORM) optical disks are typically used for archival storage.
- Tape Storage: non-volatile, used primarily for backup and export (to recover from disk failures and to restore previous data), and for archival data
  - sequential access, much slower than disk
  - very high capacity (8GB tapes are common)
  - tape can be removed from drive ; storage cost much cheaper than disk.

Storage Hierarchy

- Primary Storage: Fastest media but volatile (cache, main memory)
- Secondary Storage: next lower level in hierarchy, non-volatile, moderately fast access time, sometimes also called on-line storage (magnetic disks, flash memory)
- Tertiary Storage: lowest level in hierarchy, non-volatile, slow access time, also called o-line storage (magnetic tape, optical storage)

Magnetic Disk Mechanisms

Important here:

- Access time (time it takes from when a read or write request is issued to when the data transfer begins) is determined by seek time and Rotational Latency.
- Data-Transfer Rate: the rate at which data can be retrieved from or stored to the disk.
- Mean time to failure (MTTF): the average time the disk is expected to run continuously without any failure

Optimization of Disk-Block Access

- Block: A contiguous sequence of sectors from a single track
  - Data is transferred between main memory and disk at the granularity of blocks
  - Block size ranges from 512 bytes to several kilobytes
- File organization-optimize block access time by organizing the blocks to correspond to how data will be accessed (e.g., store related information on the same or nearby cylinder).
- Non-volatile buffers speed up disk writes by immediately writing blocks to a non-volatile RAM buffer; controller then writes to disk whenever the disk has no other requests.

RAID

- Redundant Arrays of Independent Disks: disk organization that takes advantage of utilizing large numbers of inexpensive, mass-market disks
- Main Idea: Improvement of reliability via redundancy, i.e., store extra information that can be used to rebuild information lost in case of a disk failure. Use Mirroring (or shadowing): duplicate every disk (logical disk consists of two physical disks)
- Different RAID levels (0-6) have different cost, performance, and reliability characteristics.

Storage Access

- A data base file is partitioned in to fixed-length storage units called blocks (or pages). Blocks/pages are units of both storage allocation and data transfer.
- Database system seeks to minimize the number of block transfers between disk and main memory. Transfer can be reduced by keeping as many blocks as possible in main memory.
- Buffer Pool: Portion of main memory available to store copies of disk blocks.

- Buffer Manager: System component responsible for allocating and managing buer space in main memory.

Buffer Manager

Program calls on buffer manager when it needs block from disk

- The requesting program is given the address of the block in main memory, if it is already present in the buffer.
- If the block is not in the buffer, the buffer manager allocates space in the buffer for the block, replacing (throwing out) some other blocks, if necessary to make space for new blocks.
- The block that is thrown out is written back to the disk only if it was modified since the most recent time that it was written to/fetched from the disk.
- Once space is allocated in the buffer, the buffer manager reads in the block from the disk to the buffer, and passes the address of the block in the main memory to the requesting program.

Buffer Replacement Policies

- Most operating systems replace the block least recently used (LRU strategy)
- LRU -Use past reference of block as a predictor of future references
- Queries have well-defined access patterns (such as sequential scans), and a database system can use the information in a user's query to predict future references LRU can be a bad strategy for certain access patterns involving repeated sequential scans of data files
- Mixed strategy with hints on replacement strategies provided by the query optimizer is preferable (based on the used query processing algorithm(s)).
- Pinned block: memory block that is not allowed to be written back to disk
- Toss immediate strategy: frees the space occupied by a block as soon as the final record (tuple) of that block has been processed.
- Most recently used strategy (MRU): system must pin the block currently being processed. After the final tuple of that block has been processed, the block is unpinned, and it becomes the most recently used block.
- Buffer manager can use statistical information regarding the probability that a request will reference a particular relation, e.g., the data dictionary is frequently accessed ; keep data dictionary blocks in main memory buffer

8.9 File Organization

- File organization refers to the relationship of the key of the record to the physical location of that record in the computer file.
- File organization may be either physical file or a logical file. A physical file is a physical unit, such as magnetic tape or a disk.
- A logical file on the other hand is a complete set of records for a specific application or purpose.
- A logical file may occupy a part of physical file or may extend over more than one physical file.
- Typical DBMS applications need a small subset of the DB at any given time.
- when a portion of the data is needed it must be located on disk, copied to
- Memory for processing and rewritten to disk if the data was modified.

Advantages of File Organization

1. Fast access to single record or collection of related records.
2. Easy record adding/update/removal, without disrupting.
3. Storage efficiency.

4. Redundancy as a warranty against data corruption.

Types of File Organization

1. Sequential File
2. Indexed Sequential
3. Direct file / Hash file

Sequential file

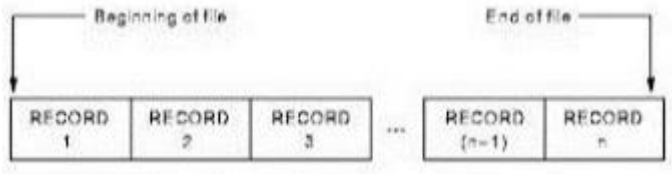
- A sequential file maintains the records in the logical sequence of its primary key values.
- A sequential file can be stored on devices like magnetic tape that allow sequential access.
- In this organization records are written consecutively when the file is created. Records in a sequential file can be stored in two ways.

• **Pile file**: Records are placed one after another as they arrive (no sorting of any kind).
• **Sorted file**: Records are placed in ascending or descending values of the primary key.
File Reorganization: In file reorganization all records, which are marked to be deleted are deleted and all inserted records are moved to their correct place (sorting).

File reorganization steps are:

1. read the entire file (all blocks) in RAM.
2. remove all the deleted records.
3. write all the modified blocks at a different place on the disk.

1. Inserting a record: To insert a record, it is placed at the end of the file. No need to sort (ascending or descending order) the file.
2. Deleting or modifying a record: This will require fetching the block containing the record, finding the record in the block and just marking it deleted, and then write the modified block to the disk. Total time required: TD or TM = TF + 2r.
3. Sorted Sequential File: In a sorted file, first the record is inserted at the end of the file and then moved to its correct location (ascending or descending). Records are stored in order of the values of the key field.



Advantages of Sequential File Organisation

1. Good for report generation, statistical computation and inventory control.
2. It is fast and efficient when dealing with large volumes of data that need to be processed periodically (batch system)
3. Simple file design
4. Very efficient when most of the records must be processed e.g. Payroll
5. Very efficient if the data has a natural order
6. Can be stored on inexpensive devices like magnetic tape

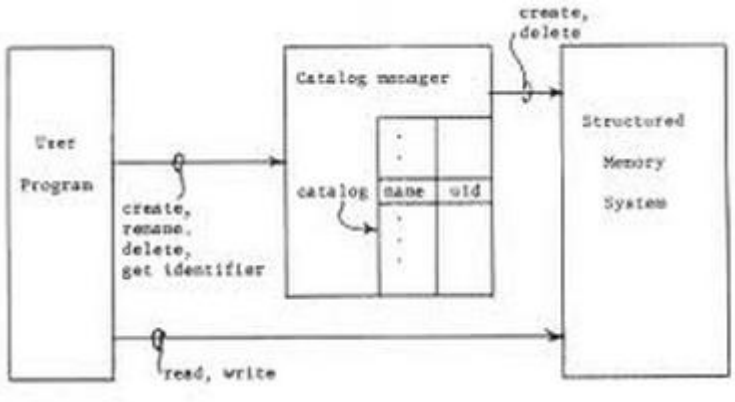
Disadvantages of sequential File Organisation

- Requires that all new transactions be sorted into the proper sequence for sequential access processing.

- Locating, storing, modifying, deleting, or adding records in the file require rearranging the file.
- This method is too slow to handle applications requiring immediate updating or responses.

Direct File Organization

1. Records are read directly from or written on to the file.
2. The records are stored at known address.
3. Address is calculated by applying a mathematical function to the key field.
4. Such file are created using some hashing function so they are called hashing organization or hashed files.



5. Files in his type are stored in direct access storage devices such as magnetic disk, using an identifying key.
6. The identifying key relates to tits actual storage position in the file.
7. The computer can directly locate the key to find the desired record without having to search through any other record first.
8. Here the records are stored randomly, hence the name random file.
9. It uses online system where the response and updation are fast.

Advantages of Direct File Organization

1. Records can be immediately accessed for updation.
2. Several files can be simultaneously updated during transaction processing.
3. Transaction need not be sorted.
4. Existing records can be amended or modified.
5. Most suitable for interactive online applications.—Very easy to handle random enquiries.

Disadvantages of Direct File Organization

1. Data may be accidentally erased or over written unless special precautions are taken.
2. Risk of loss of accuracy and breach of security.
3. Special backup and reconstruction procedures must be established.
4. Less efficient use of storage space.
5. Expensive hardware and software are required.
6. High complexity in programming.
7. File updation is more difficult when compared to that of sequential method.

Indexed File Organization

1. An indexed file contains records ordered by a record key.
2. Each record contains a field that contains the record key.

3. The record key uniquely identifies the record and determines the sequence in which it is accessed with respect to other records.
4. A record key for a record might be, for example, an employee number or an invoice number.
5. An indexed file can also use alternate indexes, that is, record keys that let you access the file using a different logical arrangement of the records.
6. For example, you could access the file through employee department rather than through employee number.
7. The record transmission (access) modes allowed for indexed files are sequential, random, or dynamic. When indexed files are read or written sequentially, the sequence is that of the key values.

Advantages of Indexed File Organization

1. Quite easy to process,
2. With proper selection of a key field, records in a large file can be searched and accessed in very quickly.
3. Any field of the records can be used as the key. The key field can be numerical or alphanumerical.

Disadvantages of Indexed Files

1. Extra data structures have to be maintained .These extra data structures maintained on the disk can use up much disk space, especially for long key values.
2. The indexed files have to be reorganized from time time to get rid of deleted records and improve performance that gets gradually decreased with addition of new records.

Self Assessment Question

1. A record consists of a……

(a). Character
(b). Collection of related records
(c). Set of one or more fields
(d). Group of files

2. Data management is a discipline that focuses on the……..

(a). Management of end users
(b). Generation, storage, and retrieval of data
(c). Help items to be used by end users
(d). Generation, storage, and retrieval of information

3. Which one is the most common classification applied to a DBMS?

(a). Number of users
(b). Database site location(s)
(c). Expected type and extent of use
(d). Software manufacturer

4. All fields for a specific entity can be grouped together as a……..

(a). File
(b). Record
(c). Field
(d). Database

5. Which of the following is considered a data anomaly?

(a). Inconsistency
(b). Timeliness
(c). Modification
(d). Obsoletion

8.10 Summary

- A database administrator (short form DBA) is a person responsible for the installation, configuration, upgrade, administration, monitoring and maintenance of physical databases.

- DBA is usually expected to have experience with one or more of the major database management products, such as Structured Query Language, SAP, and Oracle-based database management software.
- A disk block loses its contents as a result of either a head crash or failure during a data transfer. Copies of data on other disks, or archival backups on tertiary media, such as tapes, are used to recover from the failure.
- There is a hardware malfunction, or a bug in the database software or the operating system, that causes the loss of the content of volatile storage, and brings transaction processing to a halt. The content of the non-volatile storage remains intact, and is not corrupted.
- RAID Levels Schemes to provide redundancy at lower cost by using disk striping combined with parity bits Different RAID organizations, or RAID levels, have differing cost, performance and reliability characteristics RAID Level 1.

8.11 Keywords

***Atomicity*:** It states that database modifications must follow an "all or nothing" rule. Each transaction is said to be "atomic."

***Database Administrator (DBA)*:** It is the person (or group of people) responsible for overall control of the database system.

Logical Error: The transaction cannot continue with its normal execution because of such things as bad input, data not found, or resource limit exceeded.

***RAID*:** It stands for Redundant Arrays of Independent Disks. RAID is the use of multiple disks and data distribution techniques to get better Resilience and/or Performance.

System Crash: There is a hardware malfunction, or a bug in the database software or the operating system, that causes the loss of the content of volatile storage, and brings transaction processing to a halt.

8.12 Review Questions

1. Explain the database administration.
2. What is the role of database administrator? Explain in brief.
3. Explain the functions of a database administrator.
4. Write the short notes importance of DBA.
5. Discuss the failure classification.
6. Differentiate between logical error and system error.
7. Define the storage hierarchy.
8. What is the main purpose redundant array of independent disks?
9. What is the concept of transaction model?
10. Explain the data dictionary storage.

Answers for Self Assessment Questions

1. (c) 2. (b) 3. (c) 4. (b) 5. (c)

8.13 Further Studies

1. The Language of SQL By Larry Rockoff

9

Advanced DBMS

STRUCTURE

9.0 Objectives

9.1 Introduction

9.2 Database System Architectures

9.3 Centralized System

9.4 Client-server System

9.5 Parallel Database System

9.6 Distributed Database System

9.7 Database on Web Services

9.8 Concepts of ODBC and DSN

9.9 Summary

9.10 Keywords

9.11 Review Questions

9.12 Further Studies

9.0 Objectives

After studying this chapter, you will be able to:

- Discuss the database system architectures
- Explain the centralized system
- Discuss the client-server system
- Explain the parallel and distributed database system

9.1 Introduction

In this chapter we are going to discuss the data base system architecture. We also discuss the basic structure of distributed systems. Unlike parallel systems, in which the processors are tightly coupled and constitute a single database system, a distributed database system consists of loosely coupled sites that share no physical components. Furthermore, the database systems that run on each site may have a substantial degree of mutual independence.

Each site may participate in the execution of transactions that access data at one site, or several sites. The main difference between centralized and distributed database systems is that, in the former, the data reside in one single location, whereas in the latter, the data reside in several locations. This distribution of data is the cause

of many difficulties in transaction processing and query processing. In this chapter, we address these difficulties. Environments and in the schemas under which data are stored. A multidatabase system is a software layer that enables such a heterogeneous collection of databases to be treated like a homogeneous distributed database.

9.2 Database System Architectures

Following are the three types of database architectures:

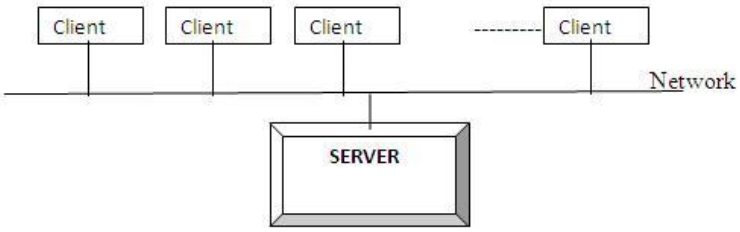
9.2.1 Client-server Database Systems

A client is defined as a requester of services and a server is defined as the provider of services. Today centralized systems act as server systems that satisfy requests generated by client systems.

Functionally database systems can be broadly divided into two parts:

Front End: Front end of a database system consists of tools such as SQL user interface, forms interfaces, reports generation tools, and analysis tools.

Back End: The back end of a database system manages query evaluation and optimization, access structures, concurrency control and recovery management.



Client Server System

Figure 9.1: Client Server System.

9.2.2 Parallel Systems

In parallel processing, many operations are performed simultaneously and the computational steps are performed sequentially. It can process thousands of transactions per second.

There are four types of Architectural Models for Parallel Processing:

Shared Memory (tightly coupled) architecture: Multiple processors share secondary (disk) storage and also share primary memory.

Shared Disk (loosely coupled) architecture: Multiple processors share secondary (disk) storage but each has their own primary memory.

Shared Nothing: The processors share neither a common memory nor common disk.

Hierarchical: This model is the combination of preceding three architectures.

9.2.3 Distributed Systems

Distributed database is a collection of multiple logically interrelated databases distributed over a computer network and a distributed database management system is a software system that manages a distributed database while making the distribution transparent to the user. Transparency means hiding the details of where the data is physically stored within the system.

Reasons for building a distributed database are sharing of data, autonomy, and availability.

Functions of distributed databases:

- Keeping track of data.
- Query Processing by a communication network.

- Security Management.
- Distributed Transaction Management
- Database Recovery.

9.3 Centralized System

A centralized distributed database management system (DDBMS) manages the database as if it were all stored on the same computer. The DDBMS synchronizes all the data periodically and, in cases where multiple users must access the same data, ensures that updates and deletes performed on the data at one location will be automatically reflected in the data stored elsewhere. The users and administrators of a distributed system, should, with proper implementation, interact with the system as if the system was centralized. This transparency allows for the functionality desired in such a structured system without special programming requirements, allowing for any number of local and/or remote tables to be accessed at a given time across the network.

Following are the main advantages of Conventional or Centralized Database management system (DBMS):

- ***Redundancy can be reduced:*** As we know that in non-database systems (file oriented systems) each application has its own private files, because of which there exists considerable redundancy in stored data, resulting in wastage of storage space. Although database systems try to eliminate all possible redundancy, but we do not mean to suggest that all redundancy can be eliminated or necessarily should be eliminated. Sometimes there is sound business or technical reasons for maintaining several distinct copies of the same stored data.
- ***Inconsistency can be avoided to some extent:*** This duplication is not only leads to redundancy but also leads to inconsistency as only one value of "ABC" might change during any modification of the data, hence two values in the database will not agree with each other. Clearly a database that is in an inconsistent state will supply incorrect or contradictory information to its users. It should be clearly noted that if the given fact is represented by a single entry, then such inconsistency can be avoided. Alternatively, if the redundancy is not removed but is controlled, then the DBMS could guarantee that the database is never inconsistent.
- ***The data can be shared:*** With DBMS it is possible to share data in the database among various applications and also it is possible to satisfy the data requirements of new applications without having to create any additional stored data.
- ***Standards can be enforced:*** With central control of database, the DBA (Database Administrator) can ensure that the data represents all applicable standards like corporate, installation, departmental, industry, national and international standards etc.
- ***Security restrictions can be applied:*** A DBA can ensure that the database is accessed only through proper channels and by authorized users. For this purpose DBA defines some security rules, which are checked whenever there is an attempt to access the sensitive data.
- ***Integrity can be maintained:*** The problem of integrity is the problem of ensuring that the data in the database is accurate and correct. Centralized control of DBMS call help in avoiding the problems of inconsistency, by permitting the DBA to define integrity rules to be checked whenever any data update, delete or insert operation is attempted.
- ***Conflicting requirements can be balanced:*** Knowing the overall requirements of the enterprise - as opposed to the requirements of individual users - the DBA can so structure the system as to provide an overall service that is "best for the enterprise".

9.4 Client-server System

The essence of client-server computing is that two kinds of processes (units of program code) that are specialized for different tasks, running on possibly different hardware, applications software and operating systems co-operate to solve a computing problem. One kind (a client) makes a request of another. The other kind (a server) performs the service requested (if the request is a valid one). Many applications have been constructed to operate in client-server mode. For example, most Internet-based services-news readers, E-mail, the World-Wide-Web have been designed this way.

In most cases, the machine on which the server is a powerful machine that resides remotely, while the client machine sits locally on the user's desktop, but this is sometimes reversed. For example, in the X-Windows model of screen display, used widely on UNIX machines, a remote application running on UNIX (e.g., a supercomputer) can perform complex computations that are eventually designed to be the raw material for graphical output. This output is sent to a (local) machine with a request to display the data, using the X protocol. Therefore, the remote machine is the client. The local machine processes this request and converts it into the appropriate display commands for the local hardware.

9.4.1 Client/Server Database Systems

Client/server systems are constructed so that the database can reside on a central computer, known as a server, and be shared among several users. Users access the server through a client or server application:

- In a two-tier client/server system, users run an application on their local computer, known as a client that connects over a network to the server running SQL Server. The client application runs both business logic and the code to display output to the user, and is also known as a thick client.
- In a multitier client/server system, the client application logic is run in two locations:
  - The thin client is run on the user's local computer and is focused on displaying results to the user.
  - The business logic is located in server applications running on a server. Thin clients request functions from the server application, which is itself a multithreaded application capable of working with many concurrent users. The server application is the one that opens connections to the database server and can be running on the same server as the database, or it can connect across the network to a separate server operating as a database server. This is a typical scenario for an Internet application. For example, a server application can run on a Microsoft Internet Information Services (IIS) and service thousands of thin clients running on the Internet or an intranet. The server application uses a pool of connections to communicate with a copy of SQL Server. SQL Server can be installed on the same computer as IIS, or it can be installed on a separate server in the network.

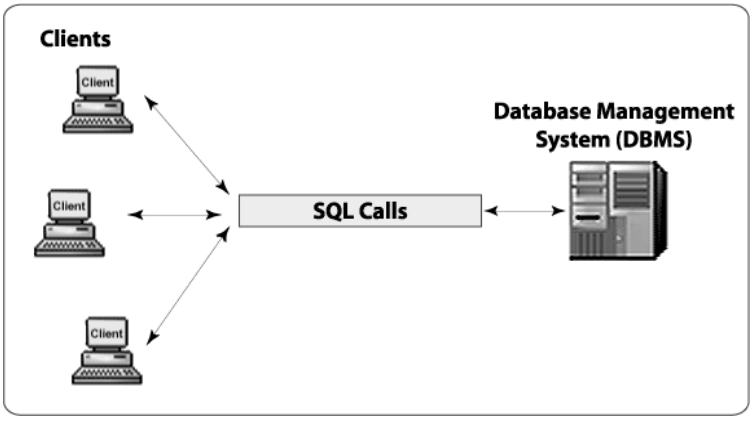


Figure 9.2: Client/Server with Database Servers.

9.4.2 Client/Server Components and Characteristics

Components of client/server computing include:

Client: A client (front-end) is a single PC or workstation associated with software that provides computer and presentation services as an interface to server resources. Presentation is usually provided by visually enhanced processing software known as a Graphical User Interface (GUI).

Server: A server (back-end) is one or more multi-user computer(s), usually a mainframe or a minicomputer, although it could be a PC. Server functions include any centrally supported role, such as file sharing, printer sharing, database access and management, communication services, facsimile services, application development, and others. Multiple functions may be supported by a single server.

Middleware: This is a client/server specific term used to describe a unique class of software employed by client/server applications. This software resides between an application and the network, and manages the interaction between the GUI front-end and data servers in the back- end. It facilitates the client/server connections over the network and also allows client applications to access and update remote databases and mainframe files.

Characteristics of client/server computing include:

Distributed: Most commonly, a server is a distinct computer that serves from a few to any number of client systems. It is feasible to have clients and servers on the same computer. The server may be in the same room as its clients, or it may be across town or around the world.

Decentralized: Client/server systems are typically installed, administered, and operated by a business unit, rather than a centralized computing facility.

Complex: Client/server systems usually involve multiple clusters of computers linked by high-speed communication lines.

Caution

When placing mission critical applications on a client/server system. The end-user computing evolution provided computing power at the workplace, and resulted in end-user demand for access to corporate data with little regard for the security of that data.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Compare and contrast distributed system and parallel system architecture.

……..……

……..……

……...

9.5 Parallel Database System

A parallel database system seeks to improve performance through parallelization of various operations, such as loading data, building indexes and evaluating queries. Although data may be stored in a distributed fashion, the distribution is governed solely by performance considerations. Parallel databases improve processing and input/output speeds by using multiple CPUs and disks in parallel. Centralized and client-server database systems are not powerful enough to handle such applications. In parallel processing, many operations are performed simultaneously, as opposed to serial processing, in which the computational steps are performed sequentially.

Parallel databases can be roughly divided into two groups; the first group of architecture is the multiprocessor architecture, the alternatives of which are the followings.

- Shared memory architecture, where multiple processors share the main memory space, as well as mass storage (e.g. hard disk drives).
- Shared disk architecture, where each node has its own main memory, but all nodes share mass storage, usually a storage area network. In practice, each node usually also has multiple processors.
- Shared nothing architecture, where each node has its own mass storage as well as main memory.
- The other architecture group is called hybrid architecture, which includes:
- Non-Uniform Memory Architecture (NUMA), which involves the Non-Uniform Memory Access.
- Cluster (shared nothing + shared disk. Local network of multiple devices (SAN) a single storage device (NAS) which is formed by a group of connected computers.

Caution

You must always create a logical-log backup after a parallel archive to make sure that you can restore the database in the event of a failure. Unlike a sequential archive, a parallel archive cannot be used for a restore without the accompanying logical-log files, so a logical-log backup is essential.

Did You Know?

Since the mid-1990s, web-based information management has used distributed and/or parallel data management to replace their centralized cousins.

Self Assessment Questions

1. Conceptual design……
(a). Is a documentation technique.
(b). Needs data volume and processing frequencies to determine the size of the database.
(c). Involves modeling independent of the DBMS.
(d). Is designing the relational model

2. Count function in SQL returns the number of…….
(a). Values (b). Distinct values (c). Groups (d). Columns

3. A DBMS query language is designed to……
(a). Support end users who use English-like commands.
(b). Support in the development of complex applications software.
(c). Specify the structure of a database.
(d). All of these.

4. The statement in SQL which allows to change the definition of a table is…….
(a). Alter (b). Update (c). Create (d). Select

9.6 Distributed Database System

All system components i.e. the database and the Database Management System (DBMS) reside at a single computer or site. Users may be able to access the Centralized Database System remotely via terminals connected to the site; however all the data access and processing takes place at the central site. The following Figure 12.3 shows a Centralized DB System.

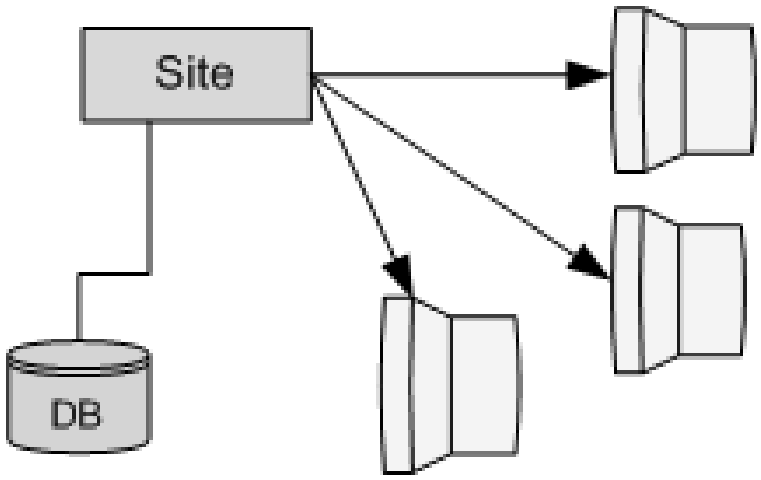


Figure 9.3: Centralized database System.

9.6.1 Distributed Database

In a Distributed Database System the database is stored/spread physically across computers or sites in different locations that are connected together by some form of data communication network. They may be spread over WAN or LAN. The computers may be of different types such as IBM Mainframes, VAXs, SUN work station, PCs etc managed by different operating systems and each fragment of the data base may be managed by a different DBMS such as Oracle, Ingress, and Microsoft SQL server.

Distributed database management system (DDBMS). In a DDS, database applications running at any of the system's sites should be able to operate on any of the database fragments transparently i.e., as if the data come from a single database managed by one DBMS. The software that manages a distributed database in such a way is called DDBMS.

The notion of distributed database is different from that of decentralized database. The latter does not imply sharing of data by a communication network. The former implies a collection of sites connected together with some kind of network and where each site has a database in its own right, but the sites work together as if data was stored at only one site.

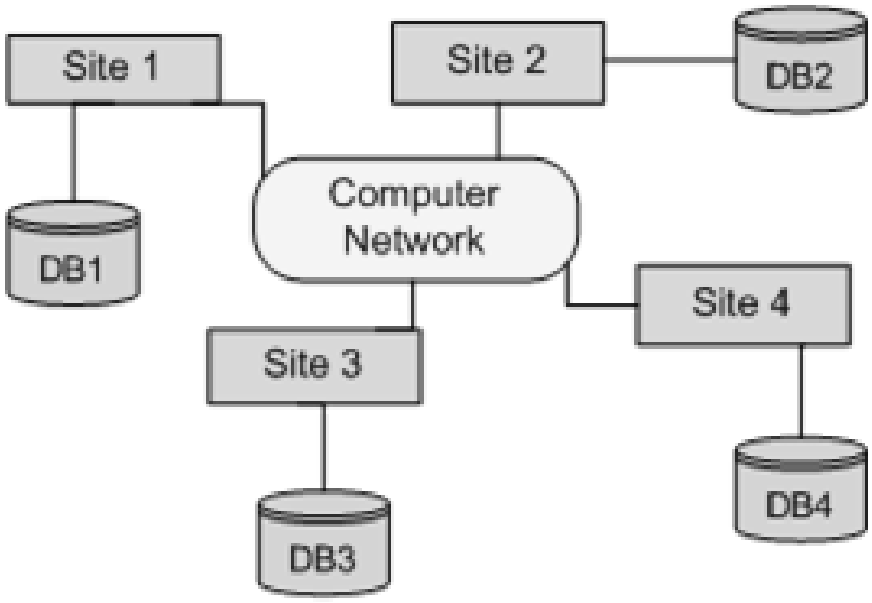


Figure 9.4: Distributed Database System.

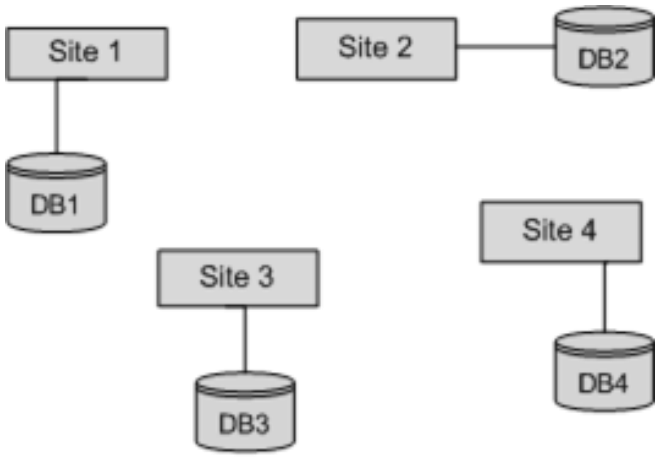


Figure 9.5: Decentralized Database System.

Distributed database design: The methodology used for the logical design of a centralized database applies to the design of the distributed one as well. However, for a distributed database three additional factors have to be considered.

Data Fragmentation: Before we decide how to distribute the data we must determine the logical units of distribution. The database may be broken up into logical units called fragments which will be stored at different sites. The simplest logical units are the tables themselves.

Horizontal fragmentation: A horizontal fragment of a table is a subset of rows in it. So horizontal fragmentation divides a table 'horizontally' by selecting the relevant rows and these fragments can be assigned to different sides in the distributed system.

Vertical fragmentation: a vertical fragment of a table keeps only certain attributes of it. It divides a table vertically by columns. It is necessary to include the primary key of the table in each vertical fragment so that the full table can be reconstructed if needed.

Mixed fragmentation: in a mixed fragmentation each fragment can be specified by a SELECT-PROJECT combination of operations. In this case the original table can be reconstructed be applying union and natural join operations in the appropriate order.

Data Replication: A copy of each fragment can be maintained at several sites. Data replication is the design process of deciding which fragments will be replicated.

Data Allocation: Each fragment has to be allocated to one or more sites, where it will be stored. There are three strategies regarding the allocation of data:

Fragmented (or partitioned): The database is partitioned into disjoint fragments, with each fragment assigned to one site (no replication). This is also called ‘non-redundant allocation’.

Complete replication: A complete copy of the database is maintained at each site (no fragmentation). Here, storage costs and communication costs for updates are most expensive. To overcome some of these problems, snapshots are sometimes used. A snapshot is a copy of the data at a given time. Copies are updated periodically. Selective replication: A combination of fragmentation and replication.

9.6.2 Types of DDBMS

A DDBMS can be classified as.

- ***Homogeneous*:** if all sites use the same DBMS product.
- ***Heterogeneous:*** if sites may run different DBMS products, which need not be based on the same underlying data model and so may be composed of Relational, Network, Hierarchical and Object-oriented DBMSs.

DBMS Transparency

The term DBMS transparency refers to the ability of DDBMS to hide the implementation details from the user. Thus the fact that a distributed database is split into several fragments that can be stored on different computers and perhaps replicated is hidden from the user. The objective of the transparency is to make the distributed system appear like a centralized system. We can identify four main types of transparency in a DDBMS:

- Distribution transparency
- Fragmentation transparency
- Location transparency
- Transaction transparency
- Performance transparency
- DBMS transparency

Advantages

- Distributed systems mirror the structure of an enterprise.
- Local autonomy (control). Security, integrity, storage representation and hardware are controlled locally. At the same the same time user can access remote data when necessary.
- No reliance on a central site. Avoid bottlenecks and system vulnerability.
- Reliability and availability. Continue to operate if one or more sites go down or communication links fail.
- Speed up of query processing. Queries about data stored locally are answered faster. Moreover, queries can be split to execute in parallel at different sites or they can be redirected to less busy sites.
- Modular growth. It is much easier to add another site than to expand a centralized system.

Disadvantages

- Software complexity and high costs. A DDBMS that hides the distributed nature from the user and provides an acceptable level of performance, reliability and availability is inherently more complex than a centralized DBMS.
- Processing overheads. Increased query processing costs, catalogue management and consistency maintenance.
- Database design more complex.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Prepare the diagram of centralized and decentralized database system.

……..……

……..……

……...

9.7 Database on Web Services

Web services enable application-to-application interaction over the Web, regardless of platform, language, or data formats. The key ingredients, including Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Universal Description, Discovery, and Integration (UDDI), have been adopted across the entire software industry. Web services usually refer to services implemented and deployed in middle-tier application servers. However, in heterogeneous and

disconnected environments, there is an increasing need to access stored procedures, as well as data and metadata, through Web services interfaces.

The Database Web services technology is a database approach to Web services. It works in the following two directions.

- Accessing database resources as a Web service.
- Consuming external Web services from the database.

Oracle Database can access Web services through PL/SQL packages and Java classes deployed within the database. Turning Oracle Database into a Web service provider leverages investment in Java stored procedures, PL/SQL packages, predefined SQL queries, and data manipulation language (DML). Conversely, consuming external Web services from the database, together with integration with the SQL engine, enables Enterprise Information Integration.

Using Oracle Database as Web Services Provider

Web Services use industry-standard mechanisms to provide easy access to remote content and applications, regardless of the platform and location of the provider and implementation and data format. Client applications can query and retrieve data from Oracle Database and call stored procedures using standard Web service protocols. There is no dependency on Oracle-specific database connectivity protocols. This approach is highly beneficial in heterogeneous, distributed, and disconnected environments.

You can call into the database from a Web service, using the database as a service provider. This enables you to leverage existing or new SQL, PL/SQL; Java stored procedures, or Java classes within Oracle Database. You can access and manipulate database tables from a Web service client.

Features of Oracle Database as a Web Service Provider

Using Oracle Database as a Web service provider offers the following features:

1. Exposes Java in the database as Web services. Exposes existing Java classes deployed in Oracle Database as Web services. Java classes implementing data-related services can be migrated between the middle tier and the database. Java portability results in database independence.
2. Provides SQL query Web services. Leverages warehousing or business intelligence queries, data monitoring queries, and any predefined SQL statements as Web services.
3. Enables DML Web services. Offers secure, persistent, transactional, and scalable logging, auditing, and tracking operations implemented through SQL DML, as Web services. DML Web services are implemented as atomic or group, or batch, INSERT, UPDATE, and DELETE operations.

Self Assessment Question

5. The RDBMS terminology for a row is……

(a). Tuple (b). Relation (c). Attribute (d). Degree

6. Which two files are used during operation of the DBMS

(a). Query languages and utilities

(b). DML and query language

(c). Data dictionary and transaction log

(d). Data dictionary and query language

9.8 Concepts of ODBC and DSN

Open Data Base Connectivity (ODBC) is a standard database access method that allows access and modification of data from any application, regardless of which database management system is handling the data. An ODBC connection establishes a link between your database, the ODBC drivers, and your application, and works as follows:

You create a database using the database program of your choice. You then use the driver (corresponding to the application) to create a data source name (DSN). The DSN links a database file to a driver and provides a central location for accessing databases on a given computer. Once this is done, you can access the database with your application.

Note

You can use a (Data Source Name) DSN connection (in which case you must ensure that your data source name is unique), or you can use a DSN-less connection. You use a DSN-less connection by coding your ASP scripts to issue commands to your Access database directly. This bypasses the ODBC software on the server, and allows for faster connection and execution time on the database interaction.

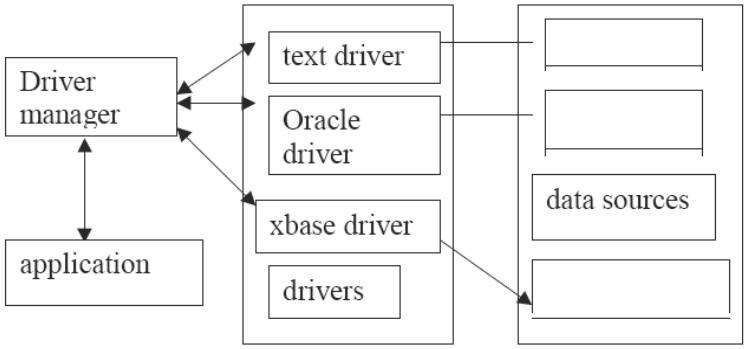


Figure 9.6: Open Database Connectivity.

There are three types of DSNs.

System DSN: System DSNs allow every application on that computer to access the data source, regardless of which user created the data source. All users on the system can see this DSN and use it (as long as they know the user name and password). The configuration for this DSN is stored in the registry. An ODBC System DSN allows you to store information about how to connect to an associated database.

User DSN*:* User DSNs are created for a specific user. Only the user who created the DSN can see this and use it. The configuration for this DSN is stored in the registry.

File DSN*:* File DSNs are similar to user DSNs, except for the fact that the configuration for this DSN is stored in a simple text file with a DSN extension.

Web appliance supports only System DSNs because these are the only ones accessible to Microsoft Windows 2000 applications (such as IIS), and hence to remote servers.

While all drivers require that you provide the database location, some drivers require additional parameters. These are often specified by the database. For example, Microsoft Access DSNs must refer to database files on the local file system.

The ODBC Manager allows you to view, add, modify, and delete data sources. It offers three options as follows:

- ***View Data Source List:*** The View Data Source List option allows you to view, modify, and delete data sources.
- ***Add SQL Server Data Source:*** The Add SQL Server Data Source option allows you to add SQL Server data sources.
- ***Add Access Data Source:*** The Add Access Data Source option allows you to add Access data sources.

Did You Know?

ODBC defines a standard C API for accessing a relational DBMS. It was developed by the SQL Access Group in 1992 to standardize the use of a DBMS by an application.

9.9 Summary

- A single database is a collection of tables, which are related to each other with the help of common fields.
- RDBMS includes features such as data independence and data abstraction that help in efficiently organizing the data.
- Prevention of data redundancy and persistent storage makes RDBMS useful for storing data.
- Data Source Name provides connectivity to a database through an ODBC driver. The DSN contains database name, directory, database driver, User ID, password, and other information.
- A client is defined as a requester of services and a server is defined as the provider of services.

9.10 Keywords

Centralization: Is the process by which the activities of an organization, particularly those regarding planning and decision-making become concentrated within a particular location and/or group.

Collusion: This group may decide to collude in order to inflate their own trust values and deflate trust values for peers that are not in the collective. Therefore, a certain level of resistance needs to be in place to limit the effect of malicious collectives.

Distributed Database: Is a collection of multiple logically interrelated databases distributed over a computer network and a distributed database management system is a software system that manages a distributed database.

Fragmentation: The relation is partitioned into several fragments. Each fragment is

Heterogeneous: if sites may run different DBMS products, which need not be based on the same underlying data model and so may be composed of Relational, Network, Hierarchical and Object-oriented DBMSs.

9.11 Review Questions

1. What are the database system architectures?
2. What is the parallel database system?
3. Explain the client-server database
4. What are the kinds distributed data base system?
5. Discuss the centralized system.
6. Explain the characteristics of client/server and components
7. What are the advantages and disadvantages of ADBMS?
8. What are the web services of database?
9. Discuss the ODBC and DSN.
10. How many types of Data Source Name (DSN)?

Answers for Self Assessment Questions

1. (c) 2. (a) 3. (d) 4. (a) 5. (d) 7. (c)

9.12 Further Studies

1. Advanced Database Management System by V.K. Jain

Data Structure

(MCA/PGDCA - 104)



Jaipur National University
Directorate of Distance Education

Established by Government of Rajasthan
Approved by UGC under Sec 2(f) of UGC ACT 1956
(**Recognised by Joint Committee of UGC-AICTE-DEC, Govt. of India**)

1

Introduction to C Language

STRUCTURE

1.0 Objectives

1.1 Introduction

1.2 Basic Concept of C Programming

1.3 The C Character Set

1.4 Constants

1.5 Data Types

1.6 Variables

1.7 Declaration of Variables

1.8 Operators

1.9 Summary

1.10 Keywords

1.11 Review Questions

1.12 Further Studies

1.0 Objectives

After studying this chapter, you will be able to:

- Discuss the basic concept of C programming
- Explain the C character set
- Understand about the data types used in C
- Define and declare the various variables in C
- Discuss the different operators used in C
- Explain the various arithmetic expressions in C
- Understand the operator precedence and their associativity

1.1 Introduction

The C is a programming language developed at AT & T's Bell Laboratories of USA in 1972. It was designed and written by a man named Dennis Ritchie. In the late seventies C began to replace the more familiar languages of that time like PL/I, ALGOL, etc. No one pushed C. It was not made the 'official' Bell Labs language. Thus, without any advertisement C's reputation spread and its pool of users grew. Ritchie seems to have been rather surprised that so many programmers preferred C to older languages like FORTRAN or PL/I,

or the newer ones like Pascal and APL. But, that is what happened.

1.2 Basic Concept of C Programming

1.2.1 Structure of C program

Every C program consists of one or more functions. A function is nothing but a group or sequence of C statements that are executed together. Each C program function performs a specific task. The 'main ()' function is the most important function and must be present in every C program. The execution of a C program begins in the main() function. The given format shows the structure of a C program.

```
main()
        function1()          function2()
{               {                    {
    statement1;      statement1;          statement1;
    statement2;      statement2;          statement2;
    ……;               ……..;                ……;
    ……;               ……..;                ……;
}               }                    }
```

Programmers are free to name C program functions (except the main () function).

Learning any programming language becomes easy with a hands-on approach, so let us get right to it. The following is a simple C program that prints a message 'Hello, world' on the screen.

```
/* First program of C language */
1.#include<stdio.h>
2.main()
3.{
4. printf("Hello, world");
5.}
```

Type this program in any text editor and then compile and run it using a C-compiler. However, your task will become much easier if you are using an IDE such as Turbo C

Steps for Starting C Program

Install Turbo C on your computer and follow the steps given below in order to type and run the program given above:

1. Go to the directory where you have installed Turbo C.
2. Type TC at the DOS command prompt.
3. In the edit window that opens, type the mentioned program above.
4. Save the program as hello.c by pressing F2 or Alt + 'S'.
5. Press Alt + 'C' or Alt + F9 to compile the program.
6. Press Alt + 'R' or Ctrl + F9 to execute the program.
7. Press Alt + F5 to see the output.

Understanding the program

In the program discussed above, the information enclosed between '/* */' is called a 'comment' and may appear anywhere in a C program. Comments are optional and are used to increase the readability of the program.

The '#include' in the first line of the program is called a preprocessor directive. A preprocessor is a program that processes the C program before the compiler. All the lines in the C program beginning with a hash (#) sign are processed by the preprocessor.

'stdio.h' refers to a file supplied along with the C compiler. It contains ordinary C statements. These statements give information about many other functions that perform input-output roles.

Thus, the statement '#include<stdio.h>' effectively inserts the file 'stdio.h' into the file hello.c making functions contained in the 'stdio.h' file available to the programmer. For example, one of the statements in the file 'stdio.h' provides the information that a function printf() exists, and can accept a string (a set of characters enclosed within the double quotes).

The next statement is the main() function. As you already know, this is the place where the execution of the C program begins. Without this function, your C program cannot execute.

Next comes the opening brace '{', which indicates the beginning of the function. The closing brace '}' indicates the end of the function.

The statement printf() enclosed within the braces'{}' informs the compiler to print (on the screen) the message enclosed between the pair of double quotes. In this case, 'Hello, world' is printed. As mentioned earlier, the statement printf() is a built-in function shipped along with the C compiler. Many other built-in functions are available that perform specific tasks. The power of C lies in these functions.

Cautious about errors!

Errors/bugs are very common while developing a program. If you do not detect them and correct them, they cause a program to produce wrong results. There are three types of errors — syntax, logical, and run-time errors.

1. Syntax errors: These errors occur because of wrongly typed statements, which are not according to the syntax or grammatical rules of the language. For example, in C, if you do not place a semi-colon after the statement, it results in a syntax error.

printf("Hello,world")–error in syntax (semicolon missing)

printf("Hello,world");-This statement is syntactically correct.

2. Logical errors: These errors occur because of logically incorrect instructions in the program. Let us assume that in a 1000 line program, if there should be an instruction, which multiplies two numbers and is wrongly written to perform addition. This logically incorrect instruction may produce wrong results. Detecting such errors is difficult.
3. Runtime errors: These errors occur during the execution of the programs though the program is free from syntax and logical errors. Some of the most common reasons for these errors are
    a. when you instruct your computer to divide a number by zero.
    b. when you instruct your computer to find logarithm of a negative number.
    c. when you instruct your computer to find the square root of a negative integer.

Unless you run the program, there is no chance of detecting such errors.

Did You Know?

The C is a case-sensitive language. It makes a distinction between letters written in lower and upper case. For example, 'A' and 'a' would be taken to mean different things. Also, remember that all C language keywords should be typed in lower case.

1.2.2 Programming Rules

The following rules that is applicable to all C programs:

1. Each instruction in a C program is written as a separate statement. Therefore a complete C program would comprise of a series of statements.

2. The statements in a program must appear in the same order in which we wish them to be executed; unless of course the logic of the problem demands a deliberate 'jump' or transfer of control to a statement, which is out of sequence.
3. Blank spaces may be inserted between two words to improve the readability of the statement. However, no blank spaces are allowed within keyword.
4. All statements are entered in small case letters.
5. The C has no specific rules for the position at which a statement is to be written.
6. Every C statement must end with a; Thus; acts as a statement terminator.
7. Comment about the program should be enclosed within /* */. For example, the first two statements in our program are comments.
8. Though comments are not necessary, it is a good practice to begin a program with a comment indicating the purpose of the program.
9. Any number of comments can be written at any place in the program. For example, a comment can be written before the statement, after the statement or within the statement as shown below:

```
/* formula */ si = p * n * r / 100 ; si = p * n * r / 100 ;
/* formula */ si = p * n * r / /* formula */ 100 ;
```

10. Sometimes it is not as obvious as to what a particular statement in a program accomplishes. At such times it is worthwhile mentioning the purpose of the statement (or a set of statements) using a comment.
For example:

```
/* formula for simple interest */
si = p * n * r / 100 ;
```

11. Often programmers seem to ignore writing of comments. But when a team is building large software well commented code is almost essential for other team members to understand it.
12. Although a lot of comments are probably not necessary in this program, it is usually the case that programmers tend to use too few comments rather than too many. An adequate number of comments can save hours of misery and suffering when you later try to figure out what the program does.
13. The normal language rules do not apply to text written within /* .. */. Thus we can type this text in small case, capital or a combination. This is because the comments are solely given for the understanding of the programmer or the fellow programmers and are completely ignored by the compiler.
14. Comments cannot be nested.

```
/* Cal of SI /* Author sam date 01/01/2010 */ */ is invalid.
```

15. A comment can be split over more than one line, as in,

```
/* This is a
Multi comment */
```

Such a comment is often called a multi-line comment.

Did You Know?

During the late 1970s and 1980s, versions of C were implemented for a wide variety of mainframe computers, minicomputers, and microcomputers, including the IBM PC, as its popularity began to increase significantly.

Caution

Using more than one main ( ) in a C program, is illegal and it creates an error during program execution.

1.3 The C Character Set

The characters that can be used to form words, numbers and expressions depend upon the computer on which the program is run. However, a subset of characters is available that can be used on most personal, micro, mini and mainframe computers. The characters in C are grouped into the following categories:

l. Letters
2. Digits
3. Special characters
4. White spaces

The complete character set is given in Table 2.1.

The compiler ignores white spaces unless they are a part of a string constant. White spaces may be used to separate words, but are prohibited between the characters of keywords and identifiers.

1.3.1 Trigraph Characters

Many non-English keyboards do not support all the characters mentioned in the Table 2.1. AMSI C introduces the concept of "trigraph" sequences to provide a way to enter certain characters that are not available on some keyboards. Each trigraph sequence consists of three characters as shown in Table 1.2. For example, if a keyboard does not support square brackets, we can still use them in a program using the trigraphs??(and ??).

Table 1.1: Character Set

Letters	**Digitals**
Uppercase A to Z	All decimal digits 0 to 9
Special Character	
, comma	& ampersand
. period	^ caret
; semicolon	* astreek
: colon	–minus
? question	+ plus
" quotation	< opening bracket
! exclamation	> closing bracket
/ slash	(left parenthesis
_ underscore	) right parenthesis
$ dollar sign	{ opening bracket
% percent sign	} closing bracket
# number sign	
White spaces	
Blank	
Horizontal	
New line	

Table 1.2: ANSI C trigraph sequences

Trgraph Sequence	Translation
??&	& ampersand
??^	^ caret
?? *	* astreek
??–	–minus
??+	+ plus
??<	< opening bracket
??>	> closing bracket
??(	(left parenthesis
??)	) right parenthesis
??{	{ opening bracket
??}	} closing bracket

1.3.2 The C Tokens

In a passage of text, individual words and punctuation marks are called tokens. Similarly, in a C program the smallest individual units are known as C tokens. The C programs are written using these tokens and the syntax of the language.

1.3.3 Keywords and Identifiers

Every C word is classified as either a keywords or an identifier. All keywords have fixed meanings and these meanings cannot be changed. Keywords serve as basic building blocks for program statements. The list of all keywords in ANSI C is listed in Table 2.3. All keywords must be written in lowercase. Some compilers may use additional keywords that must be identified from the C manual.

Table 1.3: ANSI C Keywords

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

Identifiers refer to the names of variables, functions and arrays. These are user defined names consist of sequence of letters and digits, with a letter as first character. Both upper and lower case letter are permitted. Lower case letter are commonly used. The underscore character is also permitted in identifiers.

Self Assessment Questions

1. The C is a programming language developed at AT & T's Bell Laboratories of USA in……

(a) 1965 (b) 1972.

(c) 1980 (d) 1975

2. ANSI Stands for:
(a) American National Standards Institute
(b) American National Style Institute
(c) Areas of Natural and Scientific Interest
(d) American National Standard Identification

3. A program that translates from a low level language to a higher level one is
(a) Decompiler (b) Compiler
(c) Interpreter (d) None of these.

4. Every c program requires a ……..
(a) Two main Function. (b) One Main Function.
(c) More than one (d) None of these.

1.4 Constants

Constants in C refer to fixed values that do not change during the execution of a program

Integer Constants

An integer constant refers to a sequence of digits. There are three types of integers namely decimal, octal and hexadecimal. Decimal integer consists of set of digits, 0 through 9. Valid example of decimal integer constant is:

123
_321
0
+78

Embedded spaces, commas, and non-digit characters are not permitted between digits for example

15 750
20,000
1000

An octal integer constant consists of any combination of digits from the set 0 through 7 with a leading 0. Some examples of octal integers are:

037
0
0435
0551

A sequence of digits preceded by 0x or 0X is considered as hexadecimal integer They may also include alphabets A through F or a through f. The letters A through F represent the numbers 10 through the following are the examples of valid hex integers

0x2
0x9F
0Xbcd
0x

We rarely use octal and hexadecimal numbers in programming. The largest integer value that can be stored is machine-dependent. It is 32167 on l6-bit machines and 2l47, 483, 647 on 32-bit machines. It is also possible to store larger integer constants on these machines by appending qualifiers such as U, L and UL to the constants. For example:

56789U or 56789u (unsigned integer)

987612347UL	or	987612314ul	(unsigned long integer)
9876543L	or	98765431	(long integer)

The output is shown that the integer values larger than 3276 are not properly stored on a 16-bit machine. However, when they are qualified as long integer (by appending L) the values are correctly stored.

```
Program
main( )
{
printf (“lnteger values\n\n”) ;
printf (“%d %d %d\n”, 327 67,327 67 + 1,327 67 +10);
print(“/n”);
printf (“Long integer values\n\n”);
printf (“%ld %ld %ld\n “, 32767L + 1L32767L+1L,32767L+ 10L);
}
Output
Integer values
32767 –32768 –32759
Long integer values
32767 32768 32777
```

Real Constants

Integer numbers are inadequate to represent quantities that vary continuously, such as distances, heights, temperatures, prices, and so on. These quantities are represented by number containing fractional parts like 17.548. Such numbers are called real constants. Further examples of real constants are:

0.0083
-0.75
435.36
+247.0

These numbers are shown in decimal notation, having a whole number followed by a decimal point and the fractional part. It is possible to omit digits before the decimal point or digits after the decimal point. That is,

215.
.95
.71
+.5

are all valid real numbers. A real number may also be expressed in exponential (or scientific) notation. For example, the value 215.65 may be written as 2.l565e2 in exponential notation. The general form is:

```
mantissa e exponent
```

The mantissa is either a real number expressed in decimal notation or an integer .The exponent is an integer number with an optional plus or minus sign. The letter e separating the mantissa and the exponent can be written in either lowercase or uppercase. Since the exponent causes the decimal point to “float”, this notation is said to represent a real number in floating point form. Examples of legal floating point constants are:

0.65e4
12e-2
1.5e+5
3.18E3
-1.2E-1

Embedded white space is not allowed.
Exponential notation is useful for representing numbers that small in magnitude. For example, 7500000000 may be written -0.000000368 is equivalent to -3.688-7. Floating point constants are normally represented as double-precision quantities. However, the suffixes f or F may be used to force single-precision and I or L to extend double-precision further. Some examples of valid and invalid numeric constants are given in 1.4

Table 1.4: Examples of numeric constants

Constant	Valid?	Remarks
6983541	Yes	Represents long integer
25,000	No	Comma is not allowed
+5.0E3	Yes	(ANSI C supports unary plus)
3.5+ -5	Yes	
7.1e 4	No	No white space is permitted
4.5e-2	Yes	
1.5E+2.5	No	Exponent must be an integer
$2s5	No	$ symbol is not permitted
OXTB	Yes	Hexadecimal integer

Single Character Constants

A single character constant (or simply character constant) contains a single character enclosed within a pair of single quote marks. Examples of character constants are:
'5' 'X'
Character constants have integer values known as ASCII values. For example, the statement
Printf("%d" , 'a');
would print the number 97, the ASCII value of the letter. Similarly, the statement
printf("%c" '97');
would output the letter 'a'. Since each character constant represent an integer value it is also possible to perform arithmetic operations on character constants.

String Constants

A string constant is a sequence of characters enclosed in double quotes. The characters may be letters, numbers, special characters and blank space examples are:
"Hello!"
"1 987"
"WELL DONE"
"?...t'
"5+3"
"X"
Remember that a character constant (e.g., 'X') is not equivalent to the single character string constant (e.g., "X"). Further, a single character string constant does not have an equivalent integer value while a character constant has an integer value. Character strings are often used in programs to build meaningful programs.

Backslash Character Constants

The C supports some special backslash character constants that are used in output functions. For example, the symbol "\n' stands for new-line character. A list of such backslash character constants is given in Table 1.5.

Note that each one of them represents one character, although they consist of two characters. These character combinations are known as escape sequences.

Table 1.5: Backslash character constants

Constant	Meaning
'\a'	audible alert (bell)
'\b'	back space
'\f'	form feed
'\n'	new line
'\'	carriage return
'\t'	horizontal tab
'\v'	vertical tab
'	single quote
'\'	double quote
'\" '	question mark
'\?'	backslash
'\\'	null
'\0'	

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: What is Basic Concept of C Programming?

……..……

……..……...…

……..

Ex2: What is Keywords and Identifiers?

……..……

……..……...…

……..

1.5 Data Types

The C language is rich in its data types. Storage representations and machine instructions to handle constants differ from machine to machine. The variety of data types available allows the programmer to select the type appropriate to the needs of the application as well as the machine.
The ANSI C supports four classes of data types:
1. Primary (or fundamental) data types
2. User-defined data types
3. Derived data types
4. Empty data set
The primary data types and their extensions are discussed in this section. The derived data types such as arrays, functions, structures and pointers are discussed as and when they are encountered.

All C compilers support four fundamental data types, namely ***integer (int), character (char), floating point (float), and double-precision floating point (double).*** Many of them also offer extended data types such as long int and long double. The range of the basic four types are given in Table 1.6

Table 1.6: Size and range of basic data types

Data type	Range of values
char	-128 to 127
int	-32,768 to 92,767
float	3.4e-38 to 3.4e+38
double	1.7e-308 to 1.7e+308

1.5.1 Integer Types

Integers are whole numbers with a range of values supported by a particular machine. Generally, Integers occupy one word of storage, and since the word sizes of machine vary (typically, 16 or 32 bits) the size of an integer that can be stored depends on the computer. If we use a 16 bit word length, the size of the integer value is limited to the range -32768 to + 32767. A signed integer uses one bit for sign and 15 bit for the magnitude of the number. Similarly, a 32 bit word length can store, namely word can store an integer ranging from -2, 147, 483, 648 to 2,147,483,647. In order to provide some control over the range of numbers and storage space, C has three classes of integer storage, namely short_int, int, and long_int, in both signed and unsigned forms. For example, short_in represent fairly small integer values and requires half the amount of storage as a regular int number uses. We declare long and unsigned integer to increase the range of values. The use of qualifier signed on integers is optional because the default declaration assumes a signed number. Table 1.7 shows all the allowed combinations of basic types and qualifiers and their size and range on a 16 bit machine.

Table 1.7: Size and range of data types on a 16-bit machine

Type	Size	Range
char or signed char	8	-128 to 127
unsigned char	8	0 to 255
int or signed int	16	-32,768 to 32,767
unsigned int	‘t6	0 to 65535
short int or	8	-128 to 127
signed short int		0 to 255
unsigned short int		-2,147,4A9,648 to 2,147,485.647
long int or:	8	0 to 4,294,967,295
signed long int	32	3.4E-39 to 3.4E+38
unsigned long int	32	1.7E-3OB to 1.7E+308
float	32	3.4E-4952 to .t. 1 E+49S2

1.5.2 Floating Point Types

Floating point (or real) numbers are stored in 32 bits (on all l6 bit and 32 bit machines), with 6 digits of precision. Floating point numbers are defined in C by the keyword float. When the accuracy provided by a float number is not sufficient, the type double can be used to define the number. A double data type number uses 64 bits giving a precision of 14 digits. These are known as double precision numbers. Remember that

double type represents the same data type that float represents, but with a greater precision. To extend the precision further, we may use long double which uses 80 bits.

1.5.3 Character Types

A single character can be defined as a character (char) type data. Characters are usually stored in 8 bits (one byte) of internal storage. The qualifier signed or unsigned may be explicitly applied to char. While unsigned chars have values between 0 and 255, signed chars have values from -128 to 127.

1.6 Variables

A variable is a data name that may be used to store a data value. Unlike constants that remain unchanged during the execution of a program, a variable may take different values at different times during execution.

A variable name can be chosen by the programmer in a meaningful way so as to reflect its function or nature in the program i.e.

Average
height
Total
Counter_1
class_strength

As mentioned earlier, variable names may consist of letter, digits and underscores (_) character, subject to the following conditions:

l. They must begin with a letter. Some system permits underscores as the first character.
2. ANSI standard recognizes a length of 31 characters. Since only the first eight characters are treated as significant by many compilers.
3. White space is not allowed.

Some examples of valid variable names are:

Akash	Value	T_raise
Delhi	x1	ph_value
Mark	sum1	distance

Invalid examples include:

t23	(area)
%	25^{th}

Further examples of variable names and their correctness are given in Table 1.8.

Table 1.8: Examples of variable names

Variable name	Valid?	Remark
First_tag	Valid	char is a keyword
char	Not valid	Dollar sign is illegal
Priceg	Not valid	Blank space is not permitted
group one	Not valid	First eight characters are significant
average_number	Valid	
int_type	Valid	Keyword may be part of name

If only the first eight character are recognized by a compiler, then the two names

average_height
average_weight
mean the same thing to the computer. Such names can be rewritten as
avg_heigtrt and avg_weight
ht_average and wt_average
without changing their meanings.

1.7 Declaration of Variables

After designing suitable variable names, we must declare them to the compiler. Declaration does two things:
l. It tells the compiler what the variable name is.
2. It specifies what type of data the variable will hold.
The declaration of variables must be done before they are used in the program.

Primary Type Declaration

A variable can be used to store a value of any data type. That is, the name has nothing to do with its type. The syntax for declaring a variable is as follows:
data-type vl,v2,....vn ;
vl, v2,....vn are the names of variables. Variables are separated by commas. A declaration statement must end with a semicolon. For example, valid declarations are:
int count;
int number, total;
double ratio;
int and double are the keywords to represent integer type and real type data values respectively. Table 1.9 shows various data types and their keyword equivalents.

Table 1.9: Data types and their keywords

Data type	**Keyword equivalent**
Character	char
Unsigned character	unsigned char
Signed character	signed char
Signed integer	signed int (or int)
Signed short integer	signed short int (or short int or short)
Signed long integer	signed long int (or long int or long)
Unsigned integer	unsigned int (or unsigned)
Unsigned short integer	unsigned short int (or unsigned short)
Unsigned long integer	unsigned long int (or unsigned long)
Floating point	float
Double-precision floating point	double
Extended double-precision	long double

The program segment given in Figure 1.10 illustrates declaration of variables. Main () is the beginning of the program. The opening brace {signals the execution of the program. Declaration of variables is usually done immediately after the opening brace of the program. The variable can also be declared outside (either before or after) the main function. The importance of place of declaration will be dealt in detail later while discussing functions.

```
Main( ) /*.....Program Name..... .....*/
{
/*..... Declaration..... */
float x, y;
int code;
short int count;
long int amount;
double deviation;
unsigned n;
char c;
/*.....Computation.....*/
…..
……..
……
} /*.....Program ends.....*/
```

Figure 1.10: Declaration of variables.

When an adjective (qualifier) short, long, or unsigned is used without a basic data type specifier, the C compilers treat the data type as an int. If we want to declare a character then we must do so using both the terms like unsigned char.

User-Defined Type Declaration

The C supports a feature known as "type definition" that allows users to define an identifier that would represent an existing data type. The user-defined data type identifier can later be used to declare variables. It takes the general form:

typedef *type* identifier;

Where *type* refers to an existing data type and "identifier" refers to the, new name given to the data type. The existing data type may belong to any class of type, including the user-defined ones. Remember that the new type is 'new' only in name, but not the data type. typedef cannot create a new type. Some examples of type definition are:

typedef int units;

typedef float marks;

Here, units symbolize int and marks symbolizes float. They can be later used to declare variables as follows:

units batchl, batch2;

marks namel[50], name2[50];

batchl and batch2 are declared as int variable and namel[50] and name2[50] are declared as 50 element floating point array variables. The main advantage of typedef is that we can create meaningful data type names for increasing the readability of the program.

Another user-defined data type is enumerated data type provided by ANSI standard. It is defined as follows:

enum identifier {*valuel , value2, ... valuen*};

The "identifier" is a user-defined enumerated data type which can be used to declare variables that can have one of the values enclosed within the braces (known as *enumeration constant*s). After this definition, we can declare variables to be of this, 'new' type as below:

enum identifier vl, v2, ... vn;

The enumerated variables vl, v2,...vn can only have one of the values *valuel, value2,...valuen*. The assignments of the following types are valid:

```
v1 = value3;
v5 = valuel;
```

An example:

```
enum day {Monday, Tuesday, ... Sunday};
enum day week_st, week_end;
week_st = Monday;
week_end = Friday;
if(week_st == Tuesday)
week_end = Saturday;
```

The compiler automatically assigns integer digits beginning with 0 to all the enumeration constants. That is, the enumeration constant *value1* is assigned 0, *value 2* is assigned 1, and so on. However, the automatic assignments can be overridden by assigning values explicitly to the enumeration constants. For example:

```
enum day {Monday = 1, Tuesday, …….. Sunday};
```

Here, the constant Monday is assigned the value of that increase successively by l. The remaining constants are assigned values that increase successively by 1.

The definition and declaration of enumerated variables can be combined in one statement.

Example:

```
enum day {Monday, ... Sunday} week_st, week_end;
```

Declaration of Storage Class

Variables in C can have not only *data type* but also *storage class* that provides information about their location and visibility. The storage class decides the portion of the program within which the variables are recognized. Consider the following example:

```
/* Example of storage classes */
int m;
main( )
{
        int i;
        float balance;
        function 1 0;
 }
        functionl0
        {
                int i;
                float sum;
         }
```

The variables m which has been declared before the main is called global variable. It can be used in all the functions of the program. It need not be declared in other functions. A global variable is also known as an external variable. The variables i, balance and sum are called local variables because they are declared inside a function. Local variables are visible and meaningful only inside the functions in which they are declared. They are not known to the other functions. Note that the variable i has been declared in both the functions. Any change in the value of I in one function does not affect its value in other. The C provides a variety of storage class specifiers that can be used to declare explicitly the scope and lifetime of variables. The concept of scope and lifetime are important only in multifunction and multiple file programs and therefore the storage classes are considered in detail later when functions are discussed. For now, remember that there are four storage class specifiers (auto, register, static, and extern) whose meanings are given in Table 1.11.

Table 1.11: Storage classes and their meaning

Storage class	Meaning
Auto	Local variable known to only to the function in which it is declared. *Default* is auto.
Static	Local variable which exists and retains its value even after the control is transferred to the calling function.
Extern	Global variable known to all functions in the file.
Register	Local variable which is stored in the register.

The storage class is another qualifier (like long or unsigned) that can be added to a variable declaration as shown below:

```
auto int count;
register char ch;
static int x;
extern long total;
```

Static and external (extern) variables are automatically initialized to zero. Automatic (auto) variables contain undefined values (known as 'garbage') unless they are initialized explicitly.

Assigning Values to Variables

Variables are created for use in program statements such as

```
value = amount + inrate * amount;
while (year <= PERIOD)
{
    ……
year = year + '1 ;
}
```

In the first statement, the numeric value stored in the variable inrate is multiplied by the value stored in amount and the product is added to amount. The result is stored in the variable value. This process is possible only if the variables amount and inrate have already been given values. The variable value is called the *target variable*. While all the variables are declared for their type, the variables that are used in expressions (on the right side of equal (=) sign of a computational statement) must be assigned values before they are encountered in the program. Similarly, the variable year and the symbolic constant PERIOD in the while statement must be assigned values before this statement is encountered.

1.7.1 Declaring a Variable as Constant

We may like the value of certain variables to remain constant during the execution of a program. We can achieve this by declaring the variable with the qualifier const at the time of initialization. Example:

```
const int class_size = 40;
```

const is a new data type qualifier defined by ANSI standard. This tells the compiler that the value of the int variables class_size must not be modified by the program. However, it can be used on the right_hand side of an assignment statement like any other variable.

Declaring a Variable as Volatile

ANSI standard defines another qualifier volatile that could be used to tell explicitly the compiler that a variable's value may be changed at any time by some external sources (from outside the program).

For example:

volatile int date;

The value of date may be altered by some external factors even if it does not appear on the left-hand side of an assignment statement. When we declare a variable as volatile, the compiler will examine the value of the variable each time it is encountered to see whether any external alteration has changed the value.

Remember that the value of a variable declared as volatile can be modified by its own program as well. If we wish that the value must not be modified by the program while it may be altered by some other process, then we may declare the variables as both cont and volatile as shown below:

volatile const int location = 100

Overflow and Underflow of Data

Problem of data overflow occurs when the value of a variable is either too big or too small for the data type to hold. The largest value that a variable can hold also depends on the machine. Since floating point values are rounded off to the number of significant digits allowed (or specified), an overflow normally results in the largest possible real value, whereas an underflow results in zero.

Integers are always exact within the limits of the range of the integral data types used. However, an overflow which is a serious problem may occur if the data type does not match the value of the constant. The C does not provide any warning or indication of integer overflow. It simply gives incorrect results. (Overflow normally produces a negative number.) We should therefore exercise a greater care to define correct data types for handling the input/output values.

Reading Data from Keyboard

Another way of giving values to variables is to input data through keyboard using the scanf function. It is a general input function available in C and is very similar in concept to the printf function. It works much like an INPUT statement in BASIC.

The general format of scanf is as follows:

scanf(“conlrol string”, &variablel, &variable2,....);

The control string contains the format of data being received. The ampersand symbol & before each variable name is an operator that specifies the variable name’s address. We must always use this operator, otherwise unexpected results may occur. Let us look at an example:

scanf(“%d”, &number);

When this statement is encountered by the computer, the execution stops and waits for the value of the variable number to be typed in. Since the control string “%d” specifies that an integer value is to be read from the terminal, we have to type in the value in integer form. Once the number is typed in and the ‘Return’ Key is pressed, the computer then proceeds to the next statement. Thus, the use of scanf provides an interactive feature and makes the program ‘user friendly’.

Example

The program in Figure 2.3 illustrates the use of scanf function.

The first executable statement in the program is a printf, requesting the user to enter an integer number. This is known as “prompt message” and appears on the screen like

Enter an integer number

As soon as the user types in an integer number, the computer proceeds to compare the value with 100. If the value typed is less than 100, then a message

Your number is smaller than 100

is printed on the screen. Otherwise, the message

Your number contains more than two digits

is printed. Outputs of the program, run for two different inputs are also shown in Figure 1.12.

Program

```
/*************************************************************/
/*        INTERACTING COMPUTING USING scanf FUNCTION         */
/*************************************************************/
main( )
{
int number;
printf ("Enter an integer number\n");
scanf ("%d", &number);
if ( number < 100 )
printf("Your number is smaller than 100\n\n");
else
printf("Your number contains more than two digits\n");
}
```

Output

```
Enter an integer number
54
Your number is smaller than 100
Enter an integer number
108
Your number contains more than two digits
```

Figure 1.12: Use of scanf function.

Some compilers permit the use of the 'prompt message' as a part of the control string in **scanf**, like
scanf ("Enter a number %d", &number);
In this we have used a decision statement if…else to decide whether the number is less than 100.
In this case, computer requests the user to input the values of the amount to be invested, interest rate and period of investment by printing a prompt message
lnput amount, interest rate, and period and then waits for input values. As soon as we finish entering the three values

Program

```
/*************************************************/
/*          INTERACTIVE INVESTMENT PROGRAM       */
/*************************************************/
main( )
{
int     year, period ;
float  amount, inrate, value ;
printf("lnput amount, interest rate, and period\n\n,,) ;
scanf ("%f %"d", &amount, &inrate, &period) ;
printf("\n") ;
year = 1 ;
while( year <= period )
  {
value = amount + inrate * amount ;
```

```
printt("%2d Rs %8.2f\n", year, value) ;
amount = value ;
year=year+1;
  }
}
Output
lnput amount, interest rate, and period
10000 0.14   5
1 Rs 11400.00
2 Rs 12996.00
3 Rs 14815.44
4 Rs 16889.60
5 Rs 19254.15
Input amount, interest rate, and period
20000   0.12    7
Rs 22400.00
Rs 25088.00
Rs 28098.56
Rs 31470.39
Rs 35246.84
Rs 39476.46
Rs 44213;63
```

Fig. 1.13: Interactive investment program.

corresponding to the three variables amount, inrate, and period, the computer begins to calculate the amount at the end of each year, upto 'period' and produces output.

Note that the scanf function contains three variables. In such cases, care should be exercised to see that the values entered match the order and type of the variables in the list. Any mismatch might lead to unexpected results. The compiler may not detect such errors.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Define Data type?

……..……

……..……...…

……..

Ex2: What is Variable?

……..……

……..……...…

……..

1.8 Operators

The C supports a rich set of operators. We have already used several of them, such as =, *. _, *, & and <. An operator is a symbol that tells the computer to perform certain mathematical or logical manipulations. Operators are used in programs to manipulate data and variables. They usually form a part of the mathematical of logical expressions.

The C operators can be classified into a number of categories. They include:

1. Arithmetic operators.
2. Relational operators.
3. Logical operators.
4. Assignment operators.
5. Increment and decrement operators.
6. Conditional operators.
7. Bitwise operators.
8. Special operators.

1.8.1 Arithmetic Operators

The C provides all the basic arithmetic operators. They are listed in Table 1.14. The operators +, -, *, and / all work the same way as they do in other languages. These can operate on any built-in data type allowed in C. The unary-minus operator, in effect, multiplies its single operand by -1. Therefore, a number preceded by a minus sign changes its sign.

Table 1.14: Arithmetic operators

Operator	Meaning
+	Addition or unary plus
-	Subtraction or unary minus
x	Multiplication
/	Division
%	Module division

Integer division truncates any fractional part. The module division produces the remainder of an integer division. Examples of arithmetic operators are:

a – b	a + b
a * b	a/b
a%b	-a * b

Here a and b are variables and are known as operands. The module division operator % cannot be used on floating point data.

Note that C does not have an operator for exponentiation.

Integer Arithmetic

When both the operands in a single arithmetic expression such as a+b are integers, the expression is called an integer expression and the operation is called integer arithmetic. Integer arithmetic always yields an integer value. The largest integer value depends on the machine, as pointed out earlier. In the above examples, if a and b are integers, then for a = 14 and b = 4

we have the following results:

a – b = 10
a + b = 18
a * b = 56
a / b = 3 (decimal part truncated)

a % b = 2 (remainder of division)

During integer division, if both the operands are of the same sign, the result is truncated towards zero. If one of them is negative, the direction of truncation is implementation dependent. That is,

6/7 = and -6/-7= 0

but -6/7 may be zero or -1. (Machine dependent)

Similarly, during modulo division, the sign of the result is always the sign of the first operand (the dividend.) That is

-14 % 3 = -2
-14 % -3 = -2
14 % -3 = 2

Example: The program in Figure 1.4 shows the use of integer arithmetic to convert a given number of days into months and days.

The variables months and days are declared as integers. Therefore, the statement

months = days/30;

truncates the decimal part and assigns the integer part to months. Similarly, the statement

days = days%30;

assigns the remainder part of the division to days. Thus the given number of days is converted into an equivalent number of months and days and the result is printed as shown in the output.

Program

```
/*************************************************************************/
/*              PROGRAM TO CONVERT DAYS TO MONTHS AND DAYS               */
/*************************************************************************/
main( )
{
        int     months, days;
        printl("Enter days\ n");
        scanf("%d", &days);
        months = days/30;
        days = days % 30;
        printf("Months = %d     Days =   %d", months, days);
}
```

Output

```
Enter days
265
Months=8        Days=25
Enter days
364
Months = 12     Days = 4
Enter days
45
Months = 1      Days = 15
```

Figure1.15: Illustration of integer arithmetic.

Real Arithmetic

An arithmetic operation involving only real operands is called real arithmetic. A real operand may assume values either in decimal or exponential notation. Since floating point values are rounded to the number of significant digits permissible, the final value is an approximation of the correct result. If x, y, and z are floats, then we will have:

x = 6.017.0 = 0.857143
y = 1.0/3.0 = 0.333333
z = -2.0/3.0 = -0.66667

The operator % cannot be used with real operands.

Mixed-mode Arithmetic

When one of the operands is real and the other is integer, the expression is called a mixed-mode arithmetic expression. If either operand is of the real type, then only the real operation is performed and the result is always a real number. Thus

15/10.0 = 1 .5

where as

15/10 = 1

Precedence of Arithmetic Operators

The arithmetic expression without parenthesis will be evaluated from left to right using the rules of precedence of operators. There are two distinct priority levels of arithmetic operators in c high priority and low priority. The basic evaluation procedure includes two left to right passes to the expression during the first pass the high priority operators are applied as they are encountered during the second pass the low priority operators are applied as they are encountered.

Some Computational Problems

When expressions include real values, then it is important to take necessary precautions to guard against certain computational errors. We know that the computer gives approximate values for real numbers and the errors due to such approximations may lead to serious problems. For example, consider the following statements:

a : 1.0/3.0;
b = a’3.0;

We know that (1 .0/3.0)3.0 is equal to 1. But there is no guarantee that the value of b computed in a program will equal 1.

Another problem is division by zero. On most computers, any attempt to divide a number by zero will result in abnormal termination of the program. In some cases such a division may produce meaningless results. Care should be taken to test the denominator that is likely to assume zero value and avoid any division by zero.

The third problem is to avoid overflow or underflow errors. It is our responsibility to guarantee that operands are of the correct type and range, and the result may not produce any overflow or underflow.

Example:

Output of the program in Figure.1.16 shows round-off errors that can occur in computation of floating point numbers.

Program

/********** /

```
/*                PROGRAM SHOWING ROUND-OFF ERRORS                */
/*                        Sum of n terms of l/n                   */
/*****************************************************************/
main( )
{
        float   sum, n, term;
        int     count = 1;
        sum = 0;
        printf("Enter value of n\n");
        scanf("%f", &n);
        term = 1.0/n;
        while( count <= n )
        {
                sum = sum + term;
                count++;
        }
        printf("Sum = %f\n", sum);
}
Output
Enter value of n
99
Sum = 1.000001
Enter value of n
143
Sum = 0.999999
```

Figure 1 16: Round-off errors in floating point computations.

We know that the sum of n terms of 1/n is 1. However, due to errors in floating point representation, the result is not always 1.

1.8.2 Relational Operators

We often compare two quantities and depending on their relation take certain decisions. For example, we may compare the age of two persons, or the price of two items, and so on.

These comparisons can be done with the help of relational operators. We have already used symbol <, meaning 'less than'. An expression such as

a < b or 1 < 20

containing a relational operator is termed as a relational expression. The value of a relational expression is either one or zero. It is one if the specified relation is true and zero if the relation is false. For example:

10 < 20 is true

But 20 < 10 is false

The C supports six relational operators in all. These operators and their meanings are shown in Table 1.17.

Table 1.17: Relational Operator

Operator	Meaning
<	is less than
<=	is less than or equal to

>	is greater than
>=	is greater than or equal to
= =	is equal to
!=	is not equal to

A simple relational expression contains only one relational operator and takes the following form:

ae-1 relational operator ae -2

ae-1 and ae -2 are arithmetic expressions, which may be simple constants, variables or combination of them. Given below are some examples of simple relational expressions and their values:

4.5 < =	10	True
4.5 <	-10	False
-35 >=	0	False
10 <	7+5	True
a+b = =	c + d	True

only if the sum of values of a and b is equal to the sum of values of c and d. When arithmetic expressions are used on either side of a relational operator, the arithmetic expressions will be evaluated first and then the results compared. That is, arithmetic operators have a higher priority over relational operators.

Relational expressions are used in decision statements such as, if and while to decide the course of action of a running program. We have already used the while statement.

1.8.3 Logical Operators

In addition to the relational operators, the C has the following three logical operators.

&& meaning	logical AND
¦¦ meaning	logical OR
! meaning	logical NOT

The logical operators && and ¦¦ are used when we want to test more than one condition and make decisions. An example is:

a > b && x = = 10

An expression of this kind which combines two or more relational expressions is termed as a logical expression or a compound relational expression. Like the simple relational expressions, a logical expression also yields a value of one or zero, according to the truth table shown in Table 1.18. The logical expression given above is true only if a > b is true and x == 10 is true. If either (or both) of them are false, the expression is false.

Table 1.18: Truth Table

op-1	**op-2**	**Value of the expression**	
		op-1 && op-2	**op-1 ¦¦ op-2**
Non-Zero	Non-Zero	1	1
Non-Zero	0	0	1
0	Non-Zero	0	1
0	0	0	0

Some examples of the usage of logical expressions are:

1. if (age > 55 && salary < 1000)
2. if (number < 0 ¦¦ number > 100)

1.8.4 Assignment operators

Assignment operators are used to assign the result of an expression to a variable. We have seen the usual assignment operator, ' = '. In addition, the C has a set of 'shorthand' assignment operators of the form

v op = exp;

Where v is a variable, exp is an expression and op is a C binary arithmetic operator. The operator op = is known as the shorthand assignment operator.

The assignment statement

v op: exp

is equivalent to

v = v op(exp);

with v evaluated only once. Consider an example

x + = y+l

This is same as the statement

x = x + (y+1);

The shorthand operator + = means 'add y+1 to x' or 'increment x by y + 1'. For y = 2, the above statement becomes

x + = 3;

and when this statement is executed, 3 is added to x. If the old value of x is, say 5, then the new value of x is 8. Some of the commonly used shorthand assignment operators are illustrated in Table 1.19.

Table 1.19: Shorthand Assignment Operators

Statement with simple assignment operator	Statement with shorthand operator
a = a + 1	a + = 1
a = a – 1	a - = 1
a = a * (n+1)	a * = (n+1)
a = a / (n+1)	a / = n +1
a = a % b	a % = b

The use of shorthand assignment operators has three advantages:

1. What appears on the left-hand side need not be repeated and therefore it becomes easier to write.
2. The statement is more concise and easier to read.
3. The statement is more efficient.

These advantages may be appreciated if we consider a slightly more involved statement, like

value(5*i-2) = value(5*i-2) + delta;

With the help of the + = operator, this can be written as follows:

value(5*i-2) + = delta;

It is easier to read and understand, and is more efficient because the expression 5*i-2 is evaluated only once.

Example program of Figure 1.6 prints a sequence of squares of numbers. Note the use of the shorthand operator * =

The program attempts to print a sequence of squares of numbers starting from 2. The statement

a * = a;

which is identical to

a = a*a;

replaces the current value of a by its square. When the value of a becomes equal or greater than **N** (=100) the while is terminated. Note that the output contains only three values 2, 4 and 16.

Program

```
/*************************************************************************/
/*            PROGRAM TO SHOW USE OF SHORTHAND OPERATORS          */
/*************************************************************************/
#define  N  100
#define  A  2
main()
{
      int    a;
      a =   A;
      while( a < N )
       {
            printf("%d\n", a);
            a * = a;
        }
}
Output
2
4
16
```

Figure 1.20: Use of shorthand operator.

1.8.5 Increment and decrement operators

The C has two very useful operators not generally found in other languages. These are the increment and decrement, operators:

+ + and - -

The operator + + adds 1 to the operand while - - subtracts 1. Both are unary operators and take the following form:

++m; or m++

.- - m; or m - -

+ + m; is equivalent to m = m +.1; (or m += l;)

- - m; is equivalent to m = m -.1; (or m - = l;)

We use the increment and decrement statements in for and while loops extensively.

While + + m and m + + mean the same thing when they form statements independently, they behave differently when they are used in expressions on the right-hand side of an assignment statement. Consider the following:

```
m = 5;
y = ++m;
```

In this case the value of y and m would be 6. Suppose, if we rewrite the above statements as

```
m = 5;
```

then, the value of y would be 5 and m would be 6. A prefix operator first adds 1 to the operand and then the result is assigned to the variable on left. On the other hand a postfix operator first assigns the value to the variable on left and then increment the operand.

Similar is the case, when we use + + (or - -) in subscripted variables. That is, the statement is equivalent to

```
a[1++] = 10;
```

is equivalent to

```
a[i] = 10;
```

i = i + 1;

The increment and decrement operators can be used in complex statements.

Example:

m = n + + -i + 10

Old values of n is used in evaluating the expression. N is incremented after the evaluation. Some compliers require a space on either side of n + + or + + n.

1.8.6 Conditional Operator

A ternary operator pair "? :" is available in C to construct conditional expressions of the form

exp1 ? exp2 : exp3;

where *exp1, exp2,* and *exp3* are expressions.

The operator ? : works as follows: exp1 is evaluated first. If it is nonzero (true), then the expression exp2 is evaluated and becomes the value of the expression. If exp1 is false, exp3 is evaluated and its value becomes the value of the expression. Note that only one of the expression is evaluated. For example, consider the following statements:

```
a = 10;
b = 15;
x = (a > b) ? a : b;
```

In this example, x will be assigned the value of b. This can be achieved using the if else statements as follows:

```
if(a>b)
x = a;
else
x = b;
```

1.8.7 Bitwise Operators

The C has a distinction of supporting special operators known as bitwise operators for manipulation of data at bit level. These operators are used for testing the bits, or shifting them right or left.

Bitwise operators may not be applied to float or double. Table 1.11 lists the bitwise operators and their meanings.

Table 1.21: Bitwise operators

Operator	Meaning
&	bitwise AND
!	bitwise OR
<<	shift left
>>	shift right
~	One's complement

1.8.8 Special Operators

The C supports some special operators of interest such as comma operator, **sizeof** operator, pointer operators (& and *) and member selection operators (. and ->). ANSI committee has introduced two preprocessor operators known as "stringizing" and "token-pasring" operarors (# and ##).

Comma Operator

The comma operator can be used to link the related expressions together. A comma-linked list of expressions is evaluated left to right and the value of right-mist expression is the value of the combined expression. For example, the statement

value = (x = 10, y = 5, x+y);

first assigns the value 10 to x, then assigns 5 to y, and finally assigns 15 (i.e, 10+5) to value.
Since comma operator has the lowest precedence of all operators, the parentheses are necessary.
Some applications of comma operator are:

In for loops:
for (n = 1, m = 10; n <= m; n++, m++)
In while loops:
while(c=getchar (), c! = '10')
Exchanging values:

t = x, x = y, y = t;

The sizeof Operator

The sizeof is a compile time operator and when used with an operand, it returns the number of bytes the operand occupies. The operand may be a variable a constant or a data type qualifier.
Examples:

m = sizeof(sum);
n = sizeof(long int);
k = sizeof(235L);

The sizeof operator is normally used to determine the lengths of arrays and structures when their sizes are not known to the programmer. It is also used to allocate memory space dynamically to variables during execution of a program.
Notice the way the increment operator ++ works when used in an expression in the statement c = ++a - b new value of a (= 16) is used thus giving the value 6 to c. i.e. a is incremented by 1 before it is used in the expression. However in the statement

d = b++ + a

the old value of b (=10) is used in the expression. Here b is incremented by 1 after it is used in the expression.

Self Assessment Questions

5. printf() s use for ……
(a) Store (b) Input
(c) Print (d) None of these.

6. In C There are……types of errors
(a) Three (b) Four
(c) Five (d) Two

7. Who invented C programming language?
(a) Dennis Richie at Bell Telephone
(b) Martin Richards
(c) Kernighan
(d) Ken Thompson

8. In C language all statements are entered in case letters.
(a) Small (b) Capital
(c) Only A (d) both a and b

9. Comment about the program should be enclosed within

(a) { } (b) ( )

(c) < >. (d) /* */

10 include<stdio.h>. are

(a) Preprocessor (b) Function

(c) File name (d) comment

1.9 Summary

- The C is a programming language developed at AT & T's Bell Laboratories of USA in 1972.
- ANSI C standard—in 1983, the American National Standards Institute (ANSI) commissioned a committee, X3J11, to standardize the C language.
- The character set that used to form words numbers and expression depend on the computer on which the programs run. Constants in c refer to fixed values.
- Data types in C language is reach in its data type storage representation and machine instruction to hand IC constants, variables.
- A variable character defined as a character (char) type data.

1.10 Keywords

ANSI: In 1983, the American National Standards Institute (ANSI) commissioned a committee, X3J11, to standardize the C language.

***C character set*:** The characters that can be used to form words, numbers and expressions depend upon the computer on which the program is run.

***Constants*:** Constants in C refer to fixed values that do not change during the execution of a program.

***Data-types*:** Data-type helps the programmer to select the type appropriate to the needs of the application as well as the machine.

Decompiler: A program that translates from a low level language to a higher level one is a *decompiler*.

***Integer Constants*:** An integer constant refer to a sequence of digits

Keywords and identifiers: Keywords serve as basic building blocks for program statements whereas; identifiers refer to the names of variables, functions and arrays.

main():.The 'main()' function is the most important function and must be present in every C program.

Object Oriented Programming (OOP): Object-oriented programming (OOP) is a programming paradigm using "objects" – data structures consisting of data fields and methods together with their interactions.

***Real Constants*:** Number containing fractional parts like 17.548 is called real constants.

***String Constants*:** A string constant is a sequence of characters enclosed in double quotes.

Tokens: In a passage of text, individual words and punctuation marks are called tokens.

Trigraph Character: Trigraph sequences provide a way to enter certain characters that are not available on some keyboards.

***Variables*:** A variable is a data name that may be used to store a data value.

1.11 Review Questions

1. What are the basic concepts of programming in C?
2. What is the "ANSI C Standard"?
3. Write the programming rules for C.

4. Why does the ANSI standard place limits on the length and case significance of external identifiers?
5. Discuss the C character set.
6. What are the concepts of constants in C?
7. Define the different data types in C.
8. What are the variables? What is the use of variables in C?
9. What are the types of operators used in C? Discuss their preference and associativity chart.
10. What are the arithmetic expressions? Why and how do we use it in C language?

Answers for Self Assessment Questions

1. (b)	2. (a)	3. (a)	4. (b)	5. (c)
6. (a)	7. (a)	8. (a).	9. (d)	10. (a)

1.12 Further Studies

1. http://163.25.99.51/Huang_Computer/Programming%20Language-website2.pdf
2. Let us c by Yashwant Kanatkar.

2

Decision Making and Branching

STRUCTURE

2.0 Objectives

2.1 Introduction

2.2 Sequential Statements

2.3 Unformatted I/O Functions

2.4 Formatted Input using scanf()

2.5 Formatted Output using print()

2.6 Branching Statements

2.7 The Switch Statement

2.8 Summary

2.9 Keywords

2.10 Review Questions

2.11 Furhter Studies

2.0 Objectives

After studying this chapter, you will be able to:

- Discuss the sequential statements in C
- Understand the unformatted I/O functions
- Explain the formatted input
- Explain the formatted output
- Define the branching statements in C
- Discuss the switch statement

2.1 Introduction

"Decision making" is one of the most important concepts of computer programming. Programs should be able to make logical (true/false) decisions based on the condition they are in; every program has one or few problem/s to solve; depending on the nature of the problems, important decisions have to be made in order to solve those particular problems.

In C programming "selection construct" or "conditional statement" is used for decision making. Figure 2.1 illustrates "selection construct".

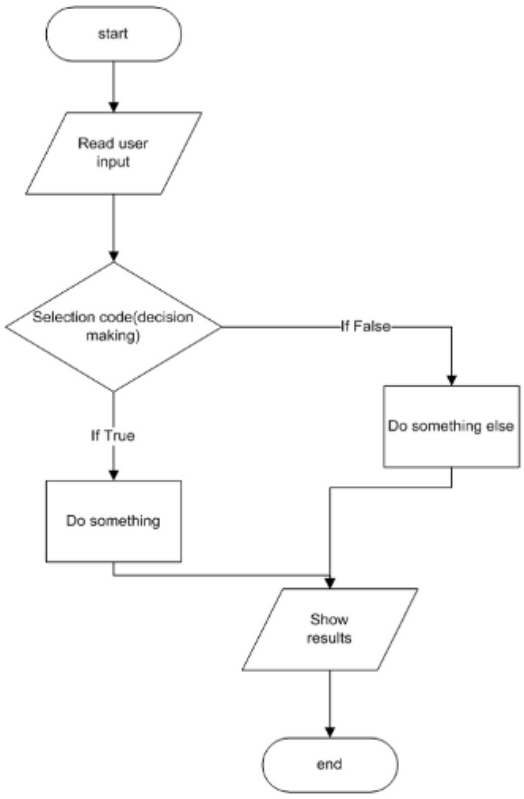


Figure 2.1: Simple selection construct.

Conditional statement is the term used by many programming languages. The importance of conditional statements should not be ignored because every program has an example of these statements. "if statement" and "switch statement" are the most popular conditional statements used in C.

Branching

Branch is the term given to the code executed in sequence as a result of change in the program's flow; the program's flow can be changed by conditional statements in that program. Figure 2.2 shows the link between selection (decision making) and branching (acting).

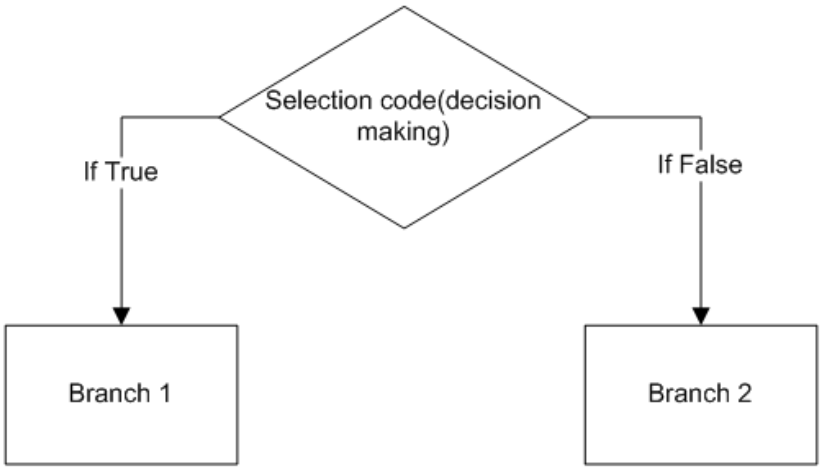


Figure 2.2: "branch" depends on the condition of the selection code.

Branching is the process of choosing the right branch for execution, depending on the result of “conditional statement”.

2.2 Sequential Statements

Sequential statements are those statements in a program that executes one by one in a given sequence.
For example following statements:

```
1. float a,b;
2. printf(“Give the first number:”);
3. scanf(“%f”,&a);
4. printf(“Give the second number:”);
5. scanf(“%f”,&b);
```

Above five statements is part of sequential statements of a C program. These statements will be executed one by one in a C program.

2.3 Unformatted I/O Functions

This is simple: only two functions provide this facility, one for reading and one for writing:

```
#include <stdio.h>
size_t fread(void *ptr, size_t size, size_t nelem, FILE *stream);
size_t fwrite(const void *ptr, size_t size, size_t nelem, FILE *stream);
```

In each case, the appropriate read or write is performed on the data pointed to by ptr. Up tonelem elements, of size size, are transferred. Failure to transfer the full number is an error only when writing; End of File can prevent the full number on input. The number of elements actually transferred is returned. To distinguish between End of File on input, or an error, use feof orferror.
If size or nelem is zero, fread does nothing except to return zero.
An example may help.

```
#include <stdio.h>
#include <stdlib.h>
struct xx{
        int xx_int;
        float xx_float;
}ar[20];
main(){
        FILE *fp = fopen(“testfile”, “w”);
        if(fwrite((const void *)ar,
                sizeof(ar[0]), 5, fp) != 5){
                fprintf(stderr,”Error writing\n”);
                exit(EXIT_FAILURE);
        }
        rewind(fp);
        if(fread((void *)&ar[10],
                sizeof(ar[0]), 5, fp) != 5){
                if(ferror(fp)){
                        fprintf(stderr,”Error reading\n”);
                        exit(EXIT_FAILURE);
                }
```

```
            if(feof(fp)){
                fprintf(stderr,"End of File\n");
                exit(EXIT_FAILURE);
            }
        }
        exit(EXIT_SUCCESS);
    }
```

2.4 Formatted Input using scanf()

Formatted input refers to an input data that has been arranged in a particular format. For example, consider the following data:

15.75 123 John

This line contains three pieces of data, arranged in a particular form. Such data has to be read conforming to the format of its appearance. For example, the first part of the data should be read into a variable float, the second into int, and the third part into char. This is possible in C using the scanf function.

We have already used this input function in a number of examples. Here, we shall explore all of the options that are available for reading the formatted data with scanf function. The general form of scanf is

```
Scanf ("control string", arg1, arg2,……argn);
```

The control string specifies the field format in which the data is to be entered and the arguments arg1 ,arg2,.....argn specify the address of locations where the data is stored. Control string and arguments are separated by commas.

Control string contains field specifications which direct the interpretation of input data. It may include:

- Field (or format) specifications, consisting of the conversion character %,a data type character (or type specifier), and an optional number, specifying field width.
- Blanks, tabs, or newlines.

Blanks tabs and newlines are ignored. The data type character indicates the type of data that is to be assigned to the variable-associated with the corresponding argument. The field width specifier is optional. The discussions that follow will clarify these concepts.

2.4.1 Inputting Integer Numbers

The field specification for reading an integer number is:

```
%wd
```

The percent sign (%) indicates that a conversion specification follows. w is an integer number that specifies the field width of the number to be read and d, known, as data type character, indicates that the number to be read is in integer mode. Consider the following example:

```
scanf("%2d %5d", &num1, &num2);
```

Data line:

```
50    31426
```

The value 50 is assigned to num1 and 31426 to num2. Suppose the input data is as follows:

```
31426  50
```

The variable num1 will be assigned 31 (because of %2d) and num2 will be assigned 426 (unread part of 31426). The value 50 that is unread will be assigned to the first variable in the next scanf call. This kind of

errors may be eliminated if we use the field specifications without the field width specifications. That is, the statement

Scanf("%d %d", &num1 &num2.)

will read the data

31426 50

correctly and assign 31426 to num1 and 50 to num2.

What happens if we enter a floating point number instead of an integer? The fractional part may be stripped away! Also, scanf may skip reading further input.

When the scanf reads a particular value, reading of the value will terminate as soon as the number of characters specified by the field width is reached (if specified) or until a character that is not valid for the value being read is encountered. In the case of integers, valid characters are an optionally signed sequence of digits.

An input field may be skipped by specifying * in the place of field width. For example, the statement

scanf("%d 7od", &num1, &num2);

will assign the data

31426 50

as the follows: 123 456 789

123 to a

456 skipped (because of *)

789 to b

The data type character d may be preceded by 'r' (letter ell) to read long integers.

Example :

Various input formatting options for reading integers are experimented with in the program shown in Figure 2.3.

The first scanf requests input data for three integer values a, b, and c, and accordingly three values 1, 2, and 3 are keyed in. Because of the specification %* d the value 2 has been skipped and 3 is assigned to the variable b. Notice that since no data is available for c, it contains garbage.

The second scanf specifies the format %2d and %4d for the variables x and y respectively. Whenever we specify field width for reading integer numbers, the input numbers should not contain more digits than the specified size. Otherwise, the extra digits on the right-hand side will be truncated and assigned to the next variable in the list. Thus, the second scanf has truncated the four digit number 6789 and assigned 67 to X and 89 to y. the value 4321 has assigned to the first variable in the immediately following scanf statement.

```
main()
  {
     int a,b,c,x,y,z;
     int p,q,r;

     printf("Enter three integer numbers\n");
     scanf("%d %*d %d",&a,&b,&c);
     printf("%d %d %d \n\n",a,b,c);

     printf("Enter two 4-digit numbers\n");
     scanf("%2d %4d",&x,&y);
     printf("%d %d\n\n", x,y);

     printf("Enter two integers\n");
     scanf("%d %d", &a,&x);
```

```
    printf("%d %d \n\n",a,x);

    printf("Enter a nine digit number\n");
    scanf("%3d %4d %3d",&p,&q,&r);
    printf("%d %d %d \n\n",p,q,r);

    printf("Enter two three digit numbers\n");
    scanf("%d %d",&x,&y);
    printf("%d %d",x,y);
}
Output
Enter three integer numbers
1 2 3
1 3 –3577
Enter two 4-digit numbers
6789  4321
67 89
Enter two integers
44  66
4321 44
Enter a nine-digit number
123456789
66 1234 567
Enter two three-digit numbers
123 456
89 123
```

Figure 2.3: Reading integers using scanf.

Caution

Input data items must be separated by spaces, tabs or newlines punctuation marks do not count as separators because when the scanf function searches the input data line for a value to be, read, it will always bypass any white space characters.

2.4.2 Inputting Real Numbers

Unlike integer numbers, the field width of real numbers is not to be specified and therefore scanf reads real numbers using the simple specification %f for both the notations, namely, decimal point notation and exponential notation For example, the statement

scanf("%f %f ", &x, &y, &z);

With the input data

475.89 43.21E-1 678

will assign the value 475.89 to x 4.321 to y and 678.0z. The input field specifications may be separated by any arbitrary blank spaces.

If the number to be read is of double type then the specification should be %1f instead of simple %f. A number may be skipped using % *f specification.

Example : Reading of real numbers (in both decimal point and exponential notation) is illustrated in Figure 2.4.

```
main()
  {
     float x,y;
     double p,q;
     printf("Values of x and y:");
     scanf("%f %e", &x, &y);
     printf("\n");
     printf("x = %f\ny = %f\n\n", x, y);
     printf("Values of p and q:");
     scanf("%lf %lf", &p, &q);
     printf("\np = %lf\nq = %e",p,q);
     printf("\n\np = %.12lf\np = %.12e", p, q);
  }
Output

  Values of x and y:12.3456  17.5e-2
  x = 12.345600
  y = 0.175000
  Values of p and q:4.142857142857  18.5678901234567890
  p = 4.142857142857
  q = 1.856789012346e+001
```

Figure 2.4: Reading of real numbers.

2.4.3 Inputting Character String

We have already seen how a single character can be read from the terminal using the getchar function. The same can be achieved using the scarf function also. In addition, a scarf function can input strings containing more than one character. Following are the specifications for reading character strings:

The corresponding argument should be a pointer to a character array. However, %c may be used to read a single character when the argument is a pointer to a char variable.

Example:

Reading of strings using %wc and %ws is illustrated in Figure. 2.5.

The program in Figure 3.5 illustrates the use of various field specifications for reading strings. When we use %wc for reading a string, the system will wait until the with character is keyed in.

Note that the specification %s terminates reading at the encounter of a blank space. Therefore, name2 has read only the first part of "New York" and the second part is automatically assigned to name3. However, during the second run, the string "New-York" is correctly assigned to name2.

Some versions of scanf support the following conversion specifications for strings:

%[characters] and%[Acharacters]

The specification %[characters] means that only the characters specified within the brackets are permissible in the input string. If the input string contains any other character, the string will be terminated at the first encounter of such a character.

```
main()
  {
     int no;
     char name1[15], name2[15], name3[15];
     printf("Enter serial number and name one\n");
```

```
    scanf("%d %15c", &no, name1);
    printf("%d %15s\n\n", no, name1);
    printf("Enter serial number and name two\n");
    scanf("%d %s", &no, name2);
    printf("%d %15s\n\n", no, name2);
    printf("Enter serial number and name three\n");
    scanf("%d %15s", &no, name3);
    printf("%d %15s\n\n", no, name3);
}
Output
Enter serial number and name one
1 123456789012345
1 123456789012345r
Enter serial number and name two
2 New York
2          New
Enter serial number and name three
2          York
Enter serial number and name one
1 123456789012
1 123456789012   r
Enter serial number and name two
2 New-York
2 New-York
Enter serial number and name three
3 London
3 London
```

Figure 2.5: Reading of strings.

The specification % [characters] does exactly the reverse. That is, the characters specified after the circumflex ( ) are not permitted in the input string. The reading of the string will be terminated at the encounter of one of these characters.

We have just seen that % specifier cannot be used to read strings with blank spaces. But, this can be done with the help of % [] specification. Blank spaces may be included within the brackets, thus enabling the scanf to read strings with spaces. Remember that the lowercase and uppercase letters are distinct.
Example illustrates the use of % [] specification.
Example
The program in Figure 3.6 illustrates the function of % [ ] specification.

2.4.4 Reading Mixed Data Types

It is possible to use one scanf statement to input a data line containing mixed mode data. In such cases, care should be exercised to ensure that the input data items match the control specifications in order and type. When an attempt is made to read an item that does not match the type expected, the scanf function does not read any further and immediately returns the values read. The statement
scanf("%d %c %f %s", &count, &code, &ratio, name);

will read the data correctly and assign the values to the variables in the order in which they appear.

15 p 1.575 coffee

Some systems accept integers in the place of real numbers and vice-versa, and the input data is converted to the type specified in the control string.

2.4.5 Detection of Errors in Input

When a scanf function completes reading its list, it returns the value of number of items that

```
main()
  {
     char address[80];

     printf("Enter address\n");
     scanf("%[a-z]", address);
     printf("%-80s\n\n", address);
  }

Output
  Enter address
  new delhi 110 002
  new delhi

main()
  {
     char address[80];
     printf("Enter address\n");
     scanf("%[^\n]", address);
     printf("%-80s\n\n", address);
  }
  Output
  Enter address
  New Delhi 110 002
  New Delhi 110 002
```

Figure 2.6: Illustration of conversion specification [.] for string.

are successfully read. This value can be used to test whether any errors occurred in reading the input. For example, the statement

scanf("%d %f %s", &a, &b, name);

will return the value 3 if the following data is typed in:

20 150.25 motor

and will return the value 1 if the following line is entered

20 motor 150.25

This is because the function would encounter a string when it was expecting a floating point value, and would therefore terminate its scan after reading the first value.

Example: The program presented in Figure 2.7 illustrates the testing for correctness of reading of data by scanf function

```
main()
  {
     int a;
     float b;
     char c;
     printf("Enter values of a, b and c\n");
     if (scanf("%d %f %c", &a, &b, &c) == 3)
       printf("a = %d  b = %f   c = %c\n" , a, b, c);
     else
       printf("Error in input.\n");
  }
  Output      Enter values of a, b and c
  12  3.45  A
  a = 12  b = 3.450000   c = A
  Enter values of a, b and c
  23  78  9
  a = 23  b = 78.000000   c = 9
  Enter values of a, b and c
  8  A  5.25
  Error in input.
  Enter values of a, b and c
  Y  12  67
  Error in input.
  Enter values of a, b and c
  15.75  23  X
  a = 15  b = 0.750000   c = 2
```

Figure 2.7: Detection of errors in scanf input.

The function scanf is expected to read three items of data and therefore, when the values for all the three variables are read correctly, the program prints out their values. During the third run' the second item does not match with the type of variable and therefore the reading is terminated and the error message is printed. Same is the case with the fourth run.

In the last run although data items do not match the variables, no error message has been printed. When we attempt to read a real number for an int variable, the integer part is assigned to the variable and the truncated decimal part to assigned to the next variable. Note the variable the character '2' is assigned to the character variable C.

Commonly used scanf format codes are given in Table 2.8

Table 2.8: scanf Format Codes

%c	read a single character
%d	read a decimal integer
%e	read a floating point value
%f	read a floating point value
%g	read a floating point value
%h	read a short integer
%i	read a decimal, hexadecimal, or octal integer
%o	read an octal integer

%s	read a string
%u	read an unsigned decimal integer
%X	read a hexadecimal integer
%[..]	read a string of word(s)

The following letters may be used as prefix for certain conversion characters.

h for short integers

I for long integers or double

L for long double

Points to Remember While Using scanf

New features are added to these routines from time to time as new versions of systems are released. We should consult the system reference manual before using these routines. Given below are some of the general points to keep in mind while writing a scanf statement.

1. All function arguments, except the control string, must be pointers to variables.
2. Format specifications contained in the control string should match the arguments in order.
3. Input data items must be separated by spaces and must match the variables receiving the input in the same order.
4. When searching for a value, scanf ignores line boundaries and simply looks for the next appropriate character.
5. Any unread data items in a line will be considered as a part of the data input line to the next scanf call.
6. When the field width specifier w is used, it should be large enough to contain the input data size.

Caution

When scanf encounters an 'invalid mismatch' of data or a character that is not valid for the value being read, the reading will be terminated.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: What is Sequential Statements?

……..……

……..……...…

……..

Ex2: What is Detection of Errors in Input?

……..……

……..……...…

……..

2.5 Formatted Output using print()

We have seen the use of printf function for printing captions and numerical results. It is highly desirable that the outputs are produced in such a way that they are understandable and are in an easy-to-use form. It is therefore necessary for the programmer to give careful consideration to the appearance and clarity of the output produced by program.

The printf statement provides certain features that can be effectively exploited to control the alignment and spacing of print-outs on the terminals. The general form of printf statement is

printf (“control string”, arg1, arg2, …….argn);

Control string consists of three types of items:

1. Characters that will be printed on the screen as they appear.
2. Format specifications that define the output format for display of each item.
3. Escape sequence characters such as \n, \t. and \b.

The control string indicates how many arguments follow and what their types are. The arguments arg1, arg2,.....argn are the variables whose values are formatted and printed according to the specifications of the control string.

The arguments should match in number, order and type with the format specifications.

A simple format specification has the following form:

% w.p type-specifier

where w is an integer number that specifies the total number of columns for the output value and p is another integer number that specifies the number of digits to the right of the decimal point (of a real number) or the number of characters to be printed from a string. Both w. and p are optional. Some examples of printf statement are:

```
printf(“Programming in C”);
printf(“ “);
printf(“ \n”);
printf(“%d”, x);
printf(“a = %f\n b = %f”, a, b);
printf(“sum = %d”, 1234);
printf(“\n\n”);
```

printf never supplies a newline automatically and therefore multiple printf statements may be used to build one line of output. A newline can be introduced by the help of a newline character ‘\n’ as shown in some of the examples above.

2.5.1 Output of Integer Numbers

The format specification for printing an integer number is % w d

Where w specifies the minimum field width for the output. However, if a number is greater than the specified field width, it will be printed in full, overriding the minimum specification. d specifies that the value to be printed is an integer. The number is written right-justified in the give field width. Leading blanks will appear as necessary. The following examples illustrated the output of the number 9876 under different formats:

Format	*Output*
printf(“%d”, 9876)	9 8 7 6
printf(“%6d”, 9876)	␣ ␣ 9 8 7 6
printf(“%2d”, 9876)	9 8 7 6
printf(“%-6d”, 9876)	9 8 7 6 ␣ ␣
printf(“%06d”, 9876)	0 0 9 8 7 6

It is possible to force the printing to be left-justified by placing a minus sign directly after the % character as shown in the fourth example above. It is also possible to pad with zeros the leading blanks by placing a 0 (zero) before the field width specifier as shown in the last item above.
Long integer may be printed by specifying Id in the place of d in the format specification.
Example:
The program in Figure2..8 illustrated the output of integer number under various formats.

Program

```
/*************************************************************************************/
/*       PRINTING OF INTEGER NUMBER ……..*/
/*************************************************************************************/
        main()
        {
          int m = 12345;
          long n = 987654;
          printf("%d\n",m);
          printf("%10d\n",m);
          printf("%010d\n",m);
          printf("%-10d\n",m);
          printf("%10ld\n",n);
          printf("%10ld\n",-n);
        }
    Output
          12345
               12345
          0000012345
          12345
               987654
             - 987654
```

Figure 2.9: Formatted output of integers.

2.5.2 Output of Real Numbers

The output of a real number may be displayed in decimal notation using the following format specification:
% w.p f
The integer w indicates the minimum number of positions that are to be used for the display of the value and the integer p indicates the number of digits to be displayed after the decimal point (precision). The value, when displayed, is rounded to p decimal places and printed right-justified in the field of w columns. Leading blanks and trailing zeros will appear as necessary. The default precision is 6 decimal places. The negative numbers will be printed with the minus sign. The number will be displayed in the form [–] mmm– nnn.
We can also display a real number in exponential notation by using the specification
% w.p e
The display takes the form
[–]m.nnnne[±]xx
where the length of the string of n's is specified by the precision p. The default precision is 6. The field width w should satisfy the condition.
$w \geq p + 7$
The value will be rounded off and printed right justified in the field of w columns.

Padding the leading blanks with zeros and printing with left-justification is also possible by introducing 0 or – before the field with specifier w.

The following examples illustrate the output of the number y = 99.7654 under different format specifications:

Format	*Output*
printf("%7.4f",y)	9 8 . 7 6 5 4
printf("%7.2f",y)	_ _ 9 8 . 7 7
printf("%-7.2f",y)	9 8 . 7 7 _ _
printf("%f",y)	9 8 . 7 6 5 4
printf("%10.2e",y)	_ _ 9 . 8 8 e + 0 1
printf("%11.4e",-y)	– 9 . 8 7 6 5 e + 0 1
printf("%-10.2e",y)	9 . 8 8 e + 0 1 _ _
printf("%e",y)	9 . 8 7 6 5 4 0 e + 0 1

print("%–10.2e",y)

9		8	8	e	+	0	1		

print("%e",y)

9		8	7	6	5	4	0	e	+	0	1

Some systems also support a special field specification character that lets the user define the field size at run-time. This takes the following form:

```
printf("%".*f", width, precision, number);
```

In this case, both the field width and the precision are given as arguments which will supply the values for w and p. For example,

```
printf("%* *f",7,2,number);
```

is equivalent to

```
printf("%7.2f",number);
```

The advantage of this format is that the values for width and precision may be supplied at run-time, thus making the format a dynamic one. For example, the above statement can be used as follows:

```
int width = 7;
int precision = 2;
printf("%*.*f", width, precision, number);
```

Example:

All the options of printing a real number are illustrated in Figure 3.9.

```
main( )
{
    float y = 98.7654;
    printf("%7.4f\n", y);
    printf("%f\n",                  y);
    printf("%7.2f\n", );
    printf("%–7.2f\n",              y);
    printf("%07.2f\n",              y);
    printf("%*.*f", 7, 2, y);
    printf(" \n"); printf("%10.2e\n";
    y); printf("%12,4e\n", –y);
```

```
        printf("%–10.2e\n",y);
        printf("%e\n", y);
    }
    Output
   98.7654
   98.765404
   98.77
   98.77
   0098.77
   98.77
   9.88e+001
  –9.8765e+001
   9.88e+001
   9.876540e+001
```

Figure.2.10: Formatted output of real numbers.

Did You Know?

Microsoft C supports only three digits in exponent part.

2.5.3 Printing of a Single Character

A single character can be displayed in a desired position using the format

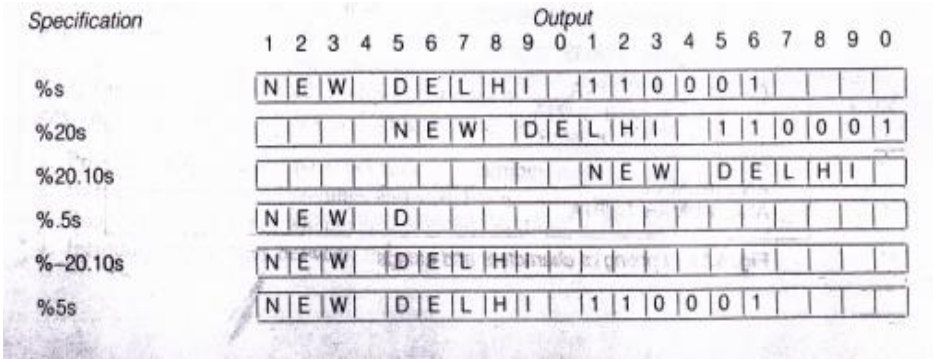
The character will be displayed right-justified in the field of w columns. We can make the display left justified by placing a minus sign before the integer w. The default value for w is 1.

2.5.4 Printing of Strings

The format specification for outputting strings is similar to that of real numbers. It is of the form

%w.ps

where *w* specifies the field width for display and *p* instructs that only the first p characters of the string are to be displayed. The display is right-justified. The following examples show the effect of a variety of specifications in printing a string "NEW DELHI 110001", containing 16 characters (including blanks)



Example: Printing of characters and strings is illustrated in Figure. 2.11.

2.5.5 Mixed Data Output

It is permitted to mix data types in one printf statement. For example, the statement of the type

printf("%d %f %s %c", a, b, c, d); is valid. As pointed out earlier, printf uses its control string to decide how many variables to be printed and what their types are. Therefore, the format specifications should match the

variables in number, order, and type. If there are not enough variables or if they are of the wrong type, incorrect results will be output.

```
Main( )
{
    char x = 'A';
    static char name[20] = "ANIL KUMAR GUPTA"; printf("OUTPUT OF
    CHARACTERS\n\n"); printf("%c\n%3c\n%5c\n", x,x,x);
    printf("%3c\n%c\n", x,x); printf("\n");
    printf("OUTPUT OF STRINGS\n\n"); printf("%s\n", name);
    printf("%20s\n", name); printf("%20,10sin", name);
    printf("%.5s\n", name);
    printf("%–20.10s\n", name);
    printf("%5s\n", name);
}
Output
Output of characters
 A
 A
 A
 A
 A
OUTPUT OF STRINGS
 ANIL KUMAR GUPTA
     ANIL KUMAR GUPTA
          ANIL KUMAR
ANIL.
ANIL KUMAR
ANIL KUMAR GUPTA
```

Figure 2.12: Printing of characters and strings.

Commonly used printf format codes are given in Table 2.12 and format flags in Table 2.13

Table 2.13: printf Format Codes

Code	Meaning
%c	print a single character
%d	print a decimal integer
%e	print a floating point value in exponent form
%f	print a floating point value without exponent
%g	print a floating point value either e-type or f-type depending on value
%i	print a signed decimal integer
%o	print an octal integer, without leading zero
%s	print a string
%u	print an unsigned decimal integer
%x	print a hexadecimal integer, without leading Ox

The following letters may be used as prefix for certain conversion characters.

h for short integers,

l for long integers or double,

L for long double

Table 2.13: Output Format Flags

Flag	Meaning
–	Output is left-justified within the field. Remaining field will be blank
+	+ or – will precede the signed numeric item.
0	Causes leading zeroes to appear.
#(with 0 or x)	Causes octal and hex items to be preceded by 0 and Ox, respectively.
# (with e,f or g)	Causes a decimal point to be present in all floating point numbers, even if it is whole number. Also prevents the truncation of trailing zeros in g-type conversion.

2.5.6 Enhancing the Readability of Output

Computer outputs are used as information for analysing certain relationships between variables and for making decisions. Therefore the correctness and clarity of outputs is of utmost importance. While the correctness depends on the solution procedure, the clarity depends on the way the output is presented, Following are some of the steps we can take to improve the clarity and hence the readability and understand ability of outputs.

1. Provide enough: blank space between two numbers.
2. Introduce; appropriate; headings and variable names in the output.
3. "Print special messages whenever a peculiar condition occurs in the output,
4. Introduce blank lines between the important sections of the output.

The system usually provides two blank spaces between the numbers. However, this can be increased by selecting a suitable field width for the numbers or by introducing a ′tab′ character between the specifications. For example, the statement

```
printf("a = %d\t  b = %d", a, b);
```

will provide four blank spaces between the two fields. We can also print them on two separate lines by using the statement

```
printf("a = %d\n b = %d", a, b);
```

Messages and headings can be printed by using the character strings, directly in the printf statement.
Examples:

```
printf("\n OUTPUT RESULTS        O:n");
printf("code\t Name\t Age\n"):
printf("Error in input data\n");
printf(Enter your name\n");
```

2.6 Branching Statements

The C language programs follow a sequential form of execution of statements. Many times it is required to alter the flow of sequence of instructions. The C language provides statements that can alter the flow of a sequence of instructions. These statements are called as control statements. To jump from one part of the program to another, these statements help. The control transfer may be unconditional or conditional. Branching Statement are of the different categories are as follows:

1. if Statement
2. The if...else Statement
3. Compound Relational tests

4. Nested if Statement
5. Switch Statement

2.6.1 if Statement

The if statement is a powerful decision making statement and is used to control the flow of execution of statements. It is basically a two-way decision statement and is used in conjunction with an expression. It takes the following form:

if (test expression)

It allows the computer to evaluate the expression first and then, depending on whether the value of the expression (relation or condition) is 'true' (non-zero) or 'false' (zero), it transfers the control to a particular statement. This point of program has two paths to follow, one for the true condition and the other for the false condition as shown in Figure. 2.14

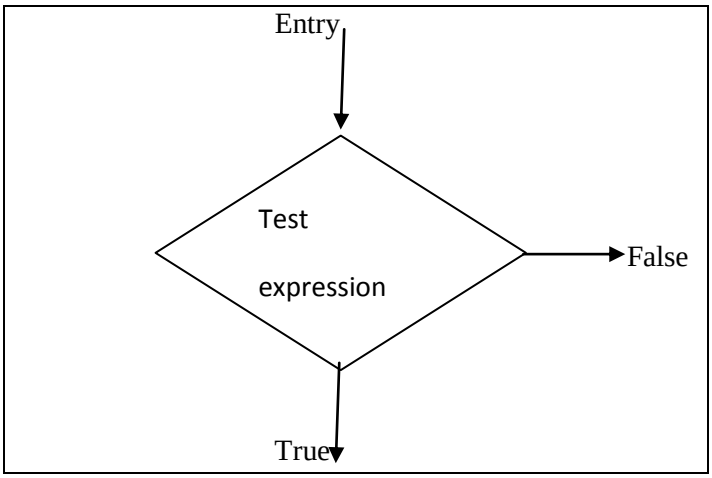


Figure 2.14: Two-way branching.

Some examples of decision making, using if statement are:

```
if (bank balance is zero)
        borrow money
if (room is dark)
        put on lights
if (code is 1)
        person is male
if (age is more than 55)
        person is retired
```

The if statement may be implemented in different forms depending on the complexity of conditions to be tested.

Simple if statement
if ...else statement
Nested if...else statement
else if ladder.

Simple if Statement

The general form of a simple if statement is

```
if(test expression)
{
```

```
    statement-block;
}
statement-x;
```

The 'statement-block' may be a single statement or a group of statements. If the test expression is true, the statement-block will be executed; otherwise the statement-block will be skipped and the execution will jump to the statement-x. Remember, when the condition is true both the statement-block and the statement-x are executed in sequence. This is illustrated in Figure. 2.15.

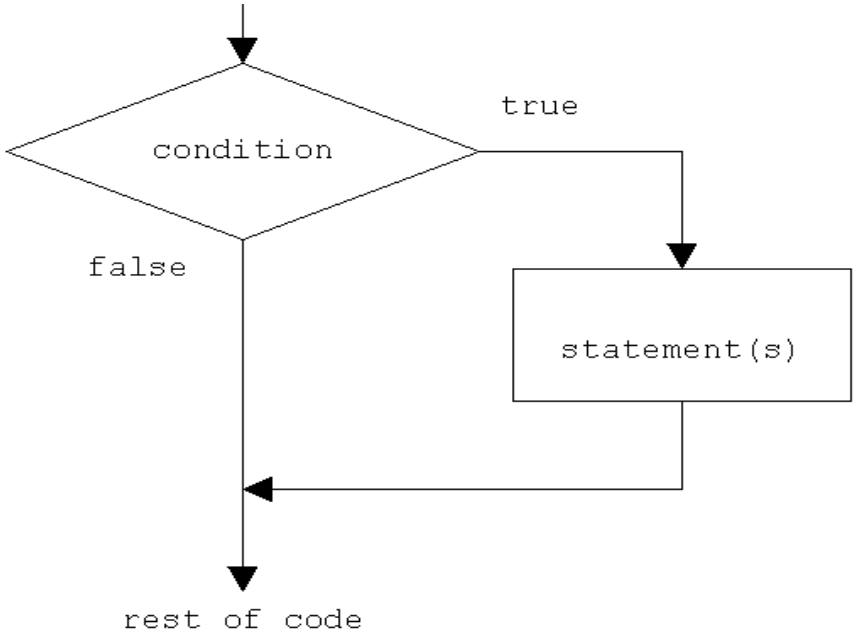


Figure 2.15: Flowchart of simple it controls.

Consider the following segment of a program that is written for processing of marks obtained in an entrance examination.

```
 if (category = SPORTS)
{
marks = marks + bonus_marks;
}
printf("7o|", marks);
```

The program tests'the type of category of the student. If the student belongs to the SPORTS category, then additional bonus_marks are added'to his marks before they are printed. For others, bonus_marks are not added.

Example:

The program in Figure 2.13 reads four values a, b, c, and d from the terminal and evaluates the ratio of (a+b) to (c–d) and prints the result, if c–d is not equal to zero.

The program given in Figure 3.13 has been run for two sets of data to see that the paths function properly. The result of the first run is printed as

Ratio = –3 181818

The second run has neither produced any results nor any message. During the second run, the value of (c–d) is equal to zero and therefore the statements contained in the statement–block are skipped. Since no other statement follows the statement–block, program stops without producing any output.

Program

```
/* ********** Illustration of if Statement  **********/
main( )
{
int a, b, c, d;
float ratio;
printf("Enter four integer values\n");
scanf("%d %d %d ", &a, &b, &c, &d);
if (c–d != 0)
{
ratio = (ftoat) (a+b) / (float) (c_d);
printf("Ratio = %f \n;', ratio);
          }
}
```

Output

```
Enter four integer values
12 23 34–45
Ratio = –3.181818
Enter four integer values
12 23 34 34
```

Figure 2.16: Illustration of simple if statement.

The simple if is often used for counting purposes.

Example :

The program in Figure 3.14 counts the number of boys whose weight is less than 50 kg and height is greater than 170 cm. The program has to test two conditions, one for weight and another for height. This is done using the compound relation

if (weight < 50 && height > 170)

This would have been equivalently done using two if statements as follows:

```
if (weight < 50)
if (height >170)
count= count +1;
```

if the value of weight is less than 50, then the following statement is executed, which in turn is another if statement. This if statement tests height and if the height is greater than 170, then the count is incremented by 1.

Program

```
/************************************/
/*Counting with if                  */
/************************************/
int count, i;
float weight, height;
count = 0;
printf("Enter weight and height for .10 boys\n");
for(i=1; i <= 10; i++)
{
```

```
scanf("%f %f", & wieght, &height)
if(Weight<50 &&height>170)
. count = count + 1;
}
printf("Number of boys,with weight < 50 kgs\n");
printf("and height > 17o cm ='%d/n",count);
}
Output
Enter.weight and height for 10 boys
45 176.5
55 174.2
47 168.0
49 170.7
54 169.0
53 170.5
49 167.0
48 175.0
47 167
5'f 170
Number of boys with weight < 50 kgs and height > 170 cm = 3
```

Figure 2.17: Use of if for counting.

Did You Know?

In imperative programming languages, the term "conditional statement" is usually used, whereas in functional programming, the terms "conditional expression" or "conditional construct" are preferred, because these terms all have distinct meanings.

The if... else statement

The **if..else** statement is an extension of the simple if statement.

```
if (test expression)
{
True-block statement(s)
}
else
{
False-block statement(s)
}
statement-xs
```

If the test expression is true, then the true-block statement (S), immediately following the if statement is executed, otherwise, the false-block (S) are executed. In either case, either true-block or false-block will be executed, not both. This is illustrated in Figure 2.15 in both the cases, the control is transferred subsequently to statement-x.

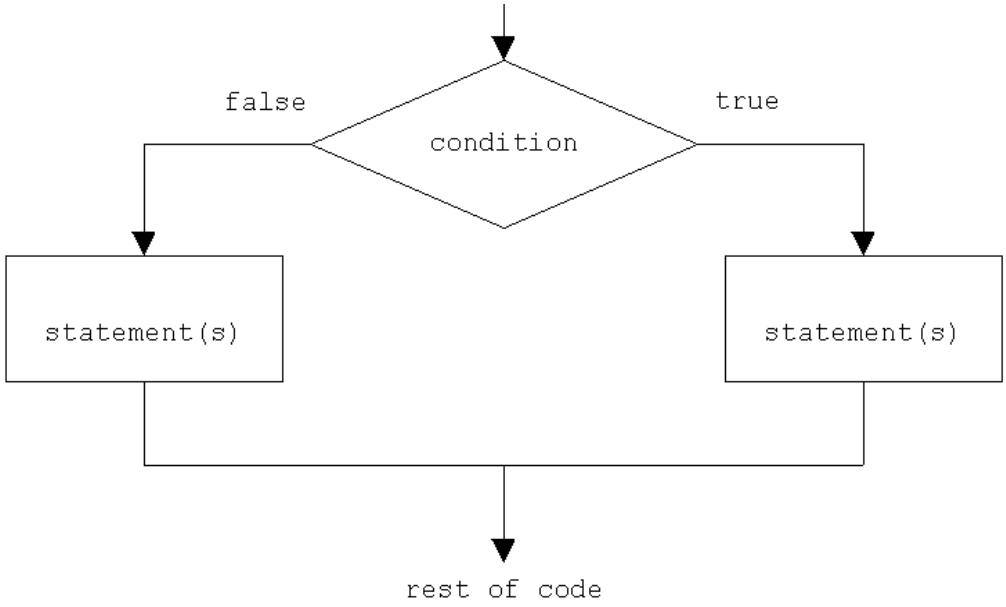


Figure 2.18: Flowchart of if else control.

Let us consider an example of counting the number of boys and girls in a class. We code 1 for a boy and 2 for a girl. The program statement to do this may be written as follows:

```
lf (code == 1;
boy=boy+1.
if (code ==2)
girl =girl +l;
```

The first test determines whether or not the student is a boy. If yes, the number of boys is increased by 1 and, the second test. The second test again determines whether the student is a girl. This is unnecessary. Once a student is identified as a boy, there is no need to test again for a girl. A student can be either a boy or a girl, not both. The program segment can be modified using the else clause as follows:

```
if (code == 1;
        boy = boy+1.
else
girl =girl +l;
xxxxxxxxxx
```

Here, if the code is equal to 1, the statement boy = boy + 1; is executed and the control is transferred to the statement xxxxxx, after skipping the else part. If the code is not equal to 1, the statement boy = boy + 1; is skipped and the statement in the else part girl = girl + 1; is executed before the control reaches the statement xxxxxx.

Consider the program given in Figure 3.16. When the value (c – d) is zero, the ratio is not calculated and the program stops without any message. In such cases we may not know whether the program stopped due to a zero value or some other error. This program can be improved by adding the else clause as follows:

```
if (c–d != 0)
{
ratio = (float)(a+b)/(float)(c–d);
```

```
printf("Ratio = %f\n", ratio);
}
else
printf("c–d is zero\n");
```

If T_{n-1} (usually known as previous term) is known, then T_n (known as present term) can be easily found by multiplying the previous term by x/n. Then
$e^x = T_o + T_1 + T_2 + ... + T_n$: sum

The program uses count to count the number of terms added. The program stops when the value of the term is less than 0.0001 (ACCURACY). Note that when a term is less than ACCURACY, the value of n is set equal to 999 (a number higher than 100) and therefore the while loop terminates. The results are printed outside the while loop.

Nesting of if... else Statements

When a series of decisions are involved, we may have to use more than one **if...else** statement in nested form as follows:

```
Program
/*************************************
/*Experiment with if else statement
/*************************************/
#define ACCURACY 0.0001
main( )
{
int n, count;
float x, term, sum;
printf("Enter value of x");
scanf("%f", &x);
n = term = sum = count = .l;
while (n <= 100)
{
Term = term*x/n;
sum=sum+term;
count=count+l;
if (term < ACCURACY)
n = 999;
else
n = n + .l;
}

printf("Terms = %d Sum = %f \n" count, sum);
}

Output
Enter value of x:0
Terms = 2 Sum = 1.000000
Enter value of x:0.1
```

```
Terms=5 Sum=1.105171
Enter value of x:0.5
Terms = 7 Sum = 1.648720
Enter value of x:0.75
Terms=8 sum:2. 116997
Enter value of x:0.99
Terms=9 Sum =2.691232
Enter value of x:1
Terms=9 Sum:2.718279
```

Figure 2.19: Illustration of if else statement.

The logic of execution is illustrated in Figure 2.17. If the condition -1 is false, the statement -3 will be executed; otherwise it continues to perform the second test. If the condition -2 is true, the statement -1 will be evaluated; otherwise the statement -2 will be evaluated and then the control is transferred to the statement -x.

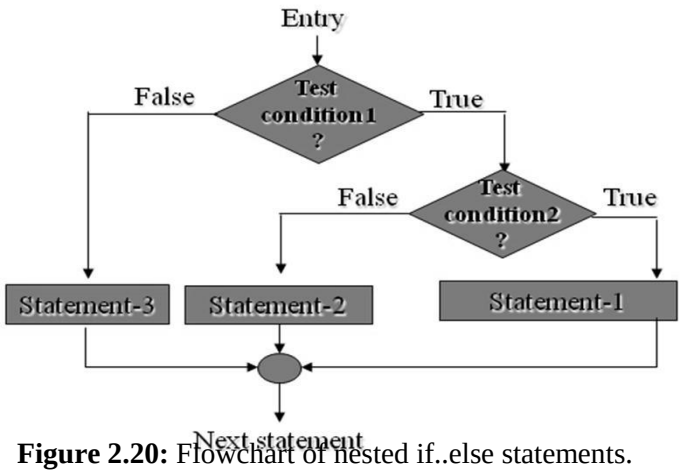


Figure 2.20: Flowchart of nested if..else statements.

A commercial bank has introduced an incentive policy of giving bonus to all its deposit holders. The policy is as follows: A bonus of 2% of the balance held on 31st December is given to everyone, irrespective of their balance, and 5% is given to female account holders if their balance is more than Rs.5000. This logic can be coded as follows:

```
if (sex is female )
{
if (balance >500)
bonus=0.05 *balance;
else
{
bonus= 0.02 * balance;
{
balance = balance +bonus
bonus:0.02–balance;
```

When nesting, care should be exercised to match every if with an else.
Consider the following alternative to the above program (which looks right at the first sight):
if (sex is female)

```
        if (balance > 5000)
bonus = 002 * balance;
else
        sbonus=0.02 *balance;
balance= balance + bonus;
```

There is an ambiguity as to over which if the else belongs to. In C, an else is linked to the closer non-terminated if. Therefore, the else is associated with the inner if and there is no else option for the outer if. This means that the computer is trying to execute the statement

balance = balance + bonus;

without really calculating the bonus for the male account holders.

Consider another alternative which also looks correct;

```
if (sex is female)
{
if (balance > 5000)
bonus = 0.05*balance;
}
else
bonus = 0.02 * balance;
balance = balance + bonus;
```

In this case, else is associated with the outer if and therefore bonus is calculated for the male account holders. However, bonus for the female account holders, whose balance is equal to or less than 5000 is not calculated because of the missing else option for the inner if.

Example: The program in Figure 2.18 selects and prints the largest of the three numbers using nested if....else statements.

Program

```
/***********************************/
/* selecting the largested of three values */
/***********************************/
main( )
{
float A, B, C;
printf("Enter three values\ n");
scanf("%f %f", &A, &B, &C);
printf("\nLargest value is");
if (A>B)
{
if (A>c)
printf("%f \n", A);
}
else
{
if (C>B)
printf("%f\n", C);
```

```
else
printf("%f \n", B);
}
}
Output
Enter three values
23445   67379   88843
Largest value is 88843.000000
```

Figure 2.21: Selecting the largest of three numbers.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Write a program to find the sum of all integers greater than 100 and less than 200 that are divisible by 8.

……..……

……..……...…

……..

2.6.2 The else…if Ladder

There is another way of putting ifs together when multipath decisions are involved. A multipath decision is a chain of ifs in which the statement associated with each else is an if. It follows general form:

This construct is known as the else if ladder. The conditions are evaluated from the top (of the ladder), downwards. As soon as a true condition is found, the statement associated with it is executed and the control is transferred to the statement-x 1skipping the rest of the ladder). When all the n conditions become false, then the final else containing the default-statement will be-executed. Figure 2.19 shows the logic of execution of else if ladder statements.

Let us consider an example of grading the students in an academic institution. The grading is done according to the following rules:

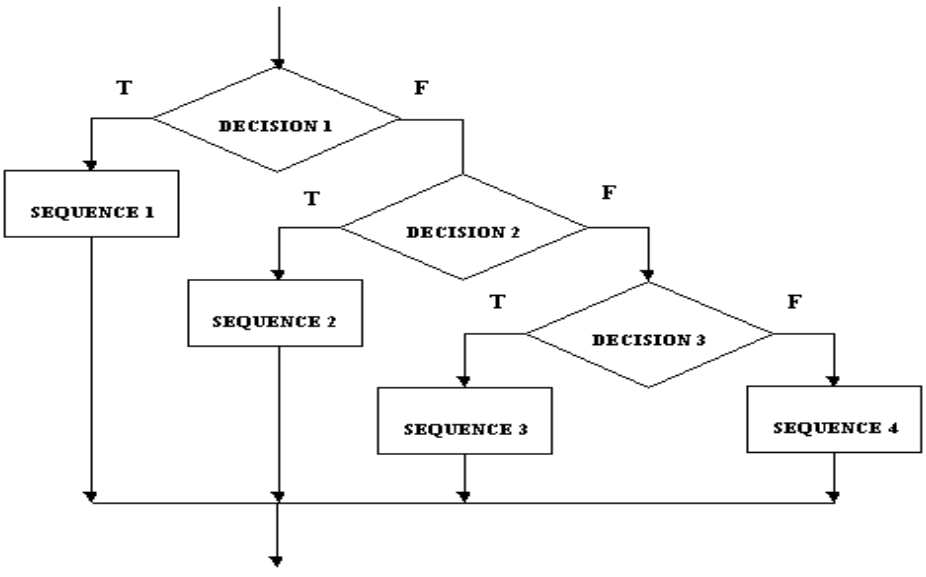


Figure 2.22: Flowchart of else...if ladder.

Average marks	Grade
80 to 100	Honours
60 to 79	First Divisions
50 to 59	Second Division
40 to 49	Third Divisions
0 to 39	Fail

This grading can be done using the else if ladder as follows:

```
if (marks > 79)
grade = „Honours’,;
else if (marks > 59)
grade = “First Division”;
else if (marks > 49)
grads = “Second Division”:
else if (marks > 39)
grade = “Third Division”;
else
grade = “Fail”;
printf (“%s\n, grade);
```

Consider another example given below:

```
if(code ==1;
colour = “RED”;
else if (code = 2)
colour = “GREEN”;
else if (code == 3)
colour = “WHITE”;
else
colour = “YELLOW”;
```

Code numbers other than 1, 2 or 3 are considered to represent YELTOW colour. The same results can be obtained by using nested if...else statements.

```
if (code !:= 1)
if (code != 2)
if (code != 3)
colour = “YELLOW”;
else
colour = “WH|TE”:
else
colour = “GREEN”;
else
colour = “RED”;
```

In such situations, the choice of the method is left to the programmer. However, in order to choose an if structure that is both effective and efficient, it is important that the programmer is fully aware of the various forms of an if statement and the rules governing their nesting.

Example: An electric Power distribution company charges its domestic consumers as follows:

Consumption Units	Rate of Charge
0–200	Rs. 0.50 per unit
201 – 400	Rs. 100 plus Rs.0.65 per unit excess of 200
401 – 600	Rs. 230 plus Rs.0.80 per unit excess of 400
601 and above	Rs. 390 plus Rs.1.00 per unit excess of 600

The program in Figure 3.19 reads the customer number and power consumed and prints the amount to be paid by the customer.

2.6.3 Nested if Statement

Using "if...else statement" within another "if...else statement" is called 'nested if statement'. "Nested if statements" is mainly used to test multiple conditions. They can be structured using following syntax:

```
if(conditionA)
      {
      Statements
      }
else if (conditionB)
      {
      statementB
      }
else
      {
      statement
      }
```

```
1
0     1
1     0     1
0     1     0     1
1     0     1     0     1
```

Self Assessment Questions

1. What is the output of the following code?

```
#include<stdio.h>
void main()
{
  int s=0;
  while(s++<10) {
    if(s<4 && s<9)
      continue;
    printf("\n%d\t",s);
  }
}
```

(a) 1 2 3 4 5 6 7 8 9
(b) 1 2 3 10
(c) 4 5 6 7 8 9 10
(d) 4 5 6 7 8 9

2. What will be output if you will execute following C code?

```
#include<stdio.h>
int main(){
   int a=0;
   #if (a==0)
      printf("Equal");
   #else if
      printf("Not equal");
   #endif
   return 0;
}
```

(a) Equal
(b) Not equal
(c) Null
(d) Compilation error

3. What will be output if you will execute following c code?

```
#include<stdio.h>
int main(){
   int x=25;
   if(!!x)
      printf("%d",!x);
   else
      printf("%d",x);
   return 0;
}
```

(a) 0
(b) 25
(c) 1
(d) –1

4. What will be output if you will execute following c code?

```
#include<stdio.h>
int main(){
   int x=5, y=10;
   if(!(!x) && x)
      printf("%d",x);
   else
      printf("%d",y);
   return 0 ;
}
```

(a) 1
(b) 0
(c) 5
(d) 10

2.7 The Switch Statement

We have seen that when one of the many alternatives is to be program using if statements to control the selection. However, the complexity of such a program increases dramatically when the number of alternatives increases the program becomes difficult to read and follow. At times, it may confuse even the person who designed it. Fortunately, the C has a built-in multiway decision statement known as a switch. The switch statement tests the value of a given variable (or expression) against a list of case values and when a match is

found. a block of statements associated with that case is executed the general form of the switch statement is as shown below:

The expression is an integer expression or characters. Value-l, Value-2 are constants or constant expressions (evaluable to an integral constant) and are known as ccs€ lables. Each of these values should be unique within a switch statement. block-l, block-2 are statement lists and may contain zero or more statements. There is no need to put braces around these blocks. Note that case labels end with a colon (:).

When the switch is executed, the value of the expression is successively compared against the values value-1, value-2,....If a case is found whose value matches with the value of the expression, then the block of statements that follows the case are executed.

Program

```
/*****************************************/
/*Use of else...if ladder                 */
/***************************************** /
main( )
{
int units, custnum;
float charges;
printf{("Enter CUSTOMEB NO. and UNITS consumed\n");
scanf("%d %d", &custnum, &units);
if (units <= 200)
charges:0.5 * units;
else if (units <= 400)
        charges = 100 + 0 65. (units - 200);
else if (units <: 600)
charges:230 + 08. (units – 400);
else
        charges = 390 + (units – 600);
printf("\n\nCustomer No: %d: Charges: %.2f \n", custnum, charges);
}
Output
Enter CUSTOMER NO. and UNITS consumed 101 .150
Customer No:101 Charges : 75.00
Enter CUSTOMER NO. and UNITS consumed 202 225
Customer No:202 Charges = 116.25
Enter CUSTOMER NO. and UNITS consumed 303 375
Customer No:303 Charges = 213.75
Enter CUSTOMER NO. and UNITS consumed 404 520
Customer No:404 Charges : 326.00
Enter CUSTOMER NO. and UNITS consumed 505 625
Customer No:505 Charges = 415.00
```

Figure 2.23: Illustration of else…if ladder.

```
Switch (expression)
{
Case value-1;
                block-1
                break;
case value-2
                block-2
                break;
default:
                default-block
                break;
}
Statement-x
```

Figure 2.24: Selection process of the switch statement.

The break statement at the end of each block signals the end of a particular case and causes an exit from the switch statement, transferring the control to the statement-x following the switch.

The default is an optional case. When present, it will be executed if the value of the expression does not match with any of the case values. If not present, no action takes place if all matches fail and the control goes to the statement-x.

The selection process of switch statement is illustrated in the flowchart shown in Figure 2.25.

The switch statement can be used to grade the students. This is illustrated below:

```
index= marks/10;
switch (index)
{
case 10:
case 9:
case 8:
        gradg = „Honours,’;
break;
case 7:
case 6:
        grade = “First Division”;
        break;
case 5:
grede = “Second Division”;
break;
case 4:
grade = “Third Division”;
break;
default:
grade = “Fail”;
break;
}
printf(“%s\n”; grade);
```

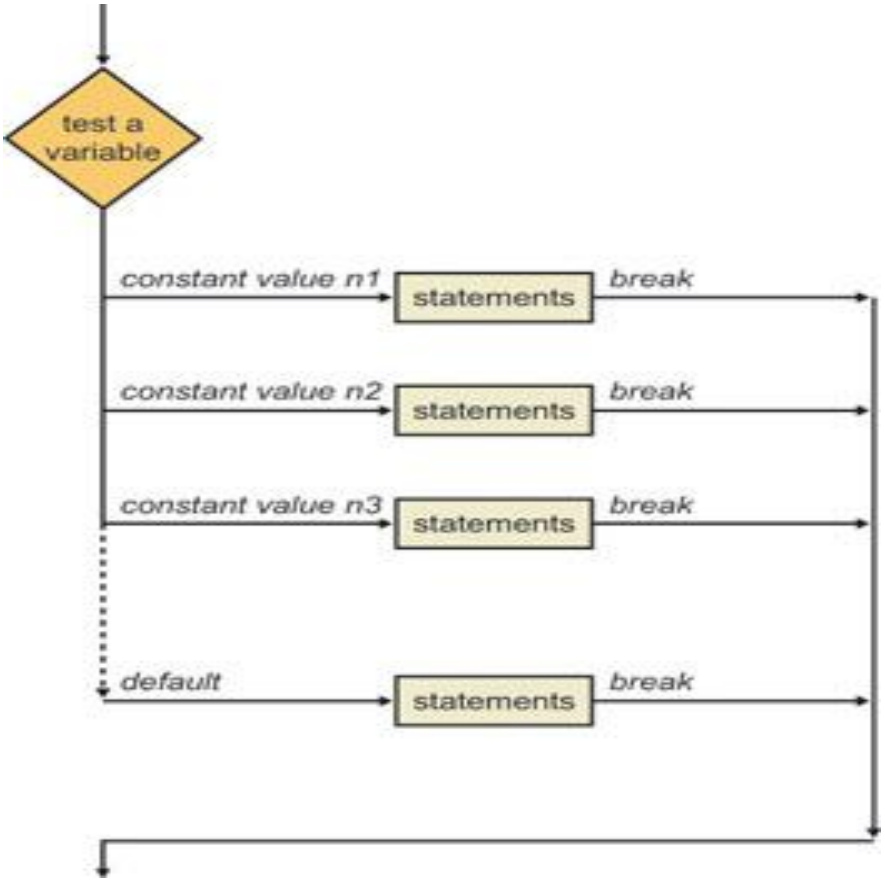


Figure 2.25: Selection process of the switch statement.

Note that we have used a conversion statement
index=marks/10;
where, index is defined as an integer. The variable index takes the following integer values.

Marks	Index
100	10
90-99	9
80-89	8
70-79	7
60–69	6
50–59	5
40–49	4
0	0

This segment of the program illustrates two important features. First, it uses empty cases. The first three cases will execute the same statements
grade = “Honours”;
break;
Same is the case with case 7 and case 6’ Second, default condition is used for all other case where marks is less than 40.
The switch statement is often used for menu selection. For example:
printf (“ TRAVEL GUIDE \n \n”);

```
printf(“ A Air Timlngs\n” );
printf(“ T Train Timings\n”);
printf(“ B Bus Service\n” );
printf(“ X To skiP\n” );
printf(“ \n Enter your choice\n”);
character = getchar( );
switch (character)
{
case ‘A’:
        air-display( );
        break:
case ‘B’ :
        bus-display( );
        break;
case ‘T’ :
        train-display( );
        break;
default :
        printf(“ No choice\n”);
}
```

It is possible to nest the switch Statements, That is, a switch may be part of a case block.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Define Branching Statement.

……..……

……..……...…

……..

Ex2: Define Switch Statement?

……..……

……..……...…

……..

2.8 Summary

- “if statement” and “switch statement” are the most popular conditional statements used in C.
- Branch is the term given to the code executed in sequence as a result of change in the program’s flow; the program’s flow can be changed by conditional statements in that program.
- Sequential statements are those statements in a program that executes one by one in a given sequence.
- Input data items must be separated by spaces, tabs or newlines punctuation marks do not count as separators.
- The function scanf is expected to read three items of data and therefore, when the values for all the three variables are read correctly, the program prints out their values.

- Computer outputs are used as information for analysing certain relationships between variables and for making decisions.
- The C language programs follow a sequential form of execution of statements.
- The if statement is powerful decision statement and used to control the flow of execution.
- The switch statement tests the value of given variables (or expression) against a list of case values and when a match is found.

2.9 Keywords

if Statement: The if statement is a powerful decision making statement and is used to control the flow of execution of statements.

Printf:The printf function is just a useful function from the standard library of functions that are accessible by C programs. The behavior of printf is defined in the ANSI standard.

Putchar : There is an analogous function putchar for writing characters one at a time to the Terminal.

Scanf: Is an input function which can read data from a terminal.

***Switch Statement*:** When the switch is executed, the value of the expression is successively compared against the values value-1, value-2 ...If a case is found whose value matches with the value of the expression, then the block of statement.

The else…if Ladder: The conditions are evaluated from the top (of the ladder), downwards. As soon as a true condition is found, the statement associated with it is executed and the control is transferred to the statement-x 1skipping the rest (of the ladder). When all the n conditions become false, then the final else containing the default-statement will be-executed.

2.10 Review Questions

1. What are the sequential statements using in C? Explain with example.
2. Explain about the unformatted input and output functions in C.
3. What are the formatted input functions in C? Explain with example.
4. What do we do to enhancing the readability of output?
5. What are the branching statements in C? Explain with example.
6. What is the difference between else...if statements and else ...if ladder?
7. What are the switch statements using in C? Explain with example.
8. Discuss the selection process of switch statement.
9. Predict the output or error(s) for the given:

(a).
```
main ()
{
int i=–1,j=–1,k=0,l=2,m;
m=i++&&j++&&k++||l++;
printf(“%d %d %d %d %d”,i,j,k,l,m);
}
```

(b).
```
main ()
{
char *p;
printf(“%d %d “,sizeof(*p),sizeof(p));
}
```

(c).
```
main ()
{
```

```
 int i=3;
switch(i)
{
  default:printf("zero");
   case 1: printf("one");
   break;
  case 2:printf("two");
  break;
  case 3: printf("three");
  break;
 }
}
```

10. Differentiate between formatted and unformatted input and output functions.

Answers for Self Assessment Questions

1. (c) 2. (d) 3. (c) 4. (c)

2.11 Further Studies

1. http://www.manipalitdubai.com/material/Lecture_Notes/ISM105/Decision_Making.pdf
2. Programing with c by Bala gura swami.

3

Looping Statements

STRUCTURE

3.0 Objectives

3.1 Introduction

3.2 Control Statement

3.3 For Statement

3.4 The Do Statement

3.5 The While Statement

3.6 Difference between While-loop and Do-while Loop

3.7 The Jumping Statement

3.8 Control Transfer Statement

3.9 Summary

3.10 Keywords

3.11 Review Questions

3.12 Furhter Studies

3.0 Objectives

After studying this chapter, you will be able to:

- Differentiate between while-loop and do-while loop
- Discuss the control statement
- Explain the while statement
- Explain the for statement
- Discuss the nesting of for loops
- Explain the do statement
- Define the while statement
- Define the?:operator
- Discuss the jumping statement
- Discuss the control transfer statement

3.1 Introduction

We have seen that a C program is a set of statements which are normally executed sequentially in the order in which they appear. This happens when no options or no repetitions of certain calculations are necessary.

However, in practice, we have a number of situations where we may have to change the order of execution of statements based on certain conditions, or repeat a group of statements until certain specified conditions are met. This involves a .kind of decision making to see whether a particular condition has occurred or not and then direct the computer to execute certain statements accordingly.

3.2 Control Statement

We have seen in the previous chapter that it is possible to execute a segment of a program repeatedly by introducing a counter and later testing it using the if statement. While this method is quite satisfactory for all practical purposes, we need to initialize and increment a counter and test its value at an appropriate place in the program for the completion of the loop. For example, suppose we want to calculate the sum of squares of all integers between 1 and 10. We can write a program using the if statement as follows:

```
sum = 0
n=1;
loop;
sum= sum+n*n;
if (n = = 10)                    n=10,
goto print;          end of loop
else
{
n=n+1;
goto loop;
}print;
```

This program does the following things:

1. Initializes the variable n.
2. Computes the square of n and adds it to sum.
3. Tests the value of n to see whether it is equal to 10 or not.

If it is equal to 10, then the program prints the results.
again.
The program evaluates the statement
sum = sum + n * n;

The 10 times. That is, the loop is executed 10 times. This number can be decreased or increased easily by modifying the relational expression appropriately in the statement if (n == 10). On such occasions where the exact number of repetitions is known, there are more convenient methods of looping in C. These looping capabilities enable us to develop concise programs containing repetitive processes without the use of goto statements.

In looping, a sequence of statements is executed until some conditions for termination of the loop are satisfied. A program loop therefore consists of two segments, one known as the body of the loop and the other known as the control statement. The control statement tests certain conditions, and then directs the repeated execution of the statements contained in the body of the loop.

Depending on the position of .the control statement in the loop, a control structure may be classified either as the entry-controlled loop or as the exit-controlled loop. The flowcharts in Figure 4.1 illustrate these structures. In the entry-controlled loop, the control conditions are tested before the start of the loop execution. If the conditions are not satisfied, then the body of the loop will not be executed. In the case of an exit-controlled loop, the test is performed at the end of the body of the loop and therefore the body is executed unconditionally for the first time.

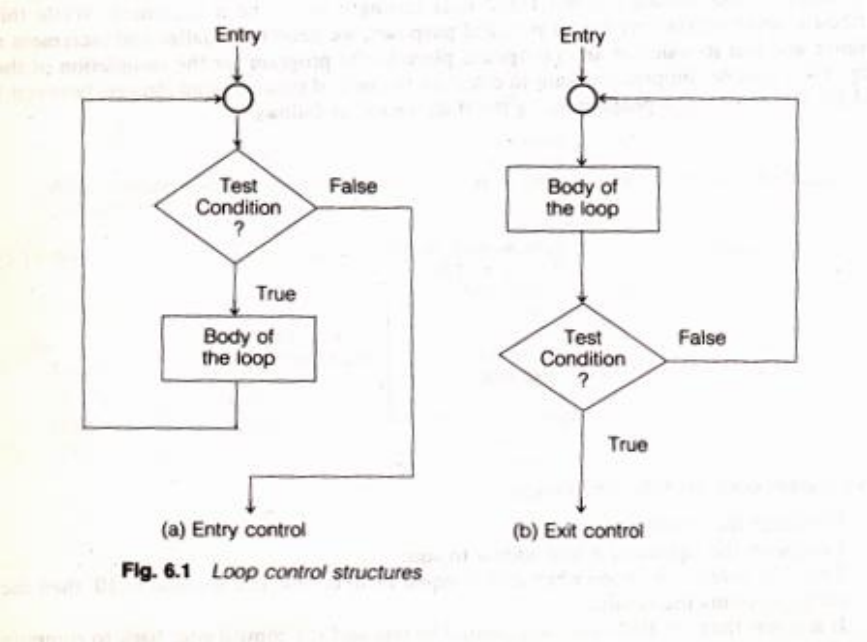


Fig. 6.1 *Loop control structures*

Figure 3.1: Loop control structure.

The test conditions should be carefully stated in order to perform the desired number of loop executions. It is assumed that the test condition will eventually transfer the control out of the loop. In case, due to some reason it does not do so, the control sets up an infinite loop and the body is executed over and over again.

A looping process, in general, would include the following four steps:
1. Setting and initialization of a counter.
2. Execution of the statements in the loop.
3. Test for a specified condition for execution of the loop.
4. Incrementing the counter.

The test may be either to determine whether the loop has been repeated the specified number of times or to determine whether a particular condition has been met. The C language provides for three loop constructs for performing loop operations. They are:
1. The while statement.
2. The do statement.
3. The for statement.

3.3 For Statement

The for loop is another entry-controlled loop that provides a more concise loop control structure. The general form of the for loop is

```
for (initialization ; test-condition ; increment)
{
body of the loop
}
```

```
/* sample program using a for statement */
        #include <stdio.h>
        main()
        {
                int  n, t_number;
                t_number = 0;
                for( n = 1; n <= 200; n = n + 1 )
                t_number = t_number + n;
                printf("The 200th triangular_number is %d\n", t_number);
        }
        Output
         The 200th triangular_number is 20100
```

Figure 3.2: sample program using a for statement.

The execution of the for statement is as follows:

1. Initialization of the control variables is done first, using assignment statements such as i = 1 and count = 0. The variables i and count are known as loop-control variables.

2. The value of the control variable is tested using the test-condition. The test-condition is a relational expression, such as i < 10 that determines when the loop will exit. If the condition is true, the body of the loop is executed; otherwise the loop is terminated and the execution continues with the statement that immediately follows the loop.

3. When the body of the loop is executed, the control is transferred back to the for statement after evaluating the last statement in the loop. Now, the control variable is incremented using an assignment statement such as i = i+1 and the new value of the control variable is again tested to see whether it satisfies the loop condition. If the condition is satisfied, the body of the loop is again executed. This process continues till the value of the control variable fails to satisfy the test-condition. Consider the following segment of a program:

```
for (x=0;x<=9;x=x+1)
{
printf("%d", x);
}
printf(" \n");
```

This for loop is executed 10 times and prints the digits 0 to 9 in one line. The three sections enclosed within parentheses must be separated by semicolons. Note that there is no semicolon at the end of the increment section, x = x+1.

The for statement allows for negative increments. For example, the loop discussed above can be written as follows:

```
for (x = 9;x >= O;x = x–1)
printf1("%d", x);
printf("\n");
```

This loop is also executed 10 times, but the output would be from 9 to 0 instead of 0 to 9. Note that braces are optional when the body of the loop contains only one statement.

Since the conditional test is always performed at the beginning of the loop, the body of the loop may not be executed at all, if the condition fails at the start. For example,

```
for (x = 9; x < 9; x = x– 1)
printf(" x");
```

will never be executed because the test condition fails at the very beginning itself.

Let us again consider the problem of sum of squares of integers discussed in section. This problem can be coded using the for statement as follows:

```
sum = 0;
for (n = 1; n <= 10; n = n+1) {
sum = sum+ n*n;
}
printf("sum = %d\n", sum);
```

The body of the loop

```
sum = sum + n*n;
```

is executed 10 times for n = 1,2, 10 each time incrementing the sum by the square of the value of n.

One of the important points about the for loop is that all the three actions, namely initialization, testing, and incrementing, are placed in the for statement itself, thus making them visible to the programmers and users, in one place.

Example:

The program in Figure 4.3 uses a for loop to print the "Powers of 2" table for the power 0 to 20, both positive and negative.

The program evaluates the value

P=2"

We have declared p as a long int and q as a double. Additional Features of for Loop, the for loop in C has several capabilities that are not found in other loop constructs. For example, more than one variable can be initialized at a time in the for statement. The statements

```
P=1;
for (n=0; n<17; ++n)
```

can be rewritten as

```
for (p=1, n=o; n<17; ++n)
```

```
main()
  {
     longint p;
     int      n;
     double   q;
     printf("-----------------------------------------\n");
     printf(" 2 to power n       n       2 to power -n\n");
     printf("-----------------------------------------\n");
     p = 1;
     for (n = 0; n < 21 ; ++n)    /* LOOP BEGINS */
     {
        if (n == 0)
          p = 1;
        else
          p = p * 2;
        q = 1.0/(double)p ;
        printf("%10ld %10d %20.12lf\n", p, n, q);
     }                          /* LOOP ENDS   */
     printf("-----------------------------------------\n");
  }
```

Output

```
------------------------------------------------
2 to powern     n               2 to power -n
------------------------------------------------
1               0               1.000000000000
2               1               0.500000000000
4               2               0.250000000000
8               3               0.125000000000
16              4               0.062500000000
32              5               0.031250000000
64              6               0.015625000000
128             7               0.007812500000
256             8               0.003906250000
512             9               0.001953125000
1024            10              0.000976562500
2048            11              0.000488281250
4096            12              0.000244140625
8192            13              0.000122070313
16384           14              0.000061035156
32768           15              0.000030517578
65536           16              0.000015258789
131072          17              0.000007629395
262144          18              0.000003814697
524288          19              0.000001907349
1048576         20              0.000000953674
```

Figure 3.3: Program to print 'Power of 2' table using for loop.

Notice that the initialization section has two parts p = 1 and n = 1 separated by a comma. Like the initialization section, the increment section may also have more than one part. For example, the loop

```
for (n=1, m=50; n<=m; n=n+1, m=m-1) {
p = m/n;
printf("%d %d %d\n", n, m, p); }
```

is perfectly valid. The multiple arguments in the increment section are separated by commas.

The third feature is that the test-condition may have any compound relation and the testing need not be limited only to the loop control variable. Consider the example below:

```
sum = 0;
for (i = 1; i < 20 && sum < 100; ++i)
{
sum = sum+i; printf("%d %d\n", sum);
}
```

The loop uses a compound test condition with the control variable i and external variable sum. The loop is executed as long as both the conditions i < 20 and sum < 100 are true. The sum is evaluated inside the loop.

It is also permissible to use expressions in the assignment statements of initialization and increment sections. For example, a statement of the type

```
for (x = (m+ny2; x > 0; x = x/2)
```

is perfectly valid.

Another unique aspect of for loop is that one or more sections can be omitted, if necessary. Consider the following statements:

```
m = 5;
for(;m != 100;) {
printf("%d\n", m);
m = m+5;
}
```

Both the initialization and increment sections are omitted in the for statement. The initialization has been done before the for statement and the control variable is incremented inside the loop. In such cases, the sections are left blank. However, the semicolons separating the sections must remain. If the test-condition is not present, the for statement sets up an infinite loop. Such loops can be broken using break or goto statements in the loop.
We can set up time delay loops using the null statement as follows:

```
for ( j = 1000; j > 0; j = j-1)
```

This loop is executed 1000 times without producing any output; it simply causes a time delay. Notice that the body of thee loop contains only a semicolon, known as a null statement. This can also be written as

```
for U=1000; j > 0; j = j-1);
```

This implies that the C compiler will not give an error message if we place a semicolon by mistake at the end of a for statement. The semicolon will be considered as a null statement and the program may produce some nonsense.

3.3.1 Nesting of for Loops

Nesting of loops, that is, one for statement within another for statement is allowed in C. For example, two loops can be nested as follows:

```
for(i = 1; i < 10; ++i) // Outer loop
{
for (j = 1; j != 5; ++j) // Inner loop
{
}
```

The nesting may continue upto 15 levels in ANSI C; many compilers allow more. The loops should be properly indented so as to enable the reader to easily determine which statements are contained within each for statement.
The program to print the multiplication table discussed in Example can be written more concisely using nested for statements as follows:

```
for (row = 1; row <= ROWMAX ; ++row)
{
for (column = 1; column < = COLMAX ; ++column) {
y = row column;
printf("%4d", y);
}
printf("fin");
}
```

The outer loop controls the rows while the inner loop controls the columns.
Example: A class of n students takes an annual examination in m subjects. A program to read the marks obtained by each student in various subjects and to compute and print the total marks obtained by each of them is given in Figure.3.4

```
#define FIRST   360
  #define SECOND  240
  main()
  {
     int n, m, i, j, roll_number, marks, total;
     printf("Enter number of students and subjects\n");
     scanf("%d %d", &n, &m);
     printf("\n");
     for (i = 1; i <= n ; ++i)
     {
        printf("Enter roll_number :");
        scanf("%d", &roll_number);
        total = 0 ;
printf("\nEnter marks of %d subjects for ROLL NO %d\n", m,roll_number);
        for (j = 1; j <= m; j++)
        {
           scanf("%d", &marks);
           total = total + marks;
        }
        printf("TOTAL MARKS = %d", total);
        if (total >= FIRST)
       printf("( First Division )\n\n");
        else if (total >= SECOND)
       printf("( Second Division )\n\n");
            else
        printf("( ***  F A I L  *** )\n\n");
     }
  }
Output
Enter number of students and subjects
      3    6
  Enter roll_number : 8701
  Enter marks of 6 subjects for ROLL NO 8701
  81  75  83  45  61  59
  TOTAL MARKS = 404 ( First Division )
 Enter roll_number : 8702
  Enter marks of 6 subjects for ROLL NO 8702
  51  49  55  47  65  41
  TOTAL MARKS = 308 ( Second Division )
  Enter roll_number : 8704
  Enter marks of 6 subjects for ROLL NO 8704
  40  19  31  47  39  25
  TOTAL MARKS = 201 ( ***  F A I L  *** )
```

Figure 3.4: Illustration of nested for loops.

The program uses two for loops, one for controlling the number of students and the other for controlling the number of subjects. Since both the number of students and the number of subjects are requested by the program, the program may be used for a class of any size and any number of subjects. The outer loop includes three parts: (1) reading of roll-numbers of students, one after another, (2) inner loop, where the marks are read and totaled for each student, and (3) printing of total marks and declaration of grades.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Write a program for nested for loop.

……..……

……..……..…

……..

3.4 The Do Statement

The body of the loop may not be executed at all if the condition is not satisfied at the very first attempt. On some occasions it might be necessary to execute the body of the loop before the test is performed. Such situations can be handled with the help of the do statement. This takes the form:

```
do {
body of the loop
}
while (test-condition);
```

On reaching the do statement, the program proceeds to evaluate the body of the loop first. At the end of the loop, the *test-condition* in the while statement is evaluated. If the condition is true, the program continues to evaluate the body of the *loop* once again. This process continues as long as *the condition* is true. When the condition becomes false, the loop will be terminated and the control goes to the statement that appears immediately after the while statement.

```
main()
{
int count, n;
float x, y;
printf("Enter the values of x and n : ");
scanf("%f %d", &x, &n);
y = 1.0;
count = 1;              /* Initialisation */
/* LOOP BEGINs */
while ( count <= n)     /* Testing */
{
y = y*x;
count++;          /* Incrementing */
}
/* END OF LOOP */
printf("\nx = %f; n = %d; x to power n = %f\n",x,n,y);
```

```
}

Output
Enter the values of x and n : 2.5  4
x = 2.500000; n = 4; x to power n = 39.062500
Enter the values of x and n : 0.5  4
x = 0.500000; n = 4; x to power n = 0.062500
```

Figure 3.5: Program to compute x to the power n using while loop.

Since the test-condition is evaluated at the bottom of the loop, the do. .while construct provides an exit-controlled loop and therefore the body of the loop is always executed at least once.
A simple example of a do.. .while loop is:

```
do {
printf("Input a number\n");
number = getnum( );
}
while(number>0);
```

This segment of a program reads a number from the keyboard until a zero
or a negative.
The test conditions may have compound relations as well. For instance, the statement while (number > 0 && number < 100);
in the above example would cause the loop to be executed as long as the number keyed in lies between 0 and 100.
Consider another example:

```
l=1;
sum=0;
do {
sum=sum+1;
l=l+2;
}While(sum<40!! l<10)
printf("%d%d\n", l, sum);
```

The loop will be executed as long as one of the two relations is true.

Caution
In the running of program, control the loop otherwise program will not be terminated.

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: Define For Statement.

……..……

……..……...…

……..

Ex2: Define Do Statement?

……..……

……..……...….

……..

3.5 The While Statement

The simplest of all the looping structures in C is the while statement. We have used while in many of our earlier programs. The basic format of the while statement is

```
while (test condition) {
body of the loop
}
```

The while is an *entry-controlled loop* statement. The *test-condition is* evaluated and if the condition is true, then the body of the loop is executed. After execution of the body, the test condition is once again evaluated and if it is true, the body is executed once again. This process of repeated execution of the body continues until the test-condition finally becomes *false* and the control is transferred out of the loop. On exit, the program continues with the statement immediately after the body of the loop. The body of the loop may have one or more statements. The braces are needed only if the body contains two or more statements. However, it is a good practice to use braces even if the body has only one statement.

```
sum = 0; n=1;
while (n <= 10) {
SUM= SUM + n ‘ n;
n = n+1;
}
printf(“sum = %d\n”, sum);
```

The body of the loop is executed 10 times for n = 1, 2, ... 10 each time adding the square of the value of n, which is incremented inside the loop. The test condition may also be written as n < 11; the result would be the same.

Another example of while statement which uses the keyboard input is shown below:

```
character = ‘ ’
while (character != ‘Y’)
character = getchar();
```

First the character is initialized to ‘ ’. The while statement then begins by testing whether character is not equal to Y. Since the character was initialized to ‘ ’, the test is true and the loop statement

```
character = getchar();
xxxxxxx;.
```

is executed. Each time a letter is keyed in, the test is carried out and the loop statement is executed until the letter Y is pressed. When Y is pressed, the condition becomes false because character equals Y, and the loop terminates, thus transferring the control to the statement xxxxxxx;.

Example:

A program to evaluate the equation

$y = x^n$
when n is a non-negative integer, is given in Figure 4.9.
The variable y is initialized to 1 and then multiplied by x, n times using the while loop. The loop control variable, count is initialized outside the loop and incremented inside the loop. When the value of count becomes greater than n, the control exits the loop.

3.6 Difference between While-loop and Do-while Loop

The difference between a "do ...while" loop and a "while {} " loop is that the while loop tests its condition before execution of the contents of the loop begins; the "do" loop tests its condition after it is been executed at least once. As noted above, if the test condition is false as the while loop is entered the block of code is never executed. Since the condition is tested at the bottom of a do loop, its block of code is always executed at least once.
To further clear your concept on this, understand the syntax and description of the two loop types:
while
The while loop is used to execute a block of code as long as some condition is true. If the condition is false from the start the block of code is not executed at al. The while loop tests the condition before it is executed so sometimes the loop may never be executed if initially the condition is not met. Its syntax is as follows.

```
while (tested condition is satisfied)
{
block of code
}
```

In all constructs, curly braces should only be used if the construct is to execute more than one line of code. The above program executes only one line of code so it not really necessary (same rules apply to if...else constructs) but you can use it to make the program seem more understandable or readable.
Here is a simple example of the use of the while loop. This program counts from 1 to 100.

```
#include <stdio.h>
int main(void)
{
int count = 1;
while (count <= 100)
{
printf("%d\n",count);
count += 1; // Notice this statement
}
return 0;
}
```

Note that no semi-colons (;) are to be used after the while (condition) statement. These loops are very useful because the condition is tested before execution begins. However it never seem to like these loops as they are not as clear to read as the do ...while loops. The while loop is the favorite amongst most programmers but as for us, it definitely prefer the do ...while loop.
do ....while
The do loop also executes a block of code as long as a condition is satisfied.
Again, The difference between a "do ...while" loop and a "while {}" loop is that the while loop tests its condition before execution of the contents of the loop begins; the "do" loop tests its condition after it is been executed at least once. As noted above, if the test condition is false as the while loop is entered the block of

code is never executed. Since the condition is tested at the bottom of a do loop, its block of code is always executed at least once.
some people do not like these loops because it is always executed at least once. Basically because the loop is always executed, it will execute no matter what value or type of data is supposed to be required. The “do ....while” loops syntax is as follows

```
do
{
block of code
} while (condition is satisfied);
```

Note that a semi-colon (;) must be used at the end of the do ...while loop. This semi-colon is needed because it instructs whether the while (condition) statement is the beginning of a while loop or the end of a do ...while loop. Here is an example of the use of a do loop.

```
#include <stdio.h>
int main(void)
{
int value, r_digit; printf(“Enter a number to be reversed.\n”);
scanf(“%d”, &value); do
{
r_digit = value % 10;
printf(“%d”, r_digit);
value = value / 10;
} while (value != 0);
printf(“\n”);
return 0;
}
```

Self Assessment Questions

1. What will be the output of the program?

```
int main()
{
   int k=1;
   printf(“%d == 1 is” “%s\n”, k, k==1?”TRUE”:”FALSE”);
```

(a). k == 1 is TRUE
(b). 1 == 1 is FALSE
(c).1 == 1 is TRUE
(d) K == 1 is FALSE return 0;

2. To print out a and b given below, which of the following printf() statement will you use?

```
#include<stdio.h>
float a=3.14;
double b=3.14;
```

(a) printf(“%f %lf”, a, b);
(b) printf(“%Lf %f”, a, b);
(c) printf(“%Lf %Lf”, a, b);
(d) printf(“%f %Lf”, a, b);

3. The for loop is another *entry-controlled loop* that provides a more concise loop control structure.
(a) True.
(b) False

4. What will be the output?

```
main(){
int i;
for(i=1;i<=10;i++){
printf("Number is=%d",i);}
```

(a). error (b). 1,2,3,4,5,6,7,8,9,10
(c). 1,2,3,4,5,6,7,8,9 (d). none

5. What will be output of following c code?

```
#include<stdio.h>
extern int x;
int main(){
   do{
      do{
           printf("%d",x);
         }
     while(!–2);
   }
   while(0);
   return 0;
}
int x=8;
```

(a) 8 (b) 10
(c) 11 (d) None of these.

3.7 The Jumping Statement

So far we have discussed ways of controlling the flow of execution based on certain specified conditions. Like many other languages, the C supports the goto statement to branch unconditionally from one point to another in the program. The goto requires a label in order to identify the place where the branch is to be made. A label is any valid variable name, and must be followed by a colon. The label is a placed immediately before the statement where the control is to be transferred. The general form of the label: can be anywhere in the program either before or after the goto label; statement.

```
goto label;



label:
statement
```

Backward jump

```
label:
statement;



goto label;
```

Forward jump

During running of a program when a statement like goto begin; is met, the flow of control will jump to the statement immediately following the label begin: This happens unconditionally.

Note that a goto breaks the normal sequential execution of the program. If the label is before the statement goto label; a loop will be formed and some statement will be executed repeatedly. Such a jump is known as backward jump. On the other hand, if the label: is placed after the goto label; some statements will be skipped

and the jump is known as a forward jump.a goto is often used at the end of a program to direct the control to the input statement, to read further data. Consider the following example:

```
main( )
{
double x, y;
        read:
        scanf(“%f”, &x)
        if (x<o)goto read;
        y=sqrt(x);
        printf(%f%f\n”, x, y);
        goto read;
}
```

This program is written to evaluate the square root of a series of numbers read from the terminal’ The program uses two goto statements, one at the end, after printing the results to transfer the control back to the input statement and the other to skip any furthi.o*f.riu,ion when the number is negative. Due to the unconditional goto statement at the end, the control is always transferred back to the input statement. In fact, this program puts the computer in a permanent loop known as an infinite loop The computer goes round and round until we take some special steps to terminate the loop. Such infinite loops should be avoided. example illustrates how such infinite loops can be eliminated.

Example: Program presented in Figure. 3.6 illustrates the use of the goto statement.

The program evaluates one square root for five numbers. The variable count keeps the count of numbers read. When count is less than or equal to 5, goto read; directs the control to the label read; otherwise, the program prints a message and stops.

```
Program
/********************************** /
/*Use of goto Statements
#include <math.h>
main( )
{
double x, y;
int count;
count = 1;
print{(“Enter FIVE real values in a LINE \n”);
read:
scanf(“%if”, &x);
printf(“ \n”);
if(x<0)
printf(“ltem - %dis negative\n”,count);
else
{
y = sqrt(x);
printf(“%d\t %lf \n”, x, Y);
}
Count = count+1;
if (count <= 5)
goto read;
```

```
printf(" \nEnd of computation");
}
Output
Enter FIVE real values in a LINE
50.70 40        36 75 11.25
50.750000       7.1 23903
40.000000       6.324555
Item -3 is negative
75.000000               8.660254
11.250000               3.354102
End of computation
```

Figure 3.6: Use of the goto statement.

Another use of the goto statement is to transfer the control out of a loop (or nested loops) when certain peculiar conditions are encountered.

Example:

```
While (     )
{
for(      )
{
If(    ) goto end_of_program;
}
}
End_of_program:
```

Jumping out of loops

We should try to avoid using goto as far as possible. But there is nothing wrong, if we use it to enhance the readability of the program or to improve the execution speed.

Did You Know?

Use of the goto statement violates the rules of structured programming.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Define While Statement.

……..……

……..……...…

……..

Ex2: Define Jumping Statement.

……..……

……..……...…

……..

3.8 Control Transfer Statement

Loops perform a set of operations repeatedly until the control variable fails to satisfy the test-condition. The number of times a loop is repeated is decided in advance and the test condition is written to achieve this. Sometimes, when executing a loop it becomes desirable to skip a part of the loop or to leave the loop as soon as a certain condition occurs. For example, consider the case of searching for a particular name in a list containing, say, 100 names. A program loop written for reading and testing the names a 100 times must be terminated as soon as the desired name is found. The C permits a jump from one statement to another within a loop as well as a jump out of a loop.

3.8.1 Break Statement

An early exit from a loop can be accomplished by using the break statement or the goto statement. We have already seen the use of the break in the switch statement and the goto in the if...else construct. These statements can also be used within while, do, or for loops. They are illustrated in Figure 4.7 and 4.8.

When the break statement is encountered inside a loop, the loop is immediately exited and the program continues with the statement immediately following the loop. When the loops are nested, the break would only exit from the loop containing it. That is, the break will exit only a single loop.

Since a goto statement can transfer the control to any place in a program, it is useful to provide branching within a loop. Another important use of goto is to exit from deeply nested loops when an error occurs. A simple break statement would not work here.

Example:

The program illustrates the use of the break statement in a C program.

The program reads a list of positive values and calculates their average. The for loop is written to read 1000 values. However, if we want the program to calculate the average of any set of values less than 1000, then we must enter a negative number after the last value in the list, to mark the end of input.

Each value, when it is read, is tested to see whether it is a positive number or not. If it is positive, the value is added to the sum; otherwise, the loop terminates. On exit, the average of the values read is calculated and the results are printed out.

```
main()
  {
     int m;
     float x, sum, average;
     printf("This program computes the average of a set of numbers\n");
     printf("Enter values one after another\n");
     printf("Enter a NEGATIVE number at the end.\n\n");
     sum = 0;
     for (m = 1 ; m < = 1000 ; ++m)
     {
        scanf("%f", &x);
        if (x < 0)
          break;
        sum += x;
     }
     average = sum/(float)(m–1);
     printf("\n");
     printf("Number of values = %d\n", m–1);
```

```
    printf("Sum          = %f\n", sum);
    printf("Average      = %f\n", average);
  }
Output
  This program computes the average of a set of numbers
  Enter values one after another
  Enter a NEGATIVE number at the end.

  21  23  24  22  26  22  –1

  Number of values = 6
  Sum          = 138.000000
  Average      = 23.000000
```

Figure 3.7: Use of break in a program.

Example: A program to evaluate the series
For –1 < x < 1 to evaluate to 0.01% accuracy is given in Figure 3.9 The goto statement is used to exit the loop on achieving the desired accuracy.

We have used the for statement to perform the repeated addition of each of the terms in the series. Since it is an infinite series, the evaluation of the function is terminated when the term x" reaches the desired accuracy. The value of n that decides the number of loop operations is not known and therefore we have decided arbitrarily a value of 100, which may or may not result in the desired level of accuracy.

```
#define  LOOP     10
#define  ACCURACY  0.0001
main()
{
int n;
float x, term, sum
printf("Input value of x : ");
scanf("%f", &x);
sum = 0;
for (term = 1, n = 1 ; n < = LOOP ; ++n)
{
sum += term ;
if (term < = ACCURACY)
goto output; EXIT FROM THE LOOP
term *= x ;
    }
    printf("\nFINAL VALUE OF N IS NOT SUFFICIENT\n");
    printf("TO ACHIEVE DESIRED ACCURACY\n");
    goto end;
    output:
    printf("\nEXIT FROM LOOP\n");
    printf("Sum = %f;  No.of terms = %d\n", sum, n);
```

```
    end:
    ;       /* Null Statement*/
}
Output
Input value of x : .21
EXIT FROM LOOP
Sum = 1.265800;  No.of terms = 7
Input value of x : .75
EXIT FROM LOOP
Sum = 3.999774;  No.of terms = 34
Input value of x : .99
FINAL VALUE OF N IS NOT SUFFICIENT
TO ACHIEVE DESIRED ACCURACY
```

Figure 3.8: Use of goto to exit from a loop.

The test of accuracy is made using an if statement and the goto statement exits the loop as soon as the accuracy condition is satisfied. If the number of loop repetitions is not large enough to produce the desired accuracy, the program prints an appropriate message.

Break statement is not very convenient to use here. Both the normal exit and the break exit will transfer the control to the same statement that appears next to the loop. But, in the present problem, the normal exit prints the message.

"FINAL VALUE OF N IS NOT SUFFICIENT TO ACHIEVE DESIRED ACCURACY"

And the forced exit prints the results of evaluation. Notice the use of a null statement at the end. This is necessary because a program should not end with a label.

Did You Know?

The "break" is almost always the desired path, but not the default behavior of the switch/case construct (at least in C and Java).

3.8.2 Continue Statement

During the loop operations, it may be necessary to skip a part of the body of the loop under certain conditions. For example, in processing of applications for some job, we might like to exclude the processing of data of applicants belonging to a certain category. On reading the category code of an applicant, a test is made to see whether his application should be considered or not. If it is not to be considered, the part of the program loop that processes the application details is skipped and the execution continues with the next loop operation.

Like the break statement, the C supports another similar statement called the continue statement. However, unlike the break which causes the loop to be terminated, the continue, as the name implies, causes the loop to be continued with the next iteration after skipping any statements in between. The continue statement tells the compiler, "SKIP THE FOLLOWING STATEMENTS AND CONTINUE WITH THE NEXT ITERATION". The format of the continue statement is simply.

Continue

The use of the continue statement in loops is illustrated in Figure 4.10. In while and do loops, continue causes the control to go directly to the test-condition and then to continue the iteration process. In the case of for loop, the increment section of the loop is executed before the test-condition is evaluated.

Example: The program illustrates the use of continue statement.
1'hr program evaluates tile square root of a series of numbers and prints the results. The process stops when the number 9999 is typed in.

In case, the series contains any negative numbers, the process of evaluation of square root should. be bypassed for such numbers because the square root of a negative number is not defined. The continue statement is used to achieve this. The program also prints a message saying that the number is negative and keeps an account of negative numbers. Final output includes the number of positive values evaluated and the number of negative items encountered.

```
#include <math.h>
  main()
 {
    int count, negative;
    double number, sqroot;
     printf("Enter 9999 to STOP\n");
    count = 0 ;
    negative = 0 ;
    while (count < = 100)
    {
       printf("Enter a number : ");
       scanf("%lf", &number);
       if (number == 9999)
       break;     /* EXIT FROM THE LOOP
       if (number < 0)
       {
         printf("Number is negative\n\n");
         negative++ ;
         continue;  /* SKIP REST OF THE LOOP*/
       }
       sqroot = sqrt(number);
       printf("Number      = %lf\n Square root = %lf\n\n", number, sqroot);
       count++ ;
    }
    printf("Number of items done = %d\n", count);
    printf("\n\nNegative items       = %d\n", negative);
    printf("END OF DATA\n");
 }
 Output
 Enter 9999 to STOP
 Enter a number : 25.0
 Number      = 25.000000
 Square root = 5.000000

 Enter a number : 40.5
 Number      = 40.500000
 Square root = 6.363961
```

```
Enter a number : –9
Number is negative
Enter a number : 16
Number      = 16.000000
Square root = 4.000000
Enter a number : –14.75
Number is negative
Enter a number : 80
Number      = 80.000000
Square root = 8.944272
Enter a number : 9999
Number of items done = 4
Negative items      = 2
END OF DATA
```

Figure 3.9: Use of continue statement.

3.9 Summary

- The for loop is entry controlled loop that provides a more concise structure.
- The simplest of all the looping structure in c is the while statement.
- When the break statement is encountered inside a loop, the loop is immediately exited and the program continues with the statement immediately.
- The continue as name implies causes the loop to be continued with the next iteration after skipping any statement in between.
- The goto requires a label in order to identify the place where the branch is to be made.

3.10 Keywords

***continue statement*:** During the loop operations, it may be necessary to skip a part of the body of the loop under certain conditions. For example, in processing of applications for some job, we might like to exclude the processing of data of applicants belonging to a certain category.

if else statement: The basic operation of if else statement is that a statement or group of statements is executed under if.

goto statement: goto statement in highly structured language like C, there may be occasions when the use of goto might be desirable.

***if else*:** The for loop is another *entry-controlled loop* that provides a more concise loop control structure.

***while loop*:** is used to execute a block of code as long as some condition is true.

3.11 Review Questions

1. Write a program to find the number of and sum of all integers greater than 100 and less than 200 that are divisible by 7?
2. What is the difference between a logical expression and a Boolean expression?
3. Explain: When the break statement is encountered inside a loop program continues with the statement immediately.
4. Write a program to multiplication using do-while loop?

5. How many types of operator? Explain it.
6. Draw the figure of Loop control structure?
7. Differentiate between while-loop and do-while loop?
8. Which type of control transfer statement?
9. Write a program to print the Fibonacci series?
10. What is the difference between else statement and goto statement?

Answers for Self Assessment Questions

1 (a) 2 (d) 3 (b) 4(b) 5 (a)

3.12 Further Studies

1. http://www.inf.unibz.it/~calvanese/teaching/ip/lecture-notes/uni06.pdf
2. Let us c by Yashwant Kanatkar.

4

Arrays

STRUCTURE

4.0 Objectives

4.1 Introduction

4.2 Single-Dimensional Arrays

4.3 Reading and writing single dimensional arrays

4.4 Examples of Complex Programs

4.5 Multi-Dimensional Arrays

4.6 Reading writing two-dimensional arrays

4.7 Manipulation in two-dimensional arrays

4.8 Summary

4.9 Keywords

4.10 Review Questions

4.11 Further Studies

4.0 Objectives

After studying this chapter, you will be able to:

- Discuss the single-dimensional arrays
- Understand how to perform the operations on array
- Defined the examples of complex programs with array
- Explain the multi-dimensional arrays

4.1 Introduction

An array is a data structure used to store a collection of data items all of the same type.
The name of the array is associated with the collection of data. To access an individual data item, you need to indicate to the computer which array element you want. This is indicated using an array index (or.subscript).
Why are arrays useful? Suppose you want to write a program which accepts 5 integers input by the user, and prints them out in reverse order. You could do it like this:

```
int first, second, third, fourth, fifth;
printf("enter 5 integers, separated by spaces: ");
scanf("%d %d %d %d %d", &first, &second, &third, &fourth, &fifth);
printf("in reverse order: %d, %d, %d, ", fifth, fourth, third);
printf("%d, %d\n", second, first); /* output is all on 1 line */
```

This works as required. But – what if you had 50 inputs? Or 500?! Or…

Using integer variables would become very cumbersome…

4.2 Single-Dimensional Arrays

The array which is used to represent and store data in a linear form is called as 'single or one dimensional array.'

Syntax:

```
<data-type> <array_name> [size];
```

Example:

```
int a[3] = {2, 3, 5};
char ch[20] = "TechnoExam" ;
float stax[3] = {5003.23, 1940.32, 123.20} ;
```

Total Size (in Bytes):

```
total size = length of array * size of data type
```

In above example, a is an array of type integer which has storage size of 3 elements. The total size would be 3 * 2 = 6 bytes.

* MEMORY ALLOCATION :

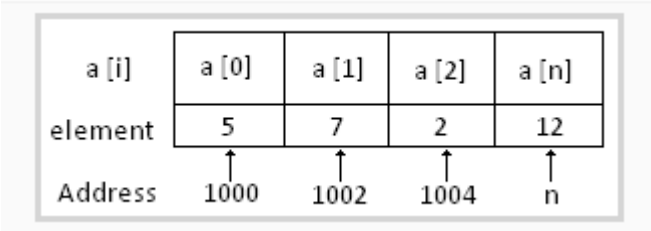


Fig : Memory allocation for one dimensional array

Program :

```
/*  Program to demonstrate one dimensional array.

#include <stdio.h>
#include <conio.h>
void main()
{
        int a[3], i;;
        clrscr();
        printf("\n\t Enter three numbers : ");
        for(i=0; i<3; i++)
        {
```

```
            scanf("%d", &a[i]);  // read array
        }
        printf("\n\n\t Numbers are : ");
        for(i=0; i<3; i++)
        {
            printf("\t %d", a[i]);  // print array
        }
        getch();
}
```

Output :

```
Enter three numbers : 9 4 6

Numbers are :   9       4       6_
```

Features :
Array size should be positive number only.
String array always terminates with null character ('\0').
Array elements are countered from 0 to n-1.
Useful for multiple reading of elements (numbers).

Disadvantages :
There is no easy method to initialize large number of array elements.
It is difficult to initialize selected elements.

Did You Know?
APL, designed by Ken Iverson, was the first programming language to provide array programming capabilities. The mnemonic APL refers to the title of his seminal book "*A Programming Language*".

Caution
Changing data types of elements in an array will cause the occurrence of error in declaration.

4.3 Reading and writing single dimensional arrays

Here is a program that will demonstrate the simple operations of the array.

```
#include <stdio.h>
void oneWay(void);
void anotherWay(void);
int main(void) {
  printf("\noneWay:\n");
  oneWay();
  printf("\nanotherWay:\n");
```

```
  anotherWay();
}

/*Array initialized with aggregate */
void oneWay(void)
{
  int vect[10] = {1,2,3,4,5,6,7,8,9,0};
  int i;
  for (i=0; i<10; i++)
{
     printf("i = %2d vect[i] = %2d\n", i, vect[i]);
  }
}
/*Array initialized with loop */
void anotherWay(void) {
  int vect[10];
  int i;
  for (i=0; i<10; i++)
      vect[i] = i+1;
  for (i=0; i<10; i++)
      printf("i = %2d vect[i] = %2d\n", i, vect[i]);
}
/* The output of this program is
  oneWay:
  i = 0 vect[i] = 1
  i = 1 vect[i] = 2
  i = 2 vect[i] = 3
  i = 3 vect[i] = 4
  i = 4 vect[i] = 5
  i = 5 vect[i] = 6
  i = 6 vect[i] = 7
  i = 7 vect[i] = 8
  i = 8 vect[i] = 9
  i = 9 vect[i] = 0

  anotherWay:
  i = 0 vect[i] = 1
  i = 1 vect[i] = 2
  i = 2 vect[i] = 3
  i = 3 vect[i] = 4
  i = 4 vect[i] = 5
  i = 5 vect[i] = 6
  i = 6 vect[i] = 7
  i = 7 vect[i] = 8
  i = 8 vect[i] = 9
  i = 9 vect[i] = 10
  */
```

Here is a more complex program that will demonstrate how to read, write and traverse the integer arrays

```
#include <stdio.h>
void intSwap(int *x, int *y);
int getIntArray(int a[], int nmax, int sentinel);
void printIntArray(int a[], int n);
void reverseIntArray(int a[], int n);

int main(void) {
  int x[10];
  int hmny;
  hmny = getIntArray(x, 10, 0);
  printf("The array was: \n");
  printIntArray(x,hmny);
  reverseIntArray(x,hmny);
  printf("after reverse it is:\n");
  printIntArray(x,hmny);
}
void intSwap(int *x, int *y)
/* It swaps the content of x and y */
{
  int temp = *x;
  *x = *y;
  *y = temp;
}
/* n is the number of elements in the array a.
These values are printed out, five per line. */
void printIntArray(int a[], int n){
  int i;
  for (i=0; i<n; ){
            printf("\t%d ", a[i++]);
            if (i%5==0)
              printf("\n");
  }
  printf("\n");
}

/* It reads up to nmax integers and stores then in a; sentinel
 terminates input. */
int getIntArray(int a[], int nmax, int sentinel)
{
  int n = 0;
  int temp;
  do {
    printf("Enter integer [%d to terminate] : ", sentinel);
    scanf("%d", &temp);
    if (temp==sentinel)
```

```
    break;
    if (n==nmax)
      printf("array is full\n");
    else
    a[n++] = temp;
  }while (1);
  return n;
}
/* It reverse the order of the first n elements of array */
void reverseIntArray(int a[], int n)
{
  int i;
  for(i=0;i<n/2;i++)
{
    intSwap(&a[i],&a[n–i–1]);
  }
}
```

4.4 Examples of Complex Programs

4.4.1 Searching Programs

Binary search: A binary search, also called a dichotomizing search, is a digital scheme for locating a specific object in a large set. Each object in the set is given a key which help us to searching special objects from a collection.

Example: Here is a c program to search an element in an array using binary search

```
#include<stdio.h>
int main(){
int a[10], i, n, m, c=0, l, u, mid;
printf("Enter the size of an array: ");
scanf("%d",&n);
printf("Enter the elements in ascending order: ");
for(i=0;i<n;i++){
scanf("%d",&a[i]);
}
printf("Enter the number to be search: ");
scanf("%d",&m);
l=0,u=n–1;
while(l<=u){
mid=(l+u)/2;
if(m==a[mid]){
c=1;
break;
}
else if(m<a[mid]){
u=mid–1;
}
else
```

```
l=mid+1;
}
if(c==0)
printf("The number is not found.");
else
printf("The number is found.");
return 0;
}
```

Output:
Enter the size of an array: 5
Enter the elements in ascending order: 4 7 8 11 21
Enter the number to be search: 11
The number is found.

Linear search: A linear search is the most basic of search algorithm you can have. A linear search sequentially moves through a collection (or data structure) looking for a matching value.
Here is a c program to search an element in an array using binary search

Example:

```
#include<stdio.h>
main()
{
int array[100], search, c, number;
printf("Enter the number of elements in array\n");
scanf("%d",&number);
printf("Enter %d numbers\n", number);
for ( c = 0 ; c < number ; c++ )
scanf("%d",&array[c]);
printf("Enter the number to search\n");
scanf("%d",&search);
for ( c = 0 ; c < number ; c++ )
{
if ( array[c] == search )     /* if required element found */
{
printf("%d is present at location %d.\n", search, c+1);
            break;
}
}
if ( c == number )
printf("%d is not present in array.\n", search);
return 0;
}
```

Output:
Enter the number of elements in array
5

Enter 5 numbers
123
56
99
–4568
957
Enter the number to search
99
99 is present at location 3.

Did You Know?

The Boyer-Moore (BM) algorithm, published in 1977, was the first algorithm for searching a simple string in a text file was one of the classic problems in the history of computing.

4.4.2 Sorting Programs

Insertion Sort

Insertion sort is one of the basic algorithms that exist among hundreds of sorting algorithms. It only performs n–1 passes, where n is the number of elements to sort.

Here is an example of simple insertion sort implementation using array in ascending order in c programming language.

Example:

```
#include<stdio.h>
int main(){
int i,j,s,temp,a[20];
printf("Enter total elements: ");
scanf("%d",&s);
printf("Enter %d elements: ",s);
for(i=0;i<s;i++)
scanf("%d",&a[i]);
for(i=1;i<s;i++){
temp=a[i];
j=i–1;
while((temp<a[j])&&(j>=0)){
a[j+1]=a[j];
j=j–1;
}
a[j+1]=temp;
}
printf("After sorting: ");
for(i=0;i<s;i++)
printf(" %d",a[i]);
return 0;
}
```

Output:

Enter total elements: 5

Enter 5 elements: 3 7 9 0 2
After sorting: 0 2 3 7 9

Bubble Sort

Bubble sort, also known as sinking sort, is a simple sorting algorithm that works by repeatedly stepping through the list to be sorted, comparing each pair of adjacent items and swapping them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted.

Here is an example of simple bubble sort implementation using array ascending order in c programming language

Example:

```
#include<stdio.h>
int main(){
int s, temp, i, j, a[20];
printf("Enter total numbers of elements: ");
scanf("%d",&s);
printf("Enter %d elements: ",s);
for(i=0;i<s;i++)
scanf("%d",&a[i]);

//Bubble sorting algorithm
for(i=s–2;i>=0;i– –){
for(j=0;j<=i;j++){
if(a[j]>a[j+1]){
temp=a[j];
a[j]=a[j+1];
a[j+1]=temp;
}
}
}

printf("After sorting: ");
for(i=0;i<s;i++)
printf(" %d",a[i]);
return 0;
}
```

Output:

```
Enter total numbers of elements: 5
Enter 5 elements: 6 2 0 11 9
After sorting: 0 2 6 9 11
```

Caution

Be aware while using pointer with array, it easily make an error, most of which are not detectable by compiler and cause program malfunction in a place that may be distant from the instruction that caused the problem.

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: Define Single-Dimensional Arrays.

……..……

……..……...…

……..

Ex2: Define Sorting Program.

……..……

……..……...…

……..

4.5 Multi-Dimensional Arrays

The array which is used to represent and store data in a tabular form is called as 'two dimensional array.' Such type of array specially used to represent data in a matrix form.
The following syntax is used to represent two dimensional array.
Syntax:

<data-type> <array_nm> [row_subscript][column-subscript];

Example:

int a[3][3];

In above example, a is an array of type integer which has storage size of 3 * 3 matrix. The total size would be 3 * 3 * 2 = 18 bytes.
It is also called as 'multidimensional array.'
* MEMORY ALLOCATION :

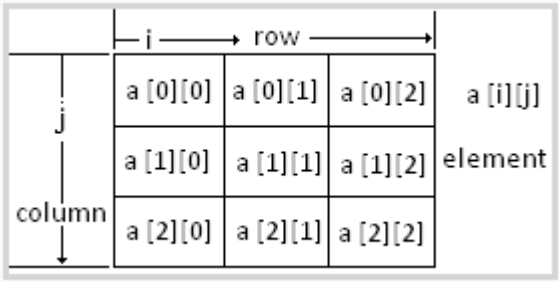


Fig : Memory allocation for two dimensional array

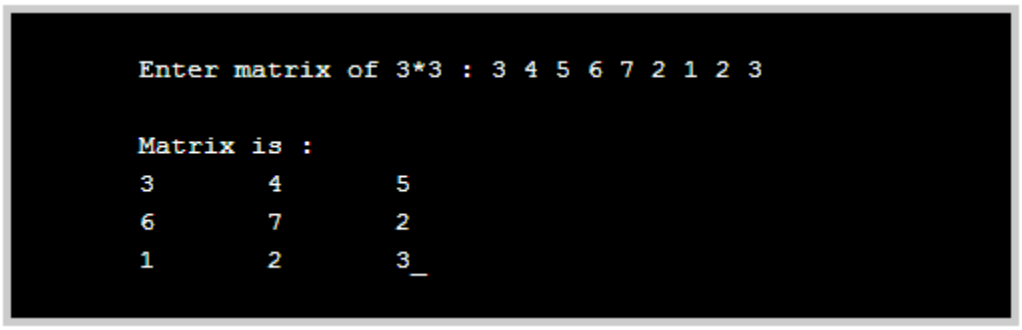
Program :

Program to demonstrate two dimensional array.

```
#include <stdio.h>
#include <conio.h>
```

```
void main()
{
        int a[3][3], i, j;
        clrscr();
        printf("\n\t Enter matrix of 3*3 : ");
        for(i=0; i<3; i++)
        {
                for(j=0; j<3; j++)
                {
                scanf("%d",&a[i][j]);  //read 3*3 array
                }
        }
        printf("\n\t Matrix is : \n");
        for(i=0; i<3; i++)
        {
                for(j=0; j<3; j++)
                {
                printf("\t %d",a[i][j]);  //print 3*3 array
                }
          printf("\n");
        }
        getch();
}
```

Output :



Limitations of two dimensional array :

- We cannot delete any element from an array.
- If we dont know that how many elements have to be stored in a memory in advance, then there will be memory wastage if large array size is specified.

4.6 Reading writing two-dimensional arrays

4.6.1 Working with Matrices:

Filling by list - Just as working with a one-dimensional array, it is possible to fill a two dimensional array by using a list at the time the array is declared. Notice the "sets" of braces "within" the list denoting the "array of arrays".

```
int [ ] [ ] scores = {  { 20, 18, 22, 20, 16 },
                        { 18, 20, 18, 21, 20 },
                        { 16, 18, 16, 20, 24 },
                        {  25, 24, 22, 24, 25 }
                     };
```

Notice the punctuation in the list. The "arrays" are separated by commas, as are ordinary elements in a list. Be sure you include the starting and ending (all inclusive) braces.

No filling when declared - When an array is created it is automatically filled with a zero (for numerical values), a false (for boolean values) or null (for String values).

Filling with user input - When working with two-dimensional arrays (such as accessing, filling, printing, etc.), it is necessary to use nested loops. The outer loop controls the number of rows and the inner loop controls the number of columns.

```
// Filling the matrix
for ( row = 0; row < 4; row ++ )
    for ( column = 0; column < 5; column + + )
    {
      scores [ row ] [ column ] = Console.readInt ("Enter score " + column + "for
                                                          contestant " + row   );
    }
```

Manipulating a matrix - Suppose you want to save the information for 30 students and 3 exam grades for each student entered at the keyboard. In addition, you want to find the average (which could be a decimal value) for each student, and then store this average in a fourth column of the same matrix. Remember, you will need to obtain the grades before you can compute the average. Here is one possibility:

```
import java.io.*;
import BreezyGUI.*;

public class matrixtest
{
  public static void main(String[] args)
  {
     double [ ] [ ] grades = new double [ 30 ] [ 4 ] ; //create memory space for entire matrix

     // Fill the matrix with user input and compute average
     int row, column;
     double sum, average;
     for ( row = 0; row < 3; row ++ )
     {
       sum = 0;
       for(column = 0; column < 3; column++)
       {
         grades[row][column] = Console.readDouble("Enter grade " + (column +1) +
                                                          "for student " + (row+1));
         sum = sum + grades[row][column];
       }
       average = sum / 3;
```

```
        grades[row][3] = average;
      }
      // Print averages only
      System.out.println("You saved the following averages: ");
      for( row = 0; row < 3; row ++ )
      {
          System.out.println("Student " + (row + 1) + ": " + grades[row][3]);
      }
   }
}
```

• Length: Just as a command such as list.length returns the length of a one dimensional array, scores.length will return the number of rows in this two-dimensional array. scores[i].length will return the number of columns of the row with subscript i in a two-dimensional array.

Working with Strings - Create a matrix of String values, fill the matrix by list, and print the matrix. Notice that the "internal" arrays are of differing sizes. Notice how the .length is used to deal with these varying lengths during printing.

```
public class ArrayOfArraysAnimalDemo
{
    public static void main(String[ ] args)
    {
        String[ ][ ] animals = {
                    { "DanaDog", "WallyDog", "JessieDog", "AlexisDog", "LuckyDog" },
                    { "BibsCat", "DoodleCat", "MillieCat", "SimonCat" },
                    { "ElyFish", "CloieFish", "GoldieFish", "OscarFish", "ZippyFish",
                                                                        "TedFish"},
                    { "RascalMule", "GeorgeMule", "GracieMule", "MontyMule",
                                                          "BuckMule", "RosieMule" }
                    };

        for (int i = 0; i < animals.length; i++)
        {
             System.out.print(animals[ i ] [ 0 ] + ": ");
             for (int j = 1; j < animals[ i ].length; j++)
             {
                  System.out.print(animals[ i ][ j ] + " ");
             }
            System.out.println( );
        }
    }
}
```

Self Assessment Questions

1. Array passed as an argument to a function is interpreted as
(a) Address of the array.

(b) Values of the first elements of the array.
(c) Address of the first element of the array.
(d) Number of element of the array.

2. How can we dynamically allocate a two-dimensional array?
(a) int **array1 = (int **)malloc(nrows * sizeof(int *));
    for(i = 0; i < nrows; i++)
    array1[i] = (int *)malloc(ncolumns * sizeof(int));
(b) int **array2 = (int **)malloc(nrows * sizeof(int *));
     array2[0] = (int *)malloc(nrows * ncolumns * sizeof(int));
     for(i = 1; i < nrows; i++)
     array2[i] = array2[0] + i * ncolumns;
(c) int *array3 = (int *)malloc(nrows * ncolumns * sizeof(int));
(d) All of the above.

3. main()
{
char thought[2][30]={"Do not walk in front of me..","I am not follow"};
printf("%c%c",*(thought[0]+9),*(*(thought+0)+5));
}
What is the output of this program?
(a) k k (b) Do not walk in front of me
(c) I may not follow (d) K
4. What will be output if you will execute following c code?
#include<stdio.h>
#include<conio.h>
void main(){
   int a[]={0,1,2,3,4,5,6,7,8,9,10};
   int i=0,num;
   num=a[++i+a[++i]]+a[++i];
   printf("%d",num);
}
(a) 6 (b) 7
(c) 8 (d) 9

5. When array elements are passed to a function with call by reference, function has pointer arguments.
(a) True (b) False

6. When array is declared with rows and columns it is called as 2-D i.e. two dimensional array
(a) True (b) False

4.7 Manipulation in two-dimensional arrays

Manipulating Multidimensional Arrays: This example shows how to work with arrays having more than two dimensions. Multidimensional arrays can be numeric, character, cell, or structure arrays.

Multidimensional arrays can be used to represent multivariate data.provides a number of functions that directly support multidimensional arrays.
Creating Multi-Dimensional Arrays: Multidimensional arrays in MATLAB are created the same way as two-dimensional arrays. For example, first define the 3 by 3 matrix, and then add a third dimension.

```
A = [5 7 8;
     0 1 9;
     4 3 6];
A(:,:,2) = [1 0 4;
            3 5 6;
            9 8 7]
```

```
A(:,:,1) =

     5     7     8
     0     1     9
     4     3     6

A(:,:,2) =

     1     0     4
     3     5     6
     9     8     7
```

The CAT function is a useful tool for building multidimensional arrays. B = cat(DIM,A1,A2,...) builds a multidimensional array by concatenating A1, A2 ... along the dimension DIM

```
B = cat( 3, [2 8; 0 5], [1 3; 7 9], [2 3; 4 6])
```

```
B(:,:,1) =
   2     8
   0     5
B(:,:,2) =

     1     3
     7     9
B(:,:,3) =

     2     3
     4     6
```

Calls to CAT can be nested.

```
A = cat(3,[9 2; 6 5], [7 1; 8 4]);
B = cat(3,[3 5; 0 1], [5 6; 2 1]);
C = cat(4,A,B,cat(3,[1 2; 3 4], [4 3; 2 1]));
```

Finding the Dimensions

SIZE and NDIMS return the size and number of dimensions of matrices.

```
SzA   = size(A)
DimsA = ndims(A)
SzC   = size(C)
DimsC = ndims(C)
```

```
SzA =

     2     2     2


DimsA =

     3


SzC =

     2     2     2     3


DimsC =

     4
```

Accessing Elements

To access a single element of a multidimensional array, use integer subscripts. For example D(1,2,2,22), using D defined in the previous slide, returns 6.

Array subscripts can also be vectors. For example:

```
K = C(:,:,1,[1 3])
```

```
K(:,:,1,1) =

     9     2
     6     5


K(:,:,1,2) =

     1     2
     3     4
```

Manipulating Multi-Dimensional Arrays

RESHAPE, PERMUTE, and SQUEEZE are used to manipulate n-dimensional arrays. RESHAPE behaves as it does for 2D arrays. The operation of PERMUTE is illustrated below.

Let A be a 3 by 3 by 2 array. PERMUTE(A,[2 1 3]) returns an array with the row and column subscripts reversed (dimension 1 is the row, dimension 2 is the column, dimension 3 is the depth and so on). Similarly, PERMUTE(A,[3,2,1]) returns an array with the first and third subscripts interchanged.

```
A = rand(3,3,2);
B = permute(A, [2 1 3]);
C = permute(A, [3 2 1]);
```

Selecting 2D Matrices From Multi-Dimensional Arrays

Functions like EIG that operate on planes or 2D matrices do not accept multi-dimensional arrays as arguments. To apply such functions to different planes of the multidimensional arrays, use indexing or FOR loops. For example:

```
A = cat( 3, [1 2 3; 9 8 7; 4 6 5], [0 3 2; 8 8 4; 5 3 5], ...
                 [6 4 7; 6 8 5; 5 4 3]);
% The EIG function is applied to each of the horizontal 'slices' of A.
for i = 1:3
    eig(squeeze(A(i,:,:)))
end
```

```
ans =

   10.3589
   -1.0000
    1.6411


ans =

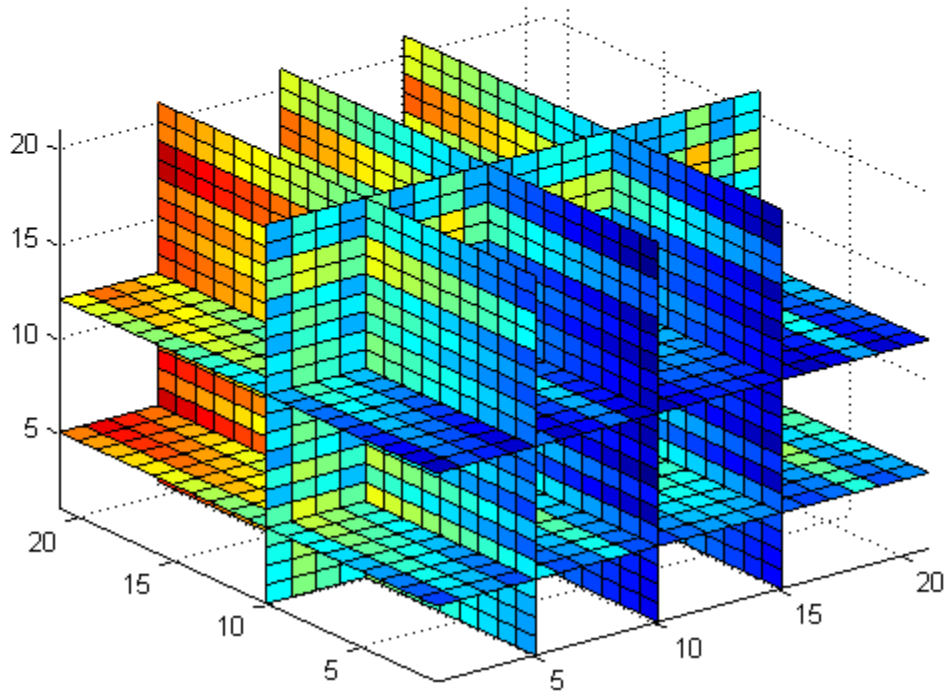
  21.2293
   0.3854 + 1.5778i
   0.3854 - 1.5778i


ans =

  13.3706
  -1.6853 + 0.4757i
  -1.6853 - 0.4757i
```

INTERP3, INTERPN, and NDGRID are examples of interpolation and data gridding functions that operate specifically on multidimensional data. Here is an example of NDGRID applied to an N-dimensional matrix.

```
x1 = -2*pi:pi/10:0;
x2 = 2*pi:pi/10:4*pi;
x3 = 0:pi/10:2*pi;
[x1,x2,x3] = ndgrid(x1,x2,x3);
z = x1 + exp(cos(2*x2.^2)) + sin(x3.^3);
slice(z,[5 10 15], 10, [5 12]); axis tight;
```



You can build multidimensional cell arrays and multidimensional structure arrays, and can also convert between multidimensional numeric and cell arrays.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Define Multi-Dimensional Arrays.

……..……

……..……

……..

Ex2: Define Manipulation in two-dimensional arrays.

……..……

……..……

……..

4.8 Summary

- An array is a collection of similar elements. These similar elements could be all integers or all floats or all characters etc.
- Array size must be declared using constant value before initialization.
- A single dimensional array will be useful for simple grouping of data that is relatively small in size.
- Sorting is the process to arrange the array elements in ascending or descending order.
- Arrays are provides a simple mechanism where more than one elements of same type are to be used.

4.9 Keywords

***Array*:** An array is the collection of elements with contiguous memory allocation.
Binary Search: It is a digital scheme for locating a specific object in a large set. Each object in the set is given a key which help us to searching special objects from a collection.
Bubble sort: It is a simple sorting algorithm that works by repeatedly stepping through the list to be sorted, comparing each pair of adjacent items and swapping them if they are in the wrong order.
Insertion sort: It is one of the basic algorithms that exist among hundreds of sorting algorithms. It only performs n–1 passes, where n is the number of elements to sort.
Linear Search: It is the most basic of search algorithm you can have. A linear search sequentially moves through a collection (or data structure) looking for a matching value.
Multi-Dimensional Arrays: Multidimensional arrays operate on the same principle as single-dimensional arrays. It shows the different dimensions of an array.

4.10 Review Questions

1. Defined the array using in C? Explain with example.
2. What is the single dimensional array? How do we declare and initialize an array? Give the example.
3. Write a C program to copy one array into another and explain its steps.
4. Differentiate between binary search and linear search.
5. Write a C program to sort the given numbers using insertion sort:
   45, 23, 54, 27, 12, 2, 56, 63
6. What is the concept of multidimensional array in C? Explain with example.
7. Write a C program for the multiplication of two arrays.
8. Enter the values in two-dimensional array and access the elements from them.
9. Create matrix of dimension 3×3 for storing 9 values in them.
10. Write a C program to find the sum of the given n integers using an array.

Answers for Self Assessment Questions

1. (c) 2. (c). 3. (b) 4. (d) 5. (a) 6. (a)

4.11 Further Studies

1. http://www.plantation-productions.com/Webster/www.artofasm.com/Linux/PDFs/Arrays.pdf
2. Let us c by Yashwant Kanatkar.

5
Strings

STRUCTURE

5.0 Objectives

5.1 Introduction

5.2 Concepts of String and String Variable

5.3 String Input/output Functions

5.4 Arrays of Strings

5.5 String Handling Functions

5.6 Summary

5.7 Keywords

5.8 Review Questions

5.9 Further Studies

5.0 Objectives

After studying this chapter, you will be able to:

- Discuss the concept of string and string variable
- Explain the string input/output functions
- Understand the arrays of strings
- Define and declare string handling functions

5.1 Introductions

In C language, strings are stored in an array of char type along with the null terminating character "\0" at the end. In other words to create a string in C you create an array of chars and set each element in the array to a char value that makes up the string. When sizing the string array you need to add plus one to the actual size of the string to make space for the null terminating character, "\0".

Syntax to declare a string in C:

char fname[4];

The above statement declares a string called fname that can take up to 3 characters. It can be indexed just as a regular array as well.

fname[] = {'t', 'w', 'o'};

Character	t	w	o	\0
ASCII Code	116	119	41	0

The last character is the null character having ASCII value zero.

5.2 Concepts of String and String Variable

Strings in C are represented by arrays of characters. The end of the string is marked with a special character, the null character, which is simply the character with the value 0. (The null character has no relation except in name to the null pointer. In the ASCII character set, the null character is named NUL.) The null or string-terminating character is represented by another character escape sequence, \0.

Because C has no built-in facilities for manipulating entire arrays (copying them, comparing them, etc.), it also has very few built-in facilities for manipulating strings.

In fact, C's only truly built-in string-handling is that it allows us to use string constants (also called string literals) in our code. Whenever we write a string, enclosed in double quotes, the C automatically creates an array of characters for us, containing that string, terminated by the \0 character. For example, we can declare and define an array of characters, and initialize it with a string constant:

char string[] = "Hello, world!";

In this case, we can leave out the dimension of the array, since the compiler can compute it for us based on the size of the initializer (14, including the terminating \0). This is the only case where the compiler sizes a string array for us, however; in other cases, it will be necessary that we decide how big the arrays and other data structures we use to hold strings are.

To do anything else with strings, we must typically call functions. The C library contains a few basic string manipulation functions, and to learn more about strings, we will be looking at how these functions might be implemented.

Did You Know?

String and character manipulations involve the manipulation of the appropriate numeric codes and not the characters themselves.

5.3 String Input/output Functions

5.3.1 Strings-Gets and Puts

We have been postponing a discussion on gets and puts, the functions associated with strings. This is the right place since; a string is an arra1, of characters. These functions are appropriate when strings are to be received from the screen or sent to the screen without errors. One can use scanf( ) to receive strings from the screen. The program using scanf for reading a name is given as follows:

char name [25];

scanf (" o/os", name);

Strings can be declared as an array of characters as shown above in the scanf function when we get the array of characters as a string, it is enough to indicate the name of the array without a subscript. When we get a string, there is no need for writing "&" in the scanf function.

We can indicate the name of the string variable itself.

Strings may contain blanks in between. If you use a scanf( ) function to get the string with a space in between such as "Rama Krishnan will not be taken note of since space is an indicator of the end of entry of the input. But gets will take all that is entered til1 the enter key is pressed. Therefore after entering the full name, the enter key can be pressed Thus using gets is a better way for strings. We can get a string in a much simpler way using gets( ).

The syntax for gets is: gets(name);

Similarly puts( ) can be used for printing a variable or a constant as given as follows:

puts (name);

puts ("Enter the word");

However, there is a limitation. printf can be used to print more than one variable and scanf to get more than one variable at a time, in a single statement. However, puts can output only one variable and gets can input only one variable in one statement.
As you know that gets and puts are unformatted I/O functions, there are no format specifications associated with them. We will take another interesting example. If a word is a palindrome, we will get the same word when we read the characters from the right to the left as well, as already discussed.

Examples are: nun
malayalam
That program uses a library."- function called strlen( ). The function strlen (str) returns the size or length of the given string.
Now let us look at the program

Example:

```
/* to check whether a string is palindrome*/
 #include <stdio.h>
#include <string.h>
#define FALSE 0
    main()
    {
          int flag=l;
           int right, left, n;
           char w[50]; /* maximum width of string 50*/
           puts("Enter string to be checked for palindrome");
           gets(w);
          n=strlen(w)– 1;
           for ((left=0, right=n); left<=n/2; ++left, – –right) {
                 if (w[left]!=w [right])
{
                       flag=FALSE;
                      break;
                 }
           }
           if (flag)
           {
                 puts (w);
                 puts (" is a palindrome");
           }
           else
                 printf ("%s is NOT a palindrome" ) ;
     }
   Output
     Enter string to be checked for palindrome
     palap
     palap
      is a palindrome
```

5.4 Arrays of Strings

In the array of strings we defined the strings in two ways:

- Single dimensional,
- Two dimensional (Multidimensional).

In the single dimensional strings of array following example:

```
char string5[20] = “Hello, “;
        char string6[] = “world!”;

        printf(“%s\n”, string5);

        strcat(string5, string6);

        printf(“%s\n”, string5);
```

Arrays of strings (arrays of character arrays) can be declared and handled in a similar manner to that described for 2-D arrays. Consider the given example:

```
#include< stdio.h>
void main(void)
{
char names[2][8] = {“Frans”, “Coenen”};

/* Output */

printf(“names = %s, %s\n”,names[0],names[1]);
printf(“names = %s\n”,names);

/* Output initials */

printf(“Initials = %c. %c.\n”,names[0][0],names[1][0]);
}
```

Here we declare a 2-D character array comprising two “roes” and 8 “columns”. We then initialize this array with two character strings. The output the array we need to index into each row using the 2-D array name on its own (strings) as a pointer cause only the first element (“row”) to be produced. Note that we can still index to individual elements using index pairs. The output from the above will be:

```
names = Frans, Coenen
names = Frans
Initials = F. C.
```

The example shows the concept of array of stings:

```
#include <stdio.h>
#include <conio.h>
void main()
{
  clrscr();
  char *array[2];
```

```
  array[0]="Hello";
  array[1]="World";
  printf("The Array of String is = %s,%s\n", array[0], array[1]);
  getch();
}
```

Caution

String comparison operators can be confusing when you are comparing numeric strings, if you are used to assuming as numbers, not strings. It may cause of error.

Did You Know?

The assignment operator may be used to initialize the array later but functions may not be used for initialization.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Define concept of string.

……..……

……..……...…

……..

Ex2: Define string as arrays.

……..……

……..……...…

……..

5.5 String Handling Functions

Here is the String handling functions as:

- strlen()
- strcpy()
- strncpy()
- strcat()
- strncat()
- strcmp()
- strncmp()

This discussion of string handling in C presumes that the following compiler directive is used.

#include <string.h>

On Linux, this file is located in /usr/include. Only the basic functions will be discussed here, so you may want to investigate string.h. In C, a string is stored as a null-terminated char array. This means that after the last truly usable char there is a null, hex 00, which is represented in C by '\0'. The subscripts used for the array start with zero (0). The following line declares a char array called *str*. The C provides fifteen consecutive bytes of memory. Only the first fourteen bytes are usable for character storage, because one must be used for the string-terminating null.

```
char str[15];
```

The following is a representation of what would be in RAM, if the string "Hello, world!" is stored in this array.
Characters: H e l l o , w o r l d !
Hex values: 48 65 6C 6C 6F 2C 20 77 6F 71 6C 64 21 00
Subscripts: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
The name of the array is treated as a pointer to the array. The subscript serves as the offset into the array, i.e., the number of bytes from the starting memory location of the array. Thus, both of the following will save the address of the 0^{th} character in the pointer variable *ptr*.
ptr = str;
ptr = &str[0];

strlen()
Syntax: len = strlen(ptr);
where len is an integer and ptr is a pointer to char. strlen() returns the length of a string, excluding the null. The following code will result in len having the value 13.
int len;
char str[15];
strcpy(str, "Hello, world!");
len = strlen(str);

strcpy()
Syntax: strcpy(ptr1, ptr2);
where ptr1 and ptr2 are pointers to char. strcpy() is used to copy a null-terminated string into a variable. Given the following declarations, several things are possible.
char S[25];
char D[25];

- Putting text into a string:
  strcpy(S, "This is String 1.");
- Copying a whole string from S to D:
  strcpy(D, S);
- Copying the tail end of string S to D:
  strcpy(D, &S[8]);

Ensure that the source string is null-terminated, very strange and sometimes very ugly things may result.

strncpy()
Syntax: strncpy(ptr1, ptr2, n);
where n is an integer and ptr1 and ptr2 are pointers to char
strncpy() is used to copy a portion of a possibly null-terminated string into a variable. Care must be taken because the '\0' is put at the end of destination string only if it is within the part of the string being copied.
Given the following declarations, several things are possible.
char S[25];
char D[25];
Assume that the following statement has been executed before each of the remaining code fragments.

- Putting text into the source string:
  strcpy(S, "This is String 1.");
- Copying four characters from the beginning of S to D and placing a null at the end:
  strncpy(D, S, 4);

D[4] = '\0';

- Copying two characters from the middle of string S to D:
  strncpy(D, &S[5], 2);
  D[2] = '\0';
- Copying the tail end of string S to D:
  strncpy(D, &S[8], 15);

which produces the same result as strcpy(D, &S[8]);

Caution

Be aware that strncpy will not automatically append a null terminator, which means that you can go from a regular, null-terminated string to a non-null-terminated string if you try to copy a string that would not present correctly at the destination.

strcat()

Syntax: strcat(ptr1, ptr2);
where ptr1 and ptr2 are pointers to char
strcat() is used to concatenate a null-terminated string to end of another string variable. This is equivalent to pasting one string onto the end of another, overwriting the null terminator. There is only one common use for strcat().
char S[25] = "world!";
char D[25] = "Hello, ";

- Concatenating the whole string S onto D:
  strcat(D, S);

strncat()

Syntax: strncat(ptr1, ptr2, n);
where n is an integer and ptr1 and ptr2 are pointers to char
strncat() is used to concatenate a portion of a possibly null-terminated string onto the end of another string variable. Care must be taken because some earlier implementations of C do not append the '\0' at the end of destination string. Given the following declarations, several things are possible, but only one is commonly used.
char S[25] = "world!";
char D[25] = "Hello";

- Concatenating five characters from the beginning of S onto the end of D and placing a null at the end:
  strncat(D, S, 5);
  strncat(D, S, strlen(S) –1);

Both would result in D containing "Hello, world".

strcmp()

Syntax: diff = strcmp(ptr1, ptr2);
where diff is an integer and ptr1 and ptr2 are pointers to char
strcmp() is used to compare two strings. The strings are compared character by character starting at the characters pointed at by the two pointers. If the strings are identical, the integer value zero (0) is returned. As soon as a difference is found, the comparison is halted and if the ASCII value at the point of difference in the first string is less than that in the second (e.g. 'a' 0x61 vs. 'e' 0x65) a negative value is returned; otherwise, a positive value is returned. Examine the following examples.
char s1[25] = "pat";

char s2[25] = “pet”;
diff will have a *negative* value after the following statement is executed.
diff = strcmp(s1, s2);
diff will have a *positive* value after the following statement is executed.
diff = strcmp(s2, s1);
diff will have a value of *zero* (0) after the execution of the following statement, which compares *s1* with itself.
diff = strcmp(s1, s1);

strncmp()
Syntax: diff = strncmp(ptr1, ptr2, n);
where diff and n are integers ptr1 and ptr2 are pointers to char. strncmp() is used to compare the first n characters of two strings. The strings are compared character by character starting at the characters pointed at by the two pointers. If the first n strings are identical, the integer value zero (0) is returned. As soon as a difference is found, the comparison is halted and if the ASCII value at the point of difference in the first string is less than that in the second (e.g. ‘a’ 0x61 vs. ‘e’ 0x65) a negative value is returned; otherwise, a positive value is returned. Examine the following examples.
char s1[25] = “pat”;
char s2[25] = “pet”;
diff will have a *negative* value after the following statement is executed.
diff = strncmp(s1, s2, 2);
diff will have a *positive* value after the following statement is executed.
diff = strncmp(s2, s1, 3);
diff will have a value of *zero* (0) after the following statement.
diff = strncmp(s1, s2, 1);

The following example show the all condition that occurs in the handling the stings function
char str[25] = “cot”;
char ch = ‘u’;
char D[25] = “pat”;

- Replacing a single character using a char variable:
  D[1] = ch;

This would result in D containing “put”.

- Replacing a single character using a char literal:
  D[1] = ‘e’;

This would result in D containing “pet”.

- Replacing a single character using a single character from a string variable:
  D[1] = str[1];

This would result in D containing “pot”.

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: Write a program to concatenate two strings, “Good” and “Morning”.
……..……

……..……...…

……..

Ex2: Write a program to compare first four charactes of two string.

……..……

……..……...….

……..

5.6 Summary

- Strings in C are represented by arrays of characters.
- The gets and puts are unformatted I/O functions, there are no format specifications associated with them.
- The printf and scanf, in order to input or output more than one variable, separate statements have to be written for each variable
- In the array of strings we defined the strings in two ways: Single dimensional and two dimensional (Multidimensional).
- In C, a string is stored as a null-terminated char array.

5.7 Keywords

Null-terminated String: String is terminated by a special character which is called as null terminator or null parameter (/0).
strcat(): It is used to concatenate a null-terminated string to end of another string variable. This is equivalent to pasting one string onto the end of another, overwriting the null terminator.
String: C string is defined as an array of characters or a pointer to characters.
strlen() It returns the length of a string, excluding the null
strncmp(): It is used to compare the first n characters of two strings. The strings are compared character by character starting at the characters pointed at by the two pointers.

5.8 Review Questions

1. How do we store a string in the array?
2. Differentiate between string and characters.
3. Write a program which would print the alphabet set a to z and A to Z in decimal and character form.
4. Write a program, which will arrange the words in the ascending order of word length.
5. What is the string handling functions in C?
6. What are the differences between strcpy() and strncpy()?
7. What are the string input/output functions?
8. Differentiate between strcat() and strncat()?
9. Define the concept of string and string variable.
10. How do we create the arrays of strings?

5.9 Further Studies

1. http://stephane.ducasse.free.fr/FreeBooks/ByExample/14%20-%20Chapter%2012%20-%20Strings.pdf
2. Programing with c by Bala gura swami.

6

User Defined Functions

STRUCTURE

6.0 Objectives

6.1 Introduction

6.2 Elements of User-defined Functions

6.3 Categories of Functions

6.4 Passing Parameters to Functions

6.5 Arrays in Functions

6.6 Nesting of Functions

6.7 Recursion

6.8 Command Line Arguments

6.9 Storage Classes

6.10 Summary

6.11 Keywords

6.12 Review Questions

6.13 Further Studies

6.0 Objectives

After studying this chapter, you will be able to:

- Discuss the basic concept of elements of user-defined functions
- Explain the categories of functions
- Define and passing parameters to functions
- Understand about the arrays in functions
- Explain the nesting of functions
- Define the recursion
- Explain the command line arguments
- Discuss the storage classes

6.1 Introduction

In C functions can be classified into two categories, namely, library functions and user-defined functions. main is an example of user defined function. printf and scanf belong to the category of library functions. The main

distinction between user-defined and library function is that the former are not required to be written by user while the latter have to be developed by the user at the time of writing a program. However the user defined function can become a part of the C program library.

6.2 Elements of User-defined Functions

A function consists of three parts:

- Function declaration
- Function call
- Function definition

6.2.1 Function Declaration

A function declaration is also called a function prototype. A function may be declared at the beginning of the main function. The function declaration is of the following type:

return data-type function name (formal argument 1, argument2,-------);

A function after execution may return a value to the function, which called it. It may not return a value at all, but, may perform some operations instead. It may return an integer, character, or float. If it returns a float we may declare the function as

```
float f1(float arg 1 int arg 2);
```

If it does not return any value we may write the above as

```
void fun2(float arg1, int arg2); /*void means nothing*/
```

If it returns a character, we may write

```
char fun3 (float arg1 int arg2);
```

If no arguments are passed into a function, an empty pair of parentheses must follow the function name.

```
Eg. char fun4 ( );
```

The arguments declared as part of the prototype are also known as formal parameters. The formal arguments indicate type of data to be transferred from the calling function. This is about the function declaration.

6.2.2 Function Call

We may call a function either directly or indirectly. When we call the function, we pass actual arguments or values. Calling a functional is also known as function reference. There must be a one to one correspondence between formal arguments declared and the actual arguments sent. They should be of the same data type and in the same order.

eg. sum=f1 (20.5, 10); fun4();

6.2.3 Function Definition

The function definition can be written anywhere in the file with a proper declaration, followed by the declaration of local variables and statements. The function definition consists of two parts namely function declarator or heading and function body the function heading is similar to function declaration, but will not terminate with a semicolon.

Let as assume that we wish to get two integers Pass them to a function *add.* Add them in the add function. Return the value to the main function and print it the algorithm for solving problem is given below:

main function

Step 1: define function add
Step 2: get 2 integers
Step 3: call add & Pass the 2 values
Step 4: get the sum

Step 5: print the value

function add

Step 1: get the value
Step 2: add them
Step 3: return the value to main

Thus we have divided the problem. The program is given below:

```
/*Example */
/*use of function*/
#include <stdio. h>
main()
{
        int a=0, b=0, sum=0;
        int add(int a, int b); /*function declaration*/
        printf ("enter 2 integers\n");
        scanf("%d%d", &a, &b);
        sum =add(a, b); /*function call*/
        printf ("sum of %d and %d =%d", a, b, sum);
}
/*function definition*/
int add (int c, int d)     /* function declarator*/
{
        int e;
        e=c+d;
        return e;
```

Result of program

```
Enter 2 integers
6667    4445
Sum of 6667 and 4445=11112
```

The explanation as to how the program works is given below:

On the filth statement (seventh line), the declaration of the function add is given. Note that the function will return an integer. Hence the return type is defined as int. The formal arguments are defined as int a and int b. The function name is add. We cannot use a variable without declaring it, as also a function without telling the compiler about it. Function declaration should appear at the beginning of the calling function. It hints to the compiler that the function is going to call the function add, later in the program. If the calling function is not going to pass any arguments, then empty parentheses are to be written after the function name. The parentheses must be present be in the function declaration. This happens when the function is called to perform an operation without passing arguments. In this case, if a & b are part of the called function(add) itself, then we need not pass any parameters. In such a case the function declaration will be as follows assuming that the called function returns an integer:

```
int add ();
```

In example ,we get the values of a & b. After that we call the function add and assign the value returned by the function to an already defined int variable sum as given below:

```
sum = add(a, b);
```

Note that odd (a, b) is the function call or function reference. Here the return type is not to be given. The types of the arguments are also not to be given. It is a simple statement without all the elements of the function

declaration. However, the function name and the names of the arguments passed, if any, should be present in the function call. When the program sees a function reference or function call, it looks for and calls the function and transfers the arguments.
The function definition consists of two parts i.e., the function declarator and function body.
The function declarator is a replica of the function declaration. The only difference is that the declaration in the calling function will end with a semicolon and the declarator in the called function will not end with a semicolon. As in main(), the entire function body will be enclosed within braces. The whole function can be assumed to be one program statement. This means, that all the statements within the body will be executed one after another before the program execution returns to the place in the main function from where it was called.

The important points to be noted are:
a) The declarator must agree totally with the declaration in the *called* function i.e., the return data type, the function name, the argument type should all appear in the same order. The declarator will not end with a semicolon.
b) You can also give the same name as in the calling function-in declaration statement or function call--or different names to the arguments in the function declarator. Here we have given the names c and d. What is important however, is that the type of arguments should appear, as it is in the declaration in the calling program. They must also appear in the same order.
c) At the time of execution, when the function encounters the closing brace }, it returns control to the calling program and returns to the same place at which the function was called.
In this program we have a specific statement return (e) before the closing brace. Therefore the program will go back to the main function with value of e. This value will be substituted as
sum=(returned value)
Therefore sum gets the value which is printed in the next statement. This is how the function works.
Assume now that the program gets a & b values, gets their sum1, gets c & d and gets their sum2 and then both the sums are passed to the function to get their total. The program for doing this is given below:

```
/*Example */
/* A function called many times */
#include <stdio.h>
main ( )
{
float a, b, c, d, sum1, sum2, sum3;
float add(f1oat a, float b); /*function declaration*/
printf("enter 2 float numbers\n");
scanf ("%f%f", &a, &b);
sum1 =add(a, b); /*function call*/
printf("enter 2 more float numbers\n");
scanf("%f%f", &c, &d);
sum2 =add(c, d); /*function call*/
sum3 =add(sum1, sum2); /*function call*/
printf(sum of %f and %f =%f\n", a, b, sum1);
printf("sum of %f and %f =%f\n", c, d, sum2);
printf("sum of %f and %f =%f\n", sum1, sum2, sum3);
}
/*function definition*/
float add (float c, float d) /*function declarator*/
```

{
float e;
e=c+d;
return e;
i

Result of program

enter 2 float numbers
1.5 3.7
enter 2 more float numbers
5.6 8.9
sum of 1.500000 and 3.700000 =5.200000
sum of 5. 600000 and 8 . 900000 :14 – 500000
sum of 5.200000 and 14.500000 :19.70000
We have defined *sum1, sum2 and sum3* as float variables.
We are calling function *add* three times with the following assignment statements:
sum1 =add(a, b);
sum2 = add(c, d);
sum3 = add(sum1, surr2);
Thus the program goes back and forth between main &, add as given below:
main()
add(a, b)
main()
add(c, d)
main()
add (sum1, sum2)
main()
Had we not used the function odd, we would have to write statements pertaining to *add* 3 times in the main program such a program would be large and difficult to read. In this method we have to code for *add* only once, and hence the program size is small. This is one of the reasons for the usage of functions.

In Example, we could add another function call by *add* (10.005 ,3.1125); This statement will also work perfectly. After the function is executed, the sum will be returned to the main function. Therefore, both variables and constants can be passed to a function by making use of the same function declaration.

Did You Know?

Older versions of the C language did not have prototypes; the function declarations only specified the return type and did not list the argument types.

6.3 Categories of Functions

The following categories of function:

- Functions with no arguments and no return values.
- Functions with arguments and no return values.
- Functions with arguments and return values.
- Functions that return multiple values.
- Functions with no arguments and return values.

6.3.1 Functions with no Arguments and no Return Value

A C function without any arguments means you cannot pass data (values like int, char etc) to the called function. Similarly, function with no return type does not pass back data to the calling function. It is one of the simplest types of function in C. This type of function which does not return any value cannot be used in an expression it can be used only as independent statement. Let's have an example to illustrate this.

```
#include<stdio.h>
#include<conio.h>
void printline()
{
int i;
printf("\n");
for(i=0;i<30;i++)
{
printf("–");
}
printf("\n");
}
void main()
{
clrscr();
printf("Welcome to function in C");
printline();
printf("Function easy to learn.");
printline();
getch();
}
```

6.3.2 Functions with Arguments and No Return Value

In the example what we have noticed that "main()" function has no control over the UDF(User define function) "printfline()", it cannot control its output. Whenever "main()" calls "printline()", it simply prints line every time. So the result remains the same.

A C function with arguments can perform much better than previous function type. This type of function can accept data from calling function. In other words, you send data to the called function from calling function but you cannot send result data back to the calling function. Rather, it displays the result on the terminal. But we can control the output of function by providing various values as arguments. Let's have an example to get it better.

```
#include<stdio.h>
#include<conio.h>
void add(int x, int y) {
int result;
result = x+y;
printf("Sum of %d and %d is %d.\n\n",x,y,result);
}
void main()
{
clrscr();
add(30,15);
```

```
add(63,49);
add(952,321);
getch();
}
```

6.3.3 Functions with Arguments and Return Value

This type of function can send arguments (data) from the calling function to the called function and wait for the result to be returned back from the called function back to the calling function. And this type of function is mostly used in programming world because it can do two way communications; it can accept data as arguments as well as can send back data as return value. The data returned by the function can be used later in our program for further calculations.

```
#include<stdio.h>
#include<conio.h>
int add(int x, int y)
{
int result;
result = x+y;
return(result);
}
void main()
{
int z;
clrscr();
z = add(952,321);
printf("Result %d.\n\n",add(30,55));
printf("Result %d.\n\n",z);
getch();
}
```

6.3.4 Functions with No Arguments but Returns Value

We may need a function which does not take any argument but only returns values to the calling function then this type of function is useful. The best example of this type of function is "getchar()" library function which is declared in the header file "stdio.h". We can declare a similar library function of own. Take a look.

```
#include<stdio.h>
#include<conio.h>
int send() {
int no_1;
printf("Enter a no : ");
scanf("%d",&no_1);
return(no_1);
}
void main()
{
int z;
clrscr();
z = send();
printf("\nYou entered : %d.", z);
```

```
getch();
}
```

6.3.5 Functions That Return Multiple Values

So far, we have learned and seen that in a function, return statement was able to return only single value. That is because; a return statement can return only one value. But if we want to send back more than one value then how we could do this?

We have used arguments to send values to the called function, in the same way we can also use arguments to send back information to the calling function. The arguments that are used to send back data are called Output Parameters.

It is a bit difficult for novice because this type of function uses pointer. Let's see an example:

```
#include<stdio.h>
#include<conio.h>
void calc(int x, int y, int *add, int *sub) {
*add = x+y;
*sub = x–y;
}
void main()
{
int a=20, b=11, p,q;
clrscr();
calc(a,b,&p,&q);
printf("Sum =  %d, Sub = %d",p,q);
getch();
}
```

6.4 Passing Parameters to Functions

In order to write correct function you should know how to pass arguments to it. The C supports a wide range of mechanisms to allow you to program functions effectively.

6.4.1 Pass by Value

With this mechanism, all arguments you pass to function are copied into copy versions and your function work in that copy versions. So parameters does not affects after the function finished.

6.4.2 Pass by Pointer

In some programming contexts, you want to change arguments you pass to function. In this case, you can use pass by pointer mechanism. Remember that a pointer is a memory address of a variable. So when you pass a pointer to a function, the function makes a copy of it and changes the content of that memory address, after function finish the parameter is changed with the changes in function body.

6.4.3 Pass an Array to Function

The C allows to pass an array to a function; in this case the array is not copied. The name of array is a pointer which points to the first entry of it. And this pointer is passed to the function when you pass an array to a function.

```
#include<stdio.h>
```

```
/* functions declaration */
/* demonstrate pass by pointer */
void swap(int *x, int *y);
/* demonstrate pass by value */
void swap(int x, int y);
/* demonstrate pass an array to the function */
void bubble_sort(int a[], int size);
void print_array(int a[],int size);
void main()
{
   int x = 10;
   int y = 20;

   printf("x,y before swapping\n");
   printf("x = %d\n",x);
   printf("y = %d\n",y);
   // pass by value
   swap(x,y);
   printf("x,y after swapping using pass by value\n");
   printf("x = %d\n",x);
   printf("y = %d\n",y);
   // pass by pointer
   swap(&x,&y);
   printf("x,y after swapping using pass by pointer\n");
   printf("x = %d\n",x);
   printf("y = %d\n",y);
   // declare an array
   const int size = 5;
   int a[size] = {1,3,2,5,4};
   printf("array before sorting\n");
   print_array(a,size);
   bubble_sort(a,size);
   printf("array after sorting\n");
   print_array(a,size);
}
/* functions implementation */
void swap(int *x, int *y){
   int temp = *x;
      *x = *y;
      *y = temp;
}
void swap(int x, int y){
   int temp = x;
      x = y;
      y = temp;
}
void bubble_sort(int a[], int size)
```

```
{
  int i,j;
  for(i=0;i<(size–1);i++)
    for(j=0;j<(size– (i+1));j++)
         if(a[j] > a[j+1])
              swap(&a[j],&a[j+1]);
}
void print_array(int a[],int size)
{
   for(int i = 0;i < size; i++)
   {
      printf("%d\t",a[i]);
      printf("\n");
   }
}
```

Here is the output

```
x,y before swapping
x = 10
y = 20
x,y after swapping using pass by value
x = 10
y = 20
x,y after swapping using pass by pointer
x = 20
y = 10
array before sorting
1
3
2
5
4
array after sorting
1
2
3
4
5
```

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Define Function.

……..……

……..……...…

……..

Ex2: Define function that return multiple values.

……..……

……..……...….

……..

6.5 Arrays in Functions

Arrays can be passed down to functions like other single variables. We use them by their name as arguments when we call the function. The parameter list of the function must indicate the fact that a parameter is an array by augmenting the [ ] at the end of the one dimensional arrays. In one-dimensional arrays the actual size does not have to be given. In case of two-dimensional arrays we recommend that you give the size of the array in both dimensions.

The following example show the concept of array of function

```
#include<stdio.h>
#include<conio.h>
  void read(int *,int);
  void dis(int *,int);
  void main()
  {
   int a[5],b[5],c[5],i;
   printf("Enter the elements of first list \n");
   read(a,5);
   printf("The elements of first list are \n");
   dis(a,5);
  }
  void read(int c[],int i)
  {
   int j;
   for(j=0;j<i;j++)
    scanf("%d",&c[j]);
   fflush(stdin);
  }
  void dis(int d[],int i)
  {
   int j;
   for(j=0;j<i;j++)
   printf("%d ",d[j]);
   printf("\n");
  }
```

Passing arrays and individual array elements to functions

```
#include <stdio.h>
#define SIZE 5
void modifyArray( int b[], int size );
void modifyElement( int e );
int main()
```

```
{
  int a[ SIZE ] = { 0, 1, 2, 3, 4 };
  int i;
  printf( "Effects of passing entire array by reference:\n\nThe "       "values of the original array are:\n" );
  for ( i = 0; i < SIZE; i++ ) {
    printf( "%3d", a[ i ] );
  }
  printf( "\n" );
  modifyArray( a, SIZE );
  printf( "The values of the modified array are:\n" );
  for ( i = 0; i < SIZE; i++ )
{
    printf( "%3d", a[ i ] );
  }
  printf( "\n\n\nEffects of passing array element "       "by value:\n\nThe value of a[3] is %d\n", a[ 3 ] );
  modifyElement( a[ 3 ] );
  printf( "The value of a[ 3 ] is %d\n", a[ 3 ] );
 return 0;
}
void modifyArray( int b[], int size )
{
  int j;
  for ( j = 0; j < size; j++ ) {
    b[ j ] *= 2;
  }
}
void modifyElement( int e )
{
  printf( "Value in modifyElement is %d\n", e *= 2 );
}
```

6.6 Nesting of Functions

Inside a function you can define and use other functions. Here is an example that shows a helper function to simplify a complicated calculation.

```
var complicated = function (x) {
   var f = function (y) {
      return y * 3 + 1;
   };
   return f(f(x));
};
var y = complicated(2);
// Now y is 22
```

In the definition of a function you can refer to the arguments of the function. You can also refer to variables *outside* the function in the enclosing function. Here is the same example but with a constant1 factored out as a variable.

```
var complicated = function (x) {
```

```
    var c, f;
    c = 1; // constant
    f = function (y) {
        // We can refer to c in here!
        return y * 3 + c;
    };
    return f(f(x));
};
var y = complicated(2);
// y is still 22 here
```

That was actually the first example of a closure! The outer function closes over the inner functions, and the inner functions can use variables from the outer function.

An important case is when the inner function refers to arguments from the enclosing outer function. Arguments are like variables, you can refer to them from inside in the same way.

Here is an example of a function create that takes an argument x and returns a new function. The returned function takes no arguments and returns x.

```
var create = function (x) {
    var f = function () {
        return x; // we can refer to x here!
    };
    return f;
};
// create takes one argument, creates a function
var g = create(42);
// g is a function that takes no arguments now
var y = g();
// y is 42 here
```

In the example, even after the create function finishes the function it returns keeps working.

6.7 Recursion

Recursion is a powerful tool that can really simplify your code if you find that you have a problem that can be solved by using it. A recursive function is one that calls itself one or more times. One example of recursive functions is operations on binary trees. Binary trees are an advanced data structure, but all operations on them are considered recursive in nature. Traversing linked lists is another recursive problem, but linked lists too are an advanced topic.

Recursion is a powerful tool, but it takes a lot of careful planning so it can be difficult to implement, and many programmers will simply pass it up because of this. In order to successfully implement a recursive function, you must identify one or more exit conditions for stopping the recursive calls. If you do this wrong, your code will enter an endless loop and cause a stack overflow because of all the function calls.

Recursion is never necessary, and many never use it in practice because of the time it takes to design the problem is often longer than just coding it iteratively. In these days where agile development is king, time is everything. It is still good to know in case you do come across a problem recursive in nature, or see it in somebody else's code.

Most examples of recursion are rather contrived, and to be honest this programmer has never used it in C since learning it in college. Calculating factorials, tower of Hanoi, and the Sieve of Eratosthenes are common ones

for explaining recursion without going into too advanced of concepts. Calculating factorials is simple enough to explain the concept so we are going to write a very quick sample to do just that.
You may remember the definition of factorials from math courses, you may not. The point to take home is that the recursive function calc_factorial in calc_factorial.c calls itself until the base case is resolved and it returns 1 instead of n – 1.

```
#include <stdio.h>
int calc_factorial(int n);
int main() {
int i;
 int n_values[5] = {1, 2, 5, 3, 9};
 int factorials[5];
for (i = 0; i < 5; i++) { factorials[i] = calc_factorial(n_values[i]);
}
for (i = 0; i < 5; i++)
 {
 printf("Factorial of %d is %d", n_values[i], factorials[i]);
 printf("n");
}
}
int calc_factorial(int n) {
int n_minus_one;
 int next_n;
//Base case for exiting the recursion is a value of 1.
if (n <= 1) {
return 1;
} else
 {
//Otherwise return the next iteration's n value.
n_minus_one = n – 1;
next_n = n * calc_factorial(n_minus_one);
return next_n;
} }
```

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Define Array in function.

……..……

……..……...….

……..

Ex2: What is recursion?.

……..……

……..……...….

……..

6.8 Command Line Arguments

In C it is possible to accept command line arguments. Command-line arguments are given after the name of a program in command-line operating systems like DOS or Linux, and are passed in to the program from the operating system. To use command line arguments in your program, you must first understand the full declaration of the main function, which previously has accepted no arguments. In fact, main can actually accept two arguments: one argument is number of command line arguments, and the other argument is a full list of all of the command line arguments.

The full declaration of main looks like this:

int main (int argc, char *argv[])

The integer, argc is the argument count. It is the number of arguments passed into the program from the command line, including the name of the program.

The array of character pointers is the listing of all the arguments. argv[0] is the name of the program, or an empty string if the name is not available. After that, every element number less than argc is a command line argument. You can use each argv element just like a string, or use argv as a two dimensional array. argv[argc] is a null pointer.

How could this be used? Almost any program that wants its parameters to be set when it is executed would use this. One common use is to write a function that takes the name of a file and outputs the entire text of it onto the screen.

```
#include <stdio.h>
int main ( int argc, char *argv[] )
{
   if ( argc != 2 ) /* argc should be 2 for correct execution */
   {
      /* We print argv[0] assuming it is the program name */
      printf( “usage: %s filename”, argv[0] );
   }
   else
   {
      // We assume argv[1] is a filename to open
      FILE *file = fopen( argv[1], “r” );
      /* fopen returns 0, the NULL pointer, on failure */
      if ( file == 0 )
      {
         printf( “Could not open file\n” );
      }
      else
      {
         int x;
/* read one character at a time from file, stopping at EOF, which indicates the end of the file.  Note that the idiom of “assign  to a variable, check the value” used below works because the assignment statement evaluates to the value assigned. */
         while  ( ( x = fgetc( file ) ) != EOF )
         {
            printf( “%c”, x );
         }
         fclose( file );
```

```
    }
  }
}
```

This program is fairly short, but it incorporates the full version of main and even performs a useful function. It first checks to ensure the user added the second argument, theoretically a file name. The program then checks to see if the file is valid by trying to open it. This is a standard operation, and if it results in the file being opened, then the return value of fopen will be a valid FILE*; otherwise, it will be 0, the NULL pointer. After that, we just execute a loop to print out one character at a time from the file.

6.9 Storage Classes

A variable has two specifiers, namely data type and storage class.

- data type (e.g. int, float, char, etc.)
- storage class (e.g. auto, static, etc.)

Data type specifies the types of data stored in a variable. Storage class specifies the segments of the program where the variable is recognized, and how long the storage of the value of the variable will last.

There are four types of storage class specifications as given below:

- automatic
- static
- extern
- register

We have so far been defining only the data type of the variables, but not the storage class. You may wonder then, how our programs worked! The programmer has to specify the type when it is required to operate in a particular manner. If the storage class is not specified, the compiler will assume the type on its own. The storage class is applicable to all types of variables and is prefixed to the data type declaration as given below:

```
auto char z;
extern int a, b, c;
static float x ;
register char y;
```

The basic characteristics of each storage class are discussed in the following sections:

6.9.1Automatic Variables

(i) ***Storage location****:* Except for register variables, the other three types will be stored in memory.

(ii) ***Scope***: Auto variables are declared within a function and are local to the function. This means the value will not be available in other functions.

Auto variables defined in different functions will be independent of each other, even if they have the same name

Auto variables are local to the block in a function. If an auto variable is defined on top of the function after the opening brace, then it is available for the entire function. If it is defined later in a block after another opening brace, it will be valid only till the end of the block i.e., up to the corresponding closing brace.

The following program illustrates the concept.

```
/*Example*/
/* to demonstrate use of auto variable*/
#include <stdio.h>
main ( )
```

```
{
        auto int x=10 ;
        void f1 (int x) ;
        int f2 (int x) ;
        {
                auto int x =20;
                printf (“x = %d in the first block\n”, x) ;
                x=f2 (x) ;/*20 is passed to f2 and returned value assigned to x*/
                printf (“x = %d after the return from f2\n”, x);
        }
        printf (“x = %d after the first block\n”, x) ;
        {
                auto int x=30;
                printf (“x = %d in the second block\n”, x) ;
        }
        printf (“x = %d after the second block\n”, x) ;
        f1 (x) ; /*x=1O is passed to the function f1*/
        printf (“x = %d after return from function will be 10\n”, x) ;
        }
        void f1 (int a)
        {
                auto float x=5.555;/*integer x wil1 be lost*/
                printf (“x = %f in the function\n”, x) ;
        }
        int f2 (int x)
        {
                auto int y=100;
                y+=x; /*y will be 120* /
                printf (“y = %d in the function\n”, y) ;
                return y;
        }
}
```

Execute the program and you will get the following results:

```
x = 20 in the first block
y = 120 in the function
x = 120 after the return from f2
x = 10 after the first block
x = 30 in the second block
x = 10 after the second block
x = 5.555000 in the function
x = 10 after return from function will be 10
```

This gives a clear idea about the scope of auto variables.

(iii) ***Initial values****:* The auto variable will contain some garbage values unless initialized. Therefore they must be initialized before use.

(iv) ***Life:*** How long will the values stored in the auto variable last?

It will last as long as the function or block in which it is defined is active. If the entire function has been executed and the value has been returned, then the value of the auto variables of the functions will be lost. We cannot call it later. This point should be noted.

6.9.2 Register Variables

Register variables have similar characteristics as auto variables. The only difference between them is that while auto variables are stored in: memory, registered variables are stored in the register of the CPU. The variables are local to the block and they will be available as long as the blocks are active.

Why then do we need to declare one more storage class? The CPU registers respond much faster than the memory. After all we want to access, store and retrieve the stored variables faster so that the computing time is reduced. Registers are faster than the memory. Therefore those variables which are used frequently can be declared as register variables. They are declared as

```
register int i;
```

A memory's basic unit may be 1 byte, but depending on the size of the variable even 10 contiguous bytes of memory can be used to store a long double variable. Such an extension of size is not however possible, in the case of registers. The registers are of fixed length like 2 bytes or 4 bytes, and therefore, only integer or char type variables can be stored as register variables. Since registers have many other tasks to do, register variables may be defined sparingly. If a register variable is declared and if it is not possible to accept it as a register variable for whatever reasons, the computer will treat it as an auto variable. Therefore the programmer may specify a frequently used variable in program as a register variable in order to speed up the execution of the program.

Did You Know?

In the register type variables initial value will be an unpredictable or garbage value.

Caution

Be carefull before using register variables in program, it may create register memory problem.

6.9.3 External Variables

External variables are also known as global variables. We can create functions in more than one file. However, for the sake of simplicity, we will assume that all the functions are in one file. The global variables will be declared like other variables with a storage class specifier extern.

```
Eg.   extern int a, b
      extern float c, d
```

The scope of the variables starts from the point of declaration to the end of the program. The value of the external variable at any point of time is that of the last assignment. Assume that the main function may assign a = 10. The function z may then use it and perform a calculation and at the end assign a value 20. If printed at that point of time, the value will be 20, It may be called by another function p where its value may become zero. If at this point of time the main function calls or z calls it, the value will be 0. Thus the external variable is accessible and transparent to all the functions below it. We will write a program to demonstrate this concept.

```
/*Example* /
/* to demonstrate use of external variable*,/
#include <stdio.h>
extern int ext_a=10;
main ()
{
```

```
        int f1(int a);
        void f2 (int a);
        void f3 ();
        printf ("ext_a = %d in the main function\n", ext_a);
        f 1 (ext_a);
        printf ("ext_a = %d after the return from f1\n", ext_a);
        f2 (ext_a);
        printf ("ext_a = %d after the return from f2\n", ext_a);
        ext_a* =ext_a;
        printf ("ext_ a = %d \n", ext_a);
        f3 ();
        printf("ext_a = %d after return from f3\n", ext_a);
}
int f1 (rnt x)
{
        ext_a–=10;
        return ext_a;
}
void f2 (int x)
{
        ext._a+=20 ,
}
void f3 ()
{
        ext_a/100;
}
```

Result of the program

ext_a = 10 in the main function

ext_a = 0 after the return from f1

ext_a = 20 after the return from f2

ext_a = 400

ext_a = 4 after return from f3

How does the program work?

ext_a is declared as an external variable with value 10 before main(). Therefore, ext_a will be recognized all through the program .

fl, f2 & f3 are functions.

f1 returns an integer and f2 returns void. i.e. it does not return a value.

f3 neither receives nor returns any value.

In the first printf, we get ext_a = 10.

Now f1 is called. In f1 ext_a = 10 is passed as an argument.

The value of ext_a is 0 now in function f1. The second printf in main prints ext_a = 0

Now f2 is called.

ext_a becomes 20 now. It does not return any value. However, the third printf prints the value as 20. How does this happen? It is because the current value of ext_a is known to main even without f2 passing it.

We discussed that a function can return only one value. However, by using a global variable, we can overcome this limitation as we have done here.

Then we square ext_a i.e., ext_a = 400 now.

The 4th print statement confirms this. Now we call f3. In spite of the fact that we neither passed an argument nor returned any value from f3, ext_a is known to f3 as 400. Then 100 divide ext_a. Therefore ext_a will be 4 as confirmed by the fifth print statement.

This program illustrates the concept of external variables in simpler situations where the name of the global variable is not assigned to the function's local variables.

It is perfectly legal to use the same name for different local variables in a function. We can even use the name and declare it as another data type.

For instance, we can define ext_a as a float in another function f11. Then how is the conflict to be resolved? We will reserve the answer to the question for a few minutes

One should be careful while handling external variables because the variables may be disturbed in a remote corner inadvertently. Global variables when declared on top of main()can be identified easily, and therefore the storage class specifier extern need not be specified in such situations. If it cannot be easily recognized by declaration elsewhere in the program, it should be specified clearly.

The life of the variable is till the termination of program execution. The scope extends from the point of declaration till the end. It will be stored in memory.

Did You Know?

The initial value of an external variable is zero if not assigned.

6.9.4 Static Variables

The initial value of static variables is zero unless otherwise specified. This is also stored in memory. Static variables are declared as follows:

```
static int x, y, z;
static char a; etc.
```

Static variables are local to the functions and exist till the termination of the program. Therefore, when the program leaves the function, the value of the static variable is not lost. If the program calls the function again, the static variable will execute the function with the value it already possesses. Assume that f1 is a function containing a static variable as given below:

```
main ()
    {
        int f1 (–);
        fl (–);
    }
    int f1 (–)
    {
        static int var = 0;
    }
```

When f1 is called the first time, *var* will be initialized to zero. If *var* is finally assigned the value 10 at the end of f1, then *var* = 10 will remain till the program stops execution, and if main calls f1 again, the value of *var* will not be initialized to 0 again, but will remain as10. The initialization *var* = 0 will not have any effect. However *var* can further be modified depending on the statements in f1. Had it been an auto variable, *var* would have been initialized each time

Example: The program presented in figure converts the given temperature in Fahrenheit to Celsius using the following conversion formula: c= F –32/1.8

Program

```
/*****************************************************************/
/* FAHRENHEIT CELSIUS CONVERSION TABLE */
/*****************************************************************/
#define F_LOW 0 /*****************************/
#define F_MAX 250 /* SYMBOLIC CONSTANTS */
#define STEP 25 /*******************************/main()
{
typedef float REAL ; /* TYPE DEFINITION */
REAL Fahrenheit,Celsius
/* DECLARATION */
Fahrenheit = F_LOW; /
* INITIALIZATION*/
printf("Fahrenheit Celsius\n\n");
while(Fahrenheit <= F_MAX )
{ Celsius = ( Fahrenheit 32.0)/1.8;
printf(" %5.1f %7.2f\n", Fahrenheit, Celsius);
Fahrenheit = Fahrenheit + STEP;
}
}
```

Output

Fahrenheit	Celsius
0.0–	17.78
25.0–	3.89
50.0	10.00
75.0	23.89
100.0	37.78
125.0	51.67
150.0	65.56
175.0	79.44
200.0	93.33
225.0	107.22
250.0	121.11

The program prints a conversion table for reading temperature in Celsius, given the Fahrenheit values. The minimum and maximum, values and step size defined as symbolic constants.
These values can be changed by redefining the #define statements. A user-defined data type name REAL is used to declare the variables Fahrenheit and Celsius.
The formation specifications %5.l f and % 7.2 in the second printf statement produces two column output as shown.

Self Assessment Questions

1. Header files includes stdio, conio, string, etc.
(a). True (b) False

2. The *reverse* function reverses the number and sends it back to the ……. function.
(a) reverse (b) main

(c) add (d) None of above.

3. A function declaration is also called a ……
(a) function prototype (b).function callin
(c).return (d) None of these.

4. The arguments declared as part of the prototype are also known as ……. parameters.
(a) informal (b) formal
(c) multiple (d) single

5. Function definition consists of two parts i.e., function declarator and ……
(a). Function prototype (b). function body
(c). Function calling (d). None of these.

6. Data type specifies the types of data stored in a arrays.
(a) False (b)True

7. Registers are …….. than the memory.
(a) slower (b) faster
(c) equal (d) None of these

6.10 Summary

- In C functions classified into two categories, namely, library functions and user-defined functions.
- Elements of user-defined functions as a function declaration function definition function call.
- A function declaration is called a function prototype.
- A recursive function is one that calls itself one or more times.
- Command-line arguments are given after the name of a program in command-line operating systems like DOS or Linux, and are passed in to the program from the operating system.

6.11 Keywords

Data type: Data type specifies the types of data stored in a variable.
Formal parameters: The arguments declared as part of the prototype are also known as formal parameters.
Function declarator: The function declarator is a replica of the function declaration.
Register variables: *Register variable* are a special case of automatic variables. Automatic variables are allocated storage in the memory of the computer.
***Static variables*:** Static variables are local to the functions and exist till the termination of the program.

6.12 Review Questions

1. Write a program to pass the radius of a circle to a function and calculate the area and perimeter and print them in the called function.
2. Write a program to pass a string along with the position number to a function and return the string deleted up to the position.
3. Describe the functions.
4. Explain the “Calling by value”.

5. Describe the following:

If func stands for a function name

i) x = func(p)

ii) float func (int x);

iii) if (func (p)) y ++;

iv) if (func (d * d) ==100) printf (“None”);

v) if (a==b) func (b);

6. What is the difference between formal and actual parameters?

7. Determine the output of the following program:

```
main( )
{
int x = 10;
int y = 20;
int p,q;
p = prod(x,y);
q = prod (p,prod(x,2));
printf(“%d %d\n”, p,q);
}
prod(a,b)
int a,b;
{
return (a*b);
}
```

8. Write a function that will generate and print the first *n* Fibonacci numbers.

9. Distinguish between the following:

a) Global and local variables

b) Automatic and static variables

10. Which of the following function headers is invalid? And why?

a) Average (x,y,z);

b) Power (a, n–1)

c) product (m, 10)

d) double minimum (float a; float b;)

Answers for Self Assessment Questions

1. (a) 2. (b) 3. (a) 4. (b) 5. (b) 6. (a) 7. (b)

6.13 Further Studies

1. Let us c by Yashwant Kanatkar.
2. http://www.wavemetrics.net/doc/igorman/IV-03%20User%20Functions.pdf

7
Pointers

STRUCTURE

7.0 Objectives

7.1 Introduction

7.2 Pointer Concepts

7.3 Pointer Variable

7.4 Accessing variables through pointers

7.5 Pointer Declaration and Initializing

7.6 Pointers on Pointer

7.7 Compatibility and Application of Pointers

7.8 Memory Allocation Functions and Mapping

7.9 Memory Management Functions

7.10 Summary

7.11 Keywords

7.12 Review Questions

7.13 Further Studies

7.0 Objectives

After studying this chapter, you will be able to:

- Discuss the concepts of pointer
- Explain about the pointer variables
- Understand how to declare and initialize the pointers
- Discuss pointers on pointer
- Explain the compatibility and application of pointers
- Discuss the memory allocation functions and memory mapping
- Explain the memory management functions

7.1 Introduction

Pointers are widely used in programming; they are used to refer to memory location of another variable without using variable identifier itself. They are mainly used in linked lists and call by reference functions. Figure 7.1 illustrates the concept of pointers. As you can see here; Yptr is pointing to memory address 100.

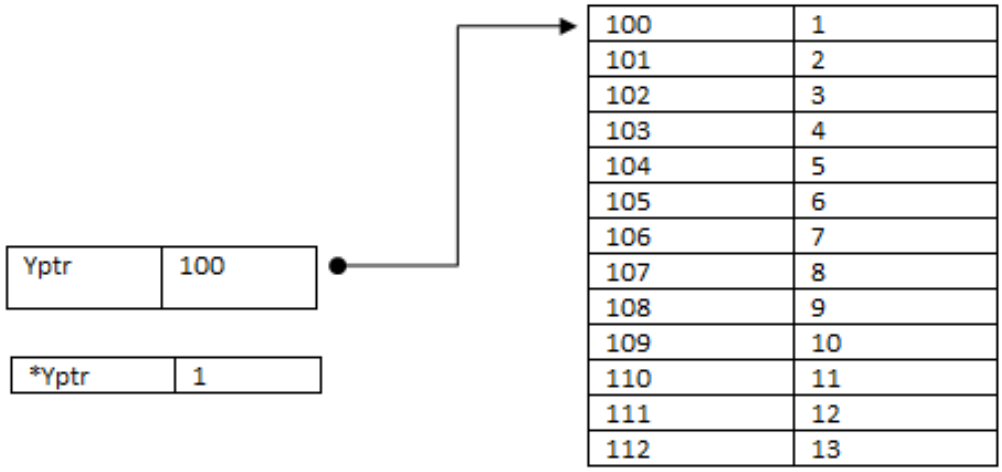


Figure 7.1: Concept of pointers.

7.2 Pointer Concepts

The pointer is a powerful concept of C. It is powerful for two reasons:

(a) It helps in achieving results, which could not otherwise be achieved such as an indirect method for returning more than one value from a function.

(b) It results in a compact code.

Pointers are closely associated with memory addressing. We know that variables are stored in memory, and that each location in memory has an address just as a person has an address. The memory locations are available in groups of 8 bits or a byte. Each byte in memory has an address. Therefore each location has an address and stores a value. The value stored can correspond to any data type such as float or char or int and their type modifiers. However, as you know, all these data types are stored in terms of 1s and 0s. We will not go into further details of storage except to state that each memory location stores a value and has an address. The memory locations are arranged in increasing order of addresses, starting from 0000, and increases one by one. We may have the address of the last location as FFFF. What does F denote? The addresses are denoted in terms of hexadecimal numbers. You can calculate the decimal equivalent of this and find the decimal address of the last location. The storage capacity varies from machine to machine.

The & operator has been used to denote the address of the variable in the scanf function. If var is the name of a datatype, &var denotes the address of the location where it is stored. Whenever a data type is declared, a memory location is allocated depending on the data type. The allocation takes place at the time of execution of the program, and therefore the address of a variable may be different at different times of execution, since the computer allots it at random depending upon the availability of a memory location. However, memory locations for each data type will be contiguous so as to make for easy handling. For instance a double needs 8 bytes of storage and therefore all the 8 bytes will be stored continuously or contiguously. Each byte of the value of the variable will be stored one after the other in continuous locations.

For, instance, this is declared as:

float ft=100.52;

Here ft is the name of the variable. Its value is 100.52. This number will be stored in 4 bytes in the memory in sequentially numbered storage locations. When you print the value of the address of a variable, it will always print the starting address. If the starting address and the data type is known it is easy to find out the locations

which are occupied by the variable. For instance, if the starting address of the float ft is 0011, then the variable will occupy upto memory location 0014. However, if you print the address of ft, i.e. &ft, you will get 0011.
A similar concept can be extended to a string. A string is an array of characters, with each character occupying a byte of memory. If Peter is stored in string w, and if w[0] i.e., p is stored in location 0020, then 'e' will be stored in 0021, 't' in 0022 and so on. Every element of an array will also be stored contiguously. The formula for finding out the location of an element or address of pth element in an array is given below:
Address of pth element = starting address of array + p * (storage space for the data type)
Example:
If an integer array *ia*. is stored from location 992 onwards, find out the location of the 10th element.
Address of 10th element = 992 + 10 * 2 = 1012. It is the starting point. However, the second byte of the integer will be stored in location 1013.

Note: The 10th element will have the subscript 9, since the 1st element has the subscript 0.

Example: If the 10th element of a long double is stored from location 2000 onwards, find the location of the 15th element and the first element.
The 1st element will be stored at location 2000 – 10 * 10=1900.
The 15th element will be stored at 1900 + 15 * 10=2050.
Let us now consider a pointer to an integer. Let the integer be mark. Then the address will be denoted as &mark. Note that all addresses will be in integers for all data types. In the case of pointers, the address of mark will be stored in another location. The pointer is a variable that contains the address of the variable. We can assign the address of the integer to an integer pointer. Usually we declare: int mark; mark =75; we can also declare int*ip; This means ip is a pointer to the integer. We can assign ip = &mark; i.e. we have assigned the address of mark to ip. Let us pictorially explain this.

1011	75
1030	1011

Here mark = 75 and the address of mark is 1011. Therefore *ip* = 1011. This value will also be stored at another location 1030. Here *ip* points to an integer mark, and holds the address of mark. Since the pointer is also a variable, it will be stored in another location. The * is called the indirection or dereferencing operator.
Similarly we can write
float f = 101.23;
float * fp;
fp = &f;
Here fp points to a float because we have assigned the address of f to fp. Remember that the pointer can point to any type of variable such as a float or char or int or string. Pointers themselves are always of type int because it is the value of the address.
It is necessary to become familiar with pointers. Therefore let us apply the concepts learnt.
We can have the definitions of the following types:
int i=204;
float f= 101. 23;
int * ip ; / * ip is a pointer to integer */
float * fp ; /* fp is a pointer to float */
This is carried out as follows:

ip = &i;
fP = &f;
By assigning *ip* to the address of i, *ip* points to integer i. Similarly, *fp* points to float *f*, suppose we now assign:
i = 100;
ip automatically points to 100.
Similarly if we assign
f = 100.05; then fp points to the new value. What actually happens? The variables i & f are assigned storage locations; ip holds the address of where i is stored fp holds the address of f. When we assign new values to i and f, the values stored in ip and fp are not affected. They continue to point to i and f, but the values of i and f have been actually changed.
If we now add the following assignment statements
int a [5];
ip = &a[0]
We have defined an array of integer a with 5 elements. When we assign the address of a [0] i.e., the 0th element of a to ip, ip will point to the array. The old assignment to ip is lost. It is irrecoverable.
We can also perform arithmetic operations on pointer variables, such as:
ip = ip + 5; /*pointer moved up by 5 locations*/
ip = ip–10; /* ip moved down by 10 locations/
ip– –; /*decremented*/
ip++; /*pointer incremented*/
*ip++; /*value incremented*/
*ip– –; /*value decremented*/
However, such operations on pointers are limited. We cannot carry out the following operations on pointers:
ip+fp; /*invalid*/
ip*fp; /*invalid*/
ip*2; /*invalid*/
fp/10; /*invalid*/
ip = rp*10; /*invalid*/

If we say ip = fp; then both fp and ip will point to the same location, and hence fp will point to the same variable pointed to by ip.

Caution
We must assign the pointers to the specific integers and floats; otherwise they will not point to value.

7.3 Pointer Variable

A pointer is a variable that contains the address of another variable. As you know, any variable has the following four properties:
a) name
b) value
c) address
d) data type

For instance, consider the following declaration of a simple integer.
int var = l0;

The name here is var, and its value is 10. Its address is not declared here since we want to give flexibility to the compiler to store it wherever it wants. If we specify an address, then the compiler must store the value at the same address. Specifying actual address is carried out during machine language programming. However this is not required in High-Level Language (HLL) programming and by printing the value of &var, we can find out the address of the variable. When the statement to find the address is executed at different times, different addresses will be printed. What is important is that the compiler allocates an address at run time for each variable and retains this till program execution is completed. This is not strictly so, in the case of auto variables. At this point you may also recall that in the case of function declarations the calling function, the compiler does not allocate memory to the variables in the declaration That is the reason why parameters in the declaration part are not recognized in the calling function. It is only a prototype.

The fourth feature of a variable is its data type. In the example, var is an integer. A pointer has all the four properties of variables. However, the data type of every pointer is always an integer because it is the value of the memory address. Memory addresses are integers. They cannot be floats or any other data types. They may point to an integer or a float or a character or a function etc. They have a name. They have a value. For instance the following is a valid declaration of a pointer to an integer.

```
int * ip;
```

Here ip is the name of a pointer. It points to or it contains the address of an integer, which is the value. It will also be stored in, another location in memory like any other variables.

Did You Know?

The compiler forgets the address of a variable when the program comes out of the block in which the variable is declared.

7.4 Accessing variables through pointers

Once a pointer has been assigned the address of a variable, the question remains as to how to access the value of the variable using the pointer. This is done by using another unary operator *(asterisk), usually known as the indirection operator. Consider the following statements:

```
int qty, *q,m;
qty=165;
q=&qty;
m=*p;
```

The first line declares qty and m as integer variables and q as a pointer variable pointing an integer. The second line assigns the value 165 to qty and the third line assigns the address of qty to the pointer variable q. The fourth line contains the indirection operator *. When the operator * is placed before a pointer variable in an expression, the pointer will return the value of the variable. The * can be called as ‘value at address’. Thus the value of q would be 165. The two statements

```
q=&qty;
m=*p;
```

are equivalent to

```
m=qty.
```

Program for accessing variables using pointers

```
main()

{
      int m,n;
      int *ptr;

      m=15;
      ptr=&m;
      y=*ptr;

      printf("Value of m is %d\n\n", m);
      printf("%d is stored at address %u \n", m,&m);
      printf("%d  is stored at address  %u \n", &m, &m);
      printf("%d is stored at address  %u \n", ptr, &ptr);
      printf("%d is stored at address  %u \n", y, &y);
       *ptr=30;
      printf("\n Now m=%d\n",x);
}
```

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: Define.pointer

……..……

……..……...…

……..

7.5 Pointer Declaration and Initializing

A pointer variable contains the memory location of another variable. You begin the declaration of a pointer by specifying the type of data stored in the location identified by the pointer. The asterisk tells the compiler that you are creating a pointer variable. Finally you give the name of the pointer variable. The pointer declaration syntax is as shown here
type * variable name

Example:
int *ptr; float *string;

7.5.1 Address Operator

Once we declare a pointer variable, we point the variable to another variable. We can do this by assigning the address of the variable to the pointer as shown here:
ptr=#
The declaration places the memory address of num variable into the pointer variable ptr. If num is stored in memory 21260 addresses then the pointer variable ptr will contain the memory address value 21260.
/* A program to illustrate pointer declaration */
main()

```
{
 int *ptr;
int sum;
sum=45;
ptr=&sum;
printf (“n sum variable value is %dn”, sum);
printf (“n The ‘ptr’ pointer variable value is %d”, ptr);
}
```

We will get the same result by assigning the address of num to a regular (non pointer) variable. The benefit is that we can also refer to the pointer variable as *ptr the asterisk tells to the computer that we are not interested in the value 21260 but in the value stored in that memory location. While the value of pointer is 21260 the value of sum is 45 however we can assign a value to the pointer * ptr as in *ptr=45.
When we place the value 45 in the memory address pointer by the variable ptr. Since the pointer contains the address 21260 the value 45 is placed in that memory location. And since this is the location of the variable num the value also becomes 45. this shows how we can change the value of pointer directly using a pointer and the indirection pointer.

```
/* Program to display the contents of the variable their address using pointer variable*/
include <stdio.h>
{
int num, *intptr;
float x, *floptr;
char ch, *cptr;
num=123;
x=12.34;
ch=‘a’;
intptr=&x;
cptr=&ch;
floptr=&x;
printf(“Num %d stored at address %un”,*intptr,intptr);
printf(“Value %f stored at address %un”,*floptr,floptr);
printf(“Character %c stored at address %un”,*cptr,cptr);
}
```

7.5.2 Pointer Expressions and Pointer Arithmetic

Like any other variable, pointer variable can be used in arithmetic expressions. For example if p1 and p2 are properly declared and initialized pointers, then the following statements are valid.

```
y = *p1 * *p2;
sum = sum + *p1;
z = 5* – *p2/p1;
*p2 = *p2 + 10;
```

The C language allows us to add integers to, subtract integers from pointers as well as to subtract one pointer from the other. We can also use short hand operators with the pointers p1+=; sum+=*p2; etc., we can also compare pointers by using relational operators the expressions such as p1 > p2 , p1==p2 and p1!=p2 are allowed.

```
/*Program to illustrate the pointer expression and pointer arithmetic*/
#include <stdio.h>
```

```
main()
{
int ptr1,ptr2;
int a,b,x,y,z;
a=30;b=6;
ptr1=&a;
ptr2=&b;
x=*ptr1+ *ptr2 –6;
y=6*– *ptr1/ *ptr2 +30;
printf(“nAddress of a +%u”,ptr1);
printf(“nAddress of b %u”,ptr2);
printf(“na=%d, b=%d”,a,b);
printf(“nx=%d,y=%d”,x,y);
ptr1=ptr1 + 70;
ptr2= ptr2;
printf(“na=%d, b=%d”,a,b);
```

For your convenience the program involving all these statements and the output is given as.

```
/*program* /
/* pointers */
#include <stdio.h>
main ( )
{
        int * var;
        int a =100;
        var = &a;
        printf (“value of * var=%d\n”, *var);
        printf (“va1ue of (*var) ++ =%d\n”, (*var)++);
         printf (“va1ue of * var=%d\n”, *var);
         printf (“address var = %d\n”, var);
         printf (“value of *var ++ = %d\n”, *var++);
         printf (“value of * var = %d\n”, *var);
         printf (“address var = %d\n”, var);
         printf (“original address var again = %d\n”, –var); /*original address restored*/
         printf(“va1ue of * var = %d\n”, *var);
         printf (“value of ++ ( *var ) =%d\n”, ++ ( *var ) );
         printf (“address var = %d\n”, var);
         printf (“address var++ = %d\n”, var++);
         printf (“ address var=%d\n, var ) ,.
          Var– –;
          printf (“address var after decrementing = %d\n”, var);
          printf (“ address ++var = %d\”, – – var);
          printf (“ address”, – – var);
          printf (“ address var = %d\n”, var);
          printf (“ value of * var = %d\n”, *var);
          printf (“value of * (var ++) = %d\n”, * (var ++));
          printf (“ value of * (– – var) = % d\n”, * (– – var));
          printf (“address var = %d\n”, var);
```

```
    printf ("value of * var = % d\", *var);
}
```

Output

```
value of * var=100
value of (*var) ++ = 100
value of * var=101
address var = 9106
value of *va++ = 101
value of * var = 9108
address var = 9108
original address var again = 9106
value of * var = 101
value of ++ (*var ) = l02
address var = 9106
address var++ = 9106
address var = 9108
address var after decrementing = 9106
address ++var = 9108
address – –var = 9106
address var = 9106
value of * var = l02
value of * (var++) = 102
value of * (– –var) = 102
address var = 9106
value of * var = 102
```

Caution

In the pointer concept do not try to perform mathematical operations such as division, multiplication, and modulus on pointers. Because adding (incrementing) and subtracting (differencing) pointers are only acceptable.

Self Assessment Questions

1. The pointer is a powerful concept of …….

(a) C++ (b) Java

(c) C (d) Visual Basic

2. Memory locations are available in groups of 1 mega bytes.

(a) True (c) False

3. A string is an array of characters, with each character occupying a……...

(a) number (b) character

(c) integer (d) None of these.

4. Variable has three properties name, value and data types.

(a) False (b) True

5. The pointer declaration syntax is……..

(a). type&&variable name (b) variable*
(c). type * variable name (d). None of these

7.6 Pointers on Pointer

While pointers provide enormous power and flexibility to the programmers, they may use cause manufactures if it not properly handled. Consider the following precautions using pointers to prevent errors. We should make sure that we know where each pointer is pointing in a program (Figure 7.2). Here are some general observations and common errors that might be useful to remember.

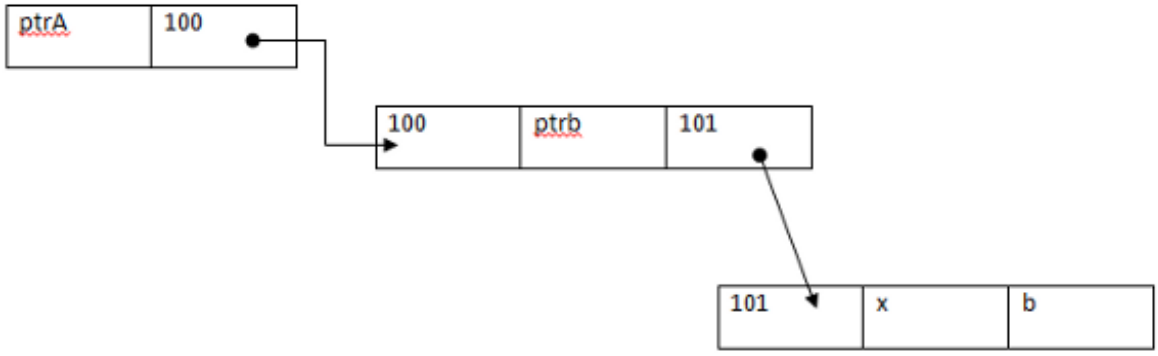


Figure 7.2: Pointers on pointer.

The abundance of C operators is another cause of confusion that leads to errors. The expressions shown as: *ptr++, *p[],(ptr).member should be carefully used. A proper understanding of the precedence and associative rules should be carefully used.

Did You Know?
A pointer contains garbage until it is initialized.

7.7 Compatibility and Application of Pointers

7.7.1 Compatibility

Two pointer types with the same type qualifiers are compatible if they point to objects of compatible types. The composite type for two compatible pointer types is the similarly qualified pointer to the composite type.
The example shows compatible declarations for the assignment operation:

```
float subtotal;
float * sub_ptr;
/* ... */
sub_ptr = &subtotal;
printf("The subtotal is %f\n", *sub_ptr);
```

The next example shows incompatible declarations for the assignment operation:

```
double league;
int * minor;
/* ... */
minor = &league;     /* error */
```

7.7.2 Application of Pointers

Accessing data by their address. A good example is parameter argv of function main.

1. Easy access
2. To return more than one value from a function.
3. To pass as arguments to functions. For e.g. consider the following structure

```
struct student
{
char name[10];
int rollno;
};
```

If you pass this structure object as argument to function then, 14 bytes (10+4) of memory will be passed to the function. Instead, if you pass the pointer to the structure as argument then only 4 bytes (or 8 bytes) of memory will be passed to the function.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Define application of pointer

……..……

……..……...…

……..

Ex2: What is pointers on pointer?.

……..……

……..……...…

……..

7.8 Memory Allocation Functions and Mapping

The malloc, calloc, or realloc are three functions used to manipulate memory. These commonly used functions are available through the stdlib library so you must include this library in order to use them.

#include stdlib.h

After including the stdlib library you can use the malloc, calloc, or realloc functions to manipulate chunks of memory for your variables.

7.8.1 Dynamic Memory Allocation Process

When a program executes, the operating system gives it a stack and a heap to work with. The stack is where global variables, static variables, and functions and their locally defined variables reside. The heap is a free section for the program to use for allocating memory at runtime.

Allocating a Block of Memory

Use the malloc function to allocate a block of memory for a variable. If there is not enough memory available, malloc will return NULL.

The prototype for malloc is:

```
void *malloc(size_t size);
```

Do not worry about the size of your variable, there is a nice and convenient function that will find it for you, called sizeof. Most calls to malloc will look like the following example:

```
ptr = (struct mystruct*)malloc(sizeof(struct mystruct));
```

In this way you can get memory for your structure variable without having to know exactly how much to allocate for all its members as well.

Allocating Multiple Blocks of Memory

You can also ask for multiple blocks of memory with the calloc function:

void *calloc(size_t num, size_t size);

If you want to allocate a block for a 10 char array, you can do this:

char *ptr;

ptr = (char *)calloc(10, sizeof(char));

The above code will give you a chunk of memory the size of 10 chars, and the ptr variable would be pointing to the beginning of the memory chunk. If the call fails, ptr would be NULL.

Releasing the Used Space

All calls to the memory allocating functions discussed here need to have the memory explicitly freed when no longer in use to prevent memory leaks. Just remember that for every call to an *alloc function you must have a corresponding call to free.

The function call to explicitly free the memory is very simple and is written as shown here:

free (ptr);

Just pass this function the pointer to the variable you want to free and you are done.

To Alter the Size of Allocated Memory, Let us get to that third memory allocation function, realloc.

void *realloc(void *ptr, size_t size);

Pass this function the pointer to the memory you want to resize and the new size you want to resize the allocated memory for the variable you want to resize.

Here is a simple and trivial example to give you a quick idea of how you might see calloc and realloc in action. You will have many chances for malloc viewing as it is the most popular of the three by far.

```
#include <stdio.h>
#include <stdlib.h>
void main() {
char *ptr, *retval;
ptr = (char *)calloc(10, sizeof(char));
if (ptr == NULL)
printf("calloc failed\n");
else
printf("calloc successful\n");
retval = realloc(ptr, 5);
if (retval == NULL)
printf("realloc failed\n");
else
printf("realloc successful\n");
free(ptr);
free(retval);
}
```

First we declared two pointers and allocated a block of memory the size of 10 chars for ptr using the calloc function. The second pointer retval is used for getting the return value from the call to realloc. Then we reallocate the size of ptr to 5 chars instead of 10. After we check whether all went well with that call, we free up both pointers.

You can play around with the values of size passed to either of the memory allocation functions to see how big a chunk you can ask for before it fails on you. Do not worry, your operating system has the ability to keep your program in check, you will not hurt it this way.

7.9 Memory Management Functions

Memory management functions handle the allocation and deallocation of dynamic memory. These functions form an abstraction layer above the standard C memory management functions malloc, free, and realloc. This block of functions can be replaced by the user with custom code to implement a different memory management scheme. For example, an embedded system application might want to use a fixed-sized static block from which to allocate.

The built-in memory management logic implements a nibble-allocation memory management algorithm that provides superior performance to calling malloc and free directly. This algorithm causes memory blocks to be allocated up front in larger sizes and then subsequently split up when future allocation requests are received. These blocks can be reset and reused in applications that are constantly allocating and freeing memory.

The key memory management function that a user might use is the following:

- **rtxMemAlloc**: This function allocates a block of memory in much the same way malloc would. The only difference from the user's perspective is that a pointer to a context structure is required as an argument. The allocated memory is tracked within this context.
- **rtxMemFreePtr**: This function releases the memory held by a pointer in much the same way the C free function would. The only difference from a user's perspective is that a pointer to a context structure is required as an argument. This context must have been used in the call to rtxMemAlloc at the time the memory was allocated.
- **rtxMemFree:** This function releases all memory held within a context.
- **rtxMemReset:** This functions resets all memory held within a context. The difference between this and the rtxMemFree function is that this function does not actually free the blocks that were previously allocated. It only resets the pointers and indexes within those blocks to allow the memory to be reused.
- **rtxMemRealloc:** This function works in the same way as the C realloc function. It reallocates an existing block of memory. As in the other cases above, a pointer to a context structure is a required argument.

Note that these memory management functions are only used in the generation of C code, not C++ (although a user can use them in a C++ application). For C++, the built-in new and delete operators are used to ensure constructors and destructors are properly executed.

Example: To Print a Substring

Let us write a function to print a substring of given length and starting position. This is illustrated in example

```
/* example gets substring beginning with
specified character position */
#include <stdio.h>
#include <string.h>
void substring(char *str, char *substr, int len) ;
void main ()
{
```

```
char text [80], substrL2O);
int len,pos;
printf (“Enter any Text :”);
gets (text),.
printf(“Enter the Length of Substring Required :”);
scanf (“%d”, &len);
printf (“Enter the position from which Required :”);
scanf (“ %d”,&pos );
substring (text+(pos–1), substr, len),.
printf (“.Substring Is %s\n” substr),.
}
void substring (char *str, char *substr, int .len)
{
int cnt = 0;
while (*str &&cnt <len)
{
* (substr++) =* (str++);
cnt++
}
* (substrl) = 0;
}
```

Result of program

Enter any Text: This is a program to get a substring
Enter the Length of Substring Required: 7
Enter the Position From which Required: 11
Substring is program.

7.10 Summary

- The memory locations are arranged in increasing order of addresses, starting from 0000, and increases one by one.
- The & operator used to denote the address of the variable in the scanf function.
- The & operator has four properties name, value, address and data type.
- The declaration of a pointer by specifying the type of data stored in the location identified by the pointer.
- The compiler allocates a base address and sufficient amount of storage to contain all the elements of the array in contiguous memory locations.
- Pointers provide enormous power and flexibility to the programmers, they use cause manufactures if it not properly handled.
- String is an array of characters, terminated with a null character. A void pointer is a C convention for a raw address.

7.11 Keywords

Array: An array is a variable that holds multiple values of the same type.

Character string: A series of characters manipulated as a group. A character string differs from a name in that it does not represent anything a name stands for some other object.

Compiler: A compiler is a computer program that transforms human readable source code of another computer program into the machine readable code that a CPU can execute.
Pointer: A pointer is a variable that contains the address of another variable.
Ragged arrays: The character arrays with the rows of varying length are called ragged arrays.
Void pointer: A void pointer is used for working with raw memory or for passing a pointer to an unspecified type.

7.12 Further Studies

1. What do you mean by pointer in C? What are its advantages and disadvantages?
2. Write a c programme to find out the area of a circle using pointer.
3. When would you use a pointer to a function?
4. Describe the following with examples?

a) Pointer arithmetic
b) Pointer and strings

5. What are the advantages of malloc and calloc?
6. Write a function to swap two float variables using call by reference.
7. Print string elements using pointer notation.
8. Write a program for binary search using pointers.
9. Explain pointers and two-dimensional arrays with examples.
10. What do you mean by function declaration and function declarator.

Answers for Self Assessment

1. (a) 2. (b) 3. (b) 4. (a) 5. (c)

7.13 Review Questions

1. http://www.cs.swarthmore.edu/~newhall/unixhelp/C_pointers.pdf

8

Structures and Unions

STRUCTURE

8.0 Objectives

8.1 Introduction

8.2 Definition of Structure

8.3 Type-Defined Structure

8.4 Structure Initialization and Accessing

8.5 Array of Structures

8.6 Structures and Functions

8.7 Passing Structures through Pointers

8.8 Uses of Structures

8.9 Definition of Union

8.10 Difference between Structures and Arrays

8.11 Pointer to Structures

8.12 Derived Data Types

8.13 Enumerated Data Types

8.14 Summary

8.15 Keywords

8 16 Review Questions

8.17 Further Studies

8.0 Objectives

After studying this chapter, you will be able to:

- Understand the definition of structure
- Explain the structures and functions
- Discuss how to passing structures to the functions
- Explain the passing structures through pointers
- Define the uses of structures
- Discuss the difference between structures and arrays
- Differentiate between structure and unions
- Explain the pointer to structures and derived data types

- Define the derived data types
- Discuss the enumerated data types

8.1 Introduction

Arrays and. structures have similarities as well as differences between them. Both arrays and structures represent collections of a number of items. While an array is a collection of items of the same data type, this does not hold good for a structure. For instance int x [10]; defines an array with dimension 100 i.e., 100 items, all of the same data type namely integers. However, structures can represent items of varying data types pertaining to an item. The only similarity between an array and a structure lies in the fact that there can be a collection of structures which is known as array of structures.

8.2 Definition of Structure

A structure is a collection of variables referenced with one name, providing a convenient means of keeping related information together. A structure declaration forms a template that may be used to create structure objects (that is, instances of a structure). Then that make up the structure are called members. (Structure members are also commonly referred to as elements or fields).

Generally, all of the members of a structure are logically related. For example, the name and address information in a mailing list would normally be represented in a structure. The following code fragment shows how to declare a structure that defines the name and address fields. The keyword struct tells the compiler that a structure is being declared.

The general form of a structure declaration is:

```
struct struct-type-name
{
type member-name;
type member-name;
type member-name;.
}
structure-variables;
```

Where either struct-type-name or structure variables may be omitted, but not both.

8.2.1 Structure Declaration

Structure is synonymous with records structure, similar to a record, contains a number of fields-or variables. The variables can be of any of the valid data types. The definition of the record book, which we call a structure, is given as:

```
struct book
{
char title [25];
char author [15];
char Publisher [25];
float Price ;
unsigned Year;
}
```

Struct is a keyword or a reserved word of "C". A structure tag or name follows which is book in this case. The beginning of the structure is indicated by opening brace. Thereafter, the fields of the record or data elements are declared one by one the variables or fields declared are also called members of the structure. The structure

consists of different types of data elements which is different from array. Let us now look at the members of struct book.
The title of the book is declared as a string with width 25; similarly the author and publisher are arrays of characters or strings of the specified width. The price is defined as a float to take care of the fractional part of the currency. The year is defined as an unsigned integer.
Thus you have to understand the following when you want to declare a structure.
(a) Struct is the header of a structure definition.
(b) It can be followed by an optional name for the structure.
(c) Then the members of the structure are declared one by one within a block.
(d) The block starts with an opening brace, but ends with a closing brace followed by a semicolon.
(e) The members can be of any data type.
(f) The members have a relation with the structure; i.e. all of them belong to the defined structure and they have identity only as members of the structure and not otherwise Therefore if you assign a name to author, it will not be accepted.

The structure declaration above is similar to the prototype in a function in so far as memory allocation is considered. The system does not allocate memory as soon as it finds structure declaration, which is for information and checking consistency later on. The allocation of memory takes place only when structure variables are declared. What is a structure variable? It is similar to other variables. For instance int i means that i is an integer variable. Similarly the given is a structure variable declaration.
Here s1 is a variable of type structure book. Suppose, we define,
struct book,s1, s2 ;

This means that there are two variables s1 and s2of type struct book. These variables can hold different values for their members.
Another point to be noted is that the structure declaration appears above all other declarations.
An example which does nothing, but define structure and declare structure variables is given as:

```
main ()
{
struct book
{
char title [25];
char author [15];
char publisher [25];
float price;
unsigned year;
};
struct book s1, s2, s3 ;
```

If you want to define a large number of books, then how will you modify the structure variable declaration? It will be as follows:
struct book s[1000];
This will allocate space for storing 1000 structures or records of books. However, how much storage space will be allocated for each element of the array? It will be the sum of storage spaces required for each member. In struct book the storage space required will be as given below:
title 25 + 1 (for null to indicate end of string)
author 15 + 1
publisher 25 + 7

price 4
year 2
Therefore the system allots Space for 1000 structure variables each with the above requirement.Space are allocated only after seeing the structure variable declaration.
Let us take another example to make the concept clear. You know that the bank account of each account holder is a record. Let us define a structure for it.

```
struct account
{
unsigned number;
char name [15];
int balance ;
} al, a2;
```

Instead of declaring separate structure variables such as struct account al, a2; we can use coding as in the example given. Here the variables are declared just after the closing brace of the structure declaration and terminated with a semicolon. This is perfectly correct. The declaration of the members of the structure is clear; the balance has been declared as an integer instead of a float to make it simple. This means that the minimum transaction is a rupee.

8.2.2 Tagged Structure

The new specification of C language allows you to use "tagged" initialises by supplying the desired member name within the {}

```
struct s = {c = 3};
```

That way you only explicitly initialize the desired member(s) (and ask the compiler to implicitly zero-initialize the rest).
This not only saves you some typing but also makes the aggregate initializes independent from the order in which the members are declared in the struct.
Aggregate initialises, as you probably know, can be used with arrays, too. And C99 supports "tagged" initialization with arrays as well. How the "tags" look in case of an array is illustrated by the example:

```
int a[10] = {[5] = 3};
```

It is worth noting one more time that C language continues to stick to the "all or nothing" approach to aggregate initialization: if you specify an explicit initializer for just one (or some) members of a struct or an array, the whole aggregate (struct or array) gets initialized, and the members without explicit initializers get zero-initialized.

8.2.3 Structure Variables

The structure variable declaration is of no use unless the variables are assigned values. Here each member has to be specifically accessed for each structure variable. For instance to assign the account number for variable a1 we have to specify as follows:

```
a1. number = 0001;
```

There is a dot operator in between the structure variable name and the member name or tag.
Suppose you then want to assign account no.2 to a2, it can be assigned as follows:

```
a2 number = 2;
```

If you want to know the address where a2 number is stored you can use

```
printf ("%u",&a2 number);
```

This is similar to other data types. The structure is a complex data type, and has to indicate which structure variable to which the number belongs, as otherwise the number is common all the structure variable such as al,

a2, a3, etc., therefore it is necessary to be specific. Assuming that you want to get the value from the keyboard, you can use scanf as:
scanf ("%u", & a1 number) ;
You can also assign initial values directly as:
struct account a1 = {0001, "Vasu" 1000};
struct account a2= {0002, "Ram", 1500};
All the members are specified. This is similar to the declaration of initial values for arrays. However, note the semicolon after the closing brace. The struct a1 will therefore receive the values for the members in the order in which they appear. Therefore you must give the values in the right order, and they will be accepted automatically as follows:

a1. number = 0001
a1. name = Vasu
a1. balance = 1000

8.3 Type-Defined Structure

The typedef feature allows users to define new data types that are equivalent to existing data types. Once a user-defined data type has been established, then new variables, arrays, structure and so on, can be declared in terms of this new data type. In general terms, a new data type is defined as
typedef type new type;
Where type refers to an existing data type and new-type refers to the new user-defined data type.

Example:

typedef int age;

In this declaration, age is user defined data type equivalent to type int. Hence, the variable declaration age male, female; is equivalent to writing int age, male, female; the typedef feature is particularly convenient when defining structures, since it eliminates the need to repeatedly write struct tag whenever a structure is referenced. As a result, the structure can be referenced more concisely.
In general terms, a user-defined structure type can be written as:

typedef struct;
{
member 1;
member 2;
member m; }new-type;
The typedef feature can be used repeatedly, to define one data type in terms of other user-defined data types.

8.4 Structure Initialization and Accessing

Structure members can be initialized when you declare a variable of your structure:
struct object player1 = {"player1", 0, 0};
The above declaration will create a struct object called player1 with an id equal to "player1", expose equal to 0, and hypos equal to 0.

To access the members of a structure, you use the "." (scope resolution) operator. Shown here is an example of how you can accomplish initialization by assigning values using the scope resolution operator.
struct object player1;

```
player1.id = “player1”;
player1.xpos = 0;
player1.ypos = 0;
```

8.4.1 Nested Structures

Structures can contain other structures as members; in other words, structures can nest. Consider the following two structure types:

```
struct first_structure_type {
  int integer_member;
 float float_member;
};

struct second_structure_type {
  double double_member;
  struct first_structure_type struct_member;
};
```

The first structure type is incorporated as a member of the second structure type. You can initialize a variable of the second type as follows:

1. struct second_structure_type demo;
2. demo.double_member = 12345.6789;
3. demo.struct_member.integer_member = 5;
4. demo.struct_member.float_member = 1023.17;

The member operator is used to access members of structures that are themselves members of a larger structure. No parentheses are needed to force a special order of evaluation; a member operator expression is simply evaluated from left to right.

In principle, structures can be nested indefinitely. Statements such as the following are syntactically acceptable, but bad style.

my_structure.member1.member2.member3.member4 = 5;

What happens if a structure contains an instance of its own type, however? For example:

```
struct regression
{
int int_member;
  struct regression self_member;
};
```

In order to compile a statement of this type, your computer would theoretically need an infinite amount of memory. In practice, however, you will simply receive an error message along the given lines:

```
struct5.c: In function ‘main’:
struct5.c:8: field ‘self_member’ has incomplete type
```

The compiler is telling you that self_member has been declared before its data type, regression has been fully declared naturally, since you are declaring self_member in the middle of declaring its own data type.

8.4.2 Program on Nested Structure

```
#include <stdio.h>
#include <conio.h”>
struct adress {
```

```
char ct[10];
char dist[10],state[5];
long int pin;
};
struct emp {
char name[10];
int age,sal;
struct adress a;
};
void main()
{
struct emp e[2];
int i;
clrscr();
for(i=0;i<=1;i++)
{
printf("enter [%d]st employee's name,age,salary :: ", i);
scanf("%s%d%d",e[i].name,&e[i].age,&e[i].sal);
printf("\n  enter city, district,state & pincode ::");
scanf("%s%s%s%ld",e[i].a.ct,e[i].a.dist,e[i].a.state,&e[i].a.pin);
}
for (i=0;i<=1;i++)
{
printf("\n[%d]st employee's name :: %s",i,e[i].name);
printf("\n[%d]st employee's age :: %d ",i,e[i].age);
printf("\n[%d]st employee's salary :: %d",i,e[i].sal);
printf("\n[%d]st employee's city :: %s ",i,e[i].a.ct);
printf("\n[%d]st employee's district :: %s",i,e[i].a.dist);
printf("\n[%d]st employee's state :: %s",i,e[i].a.state);
printf("\n[%d]st employee's pin :: %ld",i,e[i].a.pin);
}
```

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Define structure.

……..……

……..……...….

……..

Ex2: What is Structure Variables?.

……..……

……..……...….

……..

8.5 Array of Structures

Let us now create an array of structures for the account. This is nothing but an array of accounts, declared with size. Let us restrict the size to 5. The records will be created by using keyboard entry.
The program is given as:

```
/*program to demonstrate structures*/
#include<stdio h>
main()
{
struct account
{
unsigned number;
char name [15] ;
int balance;
}a [5];
int i;
for (i=0; i<=4; i++)
{
printf (“a/c no: =\t name::\t balance::\n,, ) ;
scanf (“ %u%s%d”, & a [1] .number, a [i] .name, &a [i] .balance) ;
}
for (i = 0; i<=4; 1++)
{
printf (“a/c No: =%u\t name:= %s\t balance: = %d\n, a [i] .number, a [i] .name, a [i].balance) ;

}
}
```

Output:

```
a/c No:=        name:=          balance:=
1       suresh                  5000
a/c No:=        name:=          balance: =
2.      Lesley                  3000
a,/c No: =      name: =         balance: =
3.      ahmed                   5500
a/c No: =       name: =         balance: =
4.      lakshmi         10900
a/c no: =       name: =         balance: =
5.      Thomas          29000
a/c no: =1      name: =Suresh   balance: =5000
a/c no: =2      name: =Lesley   balance: =3000
a/c no: =3      name: =Ahmed    balance: =5500
a/c no: =4      name: =Lakshmi          balance: =10900
a/c no:=5       name: =Thomas           balance: =29000
```

The structure array has been declared as part of structure declaration as a[5]you will see that the individual elements of the 5 accounts are scanned., and printed in the same order.

When we scan a name, we do not give the address but actual name of the variables as in a [1].Name, since it is a string variable. Remember this uniqueness. This program basically gets the 5 structures or, records pertaining to 5 account holders. Thereafter, the details of the 5 accounts are printed using the for statement. The first half of the result was typed by the user and the last 5 lines are the output of the program.

8.6 Structures and Functions

Since structures are of custom data types, functions can return structures and also take them as arguments. Keep in mind that when you do this, you are making a copy of the structure and all it is members so it can be quite memory intensive.

To return a structure from a function declares the function to be of the structure type you want to return. In our case a function to initialize our object structure might look like this:

```
struct object createobj(char id[], int xpos, int ypos) {
        struct object newobj;
        strcpy(newobj.id, name);
        newobj.xpos = xpos;
        newobj.ypos = ypos;
        return newobj;
      }
```

8.6.1 Example of Structure with Function

Let us now learn to pass a structure to a function. As an example let us use a function that prints members of the structure passed to it:

```
void printobj(struct object obj) {
        printf("name: %s, ", obj.id);
        printf("x position: %d, ", obj.xpos);
        printf("y position: %d", obj.ypos);
        printf("n");
      }
```

For completeness we shall include the full source of the examples shown so you may see how it all fits together.

```
#include <stdio.h>
#include <stdlib.h>
struct object {
        char id[20];
        int xpos;
        int ypos;
      };
      struct object createobj(char id[], int xpos, int ypos);
      void printobj(struct object obj);
      void main() {
        struct object player1 = createobj("player1", 0, 0);
        struct object enemy1 = createobj("enemy1", 2, 3);
        printobj(player1);
        printobj(enemy1);
      }
       struct object createobj(char id[], int xpos, int ypos) {
```

```
 struct object newobj;
 strcpy(newobj.id, id);
 newobj.xpos = xpos;
 newobj.ypos = ypos;
 return newobj;
}
void printobj(struct object obj) {
 printf("name: %s, ", obj.id);
 printf("x position: %d, ", obj.xpos);
 printf("y position: %d", obj.ypos);
 printf("n");
}
```

Self Assessment Questions

1. Structure is synonymous with records structure, similar to a record.
(a) True (b) False

2. Both arrays and structures represent collections of a number of items.
(a) True (b) False

3. The year is defined as an .....
(a) unsigned integer (b) signed integer
(c) none of the above (d) All of these

4. The price can be defined as a float.
(a) True (b) False

5. Structures can be copied .....
(a) individually, (b).member wise
(c).one go (d). All of these

6. structure accounts has been declared as a .....
(a) nested structure (b) global structure
(c) high structure (d) All of these

8.7 Passing Structures through Pointers

The beginning address of a structure can be accessed in the same manner as any other address, through the use of the address (&) operator.

Thus, if variable represents a structure type variable, then & variable represents the starting address of that variable. We can declare a pointer variable for a structure by writing

type *ptr;

Where type is a data type those identities the composition of the structure and ptr represents the name of the pointer variable. We can then assign the beginning address of a structure variable to this pointer by writing

ptr= &variable;

Let us take the following example:

```
typedef struct {
char name [ 40];
int roll_no;
float marks;
}student;
student s1,*ps;
```

In this example, s1 is a structure variable of type student, and ps is a pointer variable whose object is a structure variable of type student. Thus, the beginning address of s1 can be assigned to ps by writing.

```
ps = &s1;
```

An individual structure member can be accessed in terms of its corresponding pointer variable by writing

```
ptr →member
```

Where ptr refers to a structure- type pointer variable and the operator → is comparable to the period (.) operator. The associativity of this operator is also left-to-right.

The operator → can be combined with the period operator (.) to access a sub member within a structure. Hence, a submember can be accessed by writing

```
ptr → member.submember
```

Caution

Be cautious about errors, because errors/bugs are very common while developing a program. If you do not detect them and correct them, they cause a program to produce wrong results.

8.8 Uses of Structures

As we have seen, a structure is a good way of storing related data together. It is also a good way of representing certain types of information. Complex numbers in mathematics inhabit a two dimensional plane (stretching in real and imaginary directions). These could easily be represented here by:

```
typedef struct {
    double real;
    double imag;
} complex;
```

doubles have been used for each field because their range is greater than floats and because the majority of mathematical library functions deal with doubles by default.

Apart from holding data, structures can be used as members of other structures. Arrays of structures are possible, and are a good way of storing lists of data with regular fields, such as databases.

Did You Know?

The structures could be used to hold the locations of points in multi-dimensional space. Mathematicians and engineers might see a storage efficient implementation for sparse arrays.

8.9 Definition of Union

A union is a collection of variables of different types, just like a structure. However, with unions, you can only store information in one field at any one time.

You can picture a union as like a chunk of memory that is used to store variables of different types. Once a new value is assigned to a field, the existing data is wiped over with the new data.

A union can also be viewed as a variable type that can contain many different variables (like a structure), but only actually holds one of them at a time (not like a structure). This can save memory if you have a group of

data where only one of the types is used at a time. The size of a union is equal to the size of it is largest data member. In other words, the C compiler allocates just enough space for the largest member. This is because only one member can be used at a time, so the size of the largest, is the most you will need. Here is an example:

```
union time
{
long simpledate;
 double percisedate;
}mytime;
```

The union above could be used to either store the current time (in seconds) to hold time accurate to a second. Or it could be used to hold time accurate to a millisecond. Presumably there are times when you would want one or the other, but not both. This declaration should look familiar. It is the same as a struct definition, but with the keyword union instead of struct.

8.9.1 Accessing Union Fields

To access the fields of a union, use the dot operator(.) just as you would for a structure. When a value is assigned to one member, the other member(s) get whipped out since they share the same memory. Using the example, the precise time can be accessed like this:

```
printTime( mytime.percisedate );
```

In larger programs it may be difficult to keep track of which field is the currently used field. This is usually handled by using another variable to keep track of that. For example, you might use an integer called mode. When mode equals one, the regular date (simpledate) is used. If mode is two, then percisedate is used. This mode variable needs to be set every time a different member in the union is used. Here is a simple program to illustrate the use of unions.

```
#include <stdio.h>
int main()
{
union data
{
char a;
int x;
float f;
} mydata;
int mode = 1;
mydata.a = ‘A’;
printf(“Here is the data:\n%c\n%i\n%.3f\n”, mydata.a, mydata.x, mydata.f );
mydata.x = 42;
mode = 2;
printf(“Here is the data:\n%c\n%i\n%.3f\n”, mydata.a, mydata.x, mydata.f );
mydata.f = 101.357;
mode = 3;
printf(“Here is the data:\n%c\n%i\n%.3f\n”, mydata.a, mydata.x, mydata.f );
if( mode == 1 )
printf(“The char is being used\n”);
else if( mode == 2 )
printf(“The int is being used\n”);
```

```
else if( mode == 3 )
printf("The float is being used\n");
return 0;
}
```

This little program declares a union with an int, float, and char. It uses each field, and after each use prints out all the fields (with one ugly printf statement). Here is some sample output:

```
Here is the data: A 5770.000
Here is the data:*420.000
Here is the data: 1120581321101.357
```

The float is being used your output might be different. Clearly the data in the unused fields is just garbage. This happens because different types are treated differently by the computer. So if one type is set, the memory is not going to be in the format of the other types. Also mode is used to keep track of the type. In this program it is pretty useless, but in larger programs it would be a great help in keeping track of the unions use.

8.10 Difference between Structures and Arrays

The differences between structures and arrays are as follows:

- Array elements are homogeneous. Structure elements are of different data type.
- Array allocates static memory and uses index / subscript for accessing elements of the array. Structures allocate dynamic memory and uses (.) operator for accessing the member of a structure.
- Array is a pointer to the first element of it. Structure is not a pointer.
- Array element access takes less time in comparison with structures.

Both the arrays and structures are classified as structured data types as they provide a mechanism that enable us to access and manipulate data in a relatively easy manner. But they differ in a number of ways listed in Table 8.1:

Table 8.1: Difference between arrays and structures.

Arrays	Structures
1. An array is a collection of related data elements of same type.	1. Structure can have elements of different types
2. An array is a derived data type	2. A structure is a programmer-defined data type
3. Any array behaves like a built-in data types. All we have to do is to declare an array variable and use it.	3. But in the case of structure, first we have to design and declare a data structure before the variable of that type are declared and used.

Caution

Assignment of a struct should not be confused with the requirement of memory management when dealing with a pointer to a struct.

Exercise: Check Your Progress 3

Note: i) Use the space below for your answer.

Ex1: Define union.

……..……

……..……...….

……..

Ex2: What is the Difference between Structures and Arrays?

……..……

……..……...….

……..

8.11 Pointer to Structures

It is possible to create a pointer to almost any type in C, including user-defined types. It is extremely common to create pointers to structures. An example is shown as:

```
typedef struct
{
        char name[21];
        char city[21];
        char state[3];
} Rec;
typedef Rec *RecPointer;
RecPointer r;
r = (RecPointer)malloc(sizeof(Rec));
```

The pointer r is a pointer to a structure. Please note the fact that r is a pointer, and therefore takes four bytes of memory just like any other pointer. However, the malloc statement allocates 45 bytes of memory from the heap. *r is a structure just like any other structure of type Rec. The following code shows typical uses of the pointer variable:

```
strcpy((*r).name, "Leigh");
strcpy((*r).city, "Raleigh");
strcpy((*r).state, "NC");
printf("%s\n", (*r).city);
free(r);
```

You deal with *r just like a normal structure variable, but you have to be careful with the precedence of operators in C. If you were to leave off the parenthesis around *r the code would not compile because the "." operator has a higher precedence than the "*" operator. Because it gets tedious to type so many parentheses when working with pointers to structures, C includes a shorthand notation that does exactly the same thing:

```
strcpy(r->name, "Leigh");
```

The r-> notation is exactly equivalent to (*r), but takes two fewer characters.

8.12 Derived Data Types

In many applications the intrinsic data types are not enough to express in code the ideas behind an algorithm or solution to a specific problem. Derived data types and structures allow programmer to group different kinds of information that belong to a single entity. In a way they resemble arrays but with two important differences. First the different elements of a derived data type do not have to be of the same type, thus they may include integer, character or real. Second the different entities making up the derived data type are referred to with a name and not an integer index. The different element of a derived data type is referred to as components. The data type of the component can be any of the intrinsic data types, or a previously defined derived data type.

A derived data type is defined with a block of code enclosed within a type and end type statements followed by the name we wish to give to that derived data type. For example suppose we are writing a program that

manipulates geometric information, and we would like to define points and polygons. Furthermore, we would like a point to have some attributes associated with it.

Did You Know?
The Fortan 90 derived data type is similar to C structures and also have similarity with C++ classes.

8.13 Enumerated Data Types

Enumerated data type variables can only assume values which are declared as:

```
enum month { jan = 1, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, dec };
enum month this_month;
this_month = feb;
```

In this declaration, month is declared as an enumerated data type. It consists of a set of values, jan to dec. Numerically, jan is given the value 1, feb the value 2, and so on. The variable this_month is declared to be of the same type as month, and then is assigned the value associated with feb. This_month cannot be assigned any values outside those specified in the initialization list for the declaration of month.

```
#include <stdio.h>
main()
{
char *pwest = "west",*pnorth = "north", *peast="east", *psouth = "south";
enum location { east=1, west=2, south=3, north=4};
enum location direction;
direction = east;
if( direction == east )
printf("Cannot go %s\n", peast);
}
```

The variables defined in the enumerated variable location should be assigned initial values.

Example: Bookshop Inventory - A bookshop uses a personal computer to maintain the inventory of books that are being sold at the shop. The list includes details such as author, title, price publisher, stock position, etc. Whether a customer wants a book, the shopkeeper inputs the title and author of the book and system replies whether it is in the list or not. If it is not, an appropriate message is displayed. If book is in the list, then the system displays the book details and asks for number of copies. If the requested copies are available, the total cost of the books is displayed; otherwise the message "Required copies not in stock" is displayed.

The program uses a template to define the structure of the book. Note that the date of publication, a member of record structure, is also defined as a structure. When the title and author of a book are specified, the program searches for the book in the list using the function

look_up(table, s1, s2 m)

The parameter table which receives the structure variable book is declared as type struct record. The parameters s1 and s2 receive the string values of title and author while m receives the total number of books in the list. Total number of books is given by the expression

sizeof(book)/sizeof(struct record)

The search ends when the book is found in the list and the function returns the serial number of the first book in the list is zero. The program terminates when we respond "NO" to the question

Do you want other book?

Note that we use the function

Get(string)

To get title, author, etc. from the terminal. This enables us to input strings with spaces such as “C Language”. We can not use scanf to read this string since it contains two words. Since we are reading the quantity as string using the get (string) function, we have to convert it to as integer before using it in any expression. This is done using the atoi( )function.

```
#include <stdio.h>
#include <string.h>
struct record
{
char author[20];
char title[30];
float price;
struct
{
char month[10];
int year;
}
date;
char publisher[10];
int quantity;
};
int look_up(struct record table[],char s1[],char s2[],int m);
void get (charstring [ ] );
main()
{
char title[30], author[20];
int index, no_of_records; char response[10], quantity[10];
struct record book[] = {
{“Ritche”,”C Language”,45.00,”May”,1977,”PHI”,10},
{“Kochan”,”Programming in C”,75.50,”July”,1983,”Hayden”,5},
{“Balagurusamy”,”BASIC”,30.00,”January”,1984,”TMH”,0},
{“Balagurusamy”,”COBOL”,60.00,”December”,1988,”Macmillan”,25}
};
no_of_records = sizeof(book)/ sizeof(struct record);
do
{
printf(“Enter title and author name as per the list\n”);
printf(“\nTitle: “);
get(title);
printf(“Author: “);
get(author);
index = look_up(book, title, author, no_of_records);
if(index != –1) /* Book found */
{
printf(“\n%s %s %.2f %s %d %s\n\n”,
book[index].author,
book[index].title,
```

```
book[index].price,
book[index].date.month,
book[index].date.year,
book[index].publisher);
printf("Enter number of copies:");
get(quantity);
if(atoi(quantity) < book[index].quantity)
printf("Cost of %d copies = %.2f\n",atoi(quantity),
book[index].price * atoi(quantity));
else
printf("\nRequired copies not in stock\n\n");
}
else
printf("\nBook not in list\n\n");
printf("\nDo you want any other book? (YES / NO):");
get(response);
}
while(response[0] == 'Y' || response[0] == 'y');
printf("\n\nThank you. Good bye!\n");
}
void get(charstring [] )
{
char c;
int i = 0;
do
{
c = getchar();
string[i++] = c;
}
while(c != '\n');
string[i–1] = '\0';
}
int look_up(struct record table[],char s1[],char s2[],int m)
{
int i;
for(i = 0; i < m; i++)
if(strcmp(s1, table[i].title) == 0 && strcmp(s2, table[i].author) == 0)
return(i); /* book found */
return(–1); /* book not found */
}
```

Output

```
Enter title and author name as per the list
Title: BASIC
Author: Balagurusamy
Balagurusamy BASIC 30.00 January 1984 TMH
Enter number of copies:5
Required copies not in stock
```

Do you want any other book? (YES / NO):y
Enter title and author name as per the list
Title: COBOL
Author: Balagurusamy
Balagurusamy COBOL 60.00 December 1988 Macmillan
Enter number of copies:7
Cost of 7 copies = 420.00
Do you want any other book? (YES / NO):y
Enter title and author name as per the list
Title: C Programming
Author: Ritche
Book not in list
Do you want any other book? (YES / NO): n
Thank you. Good bye!

8.14 Summary

- A structure is usually used when we wish to store dissimilar data together.
- Structure elements can be accessed through a structure variables able using a dot(.) operator.
- It is possible to create an array of structure.
- The data type of the component is any of the intrinsic data types, or a previously defined derived data type.
- Derived data types and structures allow programmer to group different kinds of information that belong to a single entity.

8.15 Keywords

***Array of Structures*:** These are the arrays of the basic data types such as integers and floats.
***Nested structures*:** Structures can contain other structures as members; in other words, structures can nest.
***Structure Elements*:** These are the members of the structures declared as different types.
Structure: A structure is a collection of one or more variables, possibly of different data types, grouped together under a single name for convenient handling.
***Unions*:** A union is a collection of variables of different types, just like a structure.

8.16 Review Questions

1 How does a structure differ from an array?
2 Describe what is wrong in the following structure declaration:

```
        struct
{
        int number;
        float price;
}main()
{
}
```

3. Define process of structure declaration?
4. Define the terms:
   (a) Nested structure.

(b) Array of structures.

5 Write a C program to using structure array?

6. Differentiate between the union and structure?

7. Explain the union with the example?

8. Write a C program to nested structure and explain the steps?

9. Explain the union and structure in the term of memory?

10. Write a C program to store the string in array using structure?

Answers for Self Assessment

1. (a) 2. (a) 3. (a) 4. (a) 5. (d) 6. (b)

8.17 Further Studies

1) https://www.clear.rice.edu/comp221/html/pdf/04-struct-union.pdf
2) Programing with c by Bala gura swami.

9

File Handling

STRUCTURE

9.0 Objectives

9.1 Introduction

9.2 File System Basics

9.3 Standard Streams in C

9.4 File Pointers

9.5 File Handling Functions

9.6 The getw and putw Functions

9.7 Input / Output Operations on File

9.8 Working with String Using fputs() and fgets()

9.9 The fprintf and fscanf Functions

9.10 Direct Access File

9.11 Summary

9.12 Keywords

9.13 Review Questions

9.14 Further Studies

9.0 Objectives

After studying this chapter, you will be able to:

- Explain the file system basics
- Understand about the standard streams in C
- Define the file pointers
- Explain the file handling functions
- Discuss the getw and putw functions
- Understand about the input/output operations on file
- Explain the working with string using fputs() and fgets()
- Define and declare the the fprintf and fscanf functions
- Explain the direct access file

9.1 Introduction

A file represents a sequence of byte on the disk where a group of related data is stored. File is created for permanent storage of data. It is a ready made structure. In C, we use a structure pointer of file type to declare a file.
C provides a number of functions that help to perform basic file operations. They are,

Function	description
fopen()	create a new file or open a existing file
fclose()	closes a file
getc()	reads a character from a file
putc()	writes a character to a file
fscanf()	reads a set of data from a file
fprintf()	writes a set of data to a file
getw()	reads a integer from a file
putw()	writes a integer to a file
fseek()	set the position to desire point
ftell()	gives current position in the file
rewind()	set the position to the begining point

9.2 File System Basics

A file is a collection of bytes stored on a secondary storage device, which is generally a disk of some kind. The collection of bytes may be interpreted, for example, as characters, words, lines, paragraphs and pages from a textual document; fields and records belonging to a database; or pixels from a graphical image. The meaning attached to a particular file is determined entirely by the data structures and operations used by a program to process the file. It is conceivable (and it sometimes happens) that a graphics file will be read and displayed by a program designed to process textual data. The result is that no meaningful output occurs (probably) and this is to be expected. A file is simply a machine decipherable storage media where programs and data are stored for machine usage.
Essentially there are two kinds of files that programmers deal with text files and binary files. These two classes of files will be discussed in the following sections.

9.2.1 ASCII Text Files

A text file can be a stream of characters that a computer can process sequentially. It is not only processed sequentially but only in forward direction. For this reason a text file is usually opened for only one kind of operation (reading, writing, or appending) at any given time.
Similarly, since text files only process characters, they can only read or write data one character at a time. (In C Programming Language, Functions are provided that deal with lines of text, but these still essentially process data one character at a time.) A text stream in C is a special kind of file. Depending on the requirements of the operating system, newline characters may be converted to or from carriage-return/linefeed combinations depending on whether data is being written to, or read from, the file. Other character conversions

may also occur to satisfy the storage requirements of the operating system. These translations occur transparently and they occur because the programmer has signaled the intention to process a text file.

9.2.2 Binary Files

A binary file is no different to a text file. It is a collection of bytes. In C Programming Language a byte and a character are equivalent. Hence a binary file is also referred to as a character stream, but there are two essential differences.

No special processing of the data occurs and each byte of data is transferred to or from the disk unprocessed.

The C Programming Language places no constructs on the file, and it may be read from, or written to, in any manner chosen by the programmer. Binary files can be either processed sequentially or, depending on the needs of the application, they can be processed using random access techniques. In C Programming Language, processing a file using random access techniques involves moving the current file position to an appropriate place in the file before reading or writing data. This indicates a second characteristic of binary files.

They a generally processed using read and writes operations simultaneously.

For example, a database file will be created and processed as a binary file. A record update operation will involve locating the appropriate record, reading the record into memory, modifying it in some way, and finally writing the record back to disk at its appropriate location in the file. These kinds of operations are common to many binary files, but are rarely found in applications that process text files.

9.3 Standard Streams in C

A stream is a logical entity that represents a file or device that can accept input or output. All input and output functions in standard C operates on data streams. Streams can be divided into text streams and binary streams. Streams may also have an orientation towards byte I/O or wide I/O.

9.3.1 Text Streams

A stream that is opened in text mode will read and write files in the implementation's native text file format. The data read from or written to a text stream is in the form of lines of text, each terminated by a single newline character ('\n'). The implementation will take care of conversion from this format into the native format.

On Unix systems there is no conversion necessary and none is performed. On Windows or DOS systems, "\r\n" (CRLF) is translated to "\n" (LF) and vice-versa. On Macintosh systems, "\r" (CR) is translated to "\n" (LF) and vice-versa. Some older operating systems have record-based text file formats that require more extensive conversion.

9.3.2 Binary Streams

A stream that is opened in binary mode can read and write raw data, disregarding any native file format issues. There is no conversion or translation performed.

9.3.3 Predefined Streams

Three streams are defined by the implementation and automatically opened when your C program starts up. They are the standard input, standard output and standard error streams, which may be accessed through the identifiers stdin, stdout and stderr respectively.

Typically, user input typed into the program can be read through the standard input stream, while program output written to the standard output and standard error streams will appear on the user's screen. The standard

output stream is designed for normal program output, while standard error is designed for diagnostics and error messages.

Most operating systems allow the user to redirect the input or output of these streams to other files. Some implementations run on hardware that lacks keyboards or screens; in that case the predefined streams may be mapped to a serial line, printer or other file.

Did You Know?
The stream model of file I/O was popularized by the Unix operating system, which was developed concurrently with the C programming language itself.

9.4 File Pointers

The C communicates with files using a new data type called a file pointer. This type is defined within stdio.h, and written as FILE *. A file pointer called output_file is declared in a statement like.
FILE *output_file;

- Opening a file pointer using fopen
- Closing a file using fclose

9.4.1 Opening a File

If we want to store data in a file into the secondary memory, we must specify certain things about the file to the operating system. They include the fielname, data structure, purpose.
The general format of the function used for opening a file is
FILE *fp;
fp=fopen("filename","mode");
The first statement declares the variable fp as a pointer to the data type FILE. As stated earlier, File is a structure that is defined in the I/O Library. The second statement opens the file named filename and assigns an identifier to the FILE type pointer fp. This pointer, which contains all the information about the file, is subsequently used as a communication link between the system and the program.
The second statement also specifies the purpose of opening the file. The mode does this job.
R open the file for read only.
W open the file for writing only.
A open the file for appending data to it.

Consider the following statements:
FILE *p1, *p2;
p1=fopen("data","r");
p2=fopen("results","w");
In these statements the p1 and p2 are created and assigned to open the files data and results respectively the file data is opened for reading and result is opened for writing. In case the results file already exists, its contents are deleted and the files are opened as a new file. If data file does not exist error will occur.

9.4.2 Reading From or Writing To a File

Once a file has been successfully opened, you can read from it using fscanf() or write to it using fprintf(). These functions work just like scanf() and printf(), except they require an extra first parameter, a FILE * for the file to be read/written.
There are other functions in stdio.h that can be used to read or write files. Look them up in a good C reference.

Continuing our example from above, suppose the input file consists of lines with a username and an integer test score, e.g.:

in.list
foo 70
bar 98

and that each username is no more than 8 characters long.

We might use the files we opened above by copying each username and score from the input file to the output file. In the process, we will increase each score by 10 points for the output file:

```
char username[9];  /* One extra for nul char. */
int score;

while (fscanf(ifp, "%s %d", username, &score) != EOF)
{
fprintf(ofp, "%s %d\n", username, score+10);
}
```

The function fscanf(), like scanf(), normally returns the number of values it was able to read in. However, when it hits the end of the file, it returns the special value EOF. So, testing the return value against EOF is one way to stop the loop.

The bad thing about testing against EOF is that if the file is not in the right format (e.g., a letter is found when a number is expected):

in.list
foo 70
bar 98
biz A+

then fscanf() will not be able to read that line (since there is no integer to read) and it would not advance to the next line in the file. For this error, fscanf() will not return EOF (it is not at the end of the file).

Errors like that will at least mess up how the rest of the file is read. In some cases, they will cause an infinite loop. One solution is to test against the number of values we expect to be read by fscanf() each time. Since our format is "%s %d", we expect it to read in 2 values, so our condition could be:

```
while (fscanf(ifp, "%s %d", username, &score) == 2) {
```

Now, if we get 2 values, the loop continues. If we do not get 2 values, either because we are at the end of the file or some other problem occurred (e.g., it sees a letter when it is trying to read in a number with %d), then the loop will end.

Another way to test for end of file is with the library function feof(). It just takes a file pointer and returns a true/false value based on whether we are at the end of the file.

To use it in the above example, you would do:

```
while (!feof(ifp)) {
  if (fscanf(ifp, "%s %d", username, &score) != 2)
    break;
  fprintf(ofp, "%s %d", username, score+10);
}
```

Note that, like testing != EOF, it might cause an infinite loop if the format of the input file was not as expected. However, we can add code to make sure it reads in 2 values.

When you use fscanf(...) != EOF or feof(...), they will not detect the end of the file until they try to read past it. In other words, they would not report end-of-file on the last valid read, only on the one after it.

9.4.3 Closing a File

The input output library supports the function to close a file; it is in the following format.
fclose(file_pointer);
A file must be closed as soon as all operations on it have been completed. This would close the file associated with the file pointer.
Observe the following program.

```
FILE *p1 *p2;
p1=fopen (“Input”,”w”);
p2=fopen (“Output”,”r”);
fclose(p1);
fclose(p2)
```

The above program opens two files and closes them after all operations on them are completed, once a file is closed its file pointer can be reversed on other file.
The getc and putc functions are analogous to getchar and putchar functions and handle one character at a time. The putc function writes the character contained in character variable c to the file associated with the pointer fp1. ex putc(c,fp1); similarly getc function is used to read a character from a file that has been open in read mode. c=getc(fp2).
The program shown below displays use of file operations. The data enter through the keyboard and the program writes it. Character by character, to the file input. The end of the data is indicated by entering an EOF character, which is control-z. the file input is closed at this signal.

```
#include< stdio.h >
main()
{
file *f1;
printf(“Data input output”);
f1=fopen(“Input”,”w”); /*Open the file Input*/
while((c=getchar())!=EOF) /*get a character from key board*/
putc(c,f1); /*write a character to input*/
fclose(f1); /*close the file input*/
printf(“nData outputn”);
f1=fopen(“INPUT”,”r”); /*Reopen the file input*/
while((c=getc(f1))!=EOF)
printf(“%c”,c);
fclose(f1);
}
```

Caution

In string.h all the functions will not preserve ‘\0’ termination – this can have surprising results particularly when copying a string into a character array too small to hold it.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Define file handling.

……..……

……..……...…

……..

Ex2: What is the Binary file?

……..……

……..……...…

……..

9.5 File Handling Functions

File handling functions are functions for handling files. This program reads from a file named ‘data.txt’ and copies the contents to a file named ‘output.txt’.

```
#include
void main(void)
{
FILE *in, *out;
 char buffer[100];
in = fopen(“data.txt”, “r”);
if (in == NULL) {
printf(“Error opening input file\n”);
return;
 }
out = fopen(“output.txt”, “w”);
if (out == NULL) {
printf(“Error opening output file\n”);
fclose(in);
 return;
 }

while (fgets(buffer, 100, in) != NULL) {
fputs(out, buffer);
 }
}
```

Did You Know?

A function pointer is a type of pointer in C, C++, D, and other C-like programming languages, and Fortran 2003.

9.6 The getw and putw Functions

These are integer-oriented functions. They are similar to get c and putc functions and are used to read and write integer values. These functions would be usefull when we deal with only integer data. The general forms of getw and putw are:

putw(integer,fp);
getw(fp);

```
/*Example program for using getw and putw functions*/
#include< stdio.h >
main()
{
FILE *f1,*f2,*f3;
int number I;
printf(“Contents of the data filenn”);
f1=fopen(“DATA”,”W”);
for(I=1;I< 30;I++)
{
scanf(“%d”,&number);
if(number==–1)
break;
putw(number,f1);
}
fclose(f1);
f1=fopen(“DATA”,”r”);
f2=fopen(“ODD”,”w”);
f3=fopen(“EVEN”,”w”);
while((number=getw(f1))!=EOF)/* Read from data file*/
{
if(number%2==0)
putw(number,f3);/*Write to even file*/
else
putw(number,f2);/*write to odd file*/
}
fclose(f1);
fclose(f2);
fclose(f3);
f2=fopen(“ODD”,”r”);
f3=fopen(“EVEN”,”r”);
printf(“nnContents of the odd filenn”);
while(number=getw(f2))!=EOF)
printf(“%d%d”,number);
printf(“nnContents of the even file”);
while(number=getw(f3))!=EOF)
printf(“%d”,number);
fclose(f2);
fclose(f3);
}
```

9.7 Input / Output Operations on File

9.7.1. Redirection

One way to get input into a program or to display output from a program is to use *standard input* and *standard output*, respectively. All that means is that to read in data, we use scanf() (or a few other functions) and to write out data, we use printf().

When we need to take input from a file (instead of having the user type data at the keyboard) we can use input redirection:

% a.out < inputfile

This allows us to use the same scanf() calls we use to read from the keyboard. With input redirection, the operating system causes input to come from the file (e.g., input file above) instead of the keyboard.

Similarly, there is *output redirection*:

% a.out > outputfile

that allows us to use printf() as before, but that causes the output of the program to go to a file (e.g., outputfile above) instead of the screen.

Of course, the 2 types of redirection can be used at the same time...

% a.out < inputfile > outputfile

9.7.2. The C File I/O

While redirection is very useful, it is really part of the operating system (not C). In fact, the C has a general mechanism for reading and writing files, which is more flexible than redirection alone.

stdio.h

There are types and functions in the library stdio.h that are used for file I/O. Make sure you always include that header when you use files.

Type

For files you want to read or write, you need a file pointer, e.g.:

FILE *fp;

What is this type "FILE *"? Realistically, you do not need to know. Just think of it as some abstract data structure, whose details are hidden from you. In other words, the only way you can use a FILE * is via the functions that C gives you.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Define file handling function.

……..……

……..……...….

……..

9.8 Working with String Using fputs() and fgets()

The fgets() function takes three arguments to the gets() function's one. The first argument, as with gets(), is the address (type char *) where input should be stored. The second argument is an integer representing the maximum size of the input string. The final argument is the file pointer identifying the file to be read. A function call, then, looks like this:

fgets(buf, MAX, fp);

Here, buf is the name of a char array, MAX is the maximum size of the string, and fp is the pointer-to-FILE.

The fgets() function reads input through the first newline character, until one fewer than the upper limit of characters is read, or until the end-of-file is found; fgets() then adds a terminating null character to form a string. Therefore, the upper limit represents the maximum number of characters plus the null character. If fgets() reads in a whole line before running into the character limit, it adds the newline character, marking the end of the line into the string, just before the null character. Here it differs from gets(), which reads the newline but discards it.

Like gets(), fgets() returns the value NULL when it encounters EOF. You can use this to check for the end of a file. Otherwise, it returns the address passed to it.

The fputs() function takes two arguments: first, an address of a string, and then a file pointer. It writes the string found at the pointed-to location into the indicated file. Unlike puts(), fputs() does not append a newline when it prints. A function call looks like this:

fputs(buf, fp);

Here, buf is the string address, and fp identifies the target file.

Because fgets() keeps the newline and fputs() does not add one, they work well in tandem.

The parrot.c Program

```
/* parrot.c using fgets() and fputs() */
#include <stdio.h>
#define MAXLINE 20
int main(void)
{
   char line[MAXLINE];
   while (fgets(line, MAXLINE, stdin) != NULL && line[0] != '\n')
     fputs(line, stdout);
   return 0;
}
```

When you press the Enter key at the beginning of a line, fgets() reads the newline and places it into the first element of the array line. Use that fact to terminate the input loop. Encountering end-of-file also terminates it.

Self Assessment Questions

1. The FILE structure is defined in which of the following file.

(a) stdlib.h (b) stdio.c
(c) io.h (d) stdio.h.

2. get c and putc functions and are used to read and write values

(a) integer (b) string
(c) character (d) None of these.

3. A database file will be created and processed as a binary file

(a). True (b) False

4. Way of storing data in file .....

(a) binary format
(b) text format
(c) Text Format& Binary Format
(d) None of these.

5. fprintf() use for ……string.

(a) store (b) input

(c) write (d) None of these.

9.9 The fprintf and fscanf Functions

The fprintf and fscanf functions are identical to printf and scanf functions except that they work on files. The first argument of these functions is a file pointer which specifies the file to be used. The general form of fprintf is

fprintf(fp,"control string", list);

Where fp id a file pointer associated with a file that has been opened for writing. The control string is file output specifications list may include variable, constant and string.

fprintf(f1,%s%d%f",name,age,7.5);

Here name is an array variable of type char and age is an int variable

The general format of fscanf is

fscanf(fp,"controlstring", list);

This statement would cause the reading of items in the control string.

Example:

fscanf(f2,"5s%d", item, &quantity");

Like scanf, fscanf also returns the number of items that are successfully read.

```
/*Program to handle mixed data types*/
#include< stdio.h >
main()
{
FILE *fp;
int num, qty, I;
float price, value;
char item[10],filename[10];
printf("Input filename");
scanf("%s",filename);
fp=fopen(filename,"w");
printf("Input inventory datann");
printf("Item namem number price quantityn");
for (I=1;I< =3;I++)
{
fscanf(stdin,"%s%d%f%d",item,&number,&price,&quality);
fprintf(fp,"%s%d%f%d",itemnumber,price,quality);
}
fclose (fp);
fprintf(stdout,"nn");
fp=fopen(filename,"r");
printf("Item name number price quantity value");
for(I=1;I< =3;I++)
```

```
{
fscanf(fp,"%s%d%f%d",item,&number,&prince,&quality);
value=(price*quantity");
fprintf("stdout,"%s%d%f%d%dn",item,number,price,quantity,value);
}
fclose(fp);
}
```

9.9.1 Random Access to Files

Sometimes it is required to access only a particular part of the and not the complete file. This can be accomplished by using the following function:

1 > fseek

fseek function:

The general format of fseek function is as follows:

fseek(file pointer,offset, position);

This function is used to move the file position to a desired location within the file. Fileptr is a pointer to the file concerned. Offset is a number or variable of type long, and position in an integer number. Offset specifies the number of positions (bytes) to be moved from the location specified bt the position. The position can take the 3 values.

Value Meaning

0 Beginning of the file

1 Current position

2 End of the file.

Using fread() and fwrite()

How to Write Program for file I/O using fread and fwrite

```
#include
struct employee
{
int id;
int name[30];
float sal;
}emp;
int main()
{
int i,n;
FILE *fwptr,*frptr;
//open file for writing
fwptr=fopen("emp.dat","wb");
// checking if file exists or not
if(fwptr==NULL)
{
printf("File does not exist");
return 0;
}
//Accepting number of records for writing
printf("How many records do you want?\n");
scanf("%d",&n);
```

```
printf("\n--------Writing in file using fwrite----------\n\n");
//Writing records one by one in file
for(i=0;i<n;i++)
{
printf("Enter the details of employee:\n");
printf("Enter Employee Id :\n");
scanf("%d",&emp.id);
printf("Enter Employee Name :\n");
scanf("%s",emp.name);
printf("Enter Employee Salary :\n");
scanf("%f",&emp.sal);

//for writing the entire record to the file
fwrite(&emp,sizeof(emp),1,fwptr);
}


//closing the file
fclose(fwptr);
//open file for Reading
frptr=fopen("emp.dat","rb");
// checking if file exists or not
if(fwptr==NULL)
{
printf("File does not exist");
return 0;
}
printf("\n\n--------Displaying record from file using fread----------\n");
//Displaying records till end
while(fread(&emp,sizeof(emp),1,frptr)==1)
{
printf("\nEmployee Id: %d\t",emp.id);
printf("\nEmployee Name: %s\t",emp.name);
printf("\nEmployee Salary: %f\t",emp.sal);
printf("\n");
}
//closing the file
fclose(frptr);
}
```

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Write a program that merges lines alternately from two files and writes the results to new file. If one file has less number of line than the other, the remaining lines from the larger file should be simply copied into the target file.

……..……

……..……...…

……..

Did You Know?

In computing, tar (derived from tape archive) file is the concatenation of one or more files. Each file is preceded by a 512-byte header record.

9.10 Direct Access File

The C language view of a file is to see it as an array of bytes. This means that all files may be treated as direct access files and the library function fseek() may be used to provide immediate access to any particular part of the file. This assumes that the start byte address of the record is known. The following program demonstrates the construction and use of a table of record start positions using ftell().

```
/*      program to display selected records from a file */
#include <stdio.h>
#define MAXREC  1000
long index[MAXREC];
int         makeindex(FILE *);
void      showrec(FILE *,int);

main(int argc, char *argv[])
{
      int     nrec;   /* record to show */
      int     nrecs;  /* records in file */
      FILE    *d;     /* the data file */
      if(argc != 2)
      {
            fprintf(stderr,"Usage : showrec f\n");
            exit(1);
      }
      if((d=fopen(argv[1],"r"))==NULL)
      {
            fprintf(stderr,"error opening %s\n", argv[1]);
            exit(1);
      }
      if((nrecs=makeindex(d))==0)
      {
            fprintf(stderr, "file too big or empty\n");
            exit(1);
      }
      printf("%d records\n",nrecs);
      while(1)
      {
            printf("Enter record number ");
```

```
            scanf("%d",&nrec);
            if(nrec<0)
exit(0);
            if(nrec >= nrecs)
                  printf("Out of range\n");
            else showrec(d,nrec);
      }
}
int makeindex(FILE *f)
/*      builds the (global) index table and returns the number of records or 0 if the file has too many records or
is empty */
{
      int     c;
      int     i=0;
      while(1)
      {
            if((c=getc(f))=='\n')
            {
                  index[++i]=ftell(f);
                  if(i==MAXREC) return 0;
            }
            else
                  if(c==EOF)
return i;
      }
}
void      showrec(FILE *f, int n)
/*      display required record - simply copies characters to stdout*/
{
      char c;
      fseek(f,index[n],0);
      while(1)
      {
            putchar((c=getc(f)));
            if(c=='\n') return;
      }
}
```

A typical dialogue using the source file as input is shown below.

```
$ fil3 fil3.c
76 records
Enter record number 44 or 0 if the file has too many
Enter record number 11
      FILE    *d;     /* the data file */
Enter record number 76
Out of range
Enter record number 21
            exit(1);
```

Enter record number –1

There are a number of interesting points here. The direct access functions always work with long integers and it is traditional to declare the associated variables as being of type long int. The record numbering starts at zero and the file examination part of the program is terminated by a negative input. Strictly the final parameter of fseek() ought to have been SEEK_SET not zero.

The value returned by ftell() is the byte position of the byte about to be read from the file so when a newline is encountered this is the start address of the next record.

The functions fsetpos() and fgetpos() do the same things as fseek() and ftell() only they use parameters of type fpos_t rather than long int. This, potentially, allows for larger files to be handled and the use of these functions is to be preferred.

Exercise: Check Your Progress 3

Note: i) Use the space below for your answer.

Ex1: Define Direct Access File.

……..……

……..……...….

……..

9.11 Summary

- The C communicates with files using a new data type called a file pointer.
- File I/O is done using a buffer to improve the efficiency.
- A file is a text file or a binary file depending upon its contents.
- Library functions convert \n to \r\n or vice versa while writing/reading to\form a file.
- The fgets() function reads input through the first newline character.

9.12 Keywords

- ***FILE***:It is a file data structure and must be written in upper case letters.
- ***fprintf(),fputc() and fputs()***: All these function are just like the normal output function.
- ***fscanf()***:.It is a read the formatted data from the file.
- ***fseek***: It is a function is used to move the file position to a desired location within the file.
- ***Pointer_to_file***: It is a pointer veriable which holds the starting address of the data file.

9.13 Review Questions

1. What is the purpose of the library function fflush()?
2. What is a file handling?
3. Is it necessary that a file created in text mode must always be opened in text mode for subsequent operations?
4. How do you write data in to a file?
5. Explain different types of access modes.
6. Explain any two functions used to read data from the file.
7. What is a direct access file?
8. How to using fread() and fwrite().?

9. Point out the errors in the programs.

```
Main()
{
FILE *fp;
Char str[80];
fp=fopen("TRY.C","r");
while(fgets(str,80,fp)!=EOF)
fputs(str);
fclose(fp);
}
```

10. Explain the getw and putw functions.

Answers for Self Assessment Questions

1. (d) 2. (a) 3. (a) 4. (c) 5. (c)

9.14 Further Studies

1) Programing with c by Bala gura swami.
2) http://www.facweb.iitkgp.ernet.in/~isg/PDS/SLIDES/L10.pdf

10
Analysis of Algorithm

STRUCTURE

10.0 Objectives

10.1 Introducation

10.2 Definition and Properties of Algorithm

10.3 Analysis of Algorithms

10.4 Asymptotic Notation

10.5 Summary

10.6 Keywords

10.7 Review Questions

10.8 Further Studies

10.0 Objectives

After studying this chapter, you will be able to:

- Discuss the definition and properties of algorithm
- Explain the flow chart symbols
- Discuss the conversion of flow chart to language
- Discuss the example of simple algorithms
- Explain the program design
- Define the error of C programming

10.1 Introduction

When you write a program, you have to tell the computer every small detail of what to do. And you have to get everything exactly right, since the computer will blindly follow your program exactly as written. How, then, do people write any but the most simple programs? It is not a big mystery, actually. It is a matter of learning to think in the right way. A program is an expression of an idea. A programmer starts with a general idea of a task for the computer to perform. Presumably, the programme has some idea of how to perform the task by hand, at least in general outline. The problem is to flesh out that outline into a complete, unambiguous, step-by-step procedure for carrying out the task. Such a procedure is called an "algorithm". (Technically, an algorithm is an unambiguous, step-by-step procedure that terminates after a finite number of steps; we do not want to count procedures that go on forever.) An algorithm is not the same as a program. A program is written in some particular programming language. An algorithm is more like the idea behind the program, but it is the idea of the steps the program will take to perform its task, not just the idea of the task itself. When describing an algorithm, the steps do not necessarily have to be specified in complete detail, as long as the steps are unambiguous and it is clear that carrying out the steps will accomplish the assigned task. An algorithm can be

expressed in any language, including English. Of course, an algorithm can only be expressed as a program if all the details have been filled in.

10.2 Definition and Properties of Algorithm

"An algorithm is procedure consisting of a finite set of unambiguous rules (instructions) which specify a finite sequence of operations that provides the solution to a problem, or to a specific class of problems for any allowable set of input quantities (if there are inputs). In other word, an algorithm is a step-by-step procedure to solve a given problem." Alternatively, we can define an algorithm as a set or list of instructions for carrying out some process step-by-step. A recipe in a cookbook is an excellent example of an algorithm. The recipe includes the requirements for the cooking or ingredients and the method of cooking them until you end up with a nice cooked dish. In the same way, algorithms executed by a computer can combine millions of elementary steps, such as additions and subtractions, into a complicated mathematical calculation. Also by means of algorithms, a computer can control a manufacturing process or coordinate the reservations of an airline as they are received from the ticket offices all over the country. Algorithms for such large-scale processes are, of course, very complex, but they are built up from pieces. One of the obstacles to overcome in using a computer to solve your problems is that of translating the idea of the algorithm to computer code (program). People cannot normally understand the actual machine code that the computer needs to run a program, so programs are written in a programming language such as C or Pascal, which is then converted into machine code for the computer to run. In the problem solving phase of computer programming, you will be designing algorithms. This means that you will have to be conscious of the strategies you use to solve problems in order to apply them to programming problems. These algorithms can be designed though the use of flowcharts or pseudo code.

In addition every algorithm must satisfy the following criteria:

Finiteness: An algorithm terminates after a finite numbers of steps.

Definiteness: Each step in algorithm is unambiguous. This means that the action specified by the step cannot be interpreted (explain the meaning of) in multiple ways and can be performed without any confusion.

Input: An algorithm accepts zero or more inputs

Output: It produces at least one output.

Effectiveness: It consists of basic instructions that are realizable. This means that the instructions can be performed by using the given inputs in a finite amount of time

Did You Know?

The term algorithm originally referred to any computation performed via a set of rules applied to numbers written in decimal form. The word is derived from the phonetic pronunciation of the last name of *Abu Ja'far Mohammed ibn Musa al-Khowarizmi*, who was an Arabic mathematician who invented a set of rules for performing the four basic arithmetic operations (addition, subtraction, multiplication and division) on decimal numbers.

10.3 Analysis of Algorithms

There may be several different ways to solve a particular problem. For example, there are several methods for sorting numbers. How can you decide which method is the best in a certain situation? How would you define "best" – is it the fastest method or the one that takes up the least amount of memory space?

Understanding the relative efficiencies of algorithms designed to do the same task is very important in every area of computing. This is how computer scientists decide which algorithm to use for a particular application. In the 1950's and 60's, many mathematicians and computer scientists developed the field of **algorithm analysis**. One researcher in particular, Donald Knuth, wrote a three-volume text called The Art of Computer Programming that provided a foundation for the subject. Interestingly enough, Prof. Knuth also used to teach at Stanford until his retirement several years ago.

As mentioned earlier, an algorithm can be analyzed in terms of time efficiency or space utilization. We will consider only the former right now. The running time of an algorithm is influenced by several factors:

1) Speed of the machine running the program
2) Language in which the program was written. For example, programs written in assembly language generally run faster than those written in C or C++, which in turn tend to run faster than those written in Java.
3) Efficiency of the compiler that created the program
4) The size of the input: processing 1000 records will take more time than processing 10 records.
5) Organization of the input: if the item we are searching for is at the top of the list, it will take less time to find it than if it is at the bottom.

The first three items in the list are problematic. We don't want to use an exact measurement of running time: To say that a particular algorithm written in Java and running on a Pentium IV takes some number of milliseconds to run tells us nothing about the general time efficiency of the algorithm, because the measurement is specific to a given environment. The measurement will be of no use to someone in a different environment. We need a general metric for the time efficiency of an algorithm; one that is independent of processor or language speeds, or compiler efficiency.

The fourth item in the list is not environment-specific, but it is an important consideration. An algorithm will run slower if it must process more data but this decrease in speed is not because of the construction of the algorithm. It's simply because there is more work to do. As a result of this consideration, we usually express the running time of an algorithm as a function of the size of the input. Thus, if the input size is n, we express the running time as $T(n)$. This way we take into account the input size but not as a defining element of the algorithm.

Finally, the last item in the list requires us to consider another aspect of the input, which again is not part of the actual algorithm. To account for this, we express timing analyses in terms of "worst case", "average case" or "best case" based on the organization of the data, or the probability of finding an element quickly. For our purposes in the following sections, we will assume a "worst case" organization (i.e., we will not worry about the organization of the input for now).

Generalizing Running Time

The problem of generalizing a timing analysis is handled by not dealing with exact numbers but instead with order of magnitude or rate of growth. In other words:

How does the execution time of an algorithm grow as the input size grows?

Does one grow more quickly than the other – how much more? The ideal situation is one where the running time grows very slowly as you add more input. So, rather than deal with exact values, we keep it general by comparing the growth of the running time as the input grows, to the growth of known functions. The following functions are the ones typically used:

input size							
n	(1)	log n	n	n log n	n^2	n^3	2^n
5	1	3	5	15	25	125	32
10	1	4	10	33	100	10^3	10^3
100	1	7	100	664	10^4	10^6	10^{30}
1000	1	10	1000	10^4	10^6	10^9	10^{300}
10000	1	13	10000	10^5	10^8	10^{12}	10^{3000}

For small inputs, the difference in running times would hardly be noticeable on a fast computer. For inputs of 100,000, however, (if we assume one microsecond per instruction) an algorithm comparable to the (n log n) function will take about 1.7 CPU seconds, an n^2 algorithm will take about 2.8 CPU hours which is unacceptable, and an n^3 algorithm would take 31.7 CPU *years*. There is no way we would ever use such an algorithm to deal with large inputs.

Analyzing Running Time

T(n), or the running time of a particular algorithm on input of size n, is taken to be the number of times the instructions in the algorithm are executed. As an illustration, consider a pseudocode algorithm that calculates the mean (average) of a set of n numbers:

1. n = read input from user
2. sum = 0
3. i = 0
4. while i < n
5. number = read input from user
6. sum = sum + number
7. i = i + 1
8. mean = sum / n

Statements 1, 2, and 3 are each executed once. Statements 5, 6, and 7 are each executed n times. Statement 4 (which controls the loop) is executed n + 1 times (one additional check is required – why?), and statement 8 is executed once. This is summarized below:

Statement	Number of times executed
1	1
2	1
3	1
4	n+1
5	n
6	n
7	n
8	1

Thus, the computing time for this algorithm in terms of input size n is: T(n) = 4n + 5. We can see intuitively, that as n grows, the value of this expression grows linearly. We say T(n) has an "order of magnitude (or rate of growth) of n". We usually denote this using big-Oh notation: T(n) = O(n), and we say that the algorithm

has a **complexity** of O(n). In some cases, we might also say the algorithms has a "time complexity" of O(n) to distinguish the growth rate for the running time of the algorithm from the amount of memory, or space, that the algorithm would use during its execution. Of course, intuition is not enough for us skeptical computer science types – we must be able to show mathematically which of the standard functions given in the table above indicates the correct rate of growth.

10.3.1 Time complexity

In previous chapters we implemented a simple sorting algorithm. We used it there to sort characters; we also could sort numbers or strings with it because it is only based on comparisons and swapping. How long does this sorting program run? It possibly takes a very long time on large inputs (that is many strings) until the program has completed its work and gives a sign of life again. Sometimes it makes sense to be able to estimate the running time before starting a program. Obviously, the running time depends on the number n of the strings to be sorted. Can we find a formula for the running time which depends on n?

Having a close look at the program we notice that it consists of two nested for-loops. In both loops the variables run from 0 to n, but the inner variable starts right from where the outer one just stands. An if with a comparison and some assignments not necessarily executed reside inside the two loops. A good measure for the running time is the number of executed comparisons. In the first iteration n comparisons take place, in the second n-1, then n-2, then n-3 etc. So 1+2+...+n comparisons are performed altogether. According to the well known Gaussian sum formula these are exactly $1/2 \cdot (n-1) \cdot n$ comparisons. Figure 10.1 illustrates this. The screened area corresponds to the number of comparisons executed. It apparently corresponds approx. to half of the area of a square with a side length of n. So it amounts to approx. $1/2 \cdot n2$.

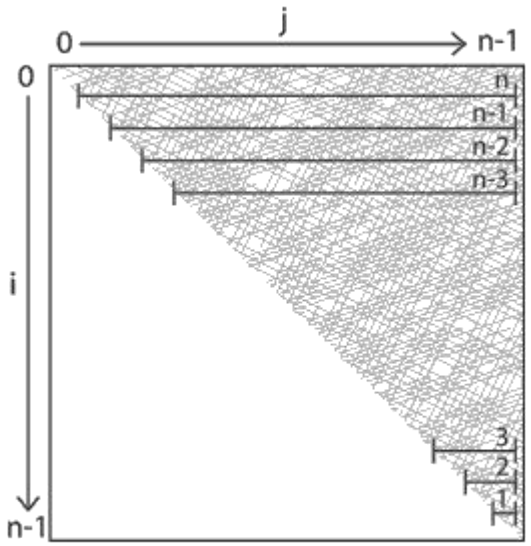


Figure 10.1. Running time analysis of sorting by minimum search

How does this expression have to be judged? Is this good or bad? If we double the number of strings to be sorted, the computing time quadruples! If we increase it ten-fold, it takes even 100 = 102 times longer until the program will have terminated! All this is caused by the expression n2. One says: Sorting by minimum search has quadratic complexity. This gives us a forefeeling that this method is unsuitable for large amounts of data because it simply takes far too much time.

So it would be a fallacy here to say: “For a lot of money, we'll simply buy a machine which is twice as fast, then we can sort twice as many strings (in the same time).” Theoretical running time considerations offer protection against such fallacies.

The number of (machine) instructions which a program executes during its running time is called its time complexity in computer science. This number depends primarily on the size of the program's input, that is approximately on the number of the strings to be sorted (and their length) and the algorithm used. So approximately, the time complexity of the program "sort an array of n strings by minimum search" is described by the expression c·n2.
c is a constant which depends on the programming language used, on the quality of the compiler or interpreter, on the CPU, on the size of the main memory and the access time to it, on the knowledge of the programmer, and last but not least on the algorithm itself, which may require simple but also time consuming machine instructions. So while one can make c smaller by improvement of external circumstances (and thereby often investing a lot of money), the term n2, however, always remains unchanged.

10.3.2 Space complexity

The better the time complexity of an algorithm is, the faster the algorithm will carry out his work in practice. Apart from time complexity, its space complexity is also important: This is essentially the number of memory cells which an algorithm needs. A good algorithm keeps this number as small as possible, too.
There is often a time-space-tradeoff involved in a problem, that is, it cannot be solved with few computing time and low memory consumption. One then has to make a compromise and to exchange computing time for memory consumption or vice versa, depending on which algorithm one chooses and how one parameterizes it.

Self Assessment Questions

1. A program is then just a specific realisation of an algorithm, which may be executed on a.....
(a) programmable device (b) physical device
(c) input (d) output

2. A program.....is a more formal and detailed description of the program's operation.
(a) flow and data structures (b) requirements
(c) bottom-up design (d) specification

3. A key limitation of .....is that the resulting functions tend to be problem-specific.
(a) flow and data structures (b) requirements
(c) top-down design (d) specification

4.....occur during the coding process.
(a) Logical errors (b) Run-time errors
(c) Device errors (d) Syntax errors

10.4 Asymptotic Notation

Suppose we are considering two algorithms, *A* and *B*, for solving a given problem. Furthermore, let us say that we have done a careful analysis of the running times of each of the algorithms and determined them to be $T_A(n)$ and $T_B(n)$, respectively, where *n* is a measure of the problem size. Then it should be a fairly simple matter to compare the two functions $T_A(n)$ and $T_B(n)$ to determine which algorithm is *the best*!

But is it really that simple? What exactly does it mean for one function, say $T_A(n)$, to be *better than* another function, $T_B(n)$? One possibility arises if we know the problem size *a priori*. For example, suppose the problem size is n_0 and $T_A(n_0) < T_B(n_0)$. Then clearly algorithm *A* is better than algorithm *B* for problem size n_0.

In the general case, we have no *a priori* knowledge of the problem size. However, if it can be shown, say, that $T_A(n) \leq T_B(n)$ for all $n \geq 0$, then algorithm *A* is better than algorithm *B* regardless of the problem size.

Unfortunately, we usually don't know the problem size beforehand, nor is it true that one of the functions is less than or equal the other over the entire range of problem sizes. In this case, we consider the ***asymptotic* behavior** of the two functions for very large problem sizes.

10.4.1 Big-Oh Notation

Definition 1: Let f(n) and g(n) be two functions. We write:

f(n) = O(g(n)) or f = O(g)

(read "f of n is big oh of g of n" or "f is big oh of g") if there is a positive integer C such that f(n) <= C * g(n) for all positive integers n.

The basic idea of big-Oh notation is this: Suppose f and g are both real-valued functions of a real variable x. If, for large values of x, the graph of f lies closer to the horizontal axis than the graph of some multiple of g, then f is of order g, i.e., f(x) = O(g(x)). So, g(x) represents an ***upper bound*** on f(x).

Example 1:

Suppose f(n) = 5n and g(n) = n. To show that f = O(g), we have to show the existence of a constant C as given in Definition 1. Clearly 5 is such a constant so f(n) = 5 * g(n). We could choose a larger C such as 6, because the definition states that f(n) must be less than or equal to C * g(n), but we usually try and find the smallest one. Therefore, a constant C exists (we only need one) and f = O(g).

Example 2:

In the previous timing analysis, we ended up with T(n) = 4n + 5, and we concluded intuitively that T(n) = O(n) because the running time grows linearly as n grows. Now, however, we can prove it mathematically:

To show that f(n) = 4n + 5 = O(n), we need to produce a constant C such that:

f(n) <= C * n for all n.

If we try C = 4, this doesn't work because 4n + 5 is not less than 4n. We need C to be at least 9 to cover *all* n. If n = 1, C has to be 9, but C can be smaller for greater values of n (if n = 100, C can be 5). Since the chosen C must work for all n, we must use 9:

4n + 5 <= 4n + 5n = 9n

Since we have produced a constant C that works for all n, we can conclude:

T(4n + 5) = O(n).

Example 3:
Say $f(n) = n^2$: We will prove that $f(n) \in O(n)$. To do this, we must show that there cannot exist a constant C that satisfies the big-Oh definition. We will prove this by contradiction.
Suppose there is a constant C that works; then, by the definition of big-Oh: $n^2 <= C * n$ for **all** n. Suppose n is any positive real number greater than C, then: $n * n > C * n$, or $n^2 > C * n$. So there exists a real number n such that $n^2 > C * n$. This contradicts the supposition, so the supposition is false. There is no C that can work for all n: $f(n) \in O(n)$ when $f(n) = n^2$.

Example 4:
Suppose $f(n) = n^2 + 3n - 1$. We want to show that $f(n) = O(n^2)$.

$f(n) = n^2 + 3n - 1$
$< n^2 + 3n$ (subtraction makes things smaller so drop it)
$<= n^2 + 3n^2$ (since $n <= n^2$ for all integers n)
$= 4n^2$

Therefore, if C = 4, we have shown that $f(n) = O(n^2)$. Notice that all we are doing is finding a simple function that is an upper bound on the original function. Because of this, we could also say that $f(n) = O(n^3)$ since (n^3) is an upper bound on n^2. This would be a much weaker description, but it is still valid.

Example 5:
Show: $f(n) = 2n^7 - 6n^5 + 10n^2 – 5 = O(n^7)$
$f(n) < 2n^7 + 6n^5 + 10n^2$
$<= 2n^7 + 6n^7 + 10n^7$
$= 18n^7$

thus, with C = 18 and we have shown that $f(n) = O(n^7)$
You are probably noticing a pattern here. **Any polynomial is big-Oh of its *term of highest degree*.** We are also ignoring constants. A proof of this is very straightforward. Any polynomial (including a general one) can be manipulated to satisfy the big-Oh definition by doing what we did in the last example: take the absolute value of each coefficient (this can only increase the function); Then since

$n^j <= n^d$ if $j <= d$

we can change the exponents of all the terms to the highest degree (the original function must be less than this too). Finally, we add these terms together to get the largest constant C we need to find a function that is an upper bound on the original one.
Big-Oh, therefore is a useful method for characterizing an approximation of the upper bound running time of an algorithm. By ignoring constants and lower degree terms, we end up with the approximate upper bound. In many cases, this is sufficient. Sometimes, however, it is insufficient for comparing the running times of two algorithms. For example, if one algorithm runs in O(n) time and another runs in $O(n^2)$ time, you cannot be sure which one is fastest for large n. Presumably the first is, but perhaps the second algorithm was not analyzed very carefully. It is important to remember that big-Oh notation gives no tight information on how good an algorithm is, it just gives an ***upper bound*** on how bad it can be.

There is another notation that is sometimes useful in the analysis of algorithms. To specify a ***lower bound*** on the growth rate of T(n), we can use big-Omega (Ω) notation. If the lower bound running time for an algorithm

is g(n) then we say T(n) = Ω(n). As an example, any algorithm with m inputs and n outputs that uses all the inputs to generate the output would require at least Ω(m + n) work. In a rough sense, big-Omega notation tells us that an algorithm requires at least this much time to run, hence it is a ***lower bound*** on the running time or alternatively can be thought of as the *best case* running time for the algorithm.

One last variation on this theme is big-Theta (Θ) notation. Big-Θ bounds a function from both ***above and below***, so two constants must be provided rather than one if we are doing formal proofs of the bounds of a given function.

We'll discuss big-Omega and big-Theta notation a bit more below. In the meantime, there is one adjustment that we must make to our definition of big-Oh. You may have noticed that we have been avoiding logarithms in the discussion so far. We cannot avoid them for long, however, because many algorithms have a rate of growth that matches logarithmic functions (a few of these include binary search, MergeSort, and QuickSort). As a brief review, recall that $\log_2 n$ is the number of times we have to divide n by 2 to get 1; or alternatively, the number of 2's we must multiply together to get n:

$$n = 2^k \quad \square \quad \log_2 n = k$$

Many "Divide and Conquer" algorithms solve a problem by dividing it into 2 smaller problems, then into 2 even smaller problems. You keep dividing until you get to the point where solving the problem is trivial. This constant division by 2 suggests a logarithmic running time.

Thus, since log(1) = 0, there is no constant C such that 1 <= C * log(n) for all n. Note, however, that for n >= 2, it is the case that 1 <= log(n) and so the constant C = 1 works for sufficiently large n (larger than 1). This suggests that we need a stronger definition of big-Oh than given previously.

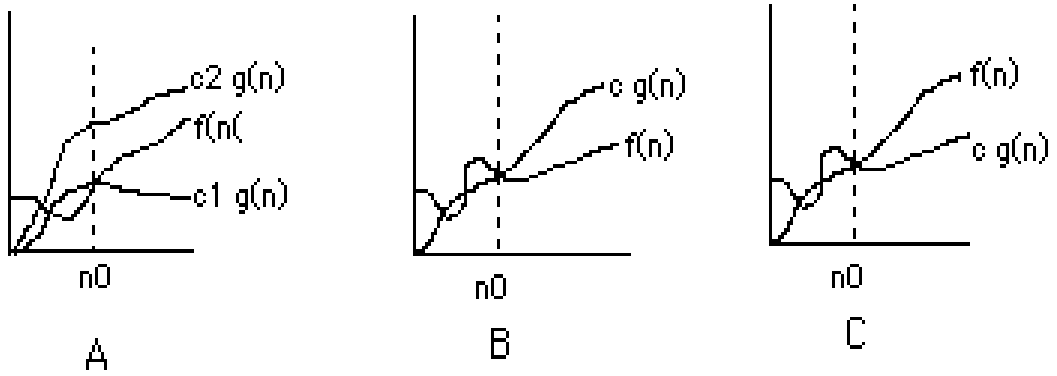
Definition 2: Let f(n) and g(n) be two functions. We write:

$$f(n) = O(g(n)) \quad \text{or} \quad f = O(g)$$

if there are positive integers C and N such that f(n) <= C * g(n) for all integers n >= N.

Using this more general definition for big-Oh, we can now say that if we have f(n) = 1, then f(n) = O(log(n)) since C = 1 and N = 2 will work.

With this definition, we can clearly see the difference between the three types of notation:



In all three graphs above, n0 is the minimal possible value to get valid bounds, but any greater value will work. Figure A shows big-Θ, which bounds a function between two constant factors (i.e., giving a tight ***lower and upper bound***). We can write f(n) = Θ(g(n)) if there exist positive constants n0, c1 and c2 such that to the right

of n0, the value of f(n) always lies between c1 * g(n) and c2 * g(n) (inclusive). Thus, in order to prove that a function is Θ(g(n)), we must prove that the function is *both* O(g(n)) and Ω (g(n)).
Figure B shows big-Oh, which gives an ***upper bound*** for a function to within a constant factor. We can write f(n) = O(g(n)) if there exist positive constants n0 and c such that to the right of n0, f(n) always lies on or below c * g(n).
Finally, Figure C shows big-Ω, which gives a ***lower bound*** for a function to within a constant factor. We can write f(n) = Ω (g(n)) if there exist positive constants n0 and c such that to the right of n0, f(n) always lies on or above c * g(n).
There is a handy theorem that relates these notations:
Theorem: For any two functions f(n) and g(n), f(n) = □ (g(n)) if and only if f(n) = O(g(n)) and f(n) = □ (g(n)).

Example 6:
Show: $f(n) = 3n^3 + 3n - 1 = \Theta(n^3)$
As implied by the theorem above, to show this result, we must show two properties:

(i) $f(n) = O(n^3)$
(ii) $f(n) = \Omega(n^3)$

First, we show (i), using the same techniques we've already seen for big-Oh. We consider N = 1, and thus we only consider n >= 1 to show the big-Oh result.

$$\begin{aligned} f(n) &= 3n^3 + 3n - 1 \\ &< 3n^3 + 3n + 1 \\ &<= 3n^3 + 3n^3 + 1n^3 \\ &= 7n^3 \end{aligned}$$

thus, with C = 7 and N = 1, we have shown that $f(n) = O(n^3)$.
Next, we show (ii). Here we must provide a lower bound for f(n). Here, we choose a value for N, such that the highest order term in f(n) will always dominate (be greater than) the lower order terms. We choose N = 2, since for n >=2, we have n^3 >= 8. This will allow n^3 to be larger than the remainder of the polynomial (3n - 1) for all n >= 2. So, by subtracting an extra n^3 term, we will form a polynomial that will always be less than f(n) for n >= 2.

$$\begin{aligned} f(n) &= 3n^3 + 3n - 1 \\ &> 3n^3 - n^3 \qquad \text{since } n^3 > 3n - 1 \text{ for any } n >= 2 \\ &= 2n^3 \end{aligned}$$

Thus, with C = 2 and N = 2, we have shown that $f(n) = \Omega(n^3)$, since f(n) is shown to always be greater than $2n^3$.

10.4.2 Big Omega Notation

The big oh notation introduced in the preceding section is an asymptotic *upper bound*. In this section, we introduce a similar notation for characterizing the asymptotic behavior of a function, but in this case it is a *lower bound*.

Definition (Omega) Consider a function *f*(*n*) which is non-negative for all integers $n \geq 0$ We say that ``*f*(*n*) is omega *g*(*n*)," which we write $f(n) = \Omega(g(n))$ if there exists an integer n_0 and a constant *c*>0 such that for all integers , $n \geq n_0$. $f(n) \geq cg(n)$

The definition of omega is almost identical to that of big oh. The only difference is in the comparison--for big oh it is $f(n) \leq cg(n)$ for omega, it is $f(n) \geq cg(n)$ All of the same conventions and caveats apply to omega as they do to big oh.

Example:
Consider the function $f(x) = 5n^2 - 64n + 256$ which is shown in Figure ❑. Clearly, *f*(*n*) is non-negative for all integers $n \geq 0$. We wish to show that $f(n) = \Omega(n^2)$. According to Definition ❑, in order to show this we need to find an integer n_0 and a constant *c*>0 such that for all integers $n \geq n_0$, $f(n) \geq cn^2$.
As with big oh, it does not matter what the particular constants are--as long as they exist! For example, suppose we choose *c*=1. Then

$$\begin{aligned} f(n) \geq cn^2 &\Rightarrow 5n^2 - 64n + 256 \geq n^2 \\ &\Rightarrow 4n^2 - 64n + 256 \geq 0 \\ &\Rightarrow 4(n-8)^2 \geq 0. \end{aligned}$$

Since $(n-8)^2 > 0$ for all values of $n \geq 0$, we conclude that $n_0 = 0$.

So, we have that for *c*=1 and $n_0 = 0$, $f(n) \geq cn^2$ for all integers $n \geq n_0$. Hence, $f(n) = \Omega(n^2)$. Figure ❑ clearly shows that the function $f(n) = n^2$ is less than the function *f*(*n*)=5*n*-64*n*+256 for all values of $n \geq 0$. Of course, there are many other values of *c* and n_0 that will do. For example, *c*=2 and $n_0 = 16$.

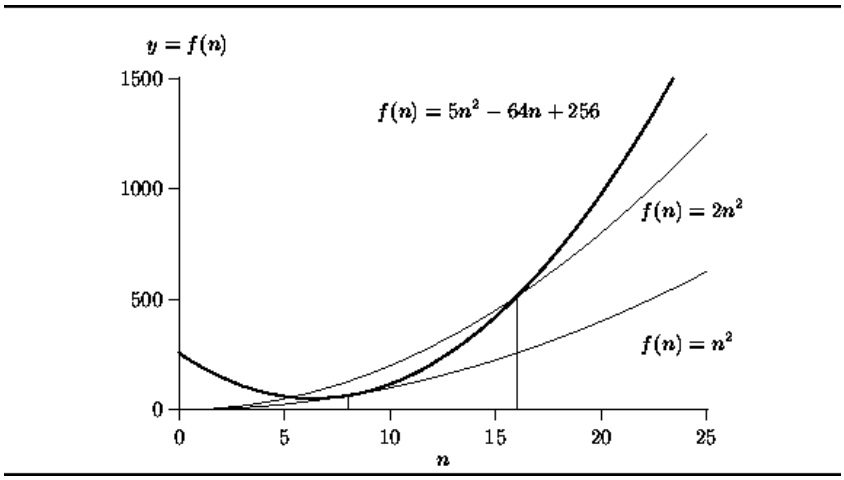


Figure: Showing that $f(n) = 4n^2 - 64n + 288 = \Omega(n^2)$

Figure 10.2 Big-Theta Notation

The Θ notation describes asymptotic tight bounds.

$f(n)$ is $\Theta(g(n))$ iff $\exists$ positive real constants C_1 and C_2 and a positive integer n_0, such that

$$C_1 g(n) \leq f(n) \leq C_2 g(n) \quad \forall \; n \geq n_0$$

Theta means that f is bounded above and below by g; Big Theta implies the "best fit". f(n) does not have to be linear itself in order to be of linear growth; it just has to be between two linear functions. We will use Theta whenever we have enough information to show that the f(n) is both an upper and lower bound. Theta is a "stronger" statement than Big-Oh or Big-Omega.

Example: f(n) = n2 - 5n + 13.

The constant 13 doesn't change as n grows, so it is not crucial. The low order term, -5n, doesn't have much effect on f compared to the quadratic term, n2.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Define Time complexity.

……..……

……..……...…

……..

Ex2: What is Big Omega Notation?

……..……

……..……...…

……..

10.5 Summary

- A program is an expression of an idea. A programmer starts with a general idea of a task for the computer to perform.
- A computer is essentially a physical device designed to carry out a collection of primitive actions.
- A program specification is a more formal and detailed description of the program's operation.
- Top-down design is useful for defining the overall structure of a program, and the dependencies between functions.

10.6 Keywords

Action: An action symbol contains a set of instructions in C code. The syntax is the one of C language.

Assembly language: An assembly language is a low-level programming language for computers, microprocessors, microcontrollers, and other programmable devices.

Pseudocode: Pseudocode is an artificial and informal language that helps programmers develops algorithms. Pseudocode is a "text-based" detail (algorithmic) design tool.

Top-down design: Top-down design is a methodology that starts at the highest level of a design concept and proceeds towards the lowest level.

10.7 Review Questions

1. What is a flow chart? Give its classification.
2. What is an algorithm? Explain its salient features.
3. What is debugging?
4. What is programming design?

5. Explain difference between syntax and logical error.
6. Explain the errors and its types.
7. Write the process of convert flow chart to program.
8. Differentiate between algorithm and flowchart.
9. Explain syntax error with suitable example.
10. Write a process to convert program to flowchart.

Answers for Self Assessment Questions

1. (c) 2. (a) 3. (c) 4. (b)

10.8 Review Questions

1) http://elearning.vtu.ac.in/12/enotes/ADA/Unit2A-RMS.pdf
2) Let us c by Yashwant Kanatkar.

Programming in 'C'

&

Data Structure

(MCA/PGDCA - 104)



Jaipur National University
Directorate of Distance Education

Established by Government of Rajasthan
Approved by UGC under Sec 2(f) of UGC ACT 1956
(**Recognised by Joint Committee of UGC-AICTE-DEC, Govt. of India**)

1
Linked List

STRUCTURE

1.0 Objectives

1.1 Introduction

1.2 List and Arrays

1.3 Linked List

1.4 Types of Linked Lists

1.5 Header Node in Linked List

1.6 Application of Linked Lists

1.7 Addition of Two Long Positive Numbers

1.8 Evaluation of a Polynomial

1.9 Summary

1.10 Keywords

1.11 Review Questions

1.12 Further Studies

1.0 Objectives

After studying this chapter, you will be able to:

- Explain the data structures
- Discuss the operations of linked list
- Explain the types of linked list
- Discuss the header node of linked list
- Explain the applications on linked lists
- Evaluate a polynomial

1.1 Introduction to Data Structures

Data structure has a different way of storing and organizing data in a computer so that it can use efficiently. Different kind of Data structure is used in different application. For example, B-tree is widely used in implementation of databases.

Data Structure in C

A memory play a vital role in C because all variable you declare are at first stored in the memory. Some memory allocation is done at compile time and some at runtime. Memory which is allocated and de-allocated during the execution of the program is known as dynamic memory. Memory spans and shrinks as per application requirement and this is one of major advantage of pointer.

A memory can be categories in following segments:

- Text Segment
- Data Segment
- Free Space
- Stack Segment

Data Segment in sub-categories in two parts:

- Static Segment
- Heap Segment

The Static Segment holds the global variables and the Heap Segment hold the dynamic variable. Dynamic memory allocation is done by using malloc () function in C.

1.2 List and Arrays

List is collection of Scalar variables (Heterogeneous), which are arranged in an order. Array is the variable (homogenous) which stores the elements. A number of connected items or names written or printed consecutively, typically one below the other. A list can also be said as array but there is a difference between arrays and list is that an array contains the elements on continuous memory allocations.

An array is a series of elements of the same type placed in contiguous memory locations that can be individually referenced by adding an index to a unique identifier.

An array is a data structure of multiple elements with the same data type. Array elements are accessed using subscript. The valid range of subscript is 0 to size –1.

1.2.1 Single Dimension Arrays

Arrays can be declared using any of the data types available in C. A single dimensional array will be useful for simple grouping of data that is relatively small in size. You can declare a single dimensional array as follows:

Declaration of Array:

int arr[10];

Example:

```
#include<stdio.h>
void main(void)
{
int a[5];
int i;
for(i = 0;i<5;i++)
{
a[i]=i;
}
for(i = 0;i<5;i++)
{
printf("%d value of i is = %d\n", i, a[i]);
```

```
}
}
```

Output:
1 value of i is = 0
2 value of i is = 1
3 value of i is = 2
4 value of i is = 3
5 value of i is = 4

1.2.2 Multidimensional Arrays

A multi-dimensional array of dimension n (i.e., an n-dimensional array or simply n-D array) is a collection of items which is accessed via n subscript expressions. Multidimensional arrays can be described as "arrays of arrays".

Example:

```
#include<stdio.h>
void main(void)
{
int a[3][2];
int i, j;
for (i = 0;i<3;i++)
{
for (j=0;j<2 ;j++)
{
scanf("%d", &a[i][j]);
}
}
for(i = 0;i<3;i++)
{
for(j=0;j<2;j++)
{
printf("value in array %d\n", a[i][j]);
}
}
}
```

Caution

Array size must be declared using constant value before initialization.

Did You Know?

In the data structure name and symbol for the floor function and ceiling function was invented by K. E. Iverson.

Self Assessment Questions

1. Direct or random access of elements is not possible in…….

(a). Linked list (b). Array (c). String (d). None of these.

1.3 Linked List

Linked list is a very common data structure often used to store similar data in memory. While the element of an array occupies contiguous memory location, those of a linked list are not constrained to be stored in adjacent location. The individual elements are stored "somewhere" in memory, rather like a family dispersed, but still bound together. The order of the elements is maintained by explicit links between them. For instance, the marks obtained by different student can stored in linked list.

1.3.1 Operation on Linked Lists

This section gives pseudo code for adding or removing nodes from linear/singly, doubly, and circularly linked lists in-place. Throughout we will use null to refer to an end-of-list marker or sentinel, which may be implemented in a number of ways.

Our node data structure will have two fields. We also keep a variable head which always points to the first node in the list, or is null for an empty list.

```
record Node
{
   data // The data being stored in the node; often a reference to the actual data
   next // A reference to the next node
 }
Node head // points to front of list
Node node Being Removed
```

1.3.2 Singly Linked List

Traversal of a linear linked list is easy, beginning at the first node and following each next link until we come to the end:

```
node := head // start at front of list
 while node not null { // loop through list
    (do something with node.data)
    node := node.next // go to next node in list }
```

The following code inserts a node after an existing node in a linear linked list. Inserting a node before an existing one cannot be done; instead, you have to locate it while keeping track of the previous node.

Singly_linked_list_insert_after.png

```
function insertAfter(Node node, Node newNode) { // insert newNode after node
    newNode.next := node.next // point new node to next node
    node.next    := newNode    // add new node to list }
```

Inserting at the beginning of the list requires a separate function. This requires updating head.

```
function insertBeginning(Node newNode) { // insert node before current head
  newNode.next := head    // point to rest on list
  head := newNode // new front of list }
```

Similarly, we have functions for removing the node after a given node, and for removing a node from the beginning of the list. To find and remove a particular node, one must again keep track of the previous element.

Singly_linked_list_delete_after.png

```
function removeAfter(Node node) { // remove node past this one
  nodeBeingRemoved:= node.next    // this is the node to delete
  node.next:= node.next.next      // point past node
 (the above statement could be node.next = nodeBeingRemoved.next)
  destroy nodeBeingRemoved        // free up deleted node }
```

```
function removeBeginning(Node node) { // remove head node
  nodeBeingRemoved:= head   // this is the node to delete
  head:= head.next          // point past deleted node
  destroy nodeBeingRemoved   // free up deleted node }
```

Notice that removeBeginning () sets head to null when removing the last node in the list.

Did You Know?

The linked lists were developed by Allen Newell, Cliff Shaw and Herbert Simon at RAND Corporation as the primary data structure for their Information Processing Language (IPL).

1.3.3 Doubly-linked Lists

With doubly-linked lists there are even more pointers to update, but also less information is needed, since we can use backwards pointers to observe preceding elements in the list. This enables new operations, and eliminates special-case functions. We will add a prevfield to our nodes, pointing to the previous element, and a tail variable which always points to the last node in the list. Both head and tail are null for an empty list.

Iterating through a doubly linked list can be done in either direction. In fact, direction can change many times, if desired.

Forwards:

```
node:= head
 while node ≠ null
    <do something with node.data>
    node:= node.next
```

Backwards:

```
node:= tail
 while node ≠ null
    <do something with node.data>
    node:= node.prev
```

These symmetric functions add a node either after or before a given node.

Doubly_linked_list_insert_after.png

```
function insertAfter(node, newNode)
    newNode.prev:= node
    newNode.next:= node.next
    if node.next = null
       tail:= newNode
    else
       node.next.prev:= newNode
    node.next:= newNode
 function insertBefore(node, newNode)
    newNode.prev := node.prev
    newNode.next := node
    if node.prev is null
       head := newNode
    else
       node.prev.next := newNode
    node.prev    := newNode
```

We also need a function to insert a node at the beginning of a possibly-empty list:

```
function insertBeginning(newNode)
```

```
    if head = null
       head := newNode
       tail := newNode
       newNode.prev := null
       newNode.next := null
    else
       insertBefore(head, newNode)
```

A symmetric function inserts at the end:

```
function insertEnd(newNode)
    if tail = null
       insert Beginning(newNode)
    else
       insert After(tail, new Node)
```

Removing a node is easier, only requiring care with head and tail:

```
function remove(node)
  if node .prev = null
     head = node. next
  else
     node.prev.next = node.next
   if node.next = null
     tail := node.prev
  else
     node.next.prev = node.prev
  destroy node
```

One subtle consequence of this procedure is that deleting the last element of a list sets both head and tail to null.

1.3.4 Combine Operation on Linked Lists

The following program shows how to build a link list by adding new nodes at the beginning, at the beginning, at the end or in the middle of the linked list. It also contains a function display() which displays all the nodes present in the linked list and function del() which can delete any node in the linked list. Go through the program carefully, a step at a time.

```
    #include <stdio.h>
    #include <conio.h>
    #include<malloc.h>
    #include <windows.h>
/* structure containing a data part and link part */
     struct node
    {
    int data ;
    struct node * link ;
    };
    void append ( struct node **, int ) ;
    void addatbeg ( struct node **, int ) ;
    void addafter ( struct node *, int, int ) ;
    void display ( struct node * ) ;
```

```
int count ( struct node * ) ;
void del ( struct node **, int ) ;
int main( )
{
struct node *p ;
p = NULL; /* empty linked list */

printf (“No. of elements in the Linked List = %d\n”, count ( p ) ) ;
append (&p, 14 ) ;
append (&p, 30 ) ;
append (&p, 25 ) ;
append (&p, 42);
append (&p, 17);
system (“cls” );
display (p);
addatbeg (&p, 999 ) ;
addatbeg ( &p, 888 ) ;
addatbeg ( &p, 777 ) ;
display (p) ;
addafter ( p, 7, 0 );
addafter ( p, 2, 1 );
addafter ( p, 5, 99);
display (p);
printf (“No. of elements in the Linked List = %d\n”, count ( p));
del ( &P, 99);
del ( &p, 1 ) ;
dell ( &p, 10);
display (P);
printf ( “No. of elements in the linked list= %d\n”, count (p));
return 0 ;
}
/* adds a node the end of a linked list */
void append (struct node **q, int num )
{
struct node *temp, *r;
if (*q = = NULL ) /* if the list is empty, create first node */
{
temp = ( struct node * ) malloc ( sizeof ( struct node ) ) ;
temp -> data = num ;
temp -> link = NULL;
*q = temp ;
}
else
{
temp = *q ;
/* go to last node */
while (temp -> link != NULL)
```

```
        temp = temp -> link ;
        /* add node at the end */
        r = ( struct node * ) malloc ( sizeof ( struct node ) ) ;
        r-> data = num ;
        r-> link = NULL ;
        temp -> link = r ;
        }
        }

/* adds a new node at the beginning of the linked list */
        void addatbeg ( struct node **q, int num )
        {
        struct node *temp;
        /*add new node*/
        temp = (struct node *) malloc (sizeof ( struct node));
        temp -> data = num ;
        temp -> link = *q ;
        *q=temp;
        }

/*adds a new node after the specified number of nodes*/
        void addafter ( struct node *q, int loc, int num )
        {
        struct node *temp, *r;
        int i;
        temp = q ;
        /* skip to desired portion */
        for ( i = 0 ; i < loc; i++)
        {
        temp = temp -> link ;

        /* if end of linked list is encountered */
        if ( temp == NULL)
        {
        printf (“There are less than %d elements in list\n”, loc);
        return;
        }
        }
       /*insert new node*/
       r = (struct node *) malloc (sizeof (struct node));
       r -> data = num;
       r ->link= temp -> link;
       temp -> link = r;
       }

/* displays the contents of the linked list*/
              void display ( struct node *q )
```

```
        {
        /* traverse the entire linked list */
        while (q != NULL)
        {
       printf ("%d", q -> data ) ;
       q=q->link ;
        }
       printf ( "In" ) ;
        }

/* counts the number of nodes present in the linked list */
        int count ( struct node * q)
        {
        int c=0;
        /* traverse the entire linked list */
        while (q != NULL)
        {
                q=q->link;
                c++ ;
         }
       return c;
       }

/* deletes the specified node from the linked list */
    void del ( struct node **q, int num)
    {
    struct node *old, *temp;
    temp = *q;

    while ( temp != NULL)
    {
    if (temp -> data = = num)
    {
    /* if node to be deleted is the first node in the linked list */
    if ( temp = = *q )
    *q = temp -> link ;
    /* deletes the intermediate nodes in the linked list */
    else
    old -> link = temp -> link ;

    /* free the memory occupied by the node */
    free ( temp);
    return;
    }           /* traverse the linked list till the last node is reached */
        else
            {
            old = temp ; /* old points to the previous node */
```

```
            temp = temp -> link; /* go to the next node */
        }
    }
    printf ("Element %d not found\n", num ) ;
    }
```

Output:

```
14 30 25 42 17
 777 888 99914 30 25 4217
777 888 999 1 14 30 99 25 42 17 0
No. of elements in the Linked List = 11
Element 10 not found
777 888 999 14 30 25 42 17 0
No. of elements in the linked list = 9
```

To begin with we have defined a structure for a node. It contains a data part and a link part. The variable p has been declared as pointer to a node. We have used this pointer as pointer to the first node in the linked list. No matter how many nodes get added to the linked list, p would continue to pointer to the first node in the list. When no node has been added to the list, p has been set to NULL to indicate that the list is empty.

The append( ) function has to deal with two situations:

(a) The node is being added to an empty list.

(b) The node is being added at the end of an existing list.

In the first case, the condition

if (*q== NULL) gets satisfied. Hence, space is allocated for the node using malloc( ).

Data and the link part of this node are set up using the statements.

```
temp -> data = num;
temp -> link = NULL;
```

Lastly, p is made to point to this node, since the first node has been added to the list and p must always point to the first node. Note that *q is nothing but equal to p.

In the other case, when the linked list is not empty, the condition

if (*q = = NULL) -

would fail, since *q (i.e. p is non-NULL). Now temp is made to point to the first node in the list through the statement.

```
temp = *q ;
```

Then using temp we have traversed through the entire linked list using the statements.

```
while (temp -> link != NULL)
temp = temp -> link ;
```

The position of the pointers before and after traversing the linked list is shown in Figure 1.1.

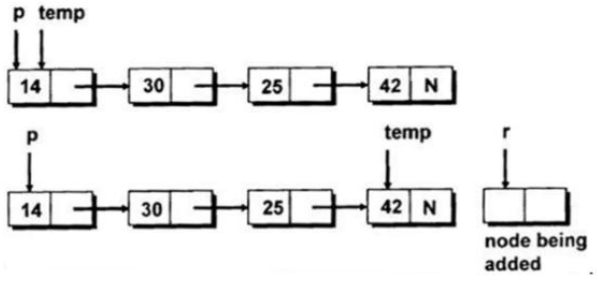


Figure 1.1: Working of append() function**.**

Each time through the loop the statement temp = temp -> link makes temp point to the next node in the list. When temp reaches the last node condition temp -> link != NULL would fail. Once outside the loop we allocate space for the new node through the statement

r = (struct node *) malloc (sizeof (struct node)) ;

Once the space has been allocated for the new node its data part is stuffed with num and the link part with NULL. Note that this node is now going to be the last Node in the list.

All that now remains to be done is connecting the previous last node with the new last node. The previous last node is being pointed to by temp and the new last node is being pointed to by r. They are connected through the statement.

temp -> link = r ;

this link gets established.

There is often confusion as to how the statement, temp = temp -> link makes temp point to the next node in the list. Let us understand this with the help of an example. Suppose in a linked list containing 4 nodes, temp is pointing at the first node. This is shown in Figure 1.2.

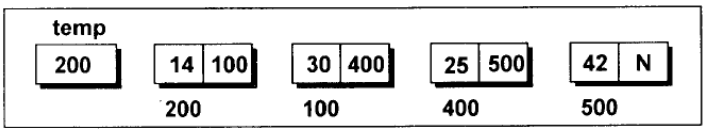


Figure 1.2: Connection of nodes.

Instead of showing the links to the next node we have shown the addresses of the next node in the link part of each node.

When we execute the statement

temp = temp -> link ;

The right hand side yields 100. This address is now stored in temp. As a result, temp starts pointing to the node present at address 100. In effect the statement has shifted temp so that it has started pointing to the next node in the list.

Let us now understand the addatbeg( ) function. Suppose there are already 5 nodes in the list and we wish to add a new node at the beginning of this existing linked list. This situation is shown in Figure 1.3.

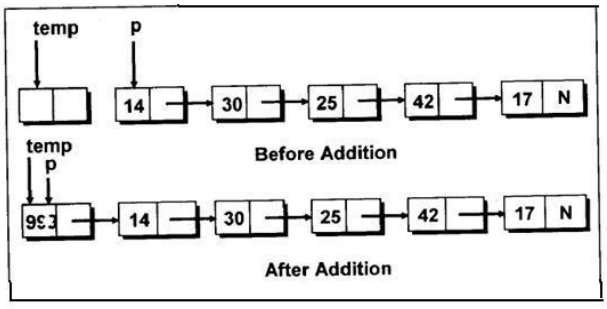


Figure 1.3: Working of addatbeg() function.

For adding a new node at the beginning, firstly space is allocated for this node and data is stored in it through the statement

temp -> data = num;

now we need to make the link part of this node point to the existing first node. This has been achieved through the statement

temp -> link = *q;

Lastly, this new node must be made the first node in the list. This has been attained through the statement
*q = temp;
The addafter( ) function permits us to add a new node after a specified number of node in the linked list.
To begin with, through a loop we skip the desired number of nodes after which a new node is to be added. Suppose we wish to add a new node containing data as 99 after the third node in the list. The position of pointers once the control reaches outside of for loop is shown in Figure 1.4. Now space is allocated for the node to be inserted and 99 is stored in the data part of it.

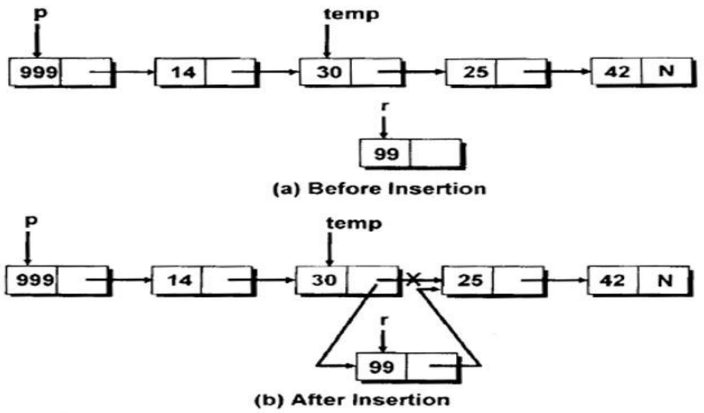


Figure 1.4: Position of pointers once the control reaches outside of for loop.

All that remains to be done is readjustment of links such that 99 go in between 30 and 25. This is achieved through the statements.
r -> link = temp -> link;
temp -> link = r;
The first statement makes link part of node containing 99 to point to the node containing 25. The second statement ensures that the link part of node containing 30 points to the node containing 99. In execution of the second statement the earlier link between 30 and 25 is severed. So now 30 no longer points to 25, it points to 99. The display( ) and count( ) functions are straight forward. We leave them for you to understand.

That brings us to the last function in the program i.e. del( ). In this function through the while loop, we have traversed through the entire linked list, checking at each node, whether it is the node to deleted. If so, we have checked if the node being deleted is the first node in the linked list. If it is so, we have simply shifted p (which is same as *q) to the next node and then deleted the earlier node.
If the node to be deleted is an intermediate node, then the position of various pointers and links before and after the deletion is shown in Figure 1.5.

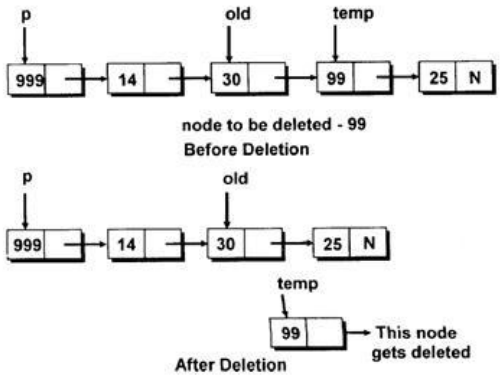


Figure 1.5: Position of various pointers and links before and after the deletion.

Caution

Make sure that your linked list functions work sensibly with the empty list. If you run into a function that will fails for the empty list.

1.3.5 Merging of Linked Lists

Suppose we have two linked lists pointed to by two independent pointers and we wish to merge the two lists into a third list. While carrying out this merging we wish to ensure that those elements which are common to both the lists occur only once in the third list. The program to achieve this is given below. It is assumed that within a list all elements are unique.

```
#include <stdio.h>
#include <conio.h>
#include <malloc.h>
#include <windows.h>
/* structure containing a data part and link part */
struct node
{
int data ;
struct node *link ;
};
void add ( struct node **, int );
void display ( struct node * );
int count ( struct node * ) ;
void merge ( struct node *, struct node *, struct node ** ) ;
int main( )
{
      struct node *first, *second, *third ;
      first = second = third = NULL; /* empty linked lists *I
      add(&first,9) ;
      add (&first, 12) ;
      add (&first, 14) ;
      add (&first, 17) ;
      add(&first,35) ;
      add (&first, 61 ) ;
      add (&first, 79 ) ;
      system (“cls” ) ;
      printf (“First linked list:\n” ) ;
      display (first ) ;
      printf (“No. of elements in Linked List: %d\n\n”, count (first ));

      add ( &second, 12) ;
      add ( &second, 17) ;
      add ( &second, 24) ;
      add ( &second, 36) ;
      add ( &second, 59) ;
      add ( &second, 64 );
      add ( &second, 87) ;
     printf ( “Second linked list\n”) ;
```

```
        display (second ) ;
        printf (“No. of elements in Linked List: %d\n\n”, count ( second )) ;

        merge ( first, second, &third ) ;

        printf ( “The merged list::\n” ) ;
       display ( tnird ) ;
       printf (“No. of elements in Linked List: %d\n”, count ( third )) ;
       return 0 ;
}

/* adds node to an ascending order linked list *I
void add ( struct node **q, int num )
{
       struct node *r, *temp = *q ;

r = ( struct node * ) malloc ( sizeof ( struct node ) ) ;
r->data =num ;

/* if list is empty or if new node is to be inserted before the first node */
if(`q==NULL ||(*q)->data >num )
{
            *q = r;
            (*q) -> link = temp ;
}
else
{
            /* traverse the entire linked list to search the position to insert the new node */
            while (temp != NULL)
            {
                 if ( temp -> data < num && ( temp -> link == NULL || temp -> link -> data > num ))
                 {
                      r -> link = temp -> link ;
                      temp -> link = r ;
                      return ;
                 }
                 temp = temp -> link ; /*go to next node */
            }
            r -> link = NULL ;
            temp -> link = r ;
      }
}

/* displays the contents of the linked list */
void display ( struct node *q )
{
    /* traverse the entire linked list */
```

```
    while (q != NULL)
    {
          printf ( "%d", q -> data ) ;
          q=q->link ;
     }
     printf ( "\n" ) ;
}

/* counts the number of nodes present in the linked list */
int count (struct node * q)
{
     int c =0;
    /* traverse the entire linked list */
    while (q != NULL)
    {
         q=q->link;
         c++;
     }
     return c;
}

/* merges the two linked lists, restricting the common elements to occur only once in the final list */
void merge ( struct node *p, struct node *q, struct node **s)
{
    struct node *z;

z = NULL;

/* if both lists are empty */
if ( p == NULL && q == NULL)
return;

/* traverse both linked lists till the end. If end of any one list is reached loop is terminated */
while ( p!= NULL && q!= NULL)
{
/* if node being added in the first node */
if (*s == NULL)
{
          *s = (struct node*) malloc (sizeof (struct node));
          z=*s;
}
else
}
          z -> link =(struct node*) malloc (sizeof (struct node));
          z=z->link;
}
```

```
if (p-> data < q-> data)
{
            z->data=p->data;
            p=p->link;
}
else
{
            if (q-> data < p-> data)
            {
            z->data=q->data;
            q=q->link;
}
else
{
    if (p->data ==q->data)
    {
         z -> data = q -> data ;
         p=p->link;
         q=q->link;
    }
    }
}
}
/* if end of first list has not been reached */
while (p!= NULL)
{
z -> link = (struct node*) malloc (sizeof ( struct node));
z = z -> link ;
z -> data = p -> data ; p = p -> link ;
}

     /* if end of second list has not been reached */
     while (q!= NULL)
      {
           z -> link = (struct node *) malloc (size of (struct node));
           z = z -> link;
           z-> data = q -> data;
           q=q->link;

z -> link = NULL
}
```

Output:

```
First linked list:
9 12 14 17 35 61 79
No. of elements in Linked List: 7
```

Second linked list:
12 17 24 36 59 64 87
No. of elements in Linked List: 7

The merged list:
9 12 14 17 24 35 36 59 61 64 79 87
No. of elements in Linked List: 12
In this program, as usual, we begin by building a structure to accommodate the data and link, which together represent a node. We have used pointers first, second and third to point to the three linked lists. Since to begin with all the three linked lists are empty, these pointers contain NULL. Next, by calling the function add( ) repeatedly two linked lists are built, one being pointed to by first and other by the pointer second. Finally, the merge( ) function is called to merge the two lists into one. This merged list is pointed by the pointer third.

While merging the two lists it is assumed that the lists themselves are in ascending order. While building the two lists the add( ) function makes sure that when a node is added the elements in the lists are maintained in ascending order.

The function merge( ) receives three parameters. The first two parameters p and q are of the type struct node * which point to the two lists that are to be merged. The third parameters is of the type struct node ** which holds the address of pointer third which is a pointer to the resultant merged list. Before calling merge( ) third contains a NULL value.
First of all we check if both the lists that are to be merged, are empty or not. If the lists are empty then the control simply returns from the function. Otherwise, a loop is executed to traverse the lists that are pointed to by p and q. If end of any of the list is reached then the loop is terminated.
To begin with, a NULL value is stored in z, which is going to point to the resultant merged list. Inside the while loop, we check the special case of adding the first node to the merged list pointed to by z. If the node being added is the first node then z is made to point to the first node of the merged list through the statement

```
z= s;
```

Next, the data from both the lists are compared and whichever is found to be smaller is stored in the data part of the first node of the merged list. The pointers that point to the merged list and to the list from where we copied the data are incremented appropriately.
During the next iteration of the while loop, if condition for first node fails and we reach the else block. Here we allocate the memory for the new node and its address is stored in z -> link. Then z is made to point to this node, through the statement

```
z = z -> link;
```

While comparing the data, if we find that the data of both the lists are equal then the data is added only once to the merged list and pointers of all the three lists are incremented, this is done through the statements.

```
if (p -> data == q-> data)
{
    z -> data = q -> data;
    p = p -> link;
    q = q -> link;
}
```

The procedure of comparing, adding the data to the merged list and incrementing the pointer of the merged list and the list from where the data is added is repeated till any of the list ends.

If we reach end of first and/or second list the while loop terminates. If we have reached end only one list then the remaining elements of the other list are simply dumped in the merged list as they are already in the ascending order. The working of the merge function is shown in Figure 1.6. Figure 1.7(a, b, c, d, e) shows the steps to merging two linked lists.

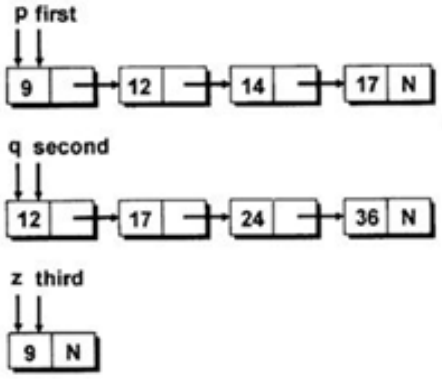


Figure 1.6: Merging of two lists.

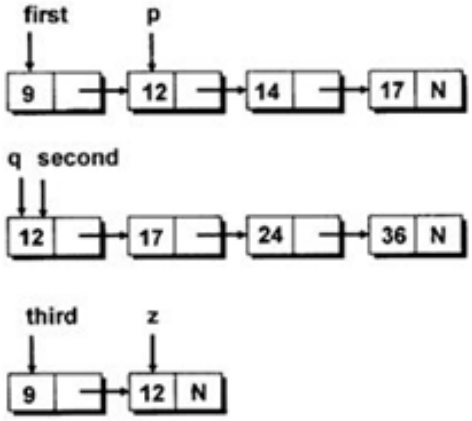


Figure 1.7(a): Merging two linked lists.

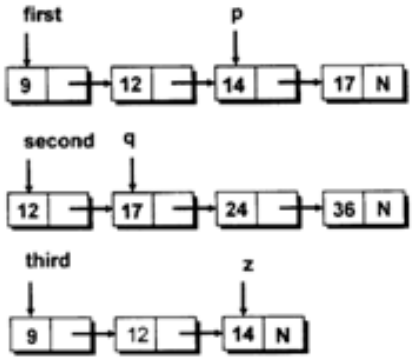


Figure 1.7(b): Merging two linked lists.

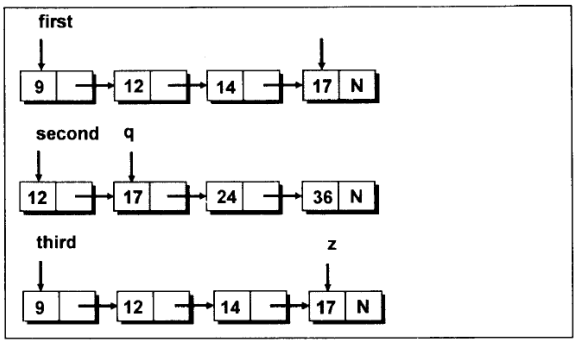


Figure 1.7(c): Merging two linked lists.

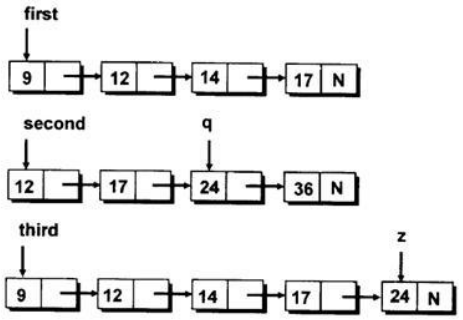


Figure 1.7(d): Merging two linked lists.

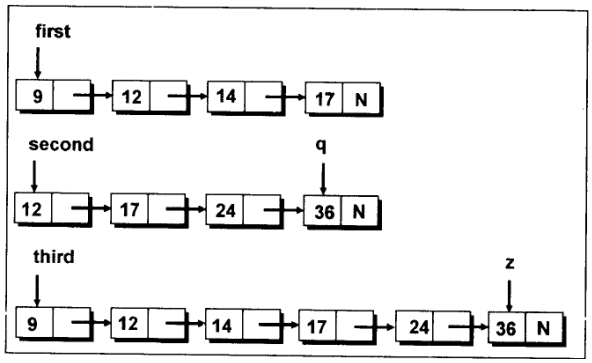


Figure 1.7(e): Merging two linked lists.

Self Assessment Questions

2. Applications of Linked List are…….
(a). Simulation, event driven systems
(b). Postfix and prefix manipulations
(c). Dictionary systems, polynomial manipulations
(d). Fixed block storage allocation, garbage collection

1.4 Types of Linked Lists

The various types of linked list are:

- Linear/Singly-linked list
- Doubly-linked list
- Circular singly-linked list
- Circular doubly-linked list

1.4.1 Singly-Linked List

As the name suggests, a singly-linked list consists of only one pointer to point to another node. It is also known as a linear list because the last node in a singly-linked list is assigned a NULL value and hence does not point to any other node. The first node in the list is known as a HEAD or first node. Figure 1.8 depicts a singly-linked list. The HEAD Node is a dummy node pointing to Node1. Node1 holds the address of Node2, and Node2 holds the address of Node3. Node3 points to NULL to indicate that there are no additional nodes present in the list.

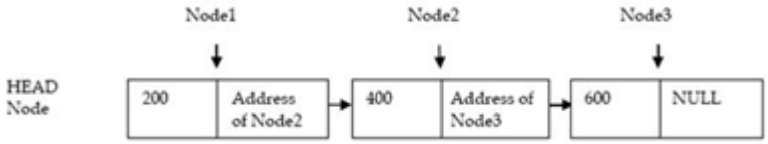


Figure 1.8: Representation of a Singly-linked List.

Example: The program shows the implementation of a singly-linked list consisting of four nodes. The program displays the value present in each node.

```
#include<stdio.h>
struct new_list
{
 int value;
struct new_list *next_element;
 } n1, n2, n3, n4; //Creates four nodes of type
new_list void main()
{
 int j;
 n1.value = 200; //Assigning value to node1
n2.value = 400; //Assigning value to node2
 n3.value = 600; //Assigning value to node3
n4.value = 800; //Assigning value to node4
n1.next_element = &n2; //Assigning address of node2 to node1
n2.next_element = &n3; //Assigning address of node3 to node2
n3.next_element = &n4; //Assigning address of node4 to node3
n4.next_element = 0; //Assigning 0 to node4 to indicate the end of the list
j = n1.next_element->value; //Storing the value of node1 in variable j
printf("%d\n", j); /* you can use this statement to print the value present in node1 or print j directly as depicted in the above statement*/
printf("%d\n", n1.next_element->value);
printf("%d/n", n4.next_element->value); //Printing the value of node4
printf("%d/n", n2.next_element->value); //Printing the value of node2
printf("%d/n", n3.next_element->value); //Printing the value of node3
}
```

Output:

After you compile the program, you will get the following output:

```
 400
0
600
```

800

In this example:

1. First a structure named new_list is created. The list contains an integer data variable named value to store data and a pointer variable named next_element to point to next node.
2. Then, four objects namely, n1, n2, n3, and n4 are created to access the structure elements. In the program they act as nodes in a list.
3. In the main () function, the value for the four nodes n1, n2, n3, and n4 are assigned.
4. Then, the address of n2 is stored in n1, address of n3 is stored in n2, and address of n4 is stored in n3. The address of n4 is assigned zero to depict the end of the list.
5. Finally, the value present in n1, n4, n2 and n3 are printed.

1.4.2 Doubly-linked List

Doubly-linked list contains two pointers for each node in the list. The first pointer points to the next element and the second pointer points to the previous element. The previous pointer for the HEAD node points to NULL and the next pointer for the last node points to NULL. Doubly-linked list is also known as a two-way list as both forward and backward traversal is possible. Figure 1.9 depicts a doubly-linked list. The HEAD Node is a dummy node pointing to Node1. Node1 has two pointers, the first pointer points to Node2 and the second pointer points to HEAD Node. Likewise, Node2 and Node3 also have two pointers to point to the next and the previous element in the list. The HEAD Node and the Node3 are assigned to NULL. The data field of Node1, Node2, and Node3 consists of values 20, 40, and 60 respectively. When you try to print the value of Node2's next element, the value present in Node3 which is 60, will be printed.

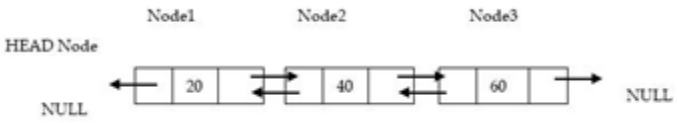


Figure 1.9: Representation of a Doubly-linked List.

Example: The program shows the implementation of a doubly-linked list consisting of three nodes. The program displays the value present in each node.

```
#include<stdio.h>
struct list
{
int value;
struct list *next; //Creating a pointer to point to the next element
struct list *previous;//Creating a pointer to point to the previous element
} n1, n2, n3; //Creating three nodes of type list
void main()
{
int j;
n1.value = 20; //Assigning value to node1
n2.value = 40; //Assigning value to node2
n3.value = 60; //Assigning value to node3
n1.next = &n2; //Assigning address of node2 to node1
n2.next = &n3; //Assigning address of node3 to node2
n2.previous = &n1; //Assigning address of node1 to node2
```

```
n3.previous = &n2; //Assigning address of node2 to node3
n3.next = 0; //Assigning 0 to node3 to indicate the end of the list
n1.previous = 0; //Assigning 0 to node1 to indicate there are no elements present before node1
j = n1.next->value; //Storing the value of node1 in variable j
printf ("%d\n", j);
printf ("%d\n", n1.next->value); // you can use this statement to print the value present in node1 or print j directly as depicted in the above statement
printf ("%d/n", n1.next->value); //Printing the next value of node1
printf ("%d/n", n2.next->value); //Printing the next value of node2
printf ("%d/n", n1.previous->value); //Printing the previous value of node1
printf ("%d/n", n2.previous->value); //Printing the previous value of node2
printf ("%d/n", n3.previous->value); //Printing the previous value of node3 }
```

Output:

After you compile the program, you will get the following output:

```
40
60
0
20
40
```

In this example:

1. First, a structure named list is created. The list contains an integer data variable named value to store data, a pointer variable named next_element to point to next node and a pointer variable named previous_element to point to previous node.
2. Then, the three objects namely, n1, n2, and n3 are created to access the structure elements. In the program they act as nodes in a list.
3. In the main () function, the value for nodes n1, n2, and n3 are assigned.
4. Then, the address of n2 is stored in n1 and address of n3 is stored in n2. In order to traverse backwards the address of n1 is stored in n2 and address of n2 is stored in n3. The address of n3 is assigned a NULL value to depict the end of the list.
5. Finally, the value present in n1, n2, and n3 are printed.

1.4.3 Circular-linked List

List in a circular-linked list, only one pointer is used to point to another node or next element. It is known as a circular list because the last node's pointer points to the HEAD node. Figure 1.10 depicts a circular-linked list. The linked list consists of four nodes like, Node1, Node2, and Node3 with values 35, 65, and 85 respectively. The last node which is Node3 points to the first node (Node1) and hence, the list continues to form a loop. When you try to print the value of Node3's next element the value present in Node1 which is 35 will be printed.

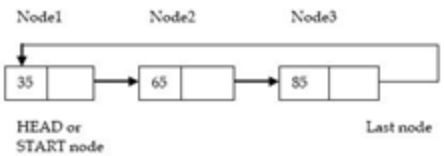


Figure 1.10: Representation of a circular-linked List.

Example: The program shows the implementation of a circular-linked list consisting of three nodes. The program displays the value present in each node.

```
#include<stdio.h>
struct list
{
int value;
struct list *next_element;
} n1, n2, n3; //Creates four nodes of type new_list
void main()
{
int j;
n1.value = 35; // Assigning value to node1
n2.value = 65; // Assigning value to node2
n3.value = 85; //Assigning value to node3
n1.next_element = &n2; //Assigning address of node2 to node1
n2.next_element = &n3; //Assigning address of node3 to node2
n3.next_element = &n1; //Assigning address of node3 to node1
j = n1.next_element->value; //Storing the value of node1 in variable j
printf("%d\n", j); //Printing the value of j

/* you can use this statement to print the value present in node1*/
printf("%d\n", n1.next_element->value);
printf("%d/n", n2.next_element->value); //Printing the value of node2
printf("%d/n", n3.next_element->value); //Printing the value of node3
}
```

Output:
After you compile the program, you will get the following output:
65
65
85
35

In this example:

1. First, a structure named list is created. The list contains an integer data variable named value to store data and a pointer variable named next_element to point to next node.
2. Then, the three objects namely, n1, n2, and n3 are created to access the structure elements. In the program they act as nodes in a list.
3. In the main () function, the value for nodes n1, n2 and n3 are assigned.
4. Then, the address of n2 is stored in n1 and address of n3 is stored in n2. Since, it is a circular list the address of n3 is assigned to n1 instead of NULL value.
5. Finally, the value present in n1, n2, and n3 are printed.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Write a simple circular linked list program to accept the elements from the user and store it in a list.

……..……

……..……

……...

1.4.4 Circular Doubly-linked List

In a circular doubly-linked list, the previous pointer of the first node and the next pointer of the last node point to the HEAD node. The HEAD node can have a dummy data or it can store the total number of nodes present in the list. Figure 1.11 depicts a circular doubly-linked list. The linked list consists of four nodes such as, HEAD node, Node1, Node2, and Node3 with values 3, 10, 15 and 20 respectively. Each node has two pointers to point to the next and previous elements. The last node (Node3) points to the HEAD node and the HEAD node in turn points to the first node (Node1).

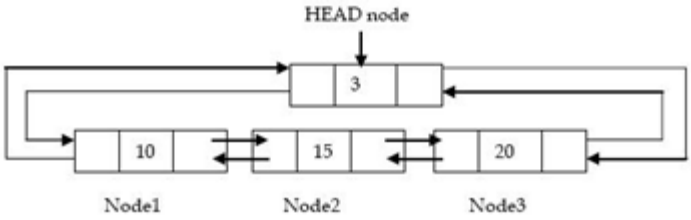


Figure 1.11: Representation of a circular doubly-linked List.

Example: The program shows the implementation of a circular doubly-linked list consisting of three nodes and a HEAD node. The program displays the value present in each node.

```
#include<stdio.h>
struct list
{
int value;
struct list *next; //Creating a pointer to point to the next element
struct list *previous; //Creating a pointer to point to the previous element
} n1, n2, n3, h; //Creates four nodes of type list
void main()
{
int j;
n1.value = 10; //Assigning value to node1
n2.value = 15; //Assigning value to node2
n3.value = 20; //Assigning value to node3
h.value = 3; //Assigning value to HEAD node
n1.next = &n2; //Assigning address of node2 to node1
n2.next = &n3; //Assigning address of node3 to node2
n3.next = &h; //Assigning address of HEAD node to node3
h.next = &n1; //Assigning address of node1 to HEAD node
n1.previous = &h; //Assigning address of node1 to HEAD node
n2.previous = &n1; //Assigning address of node1 to node2
n3.previous = &n2; //Assigning address of node2 to node3
h.previous = &n3; //Assigning address of node3 to HEAD node
j = n1.next_element->value; //Storing the value of node1 in variable j
printf("%d\n", j);
```

printf("%d\n", n1.next->value); // you can use this statement to print the value present in node1 or print j directly as depicted in the above statement
printf("%d/n", n2.next->value); //Printing the value of node2
printf("%d/n", n3.next->value); //Printing the value of node3
printf("%d/n", h.next->value); //Printing the value of HEAD node
printf("%d/n", n1.previous->value); //Printing the previous value of node1
printf("%d/n", n2.previous->value); //Printing the previous value of node2
printf("%d/n", n3.previous->value); //Printing the previous value of node3
printf("%d/n", h.previous->value); //Printing the previous value of HEAD node }
Output:
After you compile the program, you will get the following output:
15
20
3
10
3
10
15
20
In this example:

1. First, a structure named list is created. The list contains an integer data variable named value to store data, a pointer variable named next_element to point to next node, and a pointer variable named previous_element to point to previous node.
2. Then, the four objects namely, n1, n2, n3, and h are created to access the structure elements. In the program, they act as nodes in a list. The HEAD node (h) contains the total number of nodes present in the list.
3. In the main() function, the value for the nodes n1, n2, n3, and h are assigned.
4. Then, the address of n2 is stored in n1 and the address of n3 is stored in n2. In order to traverse backwards, the address of h is stored in n3 and address of n1 is stored in h.
5. Finally, the value present in n1, n2, n3, and h are printed.

Exercise: Check Your Progress 2
Note: i) Use the space below for your answer.
Ex1: What is a circular linked list? What are its advantages over linear linked list?

……..……

……..……

……...

Ex2: What are the advantages of doubly linked list over singly linked list

……..……

……..……

……...

Self Assesment Questions

3. Which of the following operations is performed more efficiently by doubly linked list than singly linked list
(a) Deleting a node whose location is given.
(b) Searching of an unsorted list for a given item.
(c) Inserting a new node after node whose location is given.
(d) Traversing the list to process each node.

4. Overflow condition in linked list may occur when attempting to…….
(a). Create a node when free space pool is empty.
(b). Traverse the nodes when free space pool is empty.
(c). Create a node when linked list is empty.
(d). None of these.

1.5 Header Node in Linked List

Sometimes it is desirable to keep an extra node at the front of a linked list. Such a node does not represent an item in the list. The information part of such node is used to store information of a different type than the values stored in the list nodes. For example a header node may contain the number of nodes in a list. A list which contains this type of node is called header linked list.(including the header node). In a header linked list where the next pointer of the last node is pointing to NULL is called a grounded header linked list.

```
# include < stdio.h >
# include < stdlib.h >
struct link
{
int info;
struct link *next;
};
int i=0;
struct link *node,*new1;
void create(struct link *n)
{
char ch='y';node=n;
while(ch!='n')
{
node->next = (struct link* ) malloc(sizeof(struct link));
node=node->next;
printf("\n Enter the value %d:-", i+1);
scanf("%d", &node->info);
node->next = NULL;
fflush(stdin);
printf("\n Enter choice—'n' for break: ");
ch = getche();
i++;
}
printf("\n Total nodes = %d\n", i);
```

```
n->info = i; /* Assign total number of nodes to the header node */
}
void display(struct link *n)
{
int count = n->info;
printf("The address and the value of the header node :-");
printf("\n 0x%x--%d", n,n->info);
node = n->next;
printf("\n The nodes are as follows:-\n");
while (count)
{
printf("\n 0x%x--%d", node, node->info);
node = node->next;
count --;
}
printf("\n");
}
void main()
{
struct link *start;
clrscr();
start=(struct link *)malloc(sizeof(struct link));
create(start);
display (start);
getch();
}
```

Here the header node does not contain any values in this program but it can hold values. Any operations like insertion or deletion of nodes will be performed after the header node.

Self Assessment Questions

5. The number of nodes in a complete binary tree of depth d (with root at depth 0) is…….

(a). $2^{d-1} + 1$ (b). $2^{d+1} - 1$ (c). $2^{d-1} + -1$ (d). $2^{d+1} + 1$

6. Linked lists are not suitable data structures for which one of the following problems…….

(a). insertion sort (b). Binary search (c). radix sort (d). polynomial manipulation

7. Which of the following is two way lists?

(a). grounded header list
(b). circular header list
(c). linked list with header and trailer nodes
(d). None of these.

8. Linked lists are best suited……..

(a). for relatively permanent collections of data
(b). for the size of the structure and the data in the structure are constantly changing
(c). for both of above situation
(d). None of these.

1.6 Application of Linked Lists

Linked lists are used as a building block for many other data structures, such as stacks, queues and their variations.

The "data" field of a node can be another linked list. By this device, one can construct many linked data structures with lists; this practice originated in the List programming language, where linked lists are a primary (though by no means the only) data structure, and is now a common feature of the functional programming style.

Sometimes, linked lists are used to implement associative arrays, and are in this context called association lists. There is very little good to be said about this use of linked lists; they are easily outperformed by other data structures such as self-balancing binary search trees even on small data sets. However, sometimes a linked list is dynamically created out of a subset of nodes in such a tree, and used to more efficiently traverse that set.

1.6.1 Insert from Front

At first initialize node type.

node *head = NULL; //empty linked list

Then we take the data input from the user and store in the node info variable. Create a temporary node node *temp and allocate space for it.

node *temp; //create a temporary node

temp = (node*)malloc(sizeof(node)); //allocate space for node

Then place info to temp->data. So the first field of the node *temp is filled. Now temp->next must become a part of the remaining linked list (although now linked list is empty but imagine that we have a 2 node linked list and head is pointed at the front) So temp->next must copy the address of the *head (Because we want insert at first) and we also want that *head will always point at front. So *head must copy the address of the node *temp. Figure 1.12 shows the inserting node at front.

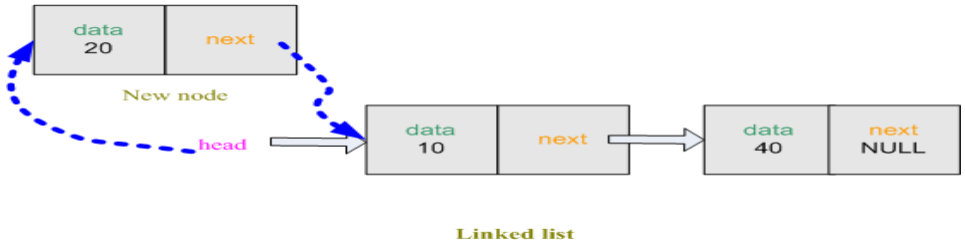


Figure 1.12: Insert at first.

temp->data = info; // store data(first field)

temp->next=head; // store the address of the pointer head(second field)

head = temp; // transfer the address of 'temp' to 'head'

1.6.2 Traversal

Now we want to see the information stored inside the linked list. We create node *temp1. Transfer the address of *head to *temp1. So *temp1 is also pointed at the front of the linked list. Linked list has 3 nodes.

We can get the data from first node using temp1->data. To get data from second node, we shift *temp1 to the second node. Now we can get the data from second node. Figure 1.13 shows the traversal of a linked lists.

```
while ( temp1!=NULL )
{
printf("temp1->data");// show the data in the linked list
temp1 = temp1->next; // transfer the address of 'temp->next' to 'temp'
}
```

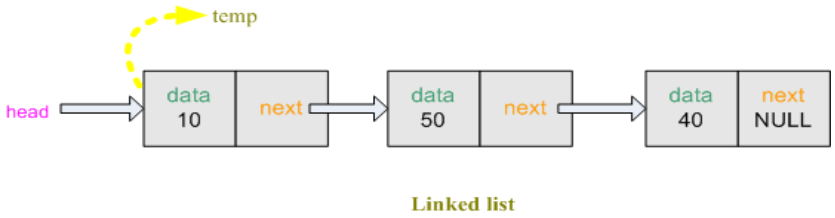


Figure 1.13: Traverse.

This process will run until the linked list's next is NULL.

1.6.3 Insert from Back

Insert data from back is very similar to the insert from front in the linked list. Here the extra job is to find the last node of the linked list.

```
node *temp1;                    // create a temporary node
temp1=(node*)malloc(sizeof(node)); // allocate space for node
temp1 = head;                   // transfer the address of 'head' to 'temp1'
while(temp1->next!=NULL) // go to the last node
    temp1 = temp1->next;//tranfer the address of 'temp1->next' to 'temp1'
```

Now, create a temporary node node *temp and allocate space for it. Then place info to temp->data, so the first field of the node node *temp is filled.node *temp will be the last node of the linked list. For this reason, temp->next will be NULL. To create a connection between linked list and the new node, the last node of the existing linked list node *temp1's second field temp1->next is pointed to node *temp. Figure 1.14 shows the process of inserting a node at the last.

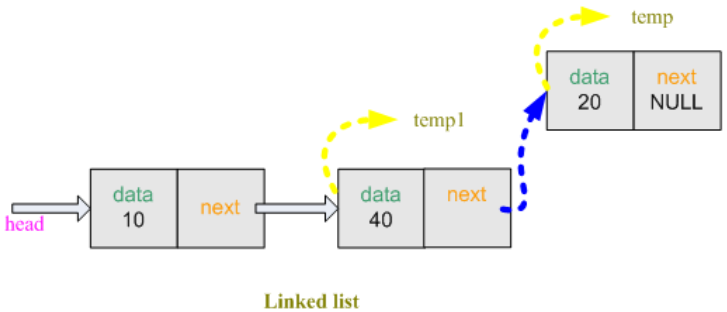


Figure 1.14: Insert at last.

```
node *temp;                     // create a temporary node
temp = (node*)malloc(sizeof(node)); // allocate space for node
temp->data = info;              // store data(first field)
temp->next = NULL;              // second field will be null(last node)
```

```
temp1->next = temp;                // 'temp' node will be the last node
```

1.6.4 Insert after Specified Number of Nodes

Insert data in the linked list after specified number of node (see Figure 1.15) is a little bit complicated. But the idea is simple. If, we add a node after 2nd position, then the new node must be in 3rd position. The first step is to go the specified number of node.

```
cout<<"ENTER THE NODE NUMBER:";
cin>>node_number;                  // take the node number from user

node *temp1;                       // create a temporary node
temp1 = (node*)malloc(sizeof(node)); // allocate space for node
temp1 = head;
for( int i = 1 ; i < node_number ; i++ )
{
    temp1 = temp1->next;           // go to the next node

    if( temp1 == NULL )
    {
        printf(%d node_number " node is not exist");
        break;
    }
}
```

Now, Create a temporary node node *temp and allocate space for it.
Then place info to temp->next, so the first field of the node node *temp is filled.

```
node *temp;                        // create a temporary node
temp = (node*)malloc(sizeof(node)); // allocate space for node
temp->data = info;                 // store data(first field)
```

To establish the connection between new node and the existing linked list, new node's next must pointed to the 2nd node's (temp1) next. The 2nd node's (temp1) next must pointed to the new node (temp).

```
temp->next = temp1->next;          //transfer the address of temp1->next to temp->next
temp1->next = temp;     //transfer the address of temp to temp1->next
```

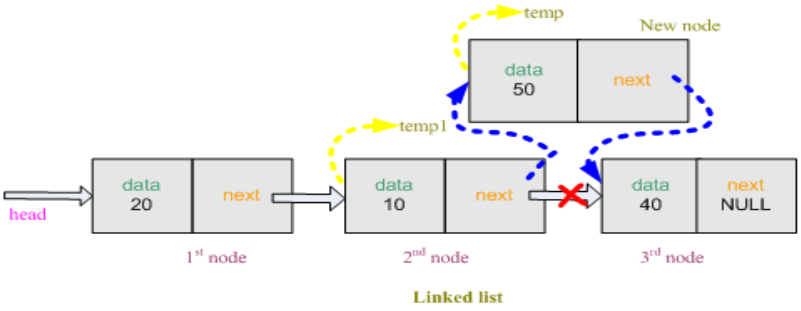


Figure 1.15: Insert after specified number of nodes.

1.6.5 Delete from Front

Delete a node from linked list is relatively easy. First, we create node *temp. Transfer the address of *head to *temp. So *temp is pointed at the front of the linked list. We want to delete the first node. So transfer the

address of temp->next to head so that it now pointed to the second node. Now free the space allocated for first node. Figure 1.16 shows how a node delete from front.

```
node *temp;                              // create a temporary node
temp = (node*)malloc(sizeof(node));  // allocate space for node
temp = head;                 // transfer the address of 'head' to 'temp'
head = temp->next;      // transfer the address of 'temp->next' to 'head'
free(temp);
```

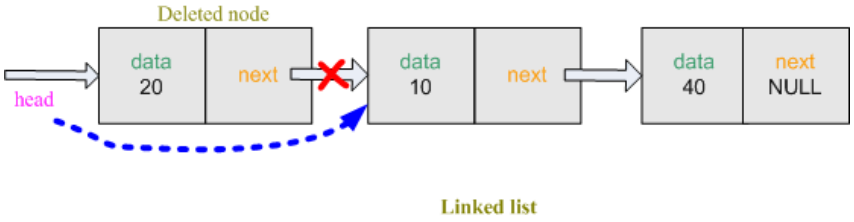


Figure 1.16: Delete at first node.

1.6.6 Delete from Back

The last node's next of the linked list always pointed to NULL. So when we will delete the last node, the previous node of last node is now pointed at NULL. So, we will track last node and previous node of the last node in the linked list. Create temporary node * temp1 and *old_temp.

```
// create a temporary node
node *temp1;
temp1 = (node*)malloc(sizeof(node)); // allocate space for node
temp1 = head;                    //transfer the address of head to temp1
node *old_temp;                  // create a temporary node
old_temp = (node*)malloc(sizeof(node));    // allocate space for node
while(temp1->next!=NULL)            // go to the last node
{
    old_temp = temp1; // transfer the address of 'temp1' to 'old_temp'
    temp1 = temp1->next;       // transfer the address of 'temp1->next' to 'temp1'
}
```

Now node *temp1 is now pointed at the last node and *old_temp is pointed at the previous node of the last node. Now rest of the work is very simple. Previous node of the last node old_temp will be NULL so it becomes the last node of the linked list. Free the space allocated for last node (see Figure 1.17).

```
old_temp->next = NULL;         // previous node of the last node is null
free(temp1);
```

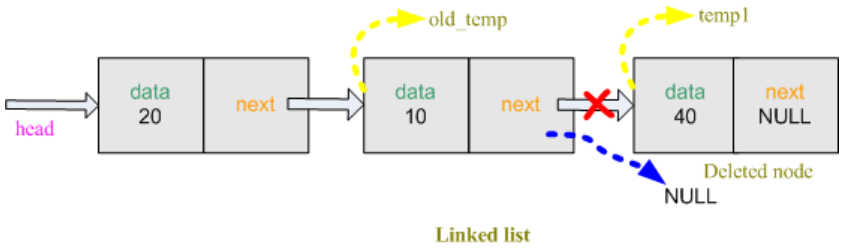


Figure 1.17: Delete at first last.

1.6.7 Delete Specified Number of Node

To delete a specified node in the linked list (see Figure 1.18), we also require finding the specified node and previous node of the specified node. Create temporary node * temp1, *old_temp and allocate space for it. Take the input from user to know the number of the node.

```
node *temp1;                    // create a temporary node
temp1 = (node*)malloc(sizeof(node)); // allocate space for node
temp1 = head;                   // transfer the address of 'head' to 'temp1'
node *old_temp;                 // create a temporary node
old_temp = (node*)malloc(sizeof(node));    // allocate space for node
old_temp = temp1;       // transfer the address of 'temp1' to 'old_temp'
cout<<"ENTER THE NODE NUMBER:";
cin>>node_number;               // take location
for( int i = 1 ; i < node_number ; i++ )
{
    old_temp = temp1;           // store previous node
    temp1 = temp1->next;        // store current node
}
```

Now node *temp1 is now pointed at the specified node and *old_temp is pointed at the previous node of the specified node. The previous node of the specified node must connect to the rest of the linked list so we transfer the address of temp1->next to old_temp->next. Now free the space allocated for the specified node.

```
old_temp->next = temp1->next; // transfer the address of 'temp1->next' to 'old_temp->next'
free(temp1);
```

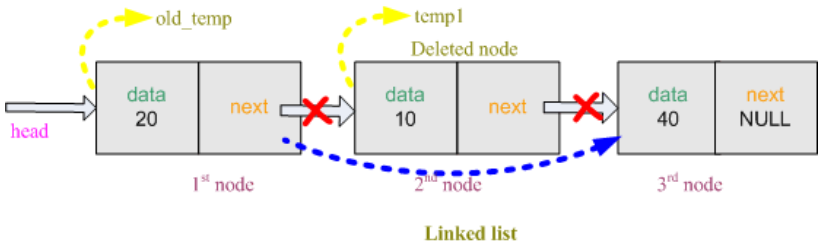


Figure 1.18: Delete a specified node in the linked list.

1.6.8 Sort Nodes

Linked list sorting is very simple. It is just like ordinary array sorting. First we create two temporary node node *temp1, *temp2 and allocate space for it. Transfer the address of first node to temp1 and address of second node to temp2. Now check if temp1->data is greater than temp2->data. If yes then exchange the data. Similarly, we perform this checking for all the nodes (see Figure 1.19).

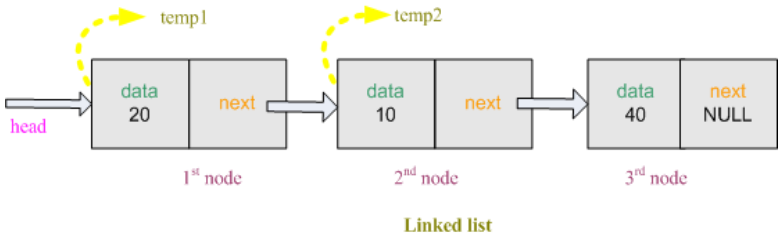


Figure 1.19: Sorting nodes.

```
node *temp1;                        // create a temporary node
temp1 = (node*)malloc(sizeof(node)); // allocate space for node
node *temp2;                        // create a temporary node
temp2 = (node*)malloc(sizeof(node)); // allocate space for node
int temp = 0;                       // store temporary data value
for( temp1 = head ; temp1!=NULL ; temp1 = temp1->next )
{
    for( temp2 = temp1->next ; temp2!=NULL ; temp2 = temp2->next )
    {
        if( temp1->data > temp2->data )
        {
            temp = temp1->data;
            temp1->data = temp2->data;
            temp2->data = temp;
        }
    }
}
```

Lab Exercise

1. Write a C program to store 20 integers in linked list in descending order.
2. Write a C program to evaluate a third degree polynomial.

1.7 Addition of Two Long Positive Numbers

The given program shows the example for the addition of two long positive numbers.
It can be derived as:

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
   unsigned char c;
   struct Node *next;
}Node;
typedef Node *slist;
slist reverse(slist);
Node *makeNode(unsigned char);
/*
*/

slist Sum(slist left, slist right) {
   if(!left || !right)
   {
      return left ? left : right;
   }
   left = reverse(left);
   right = reverse(right);
   unsigned char carry = left->c + right->c;
```

```
    slist ret = makeNode(carry % 10);
    carry /= 10;
    Node *p = left->next;
    Node *q = right->next;
    Node *r = ret;
    while(p || q) {
        carry += (p? p->c : 0)  + (q ? q->c : 0);
        r->next = makeNode(carry % 10);
        carry /= 10;
        r = r->next;
        p = p ? p->next : NULL;
        q = q ? q->next : NULL;
    }
    if(carry)
        r->next = makeNode(1);
    reverse(left);
    reverse(right);
    return reverse(ret);
}
/* utilities */
slist reverse(slist s) {
if(s->next == NULL)
        return s;
    Node *ret = reverse(s->next);
    s->next->next = s;
    s->next = NULL;
    return ret;
}
Node *makeNode(unsigned char c) {
    Node * tmp = calloc(sizeof(Node), 1);
    tmp->c = c;
    return tmp;
}
void print(slist s) {
    if(s == NULL) {
        printf("\n");
        return;
    }
    printf("%c", s->c + '0');
    print(s->next);
}
slist listFromString(const unsigned char *s) {
    if(!s || !*s) return NULL;
    slist ret = makeNode(*s++ - '0');
    Node *tmp = ret;
    unsigned char c;
    while((c = *s++)) {
```

```
        tmp->next = makeNode(c - '0');
        tmp = tmp->next;
    }
    return ret;
}
int main()
{
    slist left  = listFromString("99");
    slist right  = listFromString("233823");
    slist sum = Sum(left, right);
    print(sum);
    return 0;
}
```

Exercise: Check Your Progress 3

Note: i) Use the space below for your answer.

Ex1: What is a doubly linked list? Write program/algorithm for showing the following operations on a doubly linked list:

- Create
- Insert
- Delete

……..……

……..……

……...

1.8 Evaluation of a Polynomial

Now we consider the evaluation of a polynomial function of a single variable. We usually compute the value of an arithmetic function by replacing each arithmetic operation by its corresponding floating-point machine operation. Round off errors and cancellations sometimes cause the calculated result to be drastically wrong. For similar reasons, a naive interval evaluation of a polynomial may lead to intervals so large as to be practically useless. Round off and cancellation errors are especially dangerous if we are evaluating a function close to a root, as we will see when we compute verified enclosures of zeroes of polynomials.

We present an algorithm to evaluate a real polynomial $p : R \rightarrow R$ defined as

$$p(t) = \sum_{i=0}^{n} p_i t^i \ , \ p_i, t \in R \ , \ i = 0, ..., n \ , \ p_n \neq 0$$

We assume the coefficients p_i to be representable in the floating-point number system of the host computer. The algorithm achieves maximum accuracy, even in the neighborhood of a root where cancellation dooms an ordinary floating-point evaluation.

1.8.1 Algorithmic Description

We present the algorithm *RPolyEval* for the evaluation (see Figure 1.20) of a real polynomial $p(t) = \sum_{i=0}^{n} p_i t^i$ with maximum accuracy. Except for the special cases $n = 0$ and $n = 1$, which can be

calculated directly, an iterative solution method is used. We first compute a floating-point approximation of $p(t)$. We then carry out a residual iteration by solving a linear system of equations. The new solution interval determined in the next step is checked for being of maximum accuracy, i.e. for being exact to one unit in the last place of the mantissa (1 ulp).

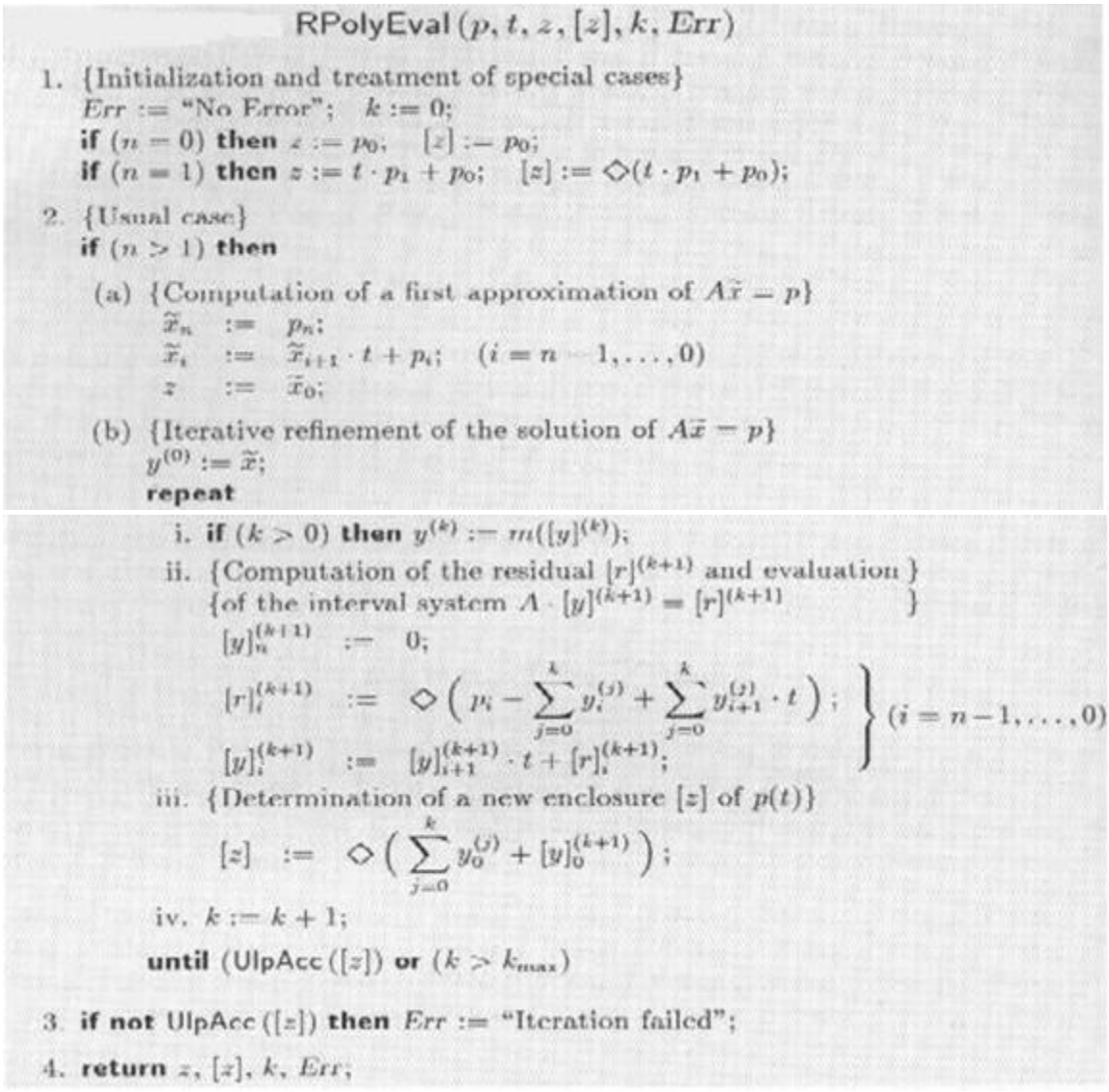


Figure 1.20: The RPolyEval Algorithm.

Example: Write a c program to evaluate a given powered polynomial.

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#define MAXSIZE 10
voidmain()
{
int a[MAXSIZE];
int i, N,power;
float x, polySum;
clrscr();
```

```
printf(“Enter the order of the polynomial\n”);
scanf(“%d”, &N);
printf(“Enter the value of x\n”);
scanf(“%f”, &x);
/*Read the coefficients into an array*/
printf(“Enter %d coefficients\n”,N+1);
for (i=0;i <= N;i++)
{
scanf(“%d”,&a[i]);
}
polySum = a[0];
for (i=1;i<= N;i++)
{
polySum = polySum * x + a[i];
}
power = N;
/*power--;*/
printf(“Given polynomial is:\n”);
for (i=0;i<= N;i++)
{
if (power < 0)
{
break;
}
/* printing proper polynomial function*/
if (a[i] > 0)
printf(“ + ”);
else if (a[i] < 0)
printf(“ – ”);
else
printf (“*”);
printf("%dx^%d ",abs(a[i]),power– –);
}
printf("\n Sum of the polynomial = %6.2f\n", polySum);
}
```

1.8.2 Evaluating a Polynomial using HORNER’s Rule

Now we solve the problem of evaluating a polynomial using Horner’s rule. Horner’s rule is a strategy for evaluating a polynomial A(x) at a point using a minimum number of multiplication.

$$a_n x^n + a_{n-1} x^{n-1} + \ldots + a_0 = ((a_n x + a_{n-1}) x + \ldots) x + a_0 .$$

The following C program is a correct one, but the thing to emphasize is trouble with the use of global variables. Let us go through the C code first:

```
#include<stdio.h>
#include<conio.h>
int horner(int,int);
int count=0;
void main()
```

```
{
/*Horner's rule for evaluating a polynomial */
/* let us take a0=0,a1=1,a2=2.. and so on */
int n,x,h=0; //n is order, x is value of X in polynomial.
scanf("%d %d",&n,&x);
h=horner(n,x);
printf("%d",h);
getch();
}
int horner(int n, int x)
{
int i;
if(count!=n)
{
i=count;
count++;
printf("%d+%d", i, x);
return (i + x*horner(n, x));
}
else
{
printf("%d))=",count);
return count;
}
}
```

Initially there was a big problem with the above program when we not used the local variable 'i' in the 'horner' function. You can check the output with and without the use of 'i'. The problem was that all the return statements were being evaluated after the last 'horner' got evaluated for the recursive function, and in the meanwhile, the global variable 'count' got changed every time 'horner' was evaluated. Since the return statement depends upon 'count', the output came out to be deviated from the expected value depending upon the size of input. Hence the variable 'i' came into existence.

Case Study: Linked List Problems

Here are some problems related with linked list. The first few are quite basic and the last few are quite advanced. Each problem starts with a basic definition of what needs to be accomplished. Many of the problems also include hints or drawings to get you started. The solutions to all the problems are in the next section. It is easy to just passively sweep your eyes over the solution — verifying its existence without lettings its details touch your brain. To get the most benefit from these problems, you need to make an effort to think them through. Whether or not you solve the problem, you will be thinking through the right issues, and the given solution will make more sense. Great programmers can visualize data structures to see how the code and memory will interact. Linked lists are well suited to that sort of visual thinking. Use these problems to develop your visualization skill. Make memory drawings to trace through the execution of code. Use drawings of the pre- and post-conditions of a problem to start thinking about a solution.

Count()

Write a Count() function that counts the number of times a given int occurs in a list. The code for this has the classic list traversal structure as demonstrated in Length().

```
void CountTest() {
List myList = BuildOneTwoThree(); // build {1, 2, 3}
int count = Count(myList, 2); // returns 1 since there's 1 '2' in the list
}
/*Given a list and an int, return the number of times that int occurs
in the list.
*/
int Count(struct node* head, int searchFor) {
// Your code
```

GetNth()

Write a GetNth() function that takes a linked list and an integer index and returns the data value stored in the node at that index position. GetNth() uses the C numbering convention that the first node is index 0, the second is index 1, ... and so on. So for the list {42, 13, 666} GetNth() with index 1 should return 13. The index should be in the range [0…length–1]. If it is not, GetNth() should assert() fail (or you could implement some other errorcase strategy).

```
void GetNthTest() {
struct node* myList = BuildOneTwoThree(); // build {1, 2, 3}
int lastNode = GetNth(myList, 2); // returns the value 3
}
```

Essentially, GetNth() is similar to an array[i] operation — the client can ask for elements by index number. However, GetNth() on a list is much slower than [ ] on an array. The advantage of the linked list is its much more flexible memory management — we can Push() at any time to add more elements and the memory is allocated as needed.

```
// Given a list and an index, return the data
// in the nth node of the list. The nodes are numbered from 0.
// Assert fails if the index is invalid (outside 0…lengh-1).
int GetNth(struct node* head, int index) {
// Your code
```

DeleteList()

Write a function DeleteList() that takes a list, deallocates all of its memory and sets its head pointer to NULL (the empty list).

```
void DeleteListTest() {
struct node* myList = BuildOneTwoThree(); // build {1, 2, 3}

DeleteList(&myList); // deletes the three nodes and sets myList to NULL
}
```

Post DeleteList() Memory Drawing

The following drawing shows the state of memory after DeleteList() executes in the above sample. Overwritten pointers are shown in gray and deallocated heap memory has an 'X' through it. Essentially DeleteList() just needs to call free() once for each node and set the head pointer to NULL.

DeleteList()

The DeleteList() implementation will need to use a reference parameter just like Push() so that it can change the caller's memory (myList in the above sample). The implementation also needs to be careful not to access the .next field in each node after the node has been deallocated.

```
void DeleteList(struct node** headRef) {
// Your code
```

1.9 Summary

- A data structure is an arrangement of data in a computer's memory or even disk storage.
- List is collection of scalar variables (heterogeneous), which are arranged in an order.
- An array is a data structure of multiple elements with the same data type. Array elements are accessed using subscript.
- A linked list is one of the fundamental data structures used in computer programming.
- A linked list is called a self-referential data.
- Linked lists are used as a building block for many other data structures, such as stacks, queues and their variations.

1.10 Keywords

Association lists: Linked lists are used to implement associative arrays, and are in this context called association lists.
Doubly-linked lists: Node has two links, one to the previous node and one to the next.
Linear linked lists: It has one link per node.
Linked list: It is one of the fundamental data structures used in computer programming.
Pointer: A pointer is a variable that contains the address of a variable.

1.11 Review Questions

1. What is the meaning of data structures?
2. Differentiate between lists and arrays?
3. What is linked list in data structure? Explain with example.
4. What are the operations can perform on the linked lists?
5. Write a c program to merge two linked lists?
6. How many types are there of linked lists?
7. Differentiate between singly and doubly-linked lists with examples.
8. What are the applications of linked lists?
9. Write a c program to add two long positive numbers.
10. How do we evaluate a polynomial using linked list?

Answers for Self Assessment Questions

1. (a) 2. (d) 3. (a) 4. (a) 5. (b) 6. (b) 7. (d) 8. (b)

1.12 Furhter Studies

1. Data Structures and Algorithms by Alfred V. Aho, Jeffrey D. Ullman, John E. Hopcroft
2. http://www.cs.toronto.edu/~gfb/csc148/2007W/07linked.pdf

2

Stacks

STRUCTURE

2.0 Objectives

2.1 Introduction

2.2 Concept of Stacks

2.3 Basic Operations of Stack

2.4 Array Implementation of a Stack

2.5 Stack as a Linked List

2.6 Stack as an Abstract Data Structure

2.7 Applications of Stacks

2.8 Evaluate a Postfix Expression and Precedence and Associativity of the Operators

2.9 Multiple Stacks

2.10 Summary

2.11 Keywords

2.12 Review Questions

2.13 Further Studies

2.0 Objectives

After studying this chapter, you will be able to:

- Discuss about a stacks
- Explain the basic operations of stack
- Discuss the array implementation of a stack
- Define the stack as a linked list
- Explain the stack as an abstract data structure
- Discuss the applications of stacks

2.1 Introduction

Stacks are simple data structures and important tools in programming language. Stacks are linear lists which have restrictions on the insertion and deletion operations. These are special cases of ordered list in which insertion and deletion is done only at the ends. The basic operations performed on stack are push and pop. The

stack implementation can be done in two ways - static implementation or dynamic implementation. Stack can be represented in the memory using a one-dimensional array or a singly linked list.
A stack is simply a list of elements with insertions and deletions permitted at one end-called the stack top. That means that it is possible to remove elements from a stack in reverse order from the insertion of elements into the stack. Thus, a stack data structure exhibits the LIFO (last in first out) property. Push and pop are the operations that are provided for insertion of an element into the stack and the removal of an element from the stack, respectively. Shown in Figure 2.2 are the effects of push and pop operations on the stack.

2.2 Concept of Stacks

A stack is a linear data structure in which allocation and deallocation are made in a last-in-first-out (LIFO) method. In the LIFO method, the insertions and deletions are done at one end which is known as the top of the stack (TOS). In a stack, the top is a variable which points to the top element in the stack. Consider a stack of books or stack of coins. The items can be added or removed only from the top. This means that the last item that is added to the stack is the first item to be removed.

2.2.1 Stack Structure

The stack data structure is used to maintain records of a file in which the order among the records of file is not important. Figure 2.1 displays the structure of a stack where stack is like a hollow cylinder with a closed bottom end and an open top end. In the stack data structure, the records are added and deleted at the top end. Last-In-First-Out (LIFO) principle is followed to retrieve records from the stack. The records added last are accessed first. In Figure 2.1, the order of entry of the records in the stack is item 1, 2, 3, 4 and the order of retrieval of the records from the stack is item 4, 3, 2, 1.

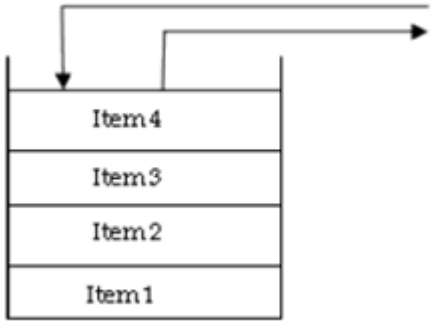


Figure 2.1: Stack.

Did You Know?

The stack was first proposed in 1955, and then patented in 1957, by the German Friedrich L. Bauer. The same concept was developed independently, at around the same time, by the Australian Charles Leonard Hamblin.

2.3 Basic Operations of Stack

The basic operations of stack are to:
1. Insert an element in the stack (Push operation)
2. Delete an element from the stack (Pop operation)

2.3.1 Push Operation

The procedure to insert a new element to the stack is called push operation. The push operation adds an element on the top of the stack. Top refers to the element on the top of stack. Push makes the Top point to the

recently added element on the stack. After every push operation, the Top is incremented by one. When the array is full, the status of stack is FULL and the condition is called stack overflow. No element can be inserted when the stack is full.

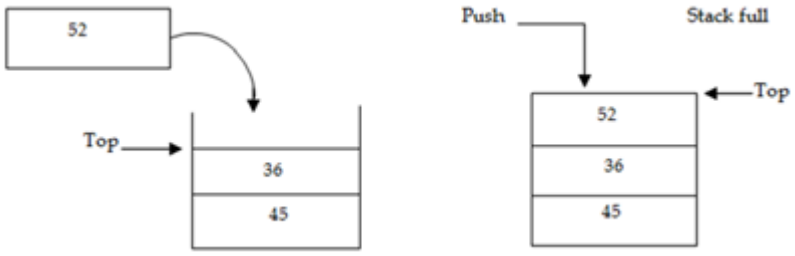


Figure 2.2: Push Operation in the Stack Data Structure.

Figure 2.2 shows the push operation in a stack. The stack has two elements 45 and 36. The 'Top' points to '36' as it is the last item in the stack. Element 52 is added on the stack through push operation. The 'Top' points to '52' after the push operation as it is the last item recently added. After adding 52, the stack is full or it is in stack overflow condition. No more items can be added in this stack. The syntax used for Push operation is PUSH (stack, item).

Algorithm to Implement Push Operation on Stack

```
PUSH (STACK, n, top, item) /* n = size of stack*/
if (top = n) then STACK_FULL; /* checks for stack overflow */
 else
 { top = top+1; /* increases the top by 1 */
 STACK [top] = item ;} /* inserts item in the new top position */
 end PUSH
```

2.3.2 Pop Operation

The procedure to delete an element from the top of the stack is called pop operation. After every pop operation, the 'Top' is decremented by one. When there is no element in the stack, the status of the stack is called empty stack or stack underflow. The pop operation cannot be performed when it is in stack underflow condition.

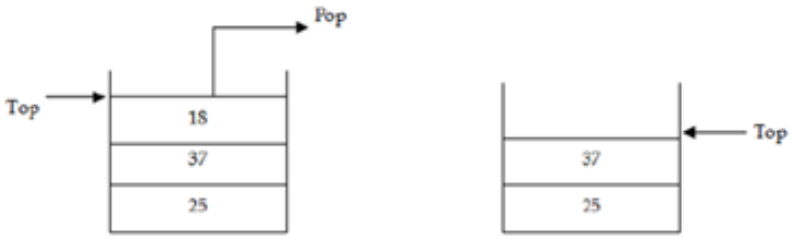


Figure 2.3: Pop Operation.

Figure 2.3 shows the pop operation in stack. The stack initially has three items, 25, 37 and 18. The 'Top' points to the last item, 18. After the pop operation, item 18 is deleted from stack. Now, the 'Top' points to 37. The syntax used for Pop operation is POP (stack).

Algorithm to Implement Pop Operation in a Stack

```
POP (STACK, top, item)
```

```
 if (top = 0) then STACK_EMPTY; /* check for stack underflow*/
else { item = STACK [top]; /* remove top element*/
top = top – 1; /* decrement stack top*/ }
 end POP
```

Representing Stacks in Memory
Stacks are represented in main memory by using one-dimensional arrays or by using a singly linked list.

2.4 Array Implementation of a Stack

When an array is used to implement a stack, the push and pop operations are realized by using the operations available on an array. The limitation of an array implementation is that the stack cannot grow and shrink dynamically as per the requirement.

Program: A complete C program to implement a stack using an array appears here:

```
#include<stdio.h>
void push(int);
int pop(void); //declare three function prototype
void display(void);
int main() //entry level of program
{
int choice=0,val=0; //initialize the variables
do
{
printf("\n\t1.Push 2.Pop 3.Display 4.Exit\n\tSelect Your Choice : ");
scanf("%d",&choice);
switch(choice)
{
```

Case 1:

```
printf("\tElement to be Pushed :");
scanf("%d",&val);
push(val); //value to be added of int data type and return void
break;
```

Case 2:

```
val=pop();
if(val!= –1) //check the condition val is not equal to –1 case of underflow
printf("\tPopped Element : %d\n",val);
break;
```

Case 3:

```
display();
break;
```

Case 4:

```
break;
```

```
default:
printf("\tWrong Choice");
break;
}
}while(choice!=4);
return 0;
}
```

Algorithm of above program:
Step 1: Three functions are declared: // a prototype
void push(int); // return void
int pop(void); // return integer value
void display(void); // return void

step 2: The main( ) method of the program is called. //entry level of the program

Step 3: Initialize the variable:
Choice =0;
Val=0;

Step 4: Take input for choice // through key board

Step 5: do-while loop is started
Switch condition is started to select the correct cases:
If user enter choice 1 case: 1 is selected.
Input the push element // push function is called and return void.
Go to step 1.

If user enter choice 2 case: 2 is selected.
Input the pop element // pop function is called and returns int.
Go to step 1.

If user enter choice 3 case: 3 is selected and display function is called. //return void.
Go to step 1.
If user enter choice 4 case: 4 at once terminate from the program and return 0.
If user enter any other number display the message "wrong choice".
Step 6: Repeat step 5
/*End of program*/

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: Write basic operation on stack?

……..……

……..……

……...

2.5 Stack as a Linked List

Initially the list is empty, so the top pointer is NULL. The push function takes a pointer to an existing list as the first parameter and a data value to be pushed as the second parameter, creates a new node by using the data value, and adds it to the top of the existing list. A pop function takes a pointer to an existing list as the first parameter, and a pointer to a data object in which the popped value is to be returned as a second parameter. Thus it retrieves the value of the node pointed to by the top pointer, takes the top point to the next node, and destroys the node that was pointed to by the top.

If this strategy is used for creating a stack with the previously used four data values: 10, 20, 30, and 40, then the stack is created as shown in Figure 2.4.

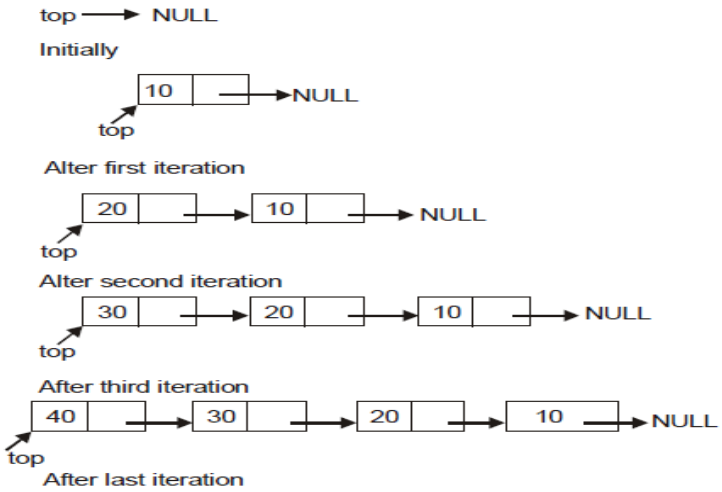


Figure 2.4: Linked Stack.

Program: A complete C program for implementation of a stack using the linked list is given here:

```
# include <stdio.h>
# include <stdlib.h>
struct node
{
int data;
struct node *link;
};
struct node *push(struct node *p, int value)
{
struct node *temp;
temp=(struct node *)malloc(sizeof(struct node));
/* creates new node
using data value
passed as parameter*/
if(temp==NULL)
{
printf("No Memory available Error\n");
exit(0);
}
temp->data = value;
```

```
temp->link = p;
p = temp;
return(p);
}
struct node *pop(struct node *p, int *value)
{
struct node *temp;
if(p==NULL)
{
printf(“ The stack is empty cannot pop Error\n”);
exit(0);
}
*value = p->data;
temp = p;
p = p->link;
free(temp);
return(p);
}
void main()
{
struct node *top = NULL;
int n,value;
do
{
do
{
printf(“Enter the element to be pushed\n”);
scanf(“%d”,&value);
top = push(top,value);
printf(“Enter 1 to continue\n”);
scanf(“%d”,&n);
} while(n == 1);
printf(“Enter 1 to pop an element\n”);
scanf(“%d”,&n);
while( n == 1)
{
top = pop(top,&value);
printf(“The value poped is %d\n”,value);
printf(“Enter 1 to pop an element\n”);
scanf(“%d”,&n);
}
printf(“Enter 1 to continue\n”);
scanf(“%d”,&n);
}
 while(n == 1);
}
```

Example: Input and Output

```
Enter the element to be pushed
10
Enter 1 to continue
1
Enter the element to be pushed
20
Enter 1 to continue
0
Enter 1 to pop an element
1
The value popped is 20
Enter 1 to pop an element
1
The value poped is 10
Enter 1 to pop an element
0
Enter 1 to continue
1
Enter the element to be pushed
30
Enter 1 to continue
1
Enter the element to be pushed
40
Enter 1 to continue
0
Enter 1 to pop an element
1
The value popped is 40
Enter 1 to pop an element
0
Enter 1 to continue
1
Enter the element to be pushed
50
Enter 1 to continue
0
Enter 1 to pop an element
1
The value popped is 50
Enter 1 to pop an element
1
The value popped is 30
Enter 1 to pop an element
0
Enter 1 to continue
```

2.6 Stack as an Abstract Data Structure

A stack is an ordered list in which insertions and deletions are made at one end called “top”. It is also known as Last-In-First-Out (LIFO) list.

Applications: Page-visited history in IE (use ’back’ button) Undo sequence in text editor Program function stack.

2.6.1 ADT Stack Operations

Table 2.5: If it is array-based, need to consider whether or not the stack is full in push( ) operation.

create an empty stack	+ create Stack ( )
destroy a stack	+ destroy Stack ( )
check whether or not the stack is empty	+ is Empty ( ): boolean {query}
insert a new element	+ push (in new Element:Stackitem Type) throw StackException
delete an element	+ pop(out stack Top: StackitemType) throw StackException +pop( ) throw StackException
retrieve an element	+ getTop(out stack Top: StackItemType) {query} throw StackException

Array-Based Implementation

Program

```
// define stack exception
#include <exception>
#include <string>
using namespace std;
class StackException: : public runtime_error
}
public:
StackException(const string & message= “”)
: exception(message.c_str())
{}
}; // end StackException

********************************************** //
// Header file StackA.h for the ADT stack.
// Array-based implementation.
********************************************** //
#include “StackException.h”
```

```
const int MAX_STACK = maximum-size-of-stack;
typedef desired-type-of-stack-item StackItemType;
class Stack
}
public:
// constructors and destructor:
Stack(); // default constructor
// copy constructor and destructor are supplied by the compiler
// stack operations:
bool isEmpty() const;
// Determines whether a stack is empty.
void push(StackItemType newItem) throw(StackException);
// Adds an item to the top of a stack.
// Exception: Throws StackException if the item cannot
// be placed on the stack
void pop() throw(StackException);
// Removes the top of a stack.
// Exception: Throws StackException if the stack is empty.
void pop(StackItemType& stackTop) throw(StackException);
// Retrieves and removes the top of a stack.
// Exception: Throws StackException if the stack is empty.
void getTop(StackItemType& stackTop) const
throw(StackException);
// Retrieves the top of a stack.
// Exception: Throws StackException if the stack is empty.
private:
StackItemType items[MAX_STACK]; // array of stack items
int top; // index to top of stack
}; // end class
// End of header file.
```

Self Assessment Questions

1. The second expression (j - k) in the following expression will be evaluated (i + 5) && (j – k)

(a) if expression (i + 5) is true. (b) if expression (i + 5) is false.

(c) irrespective of whether (i + 5) is true or false. (d) will not be evaluated in any case.

2. If a, b and c are integer variables with the values a = 8, b = 3 and c = –5. Then what is the value of the arithmetic expression: 2 * b + 3 * (a – c)

(a) 45 (b) 6 (c) –16 (d) –1

3. If the variables i, j and k are assigned the values 5, 3 and 2 respectively, then the expression i = j + (k + + = 6) + 7

(a) gives an error message (b) assigns a value 16 to i

(c) assigns a value 18 to i (d) assigns a value 19 to i

2.7 Applications of Stacks

Three applications of stacks are presented here. These examples are central to many activities that a computer must do and deserve time spent with them.
1. Expression evaluation
2. Backtracking (game playing, finding paths, exhaustive searching)
3. Memory management, run-time environment for nested language features.

2.7.1. Expression Evaluation

In particular we will consider arithmetic expressions. Understand that there are Boolean and logical expressions that can be evaluated in the same way. Control structures can also be treated similarly in a compiler.
This study of arithmetic expression evaluation is an example of problem solving where you solve a simpler problem and then transform the actual problem to the simpler one.
Aside: The NP-Complete problem. There are a set of apparently intractable problems: finding the shortest route in a graph (Traveling Salesman Problem), bin packing, linear programming, etc. that are similar enough that if a polynomial solution is ever found (exponential solutions abound) for one of these problems, then the solution can be applied to all problems.

(a) Infix, Prefix and Postfix Notation

We are accustomed to write arithmetic expressions with the operation between the two operands: a + b or c / d. If we write a + b * c, however, we have to apply precedence rules to avoid the ambiguous evaluation (add first or multiply first?).

There's no real reason to put the operation between the variables or values. They can just as well precede or follow the operands. You should note the advantage of prefix and postfix: the need for precedence rules and parentheses are eliminated

Infix	Prefix	Postfix
a + b	+ a b	a b +
a + b * c	+ a * b c	a b c * +
(a + b) * (c - d)	* + a b - c d	a b + c d - *
b * b - 4 * a * c		
40 - 3 * 5 + 1		

Postfix expressions are easily evaluated with the aid of a stack.

(b) Postfix Evaluation Algorithm

Assume we have a string of operands and operators, an informal, by hand process is
1. Scan the expression left to right
2. Skip values or variables (operands)
3. When an operator is found, apply the operation to the preceding two operands
4. Replace the two operands and operator with the calculated value (three symbols are replaced with one operand)
5. Continue scanning until only a value remains the result of the expression

The time complexity is O(n) because each operand is scanned once, and each operation is performed once.
A more formal algorithm:

```
create a new stack while(input stream is not empty)
{
token = getNextToken();
if(token instanceof operand){
push(token);
}
else
if (token instance of operator)
op2 = pop();
op1 = pop();
result = calc(token, op1, op2);
push(result);
}
}
return pop();
```

Demonstration with 2 3 4 + * 5 –

(c) Infix transformation to Postfix

This process uses a stack as well. We have to hold information that's expressed inside parentheses while scanning to find the closing ') '. We also have to hold information on operations that are of lower precedence on the stack.

The algorithm is:

1. Create an empty stack and an empty postfix output string/stream
2. Scan the infix input string/stream left to right
3. If the current input token is an operand, simply append it to the output string (note the examples above that the operands remain in the same order)
4. If the current input token is an operator, pop off all operators that have equal or higher precedence and append them to the output string; push the operator onto the stack. The order of popping is the order in the output.
5. If the current input token is '('', push it onto the stack
6. If the current input token is ')', pop off all operators and append them to the output string until a '(' is popped; discard the '('.
7. If the end of the input string is found, pop all operators and append them to the output string.

This algorithm does not handle errors in the input, although careful analysis of parenthesis or lack of parenthesis could point to such error determination.

Apply the algorithm to the above expressions.

2.7.2 Backtracking

Backtracking is used in algorithms in which there are steps along some path (state) from some starting point to some goal.

1. Find your way through a maze.
2. Find a path from one point in a graph (roadmap) to another point.
3. Play a game in which there are moves to be made (checkers, chess).

In all of these cases, there are choices to be made among a number of options. We need some way to remember these decision points in case we want/need to come back and try the alternative.

Consider the maze. At a point where a choice is made, we may discover that the choice leads to a dead-end. We want to retrace back to that decision point and then try the other (next) alternative. Again, stacks can be used as a part of the solution. Recursion is another, typically more favored, solution, which is actually implemented by a stack.

2.7.3 Memory Management

Any modern computer environment uses a stack as the primary memory management model for a running program.

When a method/function is called:-

1. An activation record is created; its size depends on the number and size of the local variables and parameters.
2. The Base Pointer value is saved in the special location reserved for it.
3. The Program Counter value is saved in the Return Address location.
4. The Base Pointer is now reset to the new base (top of the call stack prior to the creation of the AR).
5. The Program Counter is set to the location of the first byte code of the method being called.
6. Copies the calling parameters into the Parameter region.
7. Initializes local variables in the local variable region.

While the method executes, the local variables and parameters are simply found by adding a constant associated with each variable/parameter to the Base Pointer. When a method returns:-

1. Get the program counter from the activation record and replace what's in the PC.
2. Get the base pointer value from the AR and replace what's in the BP.
3. Pop the AR entirely from the stack.

Example: If the infix expression is a * b + c / d, then different snapshots of the algorithm, while scanning the expression from right to left, are shown in Table 2.2.

Table 2.6: Scanning the infix expression a * b +c / d from right to left

Steps	*Remaining Expression*	*Scanned Element*	*Queue of Operands*	*Stack of Operators*	*Prefix Output*
0	*a*b+c/d*	*nil*	*empty*	*Empty*	*nil*
1	*a*b+c/*	*d*	*d*	*Empty*	*nil*
2	*a*b+c*	*/*	*d*	*/*	*nil*
3	*a*b+*	*c*	*d c*	*/*	*nil*
4	*a*b*	*+*	*empty*	*+*	*dc/*
5	*a**	*b*	*b*	*+*	*dc/*
6	*A*	***	*b*	** +*	*dc/*
7	*Nil*	*a*	*b a*	** +*	*dc/*
8	*Nil*	*nil*	*empty*	*Empty*	*dc/ba*+*

The final prefix output that we get is d c / b a * + whose reverse is + * a b / c d, which is the prefix equivalent of the input infix expression a * b + c * d. Note that all the operands are simply pushed to the queue in steps 1, 3, 5, and 7. In step 2, the operator / is pushed to the empty stack of operators. In step 4, the operator + is checked against the elements in the stack. Since/(division) has higher priority than + (addition), the queue is emptied to the prefix output (thus we get 'dc' as the output) and then the operator/is written (thus we get 'dc/'

as the output). The operator + is then pushed to the stack. In step 6, the operator * is checked against the stack elements. Since * (multiplication) has a higher priority than + (addition), * is pushed to the stack. Step 8 signifies that all of the infix expression is scanned. Thus, the queue of operands is emptied to the prefix output (to get'd c / b a'), followed by the emptying of the stack of operators (to get'd c / b a * +').

Did You Know?

A polish mathematician Jan Lukasiewicz suggested a notation called Polish notation, which gives two alternatives to represent an arithmetic expression.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Write the operation of stack?

……..……

……..……

……...

2.8 Evaluate a Postfix Expression and Precedence and Associativity of the Operators

2.8.1 Operator Precedence and Associativity Table

Operator Name	Associativity	Operators
Primary scope resolution	left to right	::
Primary	left to right	() [ ] . ->
Unary	right to left	++ –– + – ! ~ & *
C++ Pointer to Member	left to right	.*->*
Multiplicative	left to right	* / %
Additive	left to right	+ –
Bitwise Shift	left to right	<< >>
Relational	left to right	< > <= >=
Equality	left to right	== !=
Bitwise AND	left to right	&
Bitwise Exclusive OR	left to right	^
Bitwise Inclusive OR	left to right	|
Logical AND	left to right	&&

Logical OR	left to right	||
Conditional	right to left	? :
Assignment	right to left	= += – = *= /= <<= >>= %= &= ^= |=
Comma	left to right	,

2.8.2 Evaluate a Postfix Expression

Scan the expression from left to right, use stack to hold operands if a character is an operand: push into stack if a character is an operator: pop two operands from stack evaluate the result push the result back to the stack final result is in the stack.

Example: 234+*5 – == 2 * (3+4) – 5

Program

Char	Stack
2	2
3	2, 3
4	2, 3, 4
+	2, 7 // 3 + 4
*	14 // 3 + 4
5	14, 5
–	9 //14 - 5
Result = 9	

2.8.3 Conversion of Expressions

Program: ***Convert an Infix expression to Prefix form.***

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <ctype.h>
#define MAX 50
struct infix
{
char target[MAX] ;
char stack[MAX] ;
char *s, *t ;
int top, l ; } ;
void initinfix ( struct infix * ) ;
void setexpr ( struct infix *, char * ) ;
void push ( struct infix *, char ) ;
char pop ( struct infix * ) ;
void convert ( struct infix * ) ;
int priority ( char c ) ;
void show ( struct infix ) ;
void main( )
```

```
{struct infix q ;
char expr[MAX] ;
clrscr( ) ;
initinfix ( &q ) ;
printf ( “\nEnter an expression in infix form: ” ) ;
gets ( expr ) ;
setexpr ( &q, expr ) ;
convert ( &q ) ;
printf ( “The Prefix expression is: ” ) ;
show ( q ) ;
getch( ) ;
}/* initializes elements of structure variable */
void initinfix ( struct infix *pq )
{ pq -> top = –1 ;
        strcpy ( pq -> target, “” ) ;
        strcpy ( pq -> stack, “” ) ;
        pq -> l = 0 ; } /* reverses the given expression */
void setexpr ( struct infix *pq, char *str )
{ pq -> s = str ;
        strrev ( pq -> s ) ;
        pq -> l = strlen ( pq -> s ) ;
        *( pq -> target + pq -> l ) = '\0' ;
        pq -> t = pq -> target + ( pq -> l – 1 ) ;
} /* adds operator to the stack */
void push ( struct infix *pq, char c )
{ if ( pq -> top == MAX – 1 )
printf ( “\nStack is full.\n” ) ;
else {
pq -> top++ ;
pq -> stack[pq -> top] = c ;
} } /* pops an operator from the stack */
char pop ( struct infix *pq )
{ if ( pq -> top == –1 )
{ printf ( “Stack is empty\n” ) ;
return –1 ; }
else
{
char item = pq -> stack[pq -> top] ;
pq -> top– – ;
return item ; }
} /* converts the infix expr. to prefix form */
void convert ( struct infix *pq )
{
char opr ;
while ( *( pq -> s )  ) {
if ( *( pq -> s )  == ' ' || *( pq -> s )  == '\t' )
{
```

```
pq -> s++    ;
continue;
}
if ( isdigit ( *( pq -> s )  ) || isalpha ( *( pq -> s )  ) )
{ while ( isdigit ( *( pq -> s )  ) || isalpha ( *( pq -> s )  ) )
{
*( pq -> t ) = *( pq -> s )  ;
pq -> s++   ;
pq -> t– – ;
} }
if ( *( pq -> s )  == ')' )
{
push ( pq, *( pq -> s ) ) ;
pq -> s++    ; }
if ( *( pq -> s )  == '*' || *( pq -> s )  == '+' || *( pq -> s )  == '/' || *( pq -> s )  == '%' || *( pq -> s ) == ' – ' || *(
pq -> s )  == '$' )
{
if ( pq -> top != –1 )
{
opr = pop ( pq ) ;
while ( priority ( opr ) > priority ( *( pq -> s )  ) )
{
*( pq -> t ) = opr ;
pq -> t– – ;
opr = pop ( pq ) ;
}
push ( pq, opr ) ;
push ( pq, *( pq -> s )  ) ;
}
else
push ( pq, *( pq -> s ) ) ;
pq -> s++    ;
}
if ( *( pq -> s )  == '(' )
{
opr = pop ( pq ) ;
while ( opr != ')' )
{
*( pq -> t ) = opr ;
pq -> t– – ;
opr =  pop ( pq ) ; }
pq -> s++    ;
} }
while ( pq -> top != –1 )
{ opr = pop ( pq ) ;
*( pq -> t ) = opr ;
pq -> t– – ;
```

```
}
pq -> t++ ;
} /* returns the priotity of the operator */
int priority ( char c )
{ if ( c == '$' )
        return 3 ;
    if ( c == '*' || c == '/' || c == '%' )
        return 2 ;
    else {
        if ( c == '+' || c == '-' )
            return 1 ;
        else return 0 ;} } /* displays the prefix form of given expr. */
void show ( struct infix pq )
{
    while ( *( pq.t ) )
    {
        printf ( " %c", *( pq.t ) ) ;
        pq.t++ ;}}
```

Convert an Infix form to Postfix form

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <ctype.h>
#define MAX 50
struct infix
{
    char target[MAX] ;
    char stack[MAX] ;
    char *s, *t ;
    int top ;
} ;
void initinfix ( struct infix * ) ;
void setexpr ( struct infix *, char * ) ;
void push ( struct infix *, char ) ;
char pop ( struct infix * ) ;
void convert ( struct infix * ) ;
int priority ( char ) ;
void show ( struct infix ) ;
void main( )
{
  struct infix p ;
      char expr[MAX] ;
  initinfix ( &p ) ;
  clrscr( ) ;
  printf ( "\nEnter an expression in infix form: " ) ;
      gets ( expr ) ;
```

```
        setexpr ( &p, expr ) ;
        convert ( &p ) ;
        printf ( "\nThe postfix expression is: " ) ;
        show ( p ) ;
   getch( ) ;
} /* initializes structure elements */
void initinfix ( struct infix *p )
{
        p -> top = –1 ;
        strcpy ( p -> target, "" ) ;
        strcpy ( p -> stack, "" ) ;
        p -> t = p -> target ;
        p -> s = ""  ; } /* sets s to point to given expr. */
void setexpr ( struct infix *p, char *str )
{
        p -> s = str ;
} /* adds an operator to the stack */
void push ( struct infix *p, char c )
{
        if ( p -> top == MAX )
                printf ("\nStack is full.\n" ) ;
        else
        {
                p -> top++ ;
                p -> stack[p -> top] = c ;
        }
}/* pops an operator from the stack */
char pop ( struct infix *p )
{
        if ( p -> top == –1 )
        {
                printf ("\nStack is empty.\n" ) ;
                return –1 ;
        }
        else
        {
                char item = p -> stack[p -> top] ;
                p -> top-- ;
                return item ;
        }
}/* converts the given expr. from infix to postfix form */
void convert ( struct infix *p )
{
   char opr ;

        while ( *( p -> s ) )
        {
```

```
		if ( *( p -> s ) == ' ' || *( p -> s ) == '\t' )
		{
			p -> s++ ;
			continue ;
		}
		if ( isdigit ( *( p -> s ) ) || isalpha ( *( p -> s ) ) )
		{
			while ( isdigit ( *( p -> s ) ) || isalpha ( *( p -> s ) ) )
			{
				*( p -> t ) = *( p -> s ) ;
				p -> s++ ;
				p -> t++ ;
			}
		}
		if ( *( p -> s ) == '(' )
		{
			push ( p, *( p -> s ) ) ;
			p -> s++ ;
		}
if ( *( p -> s ) == '*' || *( p -> s ) == '+' || *( p -> s ) == '/' || *( p -> s ) == '%' || *( p -> s ) == '–' || *( p -> s ) == '$' )
		{
			if ( p -> top != –1 )
			{
				opr = pop ( p ) ;
				while ( priority ( opr ) >= priority ( *( p -> s ) ) )
				{
					*( p -> t ) = opr ;
					p -> t++ ;
					opr = pop ( p ) ;
				}
				push ( p, opr ) ;
				push ( p, *( p -> s ) ) ;
			}
			else
				push ( p, *( p -> s ) ) ;
			p -> s++ ;
		}
		if ( *( p -> s ) == ')' )
		{
			opr = pop ( p ) ;
			while ( ( opr ) != '(' )
			{
				*( p -> t ) = opr ;
				p -> t++ ;
				opr =  pop ( p ) ;
			}
```

```
                        p -> s++ ;
                }
        }

        while ( p -> top != –1 )
        {
                char opr = pop ( p ) ;
                *( p -> t ) = opr ;
                p -> t++ ;
        }

        *( p -> t ) = '\0' ;
} /* returns the priority of an operator */
int priority ( char c )
{
        if ( c == '$' )
                return 3 ;
        if ( c == '*' || c == '/' || c == '%' )
                return 2 ;
        else
        {
                if ( c == '+' || c == '–' )
                        return 1 ;
                else
                        return 0 ;
        } } /* displays the postfix form of given expr. */
void show ( struct infix p )
{
   printf ("%s", p.target ) ;
}
```

3) Convert expression in postfix form to prefix form

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#define MAX 50
struct postfix
{ char stack[MAX][MAX], target[MAX] ;
char temp1[2], temp2[2] ;
char str1[MAX], str2[MAX], str3[MAX] ;
int i, top ;
} ;
void initpostfix ( struct postfix * ) ;
void setexpr ( struct postfix *, char * ) ;
void push ( struct postfix *, char * ) ;
void pop ( struct postfix *, char * ) ;
void convert ( struct postfix * ) ;
void show ( struct postfix ) ;
```

```
void main( )
{
struct postfix q ;
char expr[MAX] ;
clrscr( ) ;
initpostfix ( &q ) ;
printf ( “\nEnter an expression in postfix form: ”) ;
gets ( expr ) ;
setexpr ( &q, expr ) ;
convert ( &q ) ;
printf ( “\nThe Prefix expression is: ” ) ;
show ( q ) ;
getch( ) ;
} /* initializes the elements of the structure */
void initpostfix ( struct postfix *p )
{ p -> i = 0 ;
p -> top = –1 ;
strcpy ( p -> target, “” ) ;
} /* copies given expr. to target string */
void setexpr ( struct postfix *p, char *c )
{ strcpy ( p -> target, c ) ;
}/* adds an operator to the stack */
void push ( struct postfix *p, char *str )
{
if ( p -> top == MAX – 1 )
printf (“\nStack is full.” ) ;
else {
p -> top++ ;
strcpy ( p -> stack[p -> top], str ) ;
}
} /* pops an element from the stack */
void pop ( struct postfix *p, char *a )
{ if ( p -> top == –1 )
printf (“\nStack  is empty.” ) ;
else
{
strcpy ( a, p -> stack[p -> top] ) ;
p -> top– – ; }
} /* converts given expr. to prefix form */
void convert ( struct postfix *p )
{ while ( p -> target[p -> i] != ‘\0’ )
{ /* skip whitespace, if any */
if ( p -> target[p -> i] == ‘ ’)
p -> i++ ;
if( p -> target[p -> i] == ‘%’ || p -> target[p -> i] == ‘*’ ||
p -> target[p -> i] == ‘-’ || p -> target[p -> i] == ‘+’ ||
p -> target[p -> i] == ‘/’ || p -> target[p -> i] == ‘$’ )
```

```
{ pop ( p, p -> str2 ) ;
pop ( p, p -> str3 ) ;
p -> temp1[0] = p -> target[ p -> i] ;
p -> temp1[1] = '\0' ;
strcpy ( p -> str1, p -> temp1 ) ;
strcat ( p -> str1, p -> str3 ) ;
strcat ( p -> str1, p -> str2 ) ;
push ( p, p -> str1 ) ;
}
else
                {
                        p -> temp1[0] = p -> target[p -> i] ;
                        p -> temp1[1] = '\0' ;
                        strcpy ( p -> temp2, p -> temp1 ) ;
                        push ( p, p -> temp2 ) ;
                }
                p -> i++ ;
        } }/* displays the prefix form of expr. */
void show ( struct postfix p )
{
        char *temp = p.stack[0] ;
while ( *temp ) {
printf ("%c", *temp ) ;
temp++;
}
}
```

Convert an expression in postfix form to an infix form

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#define MAX 50
struct postfix
{ char stack[MAX][MAX], target[MAX] ;
char temp1[2], temp2[2];
char str1[MAX], str2[MAX], str3[MAX] ;
int i, top ;
} ;
void initpostfix ( struct postfix * ) ;
void setexpr ( struct postfix *, char * ) ;
void push ( struct postfix *, char * ) ;
void pop ( struct postfix *, char * ) ;
void convert ( struct postfix * ) ;
void show ( struct postfix ) ;
void main( )
{
struct postfix q ;
```

```
char expr[MAX] ;
clrscr( ) ;
initpostfix ( &q ) ;
printf (“\nEnter an expression in postfix form: ”) ;
gets ( expr ) ;
setexpr ( &q, expr ) ;
convert ( &q ) ;
printf (“\nThe infix expression is: ”) ;
show ( q ) ;
getch( ) ;
}/* initializes data member */
void initpostfix ( struct postfix *p )
{
p -> i = 0 ;
p -> top = –1 ;
strcpy ( p -> target, “” ) ;
}/* copies given expression to target string */
void setexpr ( struct postfix *p, char *c )
{
strcpy ( p -> target, c ) ;
} /* adds an expr. to the stack */
void push ( struct postfix *p, char *str )
{
if ( p -> top == MAX – 1 )
printf ( “\nStack is full.” ) ;
else
{
p -> top++ ;
strcpy ( p -> stack[p -> top], str ) ;
}
}
/* pops an expr. from the stack */
void pop ( struct postfix *p, char *a )
{
if ( p -> top == -1 )
printf (“\nStack  is empty.” ) ;
else
{
strcpy ( a, p -> stack[p -> top] ) ;
p -> top– – ;
}
}
/* converts given expr. to infix form */
void convert ( struct postfix *p )
{
while ( p -> target[p -> i] )
{
```

```
/* skip whitespace, if any */
if( p -> target[p -> i] == ‘ ’ )
p -> i++ ;
if ( p -> target[p -> i] == ‘%’ || p -> target[p -> i] == ‘*’ ||  p -> target[p -> i] == ‘-’ || p -> target[p -> i] == ‘+’ ||
p -> target[p -> i] == ‘/’ || p -> target[p -> i] == ‘$’ )
{
pop ( p, p -> str2 ) ;
pop ( p, p -> str3 ) ;
p -> temp1[0] = p -> target[p -> i] ;
p -> temp1[1] = ‘\0’ ;
strcpy ( p -> str1, p -> str3 ) ;
strcat ( p -> str1, p -> temp1 ) ;
strcat ( p -> str1, p -> str2 ) ;
push ( p, p -> str1 ) ;
}
else
{
p -> temp1[0] = p -> target[p -> i] ;
p -> temp1[1] = ‘\0’ ;
strcpy ( p -> temp2, p -> temp1 ) ;
push ( p, p -> temp2 ) ;}
p -> i++ ;
} }
/* displays the expression */
void show ( struct postfix p )
{
char *t ;
t = p.stack[0] ;
while ( *t )
{ printf (“c”, *t ) ;
t++ ; } }
```

Evaluate an epression entered in postfix form

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>
#include <ctype.h>
#define MAX 50
struct postfix
{ int stack[MAX] ;
int top, nn ;
char *s ;
} ;
void initpostfix ( struct postfix * ) ;
void setexpr ( struct postfix *, char * ) ;
void push ( struct postfix *, int ) ;
```

```
int pop ( struct postfix * ) ;
void calculate ( struct postfix * ) ;
void show ( struct postfix ) ;
void main( )
{
struct postfix q ;
char expr[MAX] ;
clrscr( ) ;
initpostfix ( &q ) ;
printf (“\nEnter postfix expression to be evaluated: ”) ;
gets ( expr ) ;
setexpr ( &q, expr ) ;
calculate ( &q ) ;
show ( q ) ;
getch( ) ; }
/* initializes data members */
void initpostfix ( struct postfix *p )
{
p -> top = –1 ;}
/* sets s to point to the given expr. */
void setexpr ( struct postfix *p, char *str )
{
p -> s = str ;
}
/* adds digit to the stack */
void push ( struct postfix *p, int item )
{
if ( p -> top == MAX – 1 )
printf (“\nStack is full.” ) ;
else
{
p -> top++ ;
p -> stack[p -> top] = item ;
}
} /* pops digit from the stack */
int pop ( struct postfix *p )
{
        int data ;
if ( p -> top == –1 ) {
printf (“\nStack is empty.” ) ;
return NULL ;
}
data = p -> stack[p -> top] ;
p -> top– – ;
return data ;
} /* evaluates the postfix expression */
void calculate( struct postfix *p )
```

```
{
int n1, n2, n3 ;
while ( *( p -> s ) )
{/* skip whitespace, if any */
if ( *( p -> s ) == ‘ ‘ || *( p -> s ) == ‘\t’ )
{
p -> s++ ;
continue;
}
/* if digit is encountered */
if ( isdigit ( *( p -> s ) ) )
{
p -> nn = *( p -> s ) – ‘0’ ;
push ( p, p -> nn ) ;
}
else
{ /* if operator is encountered */
n1 = pop ( p ) ;
n2 = pop ( p ) ;
switch ( *( p -> s ) )
{
case ‘+’ :
 n3 = n2 + n1 ;
break ;
case ‘–’ :
  n3 = n2 – n1 ;
  break ;
case ‘/’ :
  n3 = n2 / n1 ;
  break ;
case ‘*’ :
  n3 = n2 * n1 ;
  break ;
case ‘%’ :
  n3 = n2 % n1 ;
  break ;
case ‘$’ :
n3 = pow ( n2 , n1 ) ;
 break ;
default :
 printf (“Unknown operator” ) ;
  exit ( 1 ) ;
}
push ( p, n3 ) ;
}
p -> s++ ;} } /* displays the result */
void show ( struct postfix p )
```

```
{
        p.nn = pop ( &p ) ;
        printf ("Result is: %d", p.nn ) ;
}
```

Self Assessment Questions

4. The postfix form of the following infix notation is: (A + B)* (C * D – E)* F.
   (a) AB + CD – E – *F * (b) AB + CDE + – * F *
   (c) AB + CD – EF + – ** (d) ABCDEF* – + * +

5. If 'y' is of integer type then the expressions (y – 8) /9 and (y – 8)/9 – 3:
   (a) must yield the same value.
   (b) must yield different values.
   (c) may or may not yield the same value.
   (d) None of the above.

6. The result of evaluating the following postfix expression is:
   5, 7, 9, *, +, 4, 9, 3, /, +, –
   (a) 50 (b) 65 (c) 61 (d) 69

7. An infix expression can be converted to a postfix expression using a:
   (a) Stack (b) Queue (c) Dequeue (d) None of these

Exercise: Check Your Progress 3
Note: i) Use the space below for your answer.
Ex1: Transform each of the following expression to prefix and postfix.
(a) A+B-C
(b) (A + B)*(C–D)$E*F
(c) (A + B)*(C$(D – E)+F) – G

……..……

……..……

……...

Caution
In the stacks reserving a small amount of space for each stack may increase the numbers of times overflow occurs and the time required for resolving an overflow such as by adding space to the stack may be more expensive than the space saved.

2.9 Multiple Stacks

A multiple stack program is a program with more than one stack. We are talking about program defined stacks, not the implicit stack that runs behind the scenes.

One algorithm would be to create a linked list. The push operation, given the address of the pointer to the head of the stack, would allocate and insert an element in front of the head. Pop would retrieve the head and deallocate it. An incomplete example, using integers as the elements... (For readability, indicates tabs.)

Program

```
#define MAX 10
#include<stdio.h>
#include<conio.h>
int stack[MAX],topA=-1,topB=MAX;
void pushA(int no) // The push operation adds a new item to the top of the stack
{ if(topA==topB) //check for the overflow
{ printf(.\n OVERFLOW.);
return;
} stack[++(topA)]=no;
} int popA() // The pop() method removes the last element of an array, and returns that element of the stack.
{ if(topA== –1) //check for the underflow
{
printf(“\n UNDERFLOW”);
return –999;
} return stack[(topA)– –];
} void showA()
{
int i;
if(topA== –1) // check the condition for stack empty
{ printf(“\n stack Empty”);
return;
} for(i=topA;i>=0;i– –)
{ printf(“\n %d.,stack[i]”); // return stack elements by for loop
}} void pushB(int no)
{ if(topB–1==topA) //check for stack B with stack A
{ printf(“\n OVERFLO”.);
return;
} stack[.(topB)]=no;
} int popB()
{ if(topB==MAX)
{ printf(“.\n UNDERFLOW.”);
return –999;
} return stack[(topB)– –];
} void showB()
{ int i;
if(topB==MAX)
{ printf(“.\n stack Empty.”);
return;
} for(i=topB;i<MAX;i++)
{ printf(“.\n %d.,stack[i]”);
}
} void main() //program start from here
```

```
{ clrscr();
int ch,val;
do // do-while loop start
{ printf(“.\n\n\n 1 PUSH A.”);
printf(“.\n 2 PUSH B.”);
printf(“.\n 3 POP A.”);
printf(“.\n 4 POP B.”);
printf(“.\n 5 Show A.”);
printf(“.\n 6 Show B.”);
printf(“.\n 0 EXIT.”);
printf(“\nEnter your Choice”);
scanf(“%d”,&ch);
switch(ch)
{ case 1:printf(“\n enter Number”);
scanf(“%d”,&val);
pushA(val); //pushA() method called and go to declaration of pushA()
break;
case 2: printf(“\n enter Number”);
scanf(“%d”,&val);
pushB(val); //pushB() method called and go to declaration of pushB()
break;
case 3: val=popA();
if(val!=-999)
printf(“%d popped”,val);
break;
case 4 : val=popB();
if(val!= –999)
printf(“%d popped”,val);
break;
case 5: showA();
break;
case 6: showB();
break;
case 0:break;
default:printf(“\n Invalid choice”);
}}while(ch!=0);
getch();
}
```

Algorithm of Above Program:

Step 1: Declare the global varable stack[MAX], topA=–1, topB=MAX; //int data type

Step 2: PushA() method is declared of int no parameter. //void return type Check the condition: if(topA==topB) // –1=MAX

Print overflow and return void

return stack[++(topA)]=no; //increment the position of stack by 1 in pushA()

Step 3: Declare the popA() method with int return type.
Check the condition: if(topA== –1) //no element
Print underflow and return –999
Return stack[(topA) – –]; // decrement the position of stack by 1in popA()

Step 4: Declare the variable i //int data type
Check the condition: if(topA== –1) //no element
Print stack empty

Step 5: Loop for print for stack[i]
Loop start from topA to above 0
End of loop

Step 6: Declare pushB() method with int no parameter. //return type void
Check the condition: if(topB–1==topA) // the topA element equal to topB–1 element
If true print overflow
Return stack[– –(topB)]=no; //return stack (– –topB)

Step 7: Declare popB( ) method //int return type
Check condition: if(topB==MAX)
If true print underflow
return stack[(topB) – –]; //decremented the top by 1

Step 8: check the condition: if(topB==MAX)
If true print empty
Print the stack[i] value by for loop from topB to max

Step 9: Main( ) method is called //from here program is started
Do-while loop start
Initialize the variable ch, val //local variable

Step 10: Now print all the method pushA, popA, showA and so on //Menu base

Step 11: Now input the choice ch //by scanf method
If enter choice 1 case 1 is activate and input the value ‘val’
pushA(val); called go to step 1
If enter choice 2 case 2 is activate and input the value ‘val’
pushB(val); called go to step 6
If enter choice 3 case 3 is activate
val=popA( ); // popA value store in val
Now check the condition: if(val!= –999) //!= not equal to
If condition true: print poped value
Similarly in case 4

Step 12: If enter choice 5 case 5 is activate
Show A //show push and pop values

Step 13: If enter choice 6 case 6 is activate
Show B //show push and pop values

Step 14: If enter choice 0 case 0 is activate
Program at once terminated

Step 15: If enter any other choice print invalid number
Now while loop check the condition ch!=0
And return to the start of the loop and continue till condition is not false
End of loop
/*end of program*/

Exercise: Check Your Progress 4

Note: i) Use the space below for your answer.

Ex1: Write a C program to convert the following expression into prefix and postfix expressions using stacks.

(A + B) * (C – D)/ (E – F)

……..……

……..……

……...

Lab Exercise

1. Implement a stack in C in which each item on the stack is a varying number of integers. Choose a C data structure for such a stack and design push and pop routines for it.
2. Write a program for copying contents of one stack to another.

Case Study: Full-Stack Programming

There has been a lot of talk about "full-stack" programmers in the tech press recently. Along with the concept of "devops" this is an important distinction to make between pure coders and people who are capable of systems thinking. Generally speaking, in addition to a deep understanding of their core programming skills, they understand other parts of the IT stack like hardware, networking and operations.

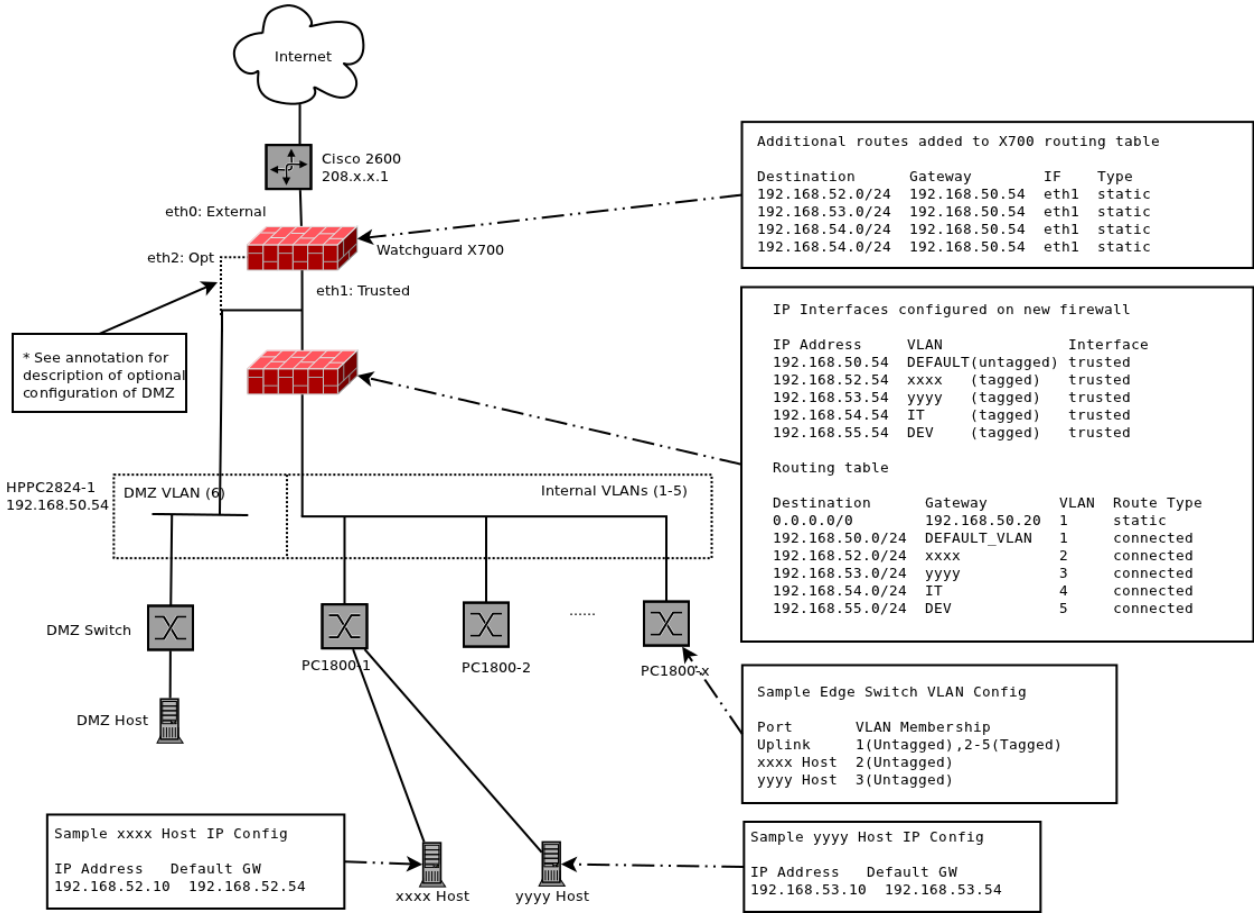
I am going to recount a story from a previous job I had where I was a programmer. We had a network team that was proficient in maintaining the existing network, but as the data center grew, things started to get unmanageable. The team argued about how to extend the network from the current single-subnet design to a multi-subnet network implemented with vlan-capable switches.

After many meetings and delays the network team had finally procured new equipment and thrown the stuff into the racks in the data center. However, no real thought had been given to how the migration would take place. Everyone was afraid to unplug anything and the data center couldn't be down for more than a few hours, so any cut over would have to be prepared well in advance.

The network team was not making any progress on the problem, so the CTO came to me to figure out if there was some solution to the migration problem. I have designed some pretty big networks in the past and I have dealt with some seriously big datacenter iron, so I figured that this network would not be too complex.

Since I had done some low level network programming in the past, I knew how multiple subnets could be subsumed in a larger subnet, and that asymmetrical routes could be created even though it is not a best practice to do so. The solution that I came up with would allow each machine on the network to be moved one at a time

by setting up some asymmetric routes ahead of time. Once the routes were in place, machines could be moved without any of the other clients on the network noticing that they were actually on a new network. This allowed the network team to spread the migration out over several days without any outage at all. Once all machines were moved, the temporary routes were removed and everything was in order with networking best practices. I found a diagram that I created for this project and redacted the IP addresses so that you can see the full solution.



2.10 Summary

- A stack is simply a list of elements with insertions and deletion permitted at one end called the stack top. Thus a stack data structure exhibits the LIFO (last in first out) property.
- Push and pop are the operations that are provided for insertion and the removal of an element from the stack.
- A multiple stack program is a program with more than one stack.
- Stack provides a restricted subset of basic container functionality. It provides insertion removal and inspection of the element at the top of the stack. It does not allow iteration through its elements.
- Stack is a container adapter, meaning that it is implemented on the top of some underlying container type. By default that underlying type is dequeue, but vector or lists may be selected explicitly.

2.11 Keywords

ADT Stack: A stack is an ordered list in which insertions and deletions are made at one end called "top". It is also known as Last-In-First-Out (LIFO) list.

Backtracking: Backtracking is used in algorithms in which there are steps along some path (state) from some starting point to some goal.

Multi stacks: A multiple stack program is a program with more than one stack.

Operation on Stacks: In modern computer languages, the stack is usually implemented with more operations than just "push" and "pop".

Stacks: A stack is simply a list of elements with insertions and deletions permitted at one end-called the stack top.

2.12 Review Questions

1. Define stack.
2. Can a stack be described as a pointer? Explain.
3. Is it possible to insert different type of elements in a stack? How?
4. Write program for the following:

(a) Copying contents of one stack to another.

(b) To check whether a string of opening parenthesis is well formed or not.

5. Transform the following infix expressions into the equivalent postfix expressions.

(A-B)*)(D/E)

(A+B^D)/(E-F) + G

A*(B+D)/E–F*(G+H/K)

(A+B)*(C–D)$ E* F

(A+B)*((C$(D–F) + F)/G) $ (H–J)

6. Transform each of the following postfix expressions to infix:

ABC+–

AB–C+DEF–+$

ABCDE–+$*EF*–

7. Explain the three applications in which stacks are used?
8. "The pop operation cannot be performed when the status of the stack is underflow". Discuss with an example.
9. "In static stack implementation, when the array is full, the status of stack is FULL and the condition is called stack overflow". Discuss.
10. "Static implementation of stack can be achieved using arrays". Explain with a program.

Answers for Self Assessment Questions

1. (a) 2. (c) 3. (c) 4. (a) 5. (a) 6. (a) 7. (a)

2.13 Further Studies

1. http://en.wikibooks.org/wiki/Data_Structures/Stacks_and_Queues
2. http://interactivepython.org/runestone/static/pythonds/BasicDS/stacks.html

3

Queue

STRUCTURE

3.0 Objectives

3.1 Introduction

3.2 Concept of Queues

3.3 Types of Queue

3.4 Linear Queues

3.5 Queues as a Linked List and Abstract Data Structure

3.6 Applications of Queues

3.7 Recursion

3.8 Fibonacci Series Using Recursion

3.9 Tower of Hanoi

3.10 Summary

3.11 Keywords

3.12 Review Questions

3.13 Furhter Studies

3.0 Objectives

After studying this chapter, you will be able to:

- Explain concept of queues
- Discuss types of queue
- Understand linear queues
- Understand queues as a linked list and abstract data structure
- Explain applications of queues

3.1 Introduction

A queue is a linear list of elements that consists of two ends known as front and rear. We can delete elements from the front end and delete elements at the rear end of a queue. A queue in an application is used to maintain a list of items that are ordered not by their values but by their sequential value.

Queue is a non-primitive linear data structure, where the homogeneous data elements are stored in sequence. In queue, data elements are inserted from one end and deleted from the other end. Hence, it is also called as

First-In First-Out (FIFO) list. Figure 3.1 shows a queue with 4 elements, where 55 are the front element and 65 is the rear element. Elements can be added from the rear and deleted from the front.

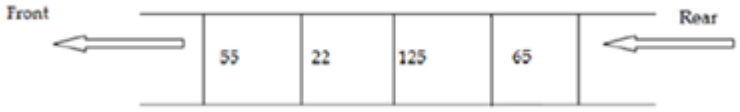


Figure 3.1: A Queue.

A queue is also a list of elements with insertions permitted at one end called the rear, and deletions permitted from the other end called the front. This means that the removal of elements from a queue is possible in the same order in which the insertion of elements is made into the queue. Thus, a queue data structure exhibits the FIFO (first in first out) property. Insert and delete are the operations that are provided for insertion of elements into the queue and the removal of elements from the queue.

3.2 Concept of Queues

A queue is an ordered collection of items in which deletion takes place at one end, which is called the front of the queue and insertion at the other end, which is called the rear of the queue. The queue is a 'first in first out' system (FIFO). In a computer, queue of task executed is similar to the queue that we see in our day to day life. In a time-sharing system, there can be many queues of tasks waiting for access to disk storage or for using the CPU. The queues in a bank or railway station counter are examples of queue. The first person in the queue is the first to be attended. The two main operations in the queue are insertion and deletion of items. The queue has two pointers, the front points to the first element of the queue and the rear points to the last element of the queue. Figure 3.2 shows the structure of a queue. A new item is inserted at the rear end and elements are removed from the queue from the front end.

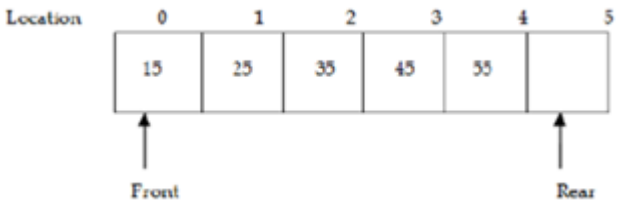


Figure 3.2: Queue Structure.

3.2.1 Basic Operations of Queue

The basic operations of queue are insertion and deletion of items which are referred as enqueue and dequeue respectively. In enqueue operation, an item is added to the rear end of the queue. In dequeue operation; the item is removed from the front end of the queue.

Insert at Rear End

To insert an item into the queue, first it should be verified whether the queue is full or not. If the queue is full, a new item cannot be inserted into the queue. The condition FRONT=NULL indicates that the queue is empty. If the queue is not full, items are inserted at the rear end. When an item is added to the queue, the value of rear is incremented by 1.

Delete from the Front End

To delete an item from the stack, first it should be verified that the queue is not empty. If the queue is not empty, the items are deleted at the front end of the queue. When an item is deleted from the queue, the value of the front is incremented by 1.

Figure 3.3 is a representation of the basic operations of a queue. The elements are inserted from the rear and deleted from the front. The first element inserted into the queue is 10, the second element inserted is 15 and so on. The 30, is the last inserted element. The first element to be deleted from the queue is 10. If a new element is added, it is inserted after 30 and it will be the last element in the queue. A new element cannot be inserted in the queue when the queue is full.

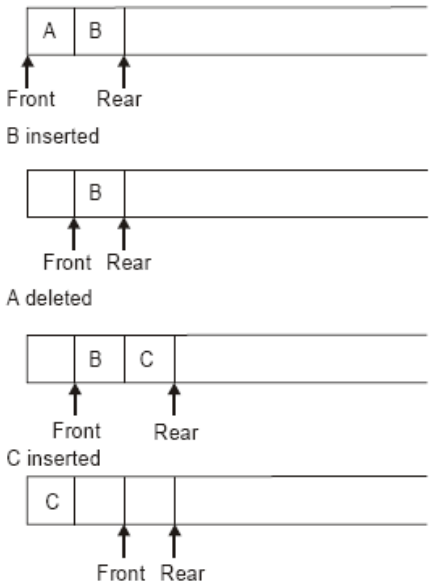


Figure 3.3: Insert and Delete Operations on the Queue.

```
/*Program of queue using array*/
/*insertion and deletion in a queue*/
/*insertion and deletion in a queue*/
 # include <stdio.h>
# define MAX 50
int queue_arr[MAX];
 int rear = –1;
int front = –1;
void ins_delete();
void insert();
 void display();
void main()
{
 int choice;
 while(1)
{
printf(“1.Insert\n”);
 printf(“2.Delete\n”);
printf(“3.Display\n”);
printf(“4.Quit\n”);
printf(“Enter your choice : \n”);
scanf(“%d”,&choice);
switch(choice)
{
case 1 : insert();
```

```
 break;
 case 2 : ins_delete();
break;
 case 3: ins_display();
break;
case 4: exit(1);
 default: printf("Wrong choice\n");
 }                    /*End of switch*/
}/*End of while*/
}/*End of main()*/
 void insert()
{
 int added_item;
 if (rear==MAX-1)
printf("Queue overflow\n");
else { if (front==-1) /*If queue is initially empty */
 front=0;
 printf("Enter an element to add in the queue : ");
scanf("%d", &added_item);
rear=rear+1; queue_arr[rear] = added_item ;
}
} /*End of insert()*/
void ins_delete()
{
 if (front == –1 || front > rear)
{
 printf("Queue underflow\n"); return ;
 }
 else
 {
 printf("Element deleted from queue is : %d\n", queue_arr[front]);
 front=front+1;
}
 } /*End of delete() */ void display() {
int i;
 if (front == -1) printf("Queue is empty\n");
 else
{
printf("Elements in the queue:\n");
for(i=front; i<= rear; i++)
printf("%d ",queue_arr[i]);
printf("\n");
}
} /*End of display() */
```

Output:

1. Insert

2. Delete
3. Display
4. Quit
Enter your choice:
1 Enter an element to add in the queue: 25
Enter your choice: 1
Enter an element to add in the queue: 36
Enter your choice: 3
Elements in the queue: 25, 36
Enter your choice: 2
Element deleted from the queue is: 25
In this example:

1. The preprocessor directives #include are given. MAXSIZE is defined as equal to 50 using the #define statement.
2. The queue is declared as an array using the declaration int queue_arr[MAX].
3. The while loop displays the different options on the screen and accepts the value entered in the variable choice.
4. The switch case compares the value entered and calls the method corresponding to it. If the value entered is invalid, it displays the message "Wrong choice".
5. Insert method: The insert method inserts item in the queue. The if condition checks whether the queue is full or not. If the queue is full, the "Queue overflow" message is displayed. If the queue is not full, the item is inserted in the queue and the rear is incremented by 1.
6. Delete method: The delete method deletes item from the queue. The if condition checks whether the queue is empty or not. If the queue is empty, the "Queue underflow" message is displayed. If the queue is not empty, the item is deleted and front is incremented by 1.
7. Display method: The display method displays the contents of the queue. The if condition checks whether the queue is empty or not. If the queue is not empty, it displays all the items in the queue.

3.2.2 Representing Queue in Memory

The main attribute of a queue data structure is the fact that it permits accessibility only to the front and back of the structure. Additionally, elements can be removed only from the front and added to the back. In this way, the appropriate metaphor used to characterize queues is the idea of a checkout line. Example: People traveling up the elevator, cars moving in line at a petrol station or machine parts on the assembly line are the real-life examples where queues are prevalent. Thereby, queues in data structures are the same as queues that you would see in any shop while waiting to pay at the checkout counter. In each of the cases, the object or customer at the front of the line was the first to enter and the one at the end of the line is the last to have entered.

Each time a customer makes payments for their goods (or the machine part is removed from the line, or a person steps off from the escalator and so on) that object leaves from the front of the queue. This represents the "dequeue" function of the queue. Each time another customer or object enters the line to wait, they join the end of the line. This represents the "enqueue" function of the queue. The "size" function of the queue returns the length of the line and "empty" function returns true only if the line is empty. Figure 3.4 depicts how a queue is represented.

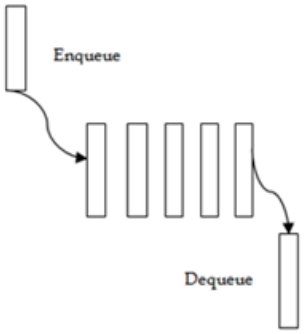


Figure 3.4: Representation of a Queue

3.3 Types of Queue

The different types of queue are

1. Double ended queue
2. Circular queue
3. Priority queue

3.3.1 Double Ended Queue

Double ended queue is also known as deque. It is a type of queue where the insertions and deletions happen at the front or the rear end of the queue. The various operations that can be performed on the double ended queue are: 1. Insert an element at the front end 2. Insert an element at the rear end 3. Delete an element at the front end 4. Delete an element at the rear end Let us now see how an element can be inserted at the front end and deleted at the rear end of a deque. Figure 3.5 depicts inserting and deleting an element from deque. The front end of the queue is identified by F and the rear end is identified by R. If we want to insert an element 20 at the front end, we can do this by checking if F is equal to zero and then increment F and insert the element. We cannot insert an element at the front if an element is already present at the first position, as queue follows 'First In First Out' method. In the figure, the element 30 is already present in the first position of the queue. Hence, we cannot insert an element at the front end. If we want to delete element 60 at the rear end, access the element at the rear end and then decrement the pointer R. When the elements are deleted and queue becomes empty, reset the pointer F to 0 and rear end pointer R to –1. An element can be deleted only if the queue is not empty.

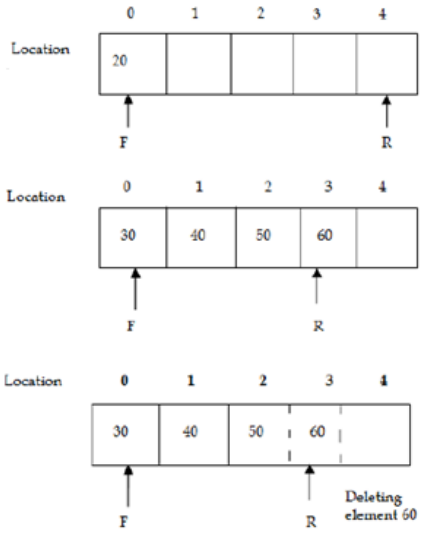


Figure 3.5: Inserting and Deleting an Element From Deque

Example: Program for the Insertion and Deletion of an Element in a Dequeue.

```
 #include<stdio.h>
 #include<conio.h>
 #define SIZE 5
 int Q_F(int R)
 { return (R==SIZE-1)?1:0; }
 int Q_E(int F, int R)
{ return(F>R)?1:0; }
 void front_insert(int num, int Q[], int *F, int *R)
{ if(*F==0 || *R== –1)
 {
Q[++(*r)]=item;
return;
 }
 if(*F!=0)
{
Q[– –(*F)]=item;
return;
 }
printf(“Front inertion not possible\n”);
 }
 void rear_delete(int Q[], int *F, int *R)
{ if(Q_E(*F, *R))
 {
 printf(“Queue underflow\n”);
 return;
 }
 printf(“The element deleted is %d\n”, Q[(*R)– –]);
if(*F>*R)
 { *F=0, *R= –1; }
 }
void display(int Q[], int F, int R)
{
int i;
 if(Q_E(F, R))
 {
 printf(“Queue is empty\n”);
 return;
 }
 printf(“Contents of the queue is:\n”);
 for(i=F;i<=R; i++) { printf(“%d\n”, Q[i]);
 }
}
void main()
{
int choice, num, F, R, Q[10];
```

```
F=0; R= –1;
for(;;)
{
printf(“1. Insert at front/n”);
printf(“2. Delete at rear end/n”);
printf(“3. Display/n”);
 printf(“4. Exit/n”);
scanf(“%d”, &choice);
 switch(choice)
{
 case 1: printf(“Enter the number to be inserted\n”);
scanf(“%d”, &num);
 front_insert(num, Q, &F, &R);
break;
case 2: rear_delete(Q, &F, &R);
break;
case 3: display(Q, F, R);
break;
default: exit(0);
}
}
}
```

Output:

```
1. Insert at front end
2. Delete at rear end
3. Display
4. Exit
1
Enter the number to be inserted
30
1. Insert at front end
2. Delete at rear end
3. Display
4. Exit
 2
Enter the number to be inserted
40
1. Insert at front end
2. Delete at rear end
3. Display
4. Exit
 3
The contents of the queue is
30 40
1. Insert at front end
2. Delete at rear end
```

3. Display
4. Exit
2

The element deleted is
40

In this example:
1. The header files are defined and a constant value 5 is defined for variable SIZE using #define header. The SIZE defines the size of the queue.
2. Four functions are created namely, Q_F( ), Q_E( ), front_insert( ), rear_delete( ), and display( ). The user has to select an appropriate function to be performed.
3. The switch statement is used to call the front_insert( ), rear_delete( ), and display( ) functions.
4. When the user enters 1, the front_insert( ) function is called. In the front_insert( ) function the if loop checks if the F pointer is equal to 0 or R pointer is equal to -1. If the result is true, then the R pointer is incremented and the value entered by the user (num) is assigned to the Q. The value of R is returned. The second if loop checks if the F pointer is not equal to 0. If the result is true, then the F pointer is decremented and the value entered by the user (num) is assigned to Q. The value of F is returned. Else, The program prints the message "front insertion not possible"
5. When the user enters 2, rear_delete( ) function is called. In the rear_delete( ) function the if loop calls the Q_E( ) function with the current pointer values of F and R. If the condition is true, the program prints the message "Queue underflow". It returns the value of F and R. The program prints the deleted element.
6. When the user enters 3, the function display( ) is called. In the function display( ) the if loop checks for the queue size. If the queue is not empty the program displays the elements.
7. When the user enters 4, the program exits.

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: Write an algorithm to perform insertion of an element at the rear end and deletion of an element at the front end for a deque.

……..……

……..……

……...

3.3.2 Circular Queue

In a circular queue, the rear end is connected to the front end forming a circular loop. An advantage of circular queue is that, the insertion and deletion operations are independent of one another. This prevents an interrupt handler from performing an insertion operation at the same time when the main function is performing a deletion operation. The Figure 3.6 depicts a circular queue. The queue elements are stored in an array. The front end of the queue is represented as F and the rear end is represented as R. Before inserting an element into the queue, the R pointer should be set to -1. The value of R is then incremented to insert the elements. In the first figure of Figure 3.6, only one element (20) is present in the queue. Hence, the value of F and R pointer will be 0. In the second figure of Figure 3.6, two elements are added (40 and 60) to the queue. This can be done by incrementing the R pointer. The following statement depicts the increment operation:
R=(R+1) % SIZE

Here,
SIZE is the queue size. In this case the size is 5.
In the third figure of figure 3.6, elements 80 and 100 are added to the queue. Now the R value will be 5. Since, the value of SIZE is also 5. The R will point to 0.

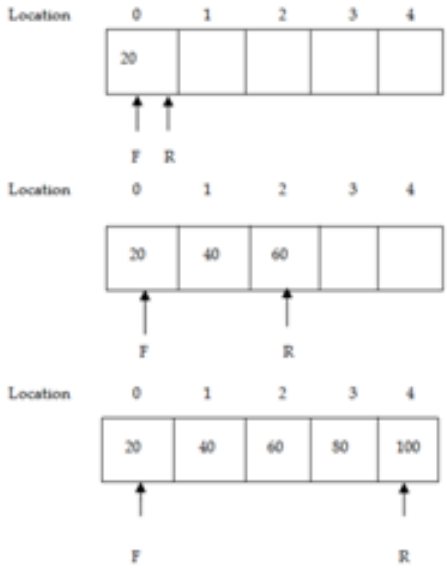


Figure 3.6: Circular Queue.

Example: Program for Implementation of Circular Queue.

```
#include<stdio.h>
#include<conio.h>
#define SIZE 5
 int Q_F(int COUNT)
{
 return (COUNT==SIZE)? 1:0;
 }
 int Q_E(int COUNT)
{ return (COUNT==0)? 1:0;
}
 void rear_insert(int item, int Q[], int *R, int *COUNT)
{ if(Q_F(*COUNT))
 { printf("Queue overflow");
return;
}
 *R= (*R+1) % SIZE;
Q[*R]=num; *COUNT+=1;
 }
 void front_delete(int Q[], int *F, int *COUNT)
{
 if(Q_E(*COUNT))
{ printf("Queue underflow");
return;
 }
 printf("The deleted element is %d\n", Q[*F]);
```

```
*F= (*F+1) % SIZE;
 *COUNT– =1;
}
 void display(int Q[], int F, int COUNT)
 {
 int i,j;
 if(Q_E(COUNT))
{
 printf(“Queue is empty\n”);
 return; }
printf(“Contents of the queue is:\n”);
 i=F; for(j=1;j<=COUNT; j++)
{
printf(“%d\n”, Q[i]);
i=(i+1) % SIZE;
}
 printf(“\n”); }
 void main()
 {
int choice, num, COUNT, F, R, Q[20];
clrscr();
 F=0; R= –1;
COUNT=0;
 for(;;)
 {
 printf(“1. iInsert at front\n”);
printf(“2. Delete at rear end\n”);
printf(“3. Display\n”);
 printf(“4. Exit\n”);
 scanf(“%d”, &choice);
switch(choice)
 {
case 1: printf(“Enter the number to be inserted\n”);
 scanf(“%d”, &num);
rear_insert(num, Q, &R, &COUNT);
break;
case 2: front_delete(Q, &F, &COUNT);
break;
case 3: display(Q, F, COUNT);
break; default: exit(0);
}
 }
}
```

Output:

1. Insert at rear end
2. Delete at front end
3. Display

4. Exit
1 Enter the number to be inserted 50
1. Insert at rear end
2. Delete at front end
3. Display
4. Exit

1Enter the number to be inserted 60
1. Insert at rear end
2. Delete at front end
3. Display
4. Exit

3 The contents of the queue is 50 60
1. Insert at rear end
2. Delete at front end
3. Display
4. Exit

2 The element deleted is 50

In this example:
1. The header files are included and a constant value 5 is defined for variable SIZE using #define statement. The SIZE defines the size of the queue.
2. A queue is created using an array named Q with an element capacity of 20. A variable named COUNT is declared to store the count of number elements present in the queue.
3. Four functions are created namely, Q_F(), Q_E(), rear_insert(), front_delete(),and display(). The user has to select an appropriate function to be performed.
4. The Switch statement is used to call the rear_insert(), front_delete(), and display() functions.
5. When the user enters 1, rear_insert() function is called. In the rear_insert() function, the if loop checks if the count is full. If the condition is true, then the program prints a message "Queue is empty". Else, it checks for the value of R and assigns the element (num) entered by the user to R. Initially, when there are no elements in the queue, R value will be 0. After every insertion the variable COUNT is incremented.
6. When the user enters 2, the front_delete() function is called. In this function, the if loop checks if the variable COUNT is empty. If the condition is true, then the program prints a message "Queue underflow". Else, the element in the 0th position will be deleted. The size of F is computed and the COUNT is set to 1.
7. When the user enters 3, the display() function is called. In this function, the if loop checks if the value of COUNT is 0. If the condition is true, the program prints a message "Queue is empty". Else, the value of F is assigned to the variable i. The for loop then displays the elements present in the queue.
8. When the user enters 4, the program exits.

3.3.3 Priority Queue

In priority queue, the elements are inserted and deleted based on their priority. Each element is assigned a priority and the element with the highest priority is given importance and processed first. If all the elements present in the queue have the same priority, then the first element is given importance.

Example: Program for Implementation of Priority Queue:

```
#include<stdio.h>
#include<malloc.h>
struct queue {
int PRI; int value; struct queue *next;
}
*F, *q, *tmp, *new;
typedef struct queue *P;
void ins()
{
int num, el_pri; new = ( P ) malloc(10 );
printf( “Enter the element to be inserted:” );
scanf( “%d”, &num );
printf( “Enter a Priority:” );
scanf( “%d”, &el_pri );
new->value = num;
new->PRI = el_pri;
if ( F == NULL || el_pri < F->PRI )
{
new->next = F;
F = new;
}
else
{
q = F;
while ( q->next != NULL && q->next->PRI <= el_pri )
q = q->next;
new->next = q->next;
q->next = new;
}
}
void del() { if ( F == NULL )
{
printf( “\n QUEUE UNDERFLOW\n” );
}
else
{ new = F; printf( “\nDeleted number is %d\n”, new->value );
F = F->next; free( F );
}
}
void disp()
{
tmp = F;
if ( F == NULL )
printf( “QUEUE IS EMPTY\n” );
else
{
```

```
printf( “QUEUE IS:\n” );
 while ( tmp != NULL )
{
printf( “\n%d[PRI=%d]”, tmp->value, tmp->PRI );
 tmp = tmp->next;
}
}
}
 int main()
{
int choice;
 clrscr();
while(1)
{
printf( “\n 1. INSERT \n 2. DELETE \n 3. DISPLAY \n 4. EXIT”);
printf( “\n Enter your choice” );
 scanf( “%d”, &choice );
switch ( choice )
 {
case 1: ins();
break;
case 2: del();
break;
case 3: disp();
break;
default: exit(1);
}
}
}
```

Output:
1. INSERT
2. DELETE
3. DISPLAY
4. EXIT

Enter your choice:
1
Enter the element to be inserted
10
Enter a priority
1

1. INSERT
2. DELETE
3. DISPLAY
4. EXIT

Enter your choice:
1
Enter the element to be inserted
20
Enter a priority
2
1. INSERT
2. DELETE
3. DISPLAY
4. EXIT

Enter your choice:
3
QUEUE IS:
10[PRI=1]
20[PRI=2]
1. INSERT
2. DELETE
3. DISPLAY
4. EXIT

Enter your choice:
2
Deleted number is
10
1. INSERT
2. DELETE
3. DISPLAY
4. EXIT

Enter your choice:
2 Deleted number is
20
1. INSERT
2. DELETE
3. DISPLAY
4. EXIT

Enter your choice:
2 QUEUE UNDERFLOW
In this example:
1. The header files namely, stdio and malloc are included.
2. A structure named queue is created which consists of three variables namely, value, PRI, and next. The variable value holds the element value. The PRI holds the element priority value and next is a pointer variable that points to the next element in the queue. Four objects are declared namely, F, q, tmp, and new to access the structure elements.

3. Three functions are created namely, ins(), del(), and disp(). The user has to select an appropriate function to perform.
4. The switch statement is used to call the ins(), del(), and disp() functions.
5. When the user enters 1, the ins() function is called. This function allocates memory of capacity 10 for the queue using malloc function. Then, the element to be inserted into the queue and its priority is inputted from the user. The entered element value and its priority are stored in the variables value and PRI. The if loop checks if the F value is equal to NULL or the priority of the entered element is less than the priority of the first element. If either of the condition is true, then the element entered is stored in the second position of the queue. Else, the value of F is assigned to q. The if loop checks if the third position of the queue is not equal to NULL and the priority of the second element is less than the third. If the condition is true, then the second element is stored in the third position and the third element is stored in the second position. The loop continues to check for the priority of all the elements in the queue and stores them accordingly.
6. When the user enters 2, the del() function is called. In this function, the if loop checks if the value of F is equal to NULL. If the condition is true, then program prints the message "QUEUE UNDERFLOW". Else, F is assigned to new, and the element in F is deleted. The pointer F is set free.
7. When the user enters 3, the function disp() is called. In this function, the if loop checks if the value of F is equal to NULL. If the condition is true, then the program prints the message "QUEUE is EMPTY". Else, it displays the elements present in the queue along with their priority.
8. When the user enters 4, the program exits.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: 1. Create a circular queue having an element storage capacity of 5. Insert 4 elements into the queue. Delete first two elements and insert an element at the position F=1 and R=3.
2. Create a priority queue having an element capacity of 3. Insert elements namely 100 having priority 2, 200 having priority 1and 300 having priority 3. Try deleting element with priority 2. Analyze the result.

……..……

……..……

……...

Caution

While using queue, Shift. IIRC it is not O(1), but O(n) and might be too slow if the queue gets large.

3.4 Linear Queues

A linear queue models the FIFO (first in first out) data structure, much like a line in real life. The first person in line will be the first person served; in queues the first element to be added is the first that can be removed. The only adding point is to the end of the list and the only removal point is the beginning of the list.
Before we can use a queue, it must be created. The purpose of initializing the queue is served by assigning -1 (as a sentinel value) to the front and rear variables. Note that the valid range of an index for the array is 0 to CAPACITY–1.
.cf {font-family: Lucida Console; font-size: 9pt; color: black; background:white;} .cl {margin: 0px;} .cb1 {color: green;} .cb2
{color: blue;} .cb3 { color: maroon;}

```
void createqueue(queue *q)
{
q->front=q->rear=-1;
}
```

3.4.1 Testing the Queue for Underflow

```
bool isempty(queue q)
{
if(q.front== –1)
return true;
else
return false;
```

3.4.2 Testing the Queue for Overflow

```
bool isfull(queue q)
{
if ((q.front==0) && (q.rear==CAPACITY–1))
return true;
else
return false;
}
```

bool can be defined as typedef enum {false, true} bool;

3.4.3 Performing the Enqueue Operation on a Linear Queue

There are two scenarios that we should consider, assuming that the queue is not full.

1. If the linear queue is empty, then the value of the front and the rear variable will be –1 (the sentinel value), then both front and rear are set to 0.
2. If the linear queue is not empty, then there are two further possibilities:

(a) If the value of the rear variable is less than CAPACITY–1, then the rear variable is incremented.

(b) If the value of the rear variable is equal to CAPACITY–1, then the elements of the linear queue are moved forward, and the front and rear variables are adjusted accordingly.

Example

```
void enqueue(queue *q, int value)
{
int i;
if(isempty(*q))
q->front=q->rear=0;
else if (q->rear==CAPACITY–1)
{
for(i=q->front;i<=q->rear;i++)
q->elements[i–q->front]=q->elements[i];
q->rear=q->rear-q->front+1;
q->front=0;
}
else
{
q->rear++;
```

```
}
&nbs;
q->elements[q->rear]=value;
}
```

Performing the Dequeue Operation on a Linear Queue

There are two possibilities:

1. If there is only one element in the linear queue then after dequeueing it will become empty. This state of the linear queue is reflected by setting the front and rear variables to –1, the sentinel value.
2. Otherwise, the value of the front variable is incremented.

Example

```
int dequeue(queue *q)
{
int temp;
temp=q->elements[q->front];
if(q->front==q->rear)
q->front=q->rear= –1;
else
q->front++;
return temp;
}
```

Exercise: Check Your Progress 3

Note: i) Use the space below for your answer.

Ex1: Write the basic operation of Queue?

……..……

……..……

……..

3.5 Queues as a Linked List and Abstract Data Structure

3.5.1 The Queue as an Abstract Data Structure

The representation of a queue as an abstract data type is straightforward. We use eltype to denote the type of the queue element and parameterize the queue type with eltype.

```
abstract typedef <<eltype>> QUEUE(eltype);
abstract empty(q)
QUEUE(eltype) q;
postcondition empty = (len(q) = 0);
abstract eltype remove(q)
QUEUE(eltype) q;
precondition empty(q) == FALSE;
postcondition remove == first(q') ;
q = sub(q', 1, len(q') – 1);
abstract insert(q, elt)
QUEUE(eltype) q;
```

eltype elt;
postcondition q == q' + <elt>;

3.5.2 Queue as a Linked List

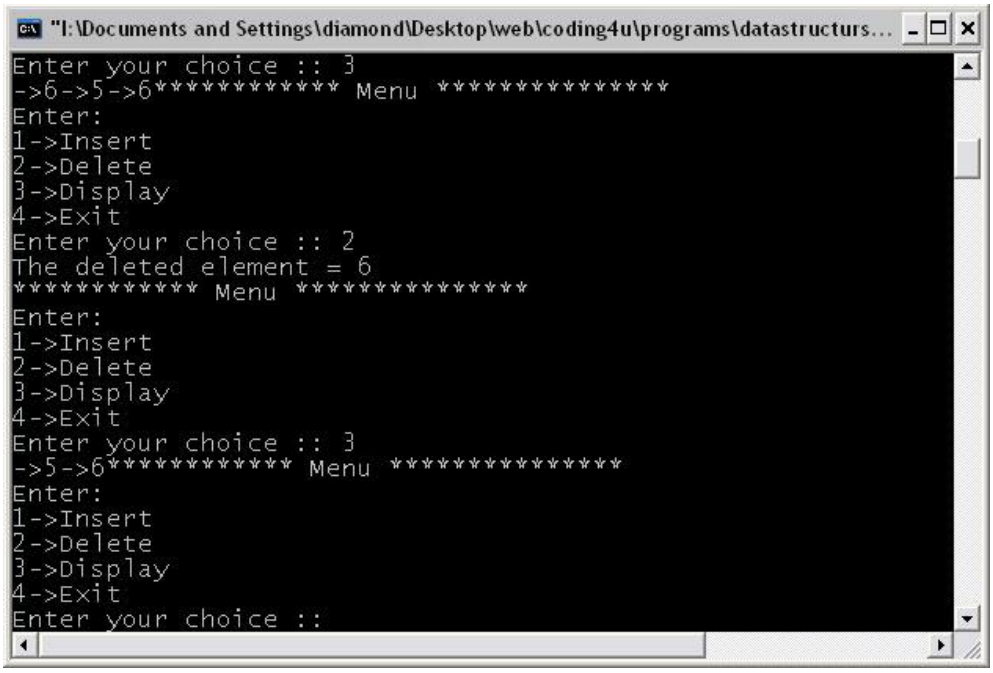
Queue can be implemented using linked list too. The main benefit in linked representation is that you need not worry about the size of the queue. In linked organization, we can add as many nodes as possible and the queue will never have a full condition. The queue that uses linked list would be similar to that of a linked list. The only difference between the two of them is that, in queues, the leftmost node and the rightmost node is called as front and rear node respectively. And we cannot remove any of the arbitrary nodes from the queue. Always the front node needs to be removed.

```
#include<stdio.h>
#include<malloc.h>
struct node
{
int info;
struct node *next;
} *front, *rear;
void enqueue(int elt);
int dequeue();
void display();
void main()
{
int ch, elt;
rear = NULL;
front = NULL;
while (1)
{
printf("********** Menu **********");
printf("\nEnter:\n1->Insert\n2->Delete\n3->Display\n4->Exit\n");
printf("Enter your choice :: ");
scanf("%d", &ch);
switch (ch)
{
case 1:
printf("Enter The Element Value\n");
scanf("%d", &elt);
enqueue(elt);
break;
case 2:
elt = dequeue();
printf("The deleted element = %d\n", elt);
break;
case 3:
display();
break;
default:
printf("~~~Exit~~~");
```

```
getch();
exit(0);
break;
}
}
}
void enqueue(int elt)
{
struct node *p;
p = (struct node*)malloc(sizeof(struct node));
p->info = elt;
p->next = NULL;
if (rear == NULL || front == NULL)
front = p;
else
rear->next = p;
rear = p;
}
int dequeue()
{
struct node *p;
int elt;
if (front == NULL || rear == NULL)
{
printf("\nUnder Flow");
getch();
exit(0);
}
else
{
p = front;
elt = p->info;
front = front->next;
free(p);
}
return (elt);
}
void display()
{
struct node *t;
t = front;
while (front == NULL || rear == NULL)
{
printf("\nQueue is empty");
getch();
exit(0);
}
```

```
while (t != NULL)
{
printf("->%d", t->info);
t = t->next;
}
}
```

Output:



Self Assessment Questions

1. Which data structure allows deleting data elements from front and inserting at rear?

(a) Stacks (b) Queues (c) Deques (d) Binary search tree

2. Identify the data structure which allows deletions at both ends of the list but insertion at only one end.

(a) Input-restricted deque (b) Output-restricted deque

(c) Priority queues (d) None of these.

3. Which of the following data structure is linear type?

(a) Strings (b) Lists (c) Queues (d) All of These.

4. What should be the value of R pointer, before inserting elements into the queue?

(a) -1 (b) 0 (c) 1 (d) Is not set any value

3.6 Applications of Queues

One application of the queue data structure is in the implementation of priority queues required to be maintained by the scheduler of an operating system. It is a queue in which each element has a priority value and the elements are required to be inserted in the queue in decreasing order of priority. This requires a change in the function that is used for insertion of an element into the queue. No change is required in the delete function. Queues that are created by an application or by the Message Queuing administrator are referred to as application queues. For information on different types of application queues, see the following links:

- Destination queues are any application queue that sending applications send messages to, or that receiving applications receive messages from. For more information, see:
  - Public and private queues
  - Transactional and nontransactional queues
- Administration queues are application-generated queues used to store system-generated negative and positive acknowledgment messages that are created by Message Queuing or connector applications. Any available nontransactional queue can be used as an administration queue.

3.6.1 Public and Private Queues

Private Queues

Private queues are queues that are not published in Active Directory and are displayed only on the local computer that contains them. Private queues have the following features:

- Message Queuing registers private queues locally by storing a description of the queue in the LQS (local queue storage) directory on the local computer. In MSMQ2.0, and
  Message Queuing3.0, the default location is %windir%\system32\msmq\storage\lqs.
  Note that a description of each public queue created on the local computer is also stored locally in a separate file in the LQS folder.
- Private queues are registered on the local computer, not in the directory service, and typically cannot be located by other Message Queuing applications.
- Private queues are accessible only by Message Queuing applications that know the full path name, the direct format name, or the private format name of the queue, as follows:
  - Path name:ComputerName\private$\QueueName.
  - Path name on local computer: \private$\QueueName.
  - Direct format name:DIRECT=ComputerAddress\PRIVATE$\PrivateQueueName.
  - Private format name: PRIVATE=ComputerGUID\QueueNumber.

  For more information on path names and format names, see Queue names.
- Private queues have the advantage of no directory service overhead, making them quicker to create, no latency in accessing them, and no replication overhead.
- Private queues are not dependent on the directory service, and thus can be created and deleted when the directory service is not working. This is useful for offline operation.
- One way that private queues can be exposed to other applications is by setting a message property. To distribute the location of a private queue, an application can send a format name of the private queue as the response queue property of a message.
- In a workgroup environment, only private queues are available.
- For a Message Queuing application on the local computer to access a private queue, the application only needs to know the path name of the queue. For a Message Queuing application on a remote computer, the application requires a direct or private format name to access such a queue. To read messages from the queue, the remote Message
  Queuing application requires the Receive Message permission for that queue. By default, everyone has permissions to send messages to public and private queues.

Public Queues

In a domain environment, public queues are queues that are published in Active Directory and hence are replicated throughout your Windows Server 2003 family forest. Note that only the properties for these queues are replicated not the actual queues themselves or their contents. Any computer within your forest can potentially access information regarding public queues provided the user has sufficient permissions to access

the applicable queue objects. Generally speaking, any user in the forest with access to Active Directory and the Send To permission for a given public queue can send messages to it. In an Active Directory environment, defining public queues ensures that queues are registered in the directory service, and that their registration is backed up. They are persistent and available to other applications.

Creating Local Public Queues

By default, in Active Directory, the Create Child Object permission that is required to create local public queues is granted to:

- The Domain Administrator account
- The Domain User account of the local administrator that installed Message Queuing
- Local System and Network Service accounts

Did You Know?

Microsoft Message Queuing or MSMQ is a Message Queue implementation developed by Microsoft and deployed in its Windows Server operating systems since Windows NT 4 and Windows 95.

3.6.2 Transactional and Nontransactional Queues

A transactional queue is one that only contains transactional messages, which are messages sent within a transaction. The requirements for sending and retrieving messages are based on the type of queue used (transactional or nontransactional) and the queue's location (local or remote).

Sending Messages

When using transactions to send messages, the messages can be sent to any transactional queue regardless of where the queue is located, but only transactional queues can accept messages sent in the context of a transaction. In a similar way, nontransactional queues can only accept messages sent outside the context of a transaction. Note that only transactional messages are delivered with Exactly-Once In-Order Delivery (EOD) guarantees. For more information on transactional messaging, see Transactional messaging.

Receiving Messages

You cannot retrieve a message from a remote transactional queue or from a local nontransactional queue within a transaction. Message Queuing does not provide support for transactional remote-read operations (accessing the contents of a queue from a computer other than the one on which the queue is located). However, this same functionality can be achieved by using a transactional read-response application.

Note that nontransactional receive operations are allowed from any queue (transactional or nontransactional, local or remote). In particular, a nontransactional receive operation from a transactional queue is essentially a transaction consisting of a single receive operation.

Caution

While using a transaction to retrieve messages, the queue must be a local transactional queue.

3.6.3 Transactional Read-Response Applications

A transactional read-response application is used when you need to retrieve transactional messages from a remote transaction queue. It uses transactions to redirect the messages from the remote queues on the sending computer to local queues on the receiving computer.

A read-response application resides on a server that contains two transactional queues:

- An input queue for collecting messages from sending applications.
- A read-request queue for collecting read-requests from the receiving application.

The read-response application does the following in a single transaction:

- Retrieves a request made by the receiving application from the read-request queue.
- Retrieves a message sent by the sending application from the input queue.
- Forwards the message from the input queue to the receiving application that made the request.

Did You Know?

The term 'queue' was first used by A.K. (Agner Krarup) Erlang published paper on queuing theory in 1909.

3.7 Recursion

• Recursion is a problem-solving approach in which a problem is solved using repeatedly applying the same solution to smaller instances.
• Each instance to the problem has size.
• An instance of size n can be solved by putting together solutions of instances of size at most n-1.
• An instance of size 1 or 0 can be solved very easily.

3.7.1 Recursive Functions

A recursive function is one which calls itself. This is another complicated idea which you are unlikely to meet frequently. We shall provide some examples to illustrate recursive functions.

Recursive functions are useful in evaluating certain types of mathematical function. You may also encounter certain dynamic data structures such as linked lists or binary trees. Recursion is a very useful way of creating and accessing these structures.

Here is a recursive version of the Fibonacci function.

```
int fib(int num)
/* Fibonacci value of a number */
{       switch(num) {
        case 0:
                return(0);
                break;
        case 1:
                return(1);
                break;
        default:  /* Including recursive calls */
                return(fib(num - 1) + fib(num - 2));
                break;
        }
}
```

We met another function earlier called power. Here is an alternative recursive version.

```
double power(double val, unsigned pow)
{
        if(pow == 0)  /* pow(x, 0) returns 1 */
                return(1.0);
        else
                return(power(val, pow - 1) * val);
}
```

3.7.2 Working of recursion,

The following program includes a call to the recursively defined function "print_backwards()", which inputs a series of characters from the keyboard, terminated with a full-stop character, and then prints them backwards on the screen.

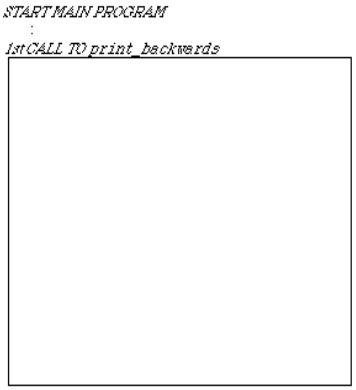
Example: 1

```
        #include<iostream>
        using namespace std;
        void print_backwards();
        int main()
        {
print_backwards();
        cout << "\n";
        return 0;
        }
        void print_backwards()
        {
        char character;
        cout << "Enter a character ('.' to end program): ";
        cin >> character;
        if (character != '.')
        {
        print_backwards();
        cout << character;
        }
}
```

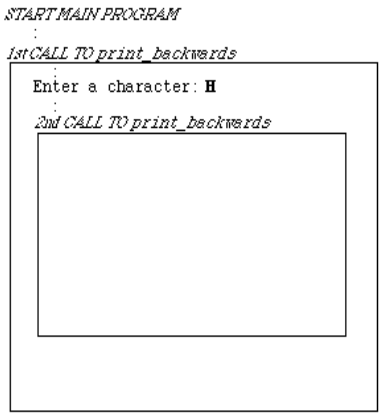
A typical input/output session is:

```
        Enter a character ('.' to end program): H
        Enter a character ('.' to end program): i
        Enter a character ('.' to end program): .
        iH
```

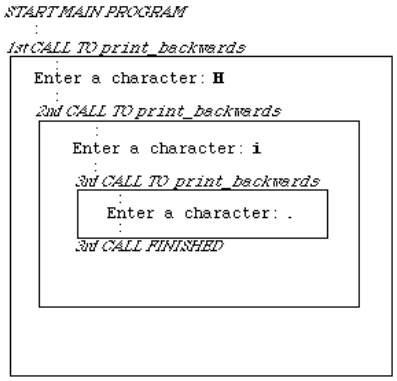
It is easy to see why **Example: 1** works with the aid of a few diagrams. When the main program executes, it begins with a call to "print_backwards()". At this point space is set aside in the computer's memory to execute this call (and in other cases in which to make copies of the value parameters). This space is represented as a box in Figure 3.7:



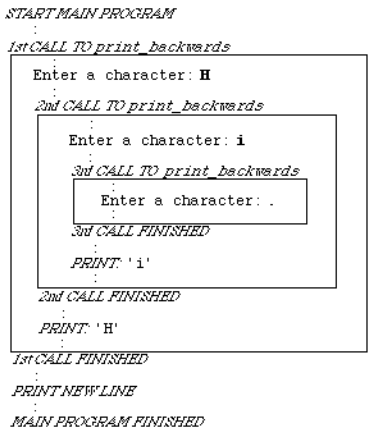
The internal execution of this call begins with a character input, and then a second call to "print_backwards()" (at this point, nothing has been output to the screen). Again, space is set aside for this second call:



The process repeats, but inside the third call to "print_backwards()" a full-stop character is input, thus allowing the third call to terminate with no further function calls:



This allows the second call to "print_backwards()" to terminate by outputting an "i" character, which in turn allows the first call to terminate by outputting an "H" character:



Exercise: Check Your Progress 4

Note: i) Use the space below for your answer.

Ex1: Write the application of Queue?

……..……

……..……

……...

3.8 Fibonacci Series Using Recursion

As a simple rule of recursion, any function can be computed using a recursive routine if :
1. The function can be expressed in its own form.
2. There exists a termination step, the point at which f(x) is known for a particular 'x'.
Therefore to write a recursive program to find the nth term of the fibonacci series, we have to express the fibonacci sequence in a recursive form using the above 2 rules :
1. fib(n) = fib(n-1) + fib(n-2) (recursive defination of fibonacci series).
2. if n=0 or n=1, return n (termination step).
Using these 2 rules, the recursive program for finding the nth term of the fibonacci series can be coded very easily as shown.

```
#include "stdio.h"
#include "conio.h"
int y;
fib(int n)
        { if(n==1 || n==0)
          return n;
         y=fib(n-1)+fib(n-2);
          return y;
        }
main()
        { int a,r;
         clrscr();
         printf("Enter any number : ");
         scanf("%d",&a);
         r=fib(a);
         printf("The no. at position %d is %d",a,r);
         getch();
         return 0;
        }
```

Self Assesment Questions

5. Which among the following is a principle of queue?
(a) Last in First Out (b) First in First Out (c) Last in Last Out (d) First in Last Out

6. Queue is a
(a) Linear data structure (b) Non-linear data structure
(c) Both (l) and (2) (d) None of these.

7. The end at which a new element gets added to a queue is called
(a) front (b) rear (c) top (d) bottom

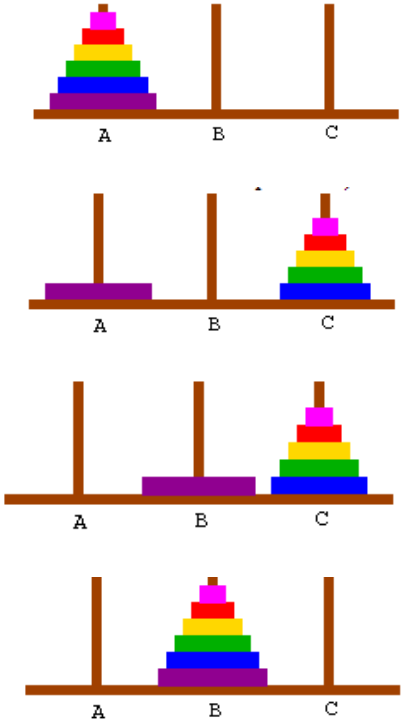
8. The end from which an element gets removed from the queue is called
(a) front (b) rear (c) top (d) bottom

3.9 Tower of Hanoi

There are 3 rods where discs can be stacked. All the discs are having different diameters. The goal of the game is to move all discs from one tower to another using the third tower. The rules are simple. Only smaller disc can go on top of the bigger discs. So the discs will be sorted according to their sizes and bigger one will be at the bottom. Only one disc can be moved at a time. you must put the removed disc in some tower, it cannot be left in any other places. At the beginning there are some discs in tower one. The goal is to move them to tower 2 following the above rules.

Solution

Suppose we are moving discs from tower 1 to tower 2 with the help of tower 3. We can see that the bigger disc of the bottom can only be moved when all top discs are moved out of tower 1. As this is the bigger disc it can only go to an empty rod. So this creates a subproblem. To move all the disc from 1 to 2, we need to move n-1 discs from 1 to 3 using 2. Now after moving the top n-1, we can move the bottom one to tower2 as it is empty at that point. After that we need to move all discs from tower 3 to tower 2 with the help of tower 1, as this is currently empty.



Lab Exercise

1. Write a C program to implement a double-ended queue, which is a queue in which insertions and deletions may be performed at either end. Use a linked representation.
2. Write program of queue and insert data in the front end.

3.10 Summary

- A queue is an ordered collection of items in which deletion takes place at the front and insertion at the rear of the queue.
- The basic operations performed on a queue include inserting an element at the rear end and deleting an element at the front end.
- Representing a queue in the memory includes representing the way in which the elements are stored in the memory and naming the address to which the front and rear pointers point to.

- The different types of queues are double ended queue, circular queue, and priority queue.
- A queue is a list of elements with insertion permitted at one end called the rear and deletion permitted from the other called the front.
- The purpose of initializing the queue is served by assigning–1 (as a sentinel value) to the front and rear variables.
- Application of the queue data structure is in the implementation of priority queues required to be maintained by the scheduler of an operating system.

3.11 Keywords

Circular queue: It is easier to represent a queue as a circular list than as a linear list. As a linear list, a queue is specified by two pointers, one to the front of the list and the other to its rear.

***Dequeue*:** Process of deleting elements from the queue.

***Enqueue*:** Process of inserting elements into queue.

***Front end*:** It refers to the first node in the queue.

Header nodes: An alternative method is to place a header. node as the first node of a circular list. This list header may be recognized by a special value in its il_fo field that cannot be the valid contents of a list node in the context of the problem, or it may contain a flag marking it as a header.

Rear end: It refers to the last node in the queue.

3.12 Review Questions

1. "In circular queue the insertion and deletion operations are independent of each other." Discuss.
2. "Using double ended queues is more advantageous than using circular queues. "Discuss
3. "Stacks are different from queues." Justify.
4. "Using priority queues is advantageous in job scheduling algorithms." Discuss.
5. Is it possible to insert an element in the front end in a queue? If yes, which method is followed? Discuss.
6. Can a basic queue be implemented to function as a dynamic queue? Discuss.
7. "It is more advantageous to implement queues using linked lists. "Discuss.
8. Write a C program to implement a double-ended queue, which is a queue in which insertions and deletions may be performed at either end. Use a linked representation.
9. Add a method Item[] toArray() to Queue that returns all N elements on the queue as an array of length N.
10. Write a recursive function that takes as input a queue, and rearranges it so that it is in reverse order.
11. What is tower of Hanoi?

Answers for Self Assessment Questions

1. (b)	2. (a)	3. (d)	4. (a)
5. (b)	6. (a)	7. (a)	8. (b)

3.13 Furhter Studies

1. http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Queue_(data_structure).html

4

Trees

STRUCTURE

4.0 Objectives

4.1 Introduction

4.2 Representation of Tree

4.3 Binary Tree

4.4 Representation of Binary Tree

4.5 Basic Operation on Binary Tree

4.6 Creation of Binary Search Tree

4.7 Types of Binary Trees

4.8 Operations on Binary Search Tree (BST)

4.9 Threaded Binary Trees

4.10 Application of Binary Tree

4.11 B-Tree and Height Balanced Tree

4.12 Summary

4.13 Keywords

4.14 Review Questions

4.15 Furhter Studies

4.0 Objectives

After studying this chapter, you will be able to:

- Explain the representation of tree
- Define the binary tree
- Understand the representation of binary tree
- Discuss the basic operation on binary tree
- Explain the creation of binary search tree

4.1 Introduction

We all know that data structure is a set of data elements grouped together under one name. A data structure can be considered as a set of rules that hold the data together. Almost all computer programs use data structures. Data structures are an essential part of algorithms. We can use it to manage huge amount of data in large databases. Some modern programming languages emphasize more on data structures than algorithms.
Choosing the best data structure for a program is a challenging task. Similar tasks may require different data structures. We derive new data structures for complex tasks using the already existing ones. We need to compare the characteristics before choosing the right data structure. A tree is a hierarchical data structure suitable for representing hierarchical information. The tree data structure has the characteristics of quick search, quick inserts, and quick deletes.

4.2 Representation of Tree

A tree is a very important data structure as it is useful in many applications. A tree structure is a way of presenting the hierarchical nature of a structure in a graphical form. It is referred as "tree structure" since its representation resembles a tree. But the chart of a tree is normally upside down compared to an actual tree, with the root at the top and the leaves at the bottom.
Example: The Figure 4.1 represents a tree structure presenting the hierarchical organization of books.

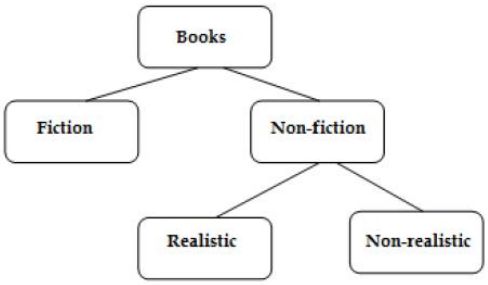


Figure 4.1: Tree Structure of Books.

In the hierarchical organization of books shown in Figure 4.1, Books is the root of the tree. Books can be classified as Fiction and Non-fiction. Non-fiction books can be further classified as Realistic and Non-realistic which are the leaves of the tree. Thus, it forms a complete tree structure.

Trees are primarily treated as data structures rather than as data types.
A tree is a widely-used data structure that depicts a hierarchical tree structure with a set of linked nodes. The elements of data structure in a tree are arranged in a non-linear fashion i.e. they use two dimensional representations. Thus, trees are known as non-linear data structures. This data structure is more efficient in inserting additional data, deleting unnecessary data, and searching new data.

4.2.1 Representation of Tree in Graphs

A graph G consists of a set of objects V = {v1, v2, v3 …} called vertices (points or nodes) and a set of objects E = {e1, e2, e3 ….} called edges (lines or arcs).
The set V (G) is called the vertex set of G and E (G) is the edge set.
The graph is denoted as G = (V, E) Let G be a graph and {u, v} an edge of G. Since {u, v} is 2-element set, we write {v, u} instead of {u, v}.

This edge can be represented as uv or vu. If e = uv is an edge of a graph G, then u and v are adjacent in G and e joins u and v.
Consider the graph in Figure 4.2.

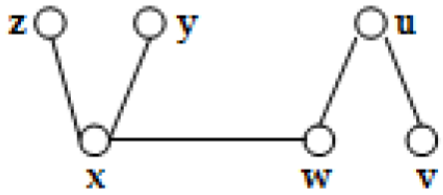


Figure 4.2: Graph.

This graph G is defined by the sets:
V (G) = {u, v, w, x, y, z} and E (G) = {uv, uw, wx, xy, xz}
Every graph has a diagram associated with it. The vertex u and an edge e are incident with each other as are v and e. If two distinct edges e and f are incident with a common vertex, then they are adjacent edges.
A tree is a set of one or more nodes T such that:

i. there is a specially designated node called a root.
ii. The remaining nodes are partitioned into n disjointed set of nodes T_1, T_2,…,T_n, each of which is a tree.

A tree structure is shown in Figure 4.3.

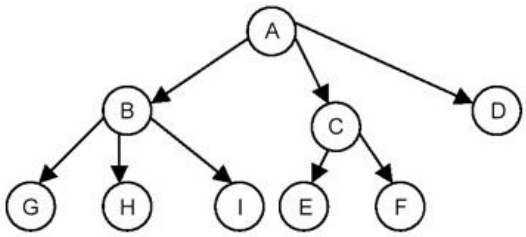


Figure 4.3: A tree structure.

This is a tree because it is a set of nodes {A,B,C,D,E,F,G,H,I}, with node A as a root node and the remaining nodes partitioned into three disjointed sets {B,G,H,I}, {C,E,F} and {D}, respectively. Each of these sets is a tree because each satisfies the aforementioned definition properly.
Shown in Figure 4.3 is a structure that is not a tree.

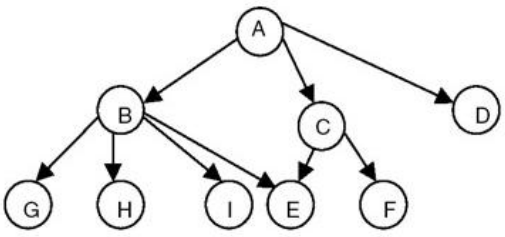


Figure 4.4: A non-tree structure.

Even though this is a set of nodes {A,B,C,D,E,F,G,H,I}, with node A as a root node, this is not a tree because the fact that node E is shared makes it impossible to partition nodes B through I into disjointed sets.

Degree of a Node of a Tree

The degree of a node of a tree is the number of subtrees having this node as a root. In other words, the degree is the number of descendants of a node. If the degree is zero, it is called a terminal or leaf node of a tree.

Degree of a Tree

The degree of a tree is defined as the maximum of degree of the nodes of the tree, that is, degree of tree = max (degree (node i) for i = 1 to n).

Level of a Node

We define the level of the node by taking the level of the root node as 1, and incrementing it by 1 as we move from the root towards the subtrees. So the level of all the descendants of the root nodes will be 2. The level of their descendants will be 3, and so on. We then define the depth of the tree to be the maximum value of the level of the node of the tree.

Did You Know?

The AVL tree is named after its two Soviet inventors, G.M. Adelson-Velskii and E.M. Landis, who published it in their 1962 paper "An algorithm for the organization of information".

4.3 Binary Tree

A tree data structure in which every node has a maximum of two child nodes is known as a binary tree. It is the most commonly used non-linear data structure. A binary tree could either have only a root node or have two disjoint binary trees called the left sub-tree or the right sub-tree. An empty tree could also be a binary tree. The Figure 4.5 represents a binary tree.

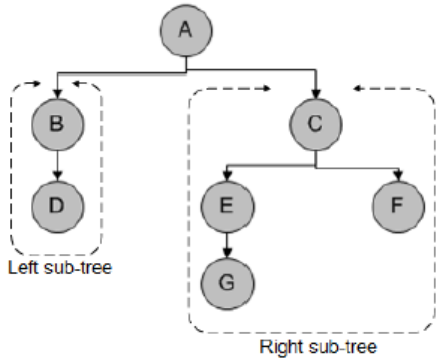


Figure 4.5: A Binary Tree.In the Figure 4.5, the node A is the root node. The nodes B and D belong to the left sub-tree and nodes C, E, F and G belong to the right sub-tree.

So, for a binary tree we find that:

i. The maximum number of nodes at level i will be 2^{i-1}.

ii. If k is the depth of the tree then the maximum number of nodes that the tree can have is $2^k - 1 = 2^{k-1} + 2^{k-2} + \ldots + 2^0$

Also, there are skewed binary trees, such as the one shown in Figure 4.6.

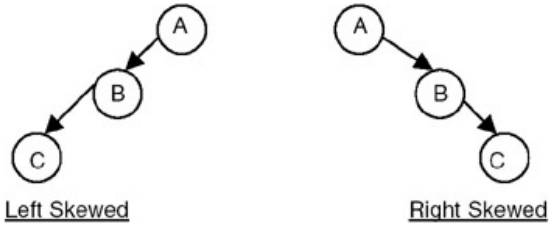


Figure 4.6: Skewed binary trees.

A full binary tree is a binary of depth k having $2^k - 1$ nodes. If it has $< 2^k - 1$, it is not a full binary tree. For example, for k = 3, the number of nodes = $2^k - 1 = 2^3 - 1 = 8 - 1 = 7$. A full binary tree with depth k = 3 is shown in Figure 4.7.

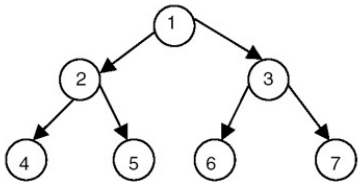


Figure 4.7: A full binary tree.

We use numbers from 1 to $2^k - 1$ as labels of the nodes of the tree.
If a binary tree is full, then we can number its nodes sequentially from 1 to 2^{k-1}, starting from the root node, and at every level numbering the nodes from left to right.
A complete binary tree of depth k is a tree with n nodes in which these n nodes can be numbered sequentially from 1 to n, as if it would have been the first n nodes in a full binary tree of depth k.
A complete binary tree with depth k = 3 is shown in Figure 4.8

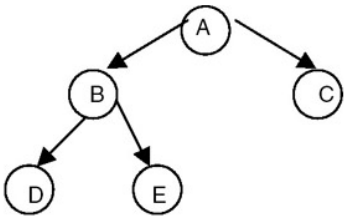


Figure 4.8: A complete binary tree.

4.4 Representation of Binary Tree

If a binary tree is a complete binary tree, it can be represented using an array capable of holding n elements where n is the number of nodes in a complete binary tree. If the tree is an array of n elements, we can store the data values of the i th node of a complete binary tree with n nodes at an index i in an array tree. That means we can map node i to the i th index in the array, and the parent of node i will get mapped at an index i/2, whereas the left child of node i gets mapped at an index 2i and the right child gets mapped at an index 2i + 1.
Since a node in a general tree has a number of children, the implementation of this tree becomes complex than that of a binary tree.
There are two ways to represent trees. They are:
1. Array representation
2. Linked representation

4.4.1 Array Representation

For example, a complete binary tree with depth k = 3, having the number of nodes n = 5, can be represented using an array of 5 as shown in Figure 4.9.

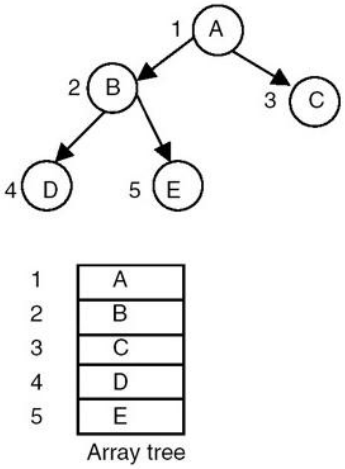


Figure 4.9: An array representation of a complete binary tree having 5 nodes and depth 3.

Shown in Figure 4.10 is another example of an array representation of a complete binary tree with depth k = 3, with the number of nodes n = 4.

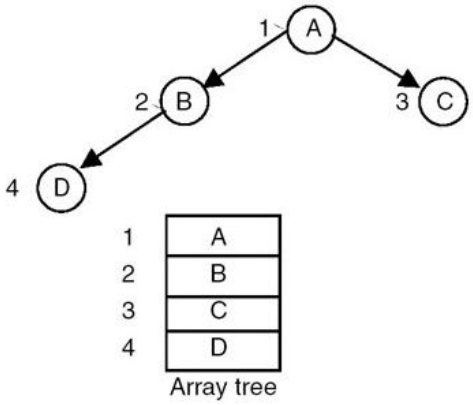
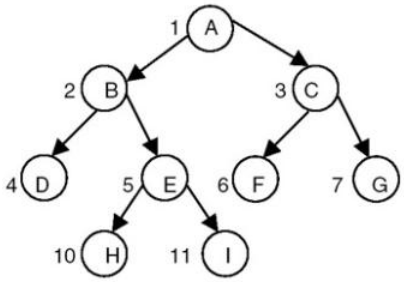


Figure 4.10: An array representation of a complete binary tree with 4 nodes and depth 3.

In general, any binary tree can be represented using an array. We see that an array representation of a complete binary tree does not lead to the waste of any storage. But if you want to represent a binary tree that is not a complete binary tree using an array representation, then it leads to the waste of storage as shown in Figure 4.11.



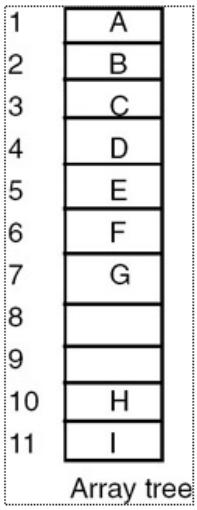


Figure 4.11: An array representation of a binary tree.

4.4.2 Linked Representation

An array representation of a binary tree is not suitable for frequent insertions and deletions, even though no storage is wasted if the binary tree is a complete binary tree. It makes insertion and deletion in a tree costly. Therefore, instead of using an array representation, we can use a linked representation, in which every node is represented as a structure with three fields: one for holding data, one for linking it with the left subtree, and the third for linking it with right subtree as shown here:

leftchild	data	Rightchild

We can create such a structure using the following C declaration:

```
struct tnode
{
  int data
  struct tnode *lchild,*rchild;
};
```

A tree representation that uses this node structure is shown in Figure 4.12.

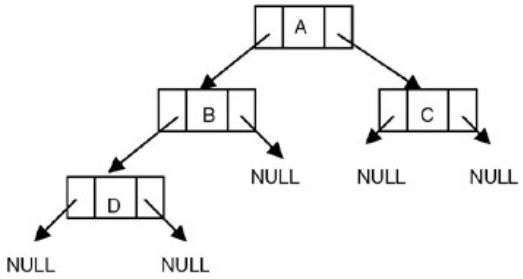


Figure 4.12: Linked representation of a binary tree.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Write the.representation of tree?

……..……

……..……

……...

4.5 Basic Operation on Binary Tree

The binary tree operations help in the logical implementation of the structure of a binary tree. The basic operations such as insertion of a node, deletion and searching of node are performed on the binary tree structure.

Traversals are the most basic operations performed on binary trees. In traversal technique, each node of a tree is visited exactly once systematically. The different traversal techniques are inorder, preorder and post order traversal.

4.5.1 Traversals and Algorithm

Binary tree traversals refer to the commonly used traversing operations on a binary tree. Traversing refers to the order in which traversing is performed on a node of the tree, its right and left sub-trees. It deals with visiting each node in a tree exactly once. A complete traversal of binary tree provides the sequential ordering of information in a tree. A binary tree can be traversed in various ways. But the standard methods of traversal are:

1. Preorder traversal
2. Inorder traversal
3. Postorder traversal

Preorder Traversal

The preorder traversal of a non-empty binary tree is defined as follows:

1. First, visit the root node.
2. Next, traverse the left sub-tree of root node in preorder.
3. Finally, traverse the right sub-tree of root node in preorder.

The Figure 4.13 depicts the functioning of preorder traversal.

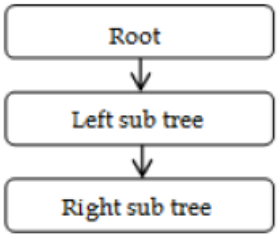


Figure 4.13: Preorder Traversal.

The algorithm for preorder traversal is as follows:

Input – ROOT is the pointer to the root node of binary tree.
Output – Visiting all nodes in preorder manner.
Data Structure – Linked structure of binary tree.
Steps

```
Preorder(Node *ptr)
1. ptr = ROOT                    //Start from ROOT
2. If (ptr ≠ null) then          //If it is non-empty node
3. Visit (ptr)                   //Visit the node
```

4. Preorder (ptr->llink) //Traverse left sub-tree of node in preorder manner
5. Preorder (ptr->rlink) //Traverse right sub-tree of node in preorder manner
6. endif
7. Stop

Example: Consider the binary tree shown in Figure 4.14. The binary tree is represented in such that the nodes in the binary tree are traversed in preorder traversal manner.

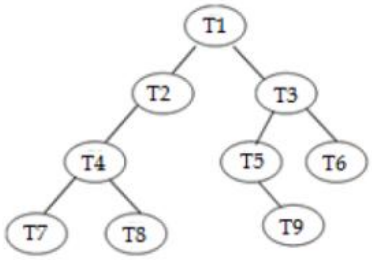


Figure 4.14: Binary Tree for Preorder Traversal.

In the Figure 4.14, T1 is the root node; T2 and T3 are the sub-trees of the root node. The preorder traversal for a binary tree traverses first the root node, then the left sub-tree and finally the right sub-tree. Since the traversing process is in the order of root node, left and right sub-trees, let us assign the alphabet N for visiting root node, L for visiting left sub-tree and R for visiting right sub-tree. The term T1NLR indicates that node T1 is the root node of the binary tree and subscript NLR indicates preorder tree traversal. The following represents the preorder traversal for binary tree present in the Figure 4.14.

T1NLR---->T1 T2 NLR T3 NLR //After visiting T1NLR

---->T1 T2T4 NLRμ T3 NLR //After visiting T2NLR, μis empty child

---->T1 T2 T4T7NLR T8NLR T3NLR//After visiting T4NLR

----> T1 T2 T4 T7 μμT8NLR T3NLR//After visiting T7NLR ,μis empty left/right child

---->T1 T2 T4 T7 T8 μμT3NLR//After visiting T8NLR

---->T1 T2 T4 T7 T8 T3 T5NLRT6NLR//After visiting T3NLR

---->T1 T2 T4 T7 T8 T3 T5 μT9NLRT6NLR//After visiting T5

---->T1 T2 T4 T7 T8 T3 T5 T9 μμT6NLR//After visiting T9NLR

---->T1 T2 T4 T7 T8 T3 T5 T9 T6 μμ//After visiting T6NLR

---->T1 T2 T4 T7 T8 T3 T5 T9 T6

Hence, the preorder traversal for the tree shown in Figure 4.14 is T1 T2 T4 T7 T8 T3 T5 T9 T6. The traversal in preorder starts with traversing root, left sub-tree and right sub-tree. But this traversing happens only during the downward movement of the traverse operation in a binary tree. If the upward traversing is required in a tree, then it takes place in a reverse manner. Due to this reason, a stack is required to save pointer variable during the tree traversal. This mode of traversing is known as iterative traverse. The general form of iterative traversal for preorder using stack is as follows:

Step 1: If the tree is empty. //Check if the tree is empty
then ("empty tree") // If tree is empty write "empty tree"

return
else
Place the pointer to the root of the tree on the stack// Move the pointer to the root of stack
Step 2:
Repeat step 3 while stack is not empty.
Step 3: Pop the top of the stack.
Repeat while the pointer value is not NULL.
Write (Node containing data).
If the right sub-tree is not empty, then stack the pointer to the right sub-tree and set pointer value to left sub-tree.

Inorder Traversal

The inorder traversal of a non-empty tree is defined as follows:
1. First, traverse the left sub-tree of the root node in inorder.
2. Next, visit the root node.
3. Finally, traverse the right sub-tree of the root node.

The Figure 4.15 depicts the functioning of inorder traversal. The figure provides a functional procedure of the inorder traversal.

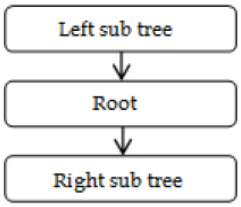


Figure 4.15: Inorder traversal.

The algorithm for inorder traversal is as follows:
1. Input – ROOT is the pointer to the root node of binary tree
2. Output – Visiting all nodes in inorder manner
3. Data Structure – Linked structure of binary tree

Steps:

Inorder(Node *ptr)

1. ptr = ROOT //Start from ROOT
2. If (ptr ≠ null) then //If it is non-empty node
3. Inorder (ptr→llink) //Traverse left sub-tree of node in inorder manner
4. Visit (ptr) //Visit the node
5. Inorder (ptr→rlink) //Traverse right sub-tree of node in inorder manner
6. EndIf
7. Stop

Consider the binary tree shown in the Figure 4.14. The inorder traversing traverses first the left sub-tree, then the node and then the right sub-tree. Since, the traversing order is left sub-tree, root node and right sub-tree, the alphabets L, N, R can be used for representing inorder traversal.
The term T1LNR indicates that node T1 is the root node of the binary tree and subscript LNR indicates inorder tree traversal.

$T1_{LNR}$---->$T2_{LNR}$ T1 $T3_{LNR}$ //After visiting $T1_{LNR}$

---->$T4_{LNR}$ T2 μ T1 $T3_{LNR}$//After visiting $T2_{LNR}$

---->$T7_{LNR}$ T4 $T8_{LNR}$ T2 μ T1 $T3_{LNR}$//After visiting $T4_{LNR}$

---->μ T7 μ T4 $T8_{LNR}$ T2 μ T1 $T3_{LNR}$//After visiting $T7_{LNR}$

---->T7 T4 μ T8 μ T2 μ T1 $T3_{LNR}$//After visiting $T8_{LNR}$

---->T7 T4 T8 T2 T1 $T5_{LNR}$ T3 $T6_{LNR}$//After visiting $T3_{LNR}$

---->T7 T4 T8 T2 T1 μ $T5T9_{LNR}$ T3 $T6_{LNR}$//After visiting $T5_{LNR}$

---->T7 T4 T8 T2 T1 T5μ T9μ T3 $T6_{LNR}$//After visiting $T9_{LNR}$

---->T7 T4 T8 T2 T1 T5T9T3 μ T6 μ//After visiting $T6_{LNR}$

---->**T7 T4 T8 T2 T1 T5T9T3 T6**

Hence, the inorder traversal for the tree shown in Figure 4.14 is T7 T4 T8 T2 T1 T5T9T3 T6.

Example: Program for Inserting Elements into the Tree and Traversing in Inorder.

```
#include<stdio.h>
#include<conio.h>
/*Define tree as a structure with data and pointers to the left and right sub-tree*/
struct tree
{long info; struct tree *left;
struct tree *right;};
/* bintree is declared as the datatype tree and initialized to Null*/
struct tree *bintree=NULL;
/*Global declaration of function insert which returns a pointer to the tree structure and accepts a pointer to tree and a long digit as parameters */
struct tree *insert(struct tree*bintree, long digit);
/*Global declaration of function inorder which does not return any value and accepts a pointer to tree as a parameter*/
void inorder (struct tree*bintree);
void main()                              // Define main function
{
long digit;
clrscr();
puts("Enter integers: and 0 to quit");
scanf("%d",&digit);                      //Reads the first number to be inserted
while (digit!=0)
{
bintree=insert(bintree,digit); // Inserts the number entered
scanf("%d",&digit); }
```

```
puts("Inorder traversing of bintree:\n");
inorder(bintree);                    //Calling inorder function to traverse the tree
}
struct tree* insert(struct tree* bintree, long digit); //insert function is defined
{
if(bintree==NULL)                    //checks if the tree is empty
{
bintree=(struct tree*) malloc(sizeof(struct tree)); //Allocates memory for the tree
bintree->left=bintree->right=NULL;                  //Left and right sub-trees is set to NULL
/* The digit entered is assigned to the info element of the tree node*/
bintree->info=digit;
}
else
{
if(digit<bintree->info)              //If the entered number is less than the data of the node
bintree->left=insert(bintree->left,digit); //insert the digit in the left sub-tree else
/*If the entered number is greater than the data of the node*/
if(digit>bintree->info)
bintree->right=insert(bintree->right,digit); //insert the digit in the right sub-tree
else
if(digit==bintree->info) //If entered number is equal to data of the node
{                            //exits program after printing that duplicate node is present puts("Duplicates
node:program exited");
exit(0);
}
}
return(bintree);
}
void inorder(struct tree*bintree) //Defining inorder function
{
if(bintree!=NULL)                    //Checks if tree is empty
{
inorder(bintree->left);    //Calls the inorder function for left sub-trees
printf("%4ld",bintree->info); // Prints data of the node
inorder(bintree->right);   // Inorder function of right sub-trees
}
}
```

Output:

```
Enter integers and 0 to quit
6 1 2 3 7 8 9 0
Inorder traversing of bintree
1 2 3 6 7 8 9
```

In this program,

1. First the structure tree is defined. It contains a variable info of long type and pointers to the right and left sub-trees.
2. The variable bintree is declared as data type tree and initialized to NULL.

3. The function insert and inorder are globally declared.
4. In the main() function,
(a) First, the numbers to be entered are declared using long data type.
(b) Then, the digit entered is read by the computer.
(c) If the digit is not 0, the insert() function is called to insert the entered digit into the binary tree. Step b, and c are executed repeatedly until the digit entered is 0.
(d) Finally, the inorder() function is called to traverse the tree.
5. The insert() function is defined. It accepts a pointer to a tree and a digit to be inserted in the tree as parameters. The insert function performs the following steps:
(a) It checks if the tree is empty or non-empty.
(b) If the tree is empty it assigns memory to the node, sets the left and right pointers of the node as NULL and assigns the digit to the info variable of the node.
(c) If the tree is non-empty, then it performs the following steps:
  (i) If the digit entered is less than the info stored in the node, it recursively calls itself to enter the digit in the left sub-tree.
  (ii) If the digit entered is greater than the info stored in the node, it recursively calls itself to enter the digit in the left sub-tree.
  (iii) If the entered digit is equal to the info stored in the node, then it displays a message "Duplicates node: program exited" and exits.
(d) The function inorder() is then defined. It accepts a pointer to a tree and a digit to be inserted in the tree as parameters
  (i) It checks if the tree is empty or non-empty.
  (ii) If the tree is non-empty, it traverses first the left sub-tree, then prints the value of the variable info stored in the node and then traverses the right sub-tree.

In the inorder traversal, the nodes traversal starts with visiting right sub-tree, root and left sub-tree. While traversing using stacks, the left sub-tree in a binary tree is traversed by moving down the tree towards left and then pushing node with data into the stack until the left sub-tree pointer node is NULL. Once the left sub-tree is traversed, the stack becomes non-empty, then pop the elements from the stack and print the data, and then traverse the pointer towards right sub-tree. This process continues until right sub-tree is NULL. The algorithm for inorder traversal using stacks is as follows:
Step 1: If the tree is empty then
{
write("empty tree")
return
} else
Place the pointer to the root of the tree on the stack.
Step 2: Repeat step 4 while stack is not empty.
Step 3: Repeat while pointer value is not NULL and stack the pointer to the left sub-tree.
Repeat while the pointer is not NULL.
Write (Node containing data)
If the right sub-tree is not empty, then stack the pointer to the right sub-tree and set pointer to the right sub-tree.

Exercise: Check Your Progress 2
Note: i) Use the space below for your answer.
Ex1: Write a recursive function to determine the sum of digits of a number that is entered through a keyboard.

……..……

……..……

……...

Postorder Traversal

The postorder traversal of a non-empty tree is defined as follows:

1. First, traverse the left sub-tree of the root node in postorder.
2. Next, the right sub-tree of the root node in postorder.
3. Finally, visit the root node.

The Figure 4.16 depicts the functioning of postorder traversal. The figure represents the functionality of postorder traversal.

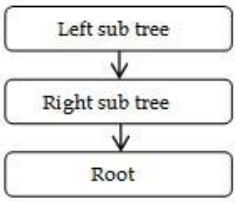


Figure 4.16: Postorder Traversal.

The algorithm for postorder traversal is as follows:

Input – ROOT is the pointer to the root node of binary tree.

Output – Visiting all nodes in postorder manner.

Data Structure – Linked structure of binary tree.

Steps

1. ptr = ROOT	//Start from ROOT
2. If (ptr ≠ null) then	//If it is non-empty node
3. Postorder (ptr→llink)	//Traverse left sub-tree of node in postorder manner
4. Postorder (ptr→rlink)	//Traverse right sub-tree of node in postorder manner
5. Visit (ptr)	//Visit the node
6. EndIf	
7. Stop	

Example: Consider the binary tree shown in the Figure 4.14. The postorder traversing depends on traversing first the left sub-tree, right sub-tree and then the node. Since, the traversing process starts with the left sub-tree, right sub-tree and root node, let us consider the alphabets L, R, N for postorder traversal.

The term T1LRN indicates that node T1 is the root node of the binary tree and subscript LRN indicates postorder tree traversal.

$T1_{LRN}$---->$T2_{LRN}T3_{LRN}T1$ //After visiting $T1_{LRN}$

---->$T4_{LRN}$ μ T2 $T3_{LRN}T1$//After visiting $T2_{LRN}$

---->$T7_{LRN}$ $T8_{LRN}T4$ T2 $T3_{LRN}T1$//After visiting $T4_{LRN}$

---->μμT7 $T8_{LRN}T4$ T2 $T3_{LRN}$ T1//After visiting $T7_{LRN}$

---->T7 μμT8 T4T2 $T3_{LRN}T1$//After visiting $T8_{LRN}$

---->T7 T8 T4 T2 $T5_{LRN}$ $T6_{LRN}T3T1$//After visiting $T3_{LRN}$

---->T7 T8 T4 T2 μ$T9_{LRN}T5T6_{LRN}T3$ T1//After visiting $T5_{LRN}$

---->T7 T8 T4 T2 μμT9T5$T6_{LRN}T3$ T1//After visiting $T9_{LRN}$

---->T7 T8 T4 T2 T9 T5μμT6 T3 T1//After visiting $T6_{LRN}$

---->**T7 T8 T4 T2 T9 T5T6 T3 T1**

Finally the postorder traversal for the tree shown in figure 4.14 is T7 T8 T4 T2 T9 T5 T6 T3 T1. The post order traversal starts with left sub-tree, then moves to the right sub-tree, and then finally to the root. Considering the postorder traversal using stacks, each node is stacked twice during the traversal of left sub-tree and right sub-tree. To distinguish between the left sub-tree and right sub-tree a traversing flag is used. During the traverse of right sub-tree, flag is set to 1. This helps in checking the flag field of the corresponding node. If the flag of a node is negative, then right sub-tree is traversed else the left sub-tree is traversed. The algorithm for postorder of binary tree using stacks is as follows:

Step 1: If the tree is empty then
{
write (“Empty tree”) return
}
else
Initialize the stack and pointer value to the root of tree.
Step 2: Start an infinite loop and repeat till step 5.
Step 3: Repeat while pointer value is no NULL, stack current pointer value. Set pointer value to left sub-tree.
Step 4: Repeat while top pointer on stack is negative.
Pop pointer off stack
write (value of pointer)
If the stack is empty
then return
Step 5: Set pointer value to the right sub-tree of the value on top of the stack.
Step 6: Stack the negative value of the pointer to right sub-tree.
Example
Program for Inorder, Preorder and PostorderTree Traversals.

```
#include<stdio.h>
#include<conio.h>
struct node                1              //Declare node as struct variable
{
int data;                                 //Declare data with data type int
```

```
struct node *right, *left;                    //The node stores pointers to the right and left sub-trees. }root,*p,*q;
                                              /*declare root as a variable of type node and p and q as pointers to node*/
struct node *make(int y)
{
struct node *newnode;          //Declare newnode as pointer to struct node newnode=(struct node *)
malloc(sizeof(struct node)); //Allocate space in memory
/*Assign object data to newnode and initialize to variable y*/
newnode->data=y;
/*Declare right newnode and left newnode to NULL*/
newnode->right=newnode->left=NULL;
return(newnode);
}
void left(struct node *r,int x)                    // Define left sub-tree function
{
/*Checks if left sub-trees is not equal to NULL*/
if(r->left!=NULL) printf("\n Invalid !");          // Prints invalid
else
r->left=make(x);                    //Initialize left sub-tree
}
void right(struct node *r, int x)          //Define right sub-tree
{
/*Checks if right sub-tree is not equal to NULL*/
if(r->right!=NULL)
printf("\n Invalid !"); // Prints invalid
else
r->right=make(x);                                  //Initialize right sub-tree
} void inorder(struct node *r)             //Define inorder traversal function
{
/*Conditional statement, check if r is not equal to NULL*/
if(r!=NULL)
{
/*Recursively call inorder passing the address of the left sub-tree*/
inorder(r->left);
printf("\t %d", r->data);          //Prints the data of the node
/*Recursively call inorder passing the address of the left sub-tree*/
inorder(r->right);
}
}
void preorder(struct node *r)              //Define preorder function
{
/*Checks if r is not equal to NULL*/
if(r!=NULL)
{
printf("\t %d", r->data);                  //Prints the data of the node
/*Recursively call preorder passing the address of the left sub-tree*/
preorder(r->left);
/*Recursively call preorder passing the address of the right sub-tree*/
```

```
preorder(r->right);
}
}
void postorder(struct node *r)              //Define postorder function
{
if(r!=NULL)                                 //Checks if r is not equal to NULL
{
/*Recursively call postorder passing the address of the left sub-tree*/
postorder(r->left); /
/*Recursively call postorder passing the address of the left sub-tree*/
postorder(r->right);
printf("\t %d", r->data);                   //Prints the data of the node
}
}
void main()
{
int no;                                     //Declare variable no
int choice;                                 //Declare variable choice
clrscr();
printf("\n Enter the root:");
scanf("%d",& no);                           //Reads the number entered
root=make(no);                              //Initialize the number to root
p=root;                                     // Value of root is then assigned to variable p
while(1)                                    //Checks the conditions provided in while loop
{
/*Prints the statement "Enter another number"*/
printf("\n Enter another number:");
scanf("%d", &no);                           //Reads the number entered
/*Conditional statement, check if no is equal to –1*/
if(no== –1)
break;                                      //If condition is true, the if loop breaks
p=root;                                     //Assign value of root to p variable
q=root;                                     //Assign value of root to q variable
/*Check if no is not equal to variable p and q not equal to NULL*/
while(no!=p->data && q!=NULL)
{
p=q;
if(no<p->data)                              //Check if no is less than variable p
/*Set q to the left sub-tree of p*/
q=p->left;
else
/*Set q to the right sub-tree of p*/
q=p->right;
}
/*Check if variable no is less than p variable with data*/
if(no<p->data)
{
```

```
/*prints the node of left tree*/
printf("\n Left branch of %d is %d", p->data, no);
left(p, no);
}
else
{
right(p,no);
/*prints the node of right tree*/
printf("\n Right Branch of %d is %d", p->data,no);
}
 while(1)
{
printf("\n 1.Inorder Traversal \n 2.Preorder Traversal \n 3.Postorder Traversal \n 4.Exit");

printf("\n Enter choice:");
scanf("%d",&choice);                // Reads the choice entered
switch(choice)
{
/*Switches to inorder function and performs inorder traversal*/
case1 :inorder(root);
break;
/*Switches to preorder function and performs preorder traversal*/
case2: preorder(root);
break;
/*Switches to postorder function and performs postorder traversal*/
case3: postorder(root);
break;
case4 : exit(0);              //Exits from the function
default:
printf("Error ! Invalid Choice "); //Prints invalid statement
break;
}
}
getch();
}
}
```

Output:

```
Enter the root: 5
Enter another number: 7
Right branch of 5 is 7
1. Inorder traversal
2. Preorder traversal
3. Postorder traversal
4. Exit
Enter choice: 1
5         7
```

1. Inorder traversal
2. Preorder traversal
3. Postorder traversal
4. Exit
Enter choice: 2
5 7
1. Inorder traversal
2. Preorder traversal
3. Postorder traversal
4. Exit
Enter choice: 3
7 5

In this program,

1. First, the header file stdio.h is included using include keyword.

(a) The variable node is defined as a structure. It has an integer variable data and pointers to its left and right sub-trees, root is a declared as a variable of type node. The variables p and q are declared as pointers to node.

2. Then, the make() function is defined. It returns a pointer to the structure node and accepts as parameter an integer variable y. It executes the following steps:

(a) It declares newnode as pointer to the structure node and assigns memory to it.

(b) It assigns the integer y to the data variable of the newnode.

(c) It sets the right and left sub-tree pointers of the newnode to NULL

3. Then, the left() sub-tree function is defined. It accepts as parameters r which is a pointer to the structure node and an integer variable x. It executes the following steps:

(a) It checks if the left pointer of r is empty. If it is not empty, then it calls the make function passing x as a parameter.

4. Then, the right() sub-tree function is defined. It accepts as parameters r which is a pointer to the structure node and an integer variable x.:

(a) It checks if the right pointer of r is empty. If it is not empty, then it calls the make function passing x as a parameter.

5. Then, the inorder() function is defined. It accepts as parameters r which is a pointer to the structure node.

(a) It checks if r is non-empty. If it is non-empty, it then traverses the left sub-tree, prints the data and then traverses the right sub-tree.

6. Then, the preorder() function is defined. It accepts as parameters r which is a pointer to the structure node.

(a) It checks if r is non-empty. If it is non-empty, it prints the data, traverses the left sub-tree, and then traverses the right sub-tree.

7. Then, the postorder() function is defined. It accepts as parameters r which is a pointer to the structure node.

(a) It checks if r is non-empty. If it is non-empty, it then traverses the left sub-tree, then the right sub-tree and then prints the data.

8. In the main() function,

(a) The variables no, choice are declared as integer variables.

(b) The value for the root node is accepted and added to the tree by calling the make() function.

(c) The value of root is then assigned to variable p and q.

(d) The program execution enters a while loop in which the following steps are performed:

(i) First, it accepts another integer no.

(ii) Then, the while loop is exited if no is equal to –1.

(iii) Then, the following steps are repeatedly performed if the no. entered is not equal to variable data of p and q is not equal to NULL.

I. First, p is assigned the value of q.

II. If the number is lesser than the data of p, q is assigned the address of the left sub-tree else q is assigned the address of the right sub-tree

(iv) If no is less than data of variable p the function left() is called passing p and no as the parameters else the function right () is called.

(e) A while loop is used to obtain the choice of traversal.

(i) If 1 is entered, inorder traversal is selected and the inorder function is executed.

(ii) If 2 is entered, preorder traversal is selected and preorder function is executed.

(iii) If 3 is selected, preorder traversal is selected and postorder function is executed.

(iv) If 4 is entered, the while loop is exited.

(v) If wrong digit is entered, an error message is printed on the screen.

(f) The getch() prompts the user to enter a key to exit the program.

Exercise: Check Your Progress 3

Note: i) Use the space below for your answer.

Ex1: Consider the algebraic expression (A + (B–C)) / ((D–E) * (F+G–H))

Construct a binary tree with the above expression and traverse the tree in inorder, preorder, and postorder manner by applying the tree traversal algorithms.

……..……

……..……

……...

4.6 Creation of Binary Search Tree

You have already learnt about binary trees. The disadvantage of binary trees is that data is stored in these trees in any order and hence, the time taken to search these trees is longer. Search trees are data structures that support many dynamic-set operations such as searching, finding the minimum or maximum values, insertion, and deletion. Binary search trees, AVL trees and B+ trees are examples of search trees. A binary search tree (BST) has binary nodes. In a binary search tree, for a given node with value n, each node to the left has a value lesser than n and each node to the right has a value greater than n. This applies recursively down the left and right sub-trees. Figure 4.17 shows a binary search tree where characters are stored in the nodes.

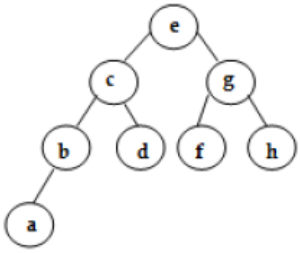


Figure 4.17: A Binary Search Tree.

Binary search trees provide an efficient way to search through an ordered collection of objects. Consider searching an ordered list. The search must proceed successively from one end of the list to the other. On an average, n/2 nodes must be compared for an ordered list that contains n nodes. In the worst case, all n nodes need to be compared. For a large collection of objects, this is very expensive.

Binary search tree enables searching quickly through the nodes. The longest path to search is equal to the height of the tree. Thus, the efficiency of a binary search tree depends on the height of the tree. For a tree with n nodes, the smallest possible height is log n and that is the number of comparisons that are needed on an average to search the tree.

Thus, binary search trees are node based data structures used in many system programming applications for managing dynamic sets. Another example for a binary search tree is given in Figure 4.18. We know that, all the elements in the left sub-tree are lesser than the root node and the elements in the right sub-tree are greater than the root node.

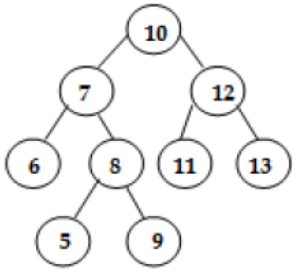


Figure 4.18: Example for Binary Search Tree.

In the binary search tree represented in Figure 4.18, 10 is the root node and all the elements in the left sub-tree are lesser than 10 and the elements in the right sub-tree are greater than 10. Every node in the tree satisfies this condition for the existing left and right sub-trees.

Caution

A tree must be balanced to obtain the smallest height, i.e., both the left and right sub-trees must have the same number of nodes.

Self Assessment Questions

1. A tree data structure in which every node has a maximum of two child nodes is known as a ……. tree.

(a) Binary (b) AVL (c) Game (d) Huffman

2. The…… of the graph are used to represent objects.

(a) Arcs (b) Edge (c) Vertices (d) Directions

3. The nodes which have the same parent node are known as ……...

(a) Root (b) Siblings (c) Left nodes (d) Right nodes

4. The height of an empty tree is ……..

(a) 0 (b) 1 (c) –1 (d) Null

4.7 Types of Binary Trees

Binary trees can be classified on the basis of the level of nodes and the number of nodes that are present at each level of the tree. The types of binary trees are:

1. Complete binary tree
2. Strictly binary tree

3. Extended binary tree

4.7.1 Complete Binary Tree

A tree is known as a complete binary tree if each of its level, except the last level, is completely filled and all nodes are as far left as possible. Therefore, at any level 'l', the maximum number of nodes must be equal to 2l. Figure 4.19 depicts a complete binary tree. The figure depicts the nodes in the binary tree with their associated levels.

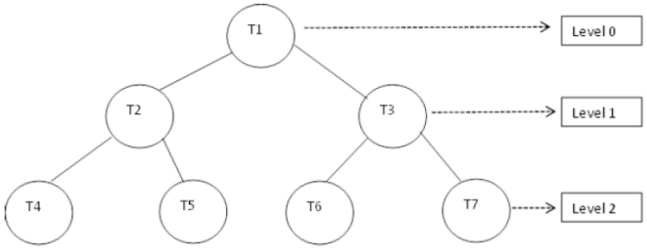


Figure 4.19: A Complete Binary Tree.

In the Figure 4.19, the node T1 at level 0 represents the root node. The two sub nodes T2 and T3 are the child nodes at level 1. The successor nodes T4, T5, T6 and T7 are the terminal nodes. The number of nodes at level 1 is equal to 2. Similarly the number of nodes at level 2 is equal to 4.

The main advantage of a complete binary tree is that the position of the parent node and child node can be mapped easily in an array. The mapping for a binary tree is defined by assigning a number to every node in the tree. The root node is assigned the number 1.For the other nodes, if i is its number, then the left child node is assigned the position 2i and the right child node is assigned the position 2i+1. The mapping of binary tree provides a simple form of array storage representation. Hence the nodes in array can be stored as a[i], where a[ ] is an array.

Caution

In a complete binary tree, at level 0 there must be only one node known as root node and it should have a maximum of two sub nodes at level 1, and at level 2 there must be a maximum of 4 sub nodes.

Example: Figure 4.20 depicts the representation of complete binary tree in an array. An array representation of binary tree allocates nodes of the tree in a memory. Each node is indexed such that the nodes are associated with the index number of the array for allocation.

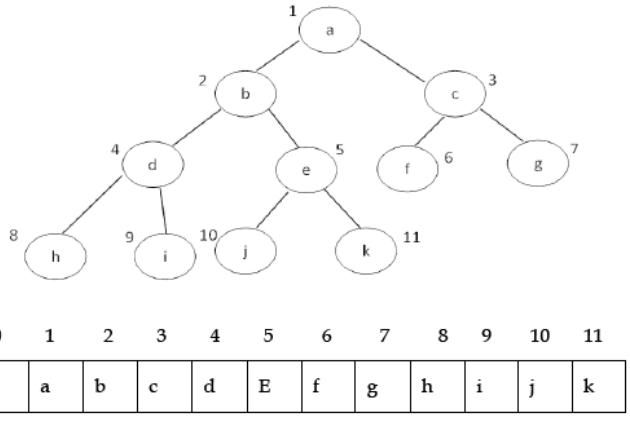


Figure 4.20: Representation of Complete Binary Tree into an Array.

In the Figure 4.20, consider the node e stored at index i=5. The parent node of node e is node b which is stored at the index i/2; 5/2=2. The left child node of node e is node j and it is stored at the index 2i; 2*5= 10 and right child node k is stored at the index 2i+1 i.e., 2*5+1=11.

4.7.2 Strictly Binary Tree

A binary tree is called a strictly binary tree when the non-terminal nodes have exactly two child nodes forming left and the right sub-trees.

Figure 4.21 depicts a strictly binary tree. A strictly binary tree with five nodes is provided such that the non-terminal nodes form the left and right sub-trees.

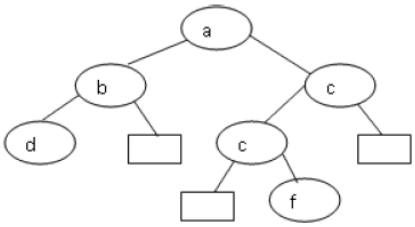


Figure 4.21: Strictly Binary Tree.

In the Figure 4.21, nodes a and b have two child nodes. The parent node a has two child nodes b and c forming the left sub-tree and right sub-tree respectively. Similarly, node b is the parent node for nodes d and e. A binary tree in which its non-leaf nodes possess exactly two child nodes represents a strictly binary tree.

4.7.3 Extended Binary Tree

A binary tree is called an extended binary tree when every node in the tree either has 0 or 2 sub nodes (child nodes). The nodes with two child nodes are called internal nodes and nodes without child nodes are called external nodes. The extended binary tree is also called a 2-tree. The nodes in the binary tree possessing only one child node can be extended with one more child node. The extended binary tree plays a very important role in implementing algebraic expressions. The algebraic expressions are represented by operands and operators. Hence, the left and right child nodes represent the operands and parent node represents the operator.

Example: Consider Figure 4.22 that depicts a binary tree with a single child node. The binary tree with single child node is initial phase of designing the binary tree which can be made a complete binary tree by extending the nodes.

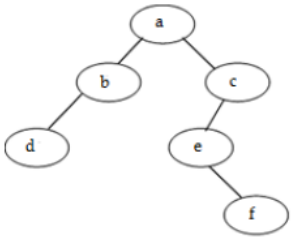


Figure 4.22: Binary Tree with Single Child Node.

In the Figure 4.22, the parent nodes b and c have only one child, node d and e respectively. The node e also has single child node f. By adding another child node to the parent nodes b, c and e, we can obtain an extended binary tree.

Figure 4.23 depicts an extended binary tree.

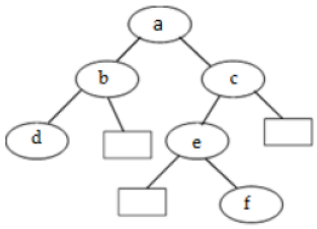


Figure 4.23: Extended Binary Tree.

Exercise: Check Your Progress 4
Note: i) Use the space below for your answer.
Ex1: What is extended binary tree?

……..……

……..……

……...

4.8 Operations on Binary Search Tree (BST)

The four main operations that we perform on binary trees are:
1. Searching
2. Insertion
3. Deletion
4. Traversal

4.8.1 Searching in Binary Search Trees

In searching, the node which we want to search is called as key node. We first match the key node with the root node. If the value of the key node is greater than the current node, then we search for it in the right sub-tree, else we search in the left sub-tree. We continue this process until we find the node or till no nodes are left. The pseudo code for searching a binary search tree is as follows:

Pseudocode for Searching a Binary Search Tree:

```
find(X, node){
if(node = NULL)
    return NULL
if(X = node:data)
    return node
else if(X<node:data)
    return find(Y,node:leftChild)
else if(X>node:data)
    return find(X,node:rightChild)
}
```

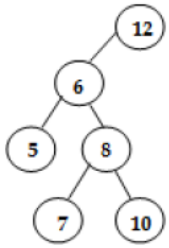


Figure 4.24: Binary Search Tree.

4.8.2 Inserting in Binary Search Trees

To insert a new element in an existing binary search tree, first we compare the value of the new node with the current node value. If the value of the new node is lesser than the current node value, we insert it as a left sub-node. If the value of the new node is greater than the current node value, then we insert it as a right sub-node. If the root node of the tree does not have any value, we can insert the new node as the root node.

Algorithm for Inserting a Value in a Binary Search Tree

1. Read the value for the node that needs to be created and store it in a node called NEW.
2. At first, if (root! =NULL) then root = NEW.
3. If (NEW->value < root->value) then attach NEW node as a left child node of root, or else attach NEW node as a right child node of root.
4. Repeat steps 3 and 4 for creating the desired binary search tree completely.

When inserting any node in a binary search tree, it is necessary to look for its proper position in the binary search tree. The new node is compared with every node of the tree. If the value of the node which is to be inserted is more than the value of the current node, then the right sub-tree is considered, else the left sub-tree is considered. Once the proper position is identified, the new node is attached as the left or right child node.

Let us now discuss the pseudo code for inserting a new element in a binary search tree.

Pseudocode for Inserting a Value in a Binary Search Tree

```
//Purpose: Insert data object X into the Tree
//Inputs: Data object X (to be inserted), binary-search-tree node
//Effect: Do nothing if tree already contains X;
// otherwise, update binary search tree by adding a new node containing data object X
insert(X, node){
   if(node = NULL){
     node = new binaryNode(X,NULL,NULL)
     return
     }
     if(X = node:data)
      return
```

```
    else if(X<node:data)
      insert(X, node:leftChild)
    else                          // X>node:data
      insert(X, node:rightChild)
}
```

Example: Consider Figure 4.25. In this figure, 35 is to be inserted. First, 35 compared with the value of root node i.e., 10. As 35 greater than 10, the right sub-tree is considered. Now, 35 compared with 22. As it greater than 22, the search moves to the right sub-tree. Then 35 compared with 34 and again the search moves to the right sub-tree. Now, 35 is compared with 40, but as it is lesser than 40, we need to move to the left branch of 40. But since the node 40 has no left child, 35 attached as the left child of 40. After the insertion of 35, the tree looks as shown in the Figure 4.26.

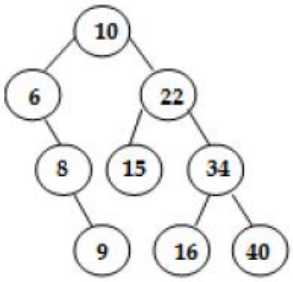


Figure 4.25: Binary Tree before Insertion.

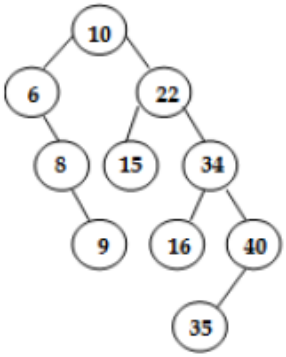


Figure 4.26: Binary Tree after Insertion.

4.8.3 Deleting in Binary Search Trees

If the node to be deleted has no children, we can just delete it. If the node to be deleted has one child, then the node is deleted and the child is connected directly to the parent node.

There are mainly three cases possible for deletion of any node from a binary search tree. They are:

1. Deletion of the leaf node
2. Deletion of a node that has one child
3. Deletion of a node that has two children.

We can delete an existing element from a binary search tree using the following pseudocode:

Pseudocode for Deleting a Value from a Binary Search Tree

```
//Purpose: Delete data object X from the Tree
//Inputs: Data object X (to be deleted), binary-search-tree node
//Effect: Do nothing if tree does not contain X;
// else, update binary search tree by deleting the node containing data object X
delete(X, node){
if(node = NULL) //nothing to do
return
if(X<node: data)
delete(X, node: leftChild)
else if(X>node: data)
delete(X, node: rightChild)
else { // found the node to be deleted. Take action based on number of node children
if(node: leftChild = NULL and node: rightChild = NULL){
delete node
node = NULL return }
else if(node: leftChild = NULL){
tempNode = node
node = node: rightChild
delete tempNode}
else if(node: rightChild = NULL)
{ tempNode = node
node = node: leftChild
delete tempNode }
else { //replace node:data with minimum data from right sub-tree
tempNode = findMin(node: rightChild)
node:data = tempNode: data
delete(node:data,node: rightChild)
}
}
```

Pseudocode for Finding Minimum Value from a Binary Search Tree

```
//Purpose: return least data object X in the Tree
//Inputs: binary-search-tree node
// Output: BST-node n containing least data object X, if it exists; NULL otherwise
findMin(node)
{
if(node = NULL)
//empty tree
return NULL
 if(node: leftChild = NULL)
return node
 return findMin(node: leftChild)
}
```

Deletion of a Leaf Node

Deletion of a leaf node is considered to be the simplest form of deletion, wherein the left or right pointer of the parent node is set as NULL. From the given tree in Figure 4.27, the node with value 6 has to be deleted. Hence, the left pointer of its parent node is set as NULL, i.e., left pointer of node with value 7 is set to NULL. The Figure 4.28 represents the tree after deletion of node holding value 6.

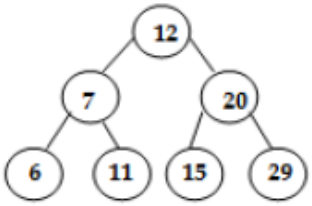


Figure 4.27: Before Deletion.

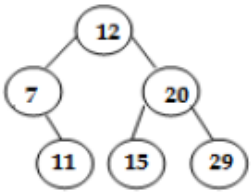


Figure 4.28: After Deletion.

Deletion of a Node That Has One Child

Consider Figure 4.29, to understand deletion of a node that has one child node. If the node 15 has to be deleted, node 18 must be copied to the place of 15 and then the node must be set free. It is noted that the inorder successor is always copied at the position of a node being deleted. The Figure 4.30 represents the tree after deletion of node 15.

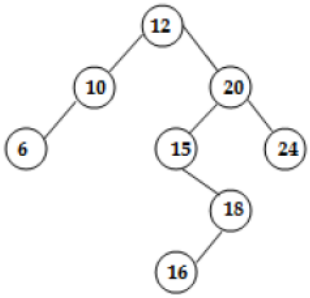


Figure 4.29: Before Deletion.

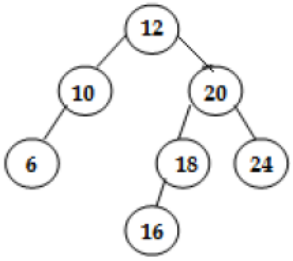


Figure 4.30: After Deletion.

Deletion of a Node That Has Two Children

Consider Figure 4.31 to understand deletion of a node that has two children. The node with the value 8 needs to be deleted and also the inorder successor of node 8 is to be found. The inorder successor will be copied at the location of the node. The Figure 4.32 represents the tree after deletion of node holding value 8.

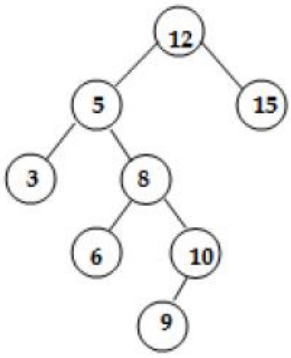


Figure 4.31: Before Deletion.

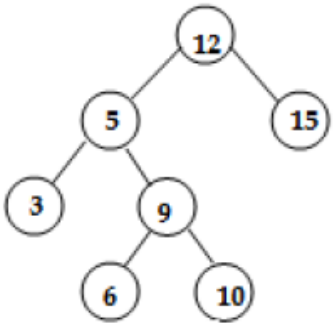


Figure 4.32: After Deletion.

Thus, 9 must be copied at the position where the value of node was 8 and the left pointer of 10 must be set as NULL. This completes the entire deletion procedure.

Exercise: Check Your Progress 5

Note: i) Use the space below for your answer.

Ex1: Write the pseudocode for traversing a binary search tree recursively.

……..……

……..……

……...

4.8.4 Other Operations

Finding the Height of a Tree The maximum level is referred as the height of the tree. The root is always at zero level, the adjacent nodes to the root are the first level, and so on. In the given Figure 4.33, the height of the tree is 3. The height of the tree is also referred as the depth of the tree.

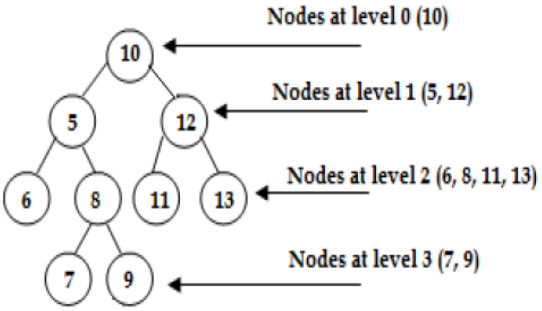


Figure 4.33: Binary Tree.

The C function to find the height of a tree is shown in the given example.

Example

```
struct node
{
    int element;
    struct node *L;
    struct node *R;
};
typedefstruct node* NODE;
int max(int x, int y)              // function to find maximum of two numbers
{
    return(x>y)?x:y;
 }
int height(NODE root)              // function to find the height of the tree
 {
    if(root==NULL)
       return 0;
     return 1+max(height(root->L),height(root->R));
}
```

In this example:

1. A structure named node is created. It consists of two pointer variables named **L** and **R** that point to the left node and right node.
2. An object called NODE is created to access the structure element.
3. Two functions are created namely max and height.
4. The max function compares its two arguments **x** and **y** and returns the greater number.
5. The height function checks if the **root** is NULL and returns 0 if the condition is true.
6. The height function recursively calls max function passing the left and right sub-trees as parameters. The value returned by the max function is incremented and returned.

To Find the Maximum and Minimum Value in a tree

In a binary search tree, a node with maximum value is found by traversing and obtaining the extreme right node of the tree. If there is no right sub-tree, then the root is returned as the node which holds the item of the

highest value. The C function to return the address of highest item in binary search tree is shown in given example.
Example

```
struct node
{
   int element;
   struct node *L;
   struct node *R;
 };
typedef struct node* NODE;
NODE maximum(NODE root)
{


  NODE data;
  if(root==NULL)
    return root;
  data=root;
  while(data->R!=NULL)
    data=data->R;        // find right most node in binary search tree
  return data;
}
```

In this example:

1. A structure named node is created. It consists of two pointer variables named **L** and **R** that point to the left node and right node.
2. An object called NODE is created to access the structure element.
3. The function maximum is defined which accepts the address of the root node as the parameter.
4. In the function maximum():
   (e) A variable data is declared as type NODE.
   (f) If root is entered as null, then the function returns root. Else, the value of root is assigned to the variable data.
   (g) Using the while loop, the extreme right node in the tree is found and its address is returned.

In a binary search tree, a node with minimum value is found by traversing and obtaining the extreme left node of the tree. If there is no left sub-tree, then the root is returned as the node which holds the item of the least value.

Self Assessment Questions

5. (d) In sequential representation, the level of the nodes starts with level one that contains the root node.
(a) True (b) False

6. In a binary tree, at level 1, there must be only one node known as root node.
(a) True (b) False

7. Which among the following traversal of binary tree starts with traversing the root node?
(a) Preorder (b) Inorder (c) Postorder (d) All of these.

8. Which among the following operations is performed before performing insertion operation?
(a) Searching (b) Deletion (c) Modification (d) None of these.

4.9 Threaded Binary Trees

In a binary tree with n nodes, there exists n+1 NULL links and 2n number of total links. However, a large value of n, the n+1 NULL and 2n number of links results in more space wastage. Hence, the solution will be to change the node structure for leaf nodes so that the nodes only consist of data field. But, this solution provides complex algorithms. Hence, the only solution is to consider the fixed node structure and use the NULL links to simplify some of the operations. This solution provides the concept of threaded binary trees.
In the threaded binary tree, the NULL links are replaced by pointers known as threads. These threads point to other nodes of a tree. When the tree is traversed in inorder, and if the left child of a node p in a binary tree is NULL, then it will be replaced with a thread. The thread will point to the node which appears just before the node p. Similarly, if the right child of node p is NULL, then it will be replaced with a thread. The thread will then point to a node that appears just after the node p after the inorder traversal of a tree. Such threads are known as inorder threads.
Example

Consider the binary tree in Figure 4.34. The figure is a simple binary tree with four levels of nodes.

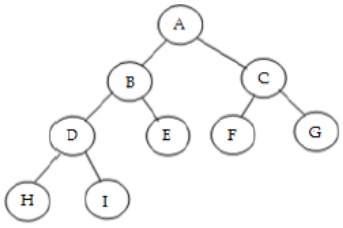


Figure 4.34: Binary Tree.

In the Figure 4.34, the inorder traversal of the binary tree is “H D I B E A F C G”. The equivalent threaded binary tree is shown in Figure 4.35.

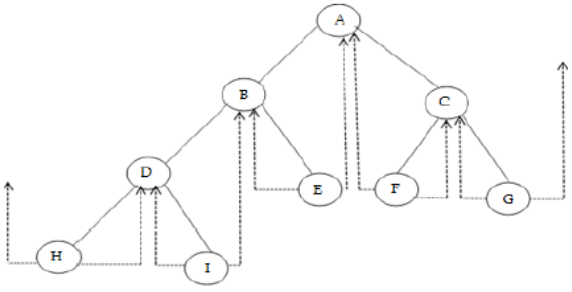


Figure 4.35: Threaded Binary Tree.

For a threaded binary tree leaf nodes are considered. In Figure 4.35, if we consider leaf node I then the inorder predecessor of I is D and the left thread will point at node D.
Similarly, the inorder successor of I is B and the right thread will point at node B. All the nodes in the binary tree will be traversed in the similar format.

But, the left thread of node H does not have an inorder predecessor and right thread of node G does not have inorder successor. In such situations, the threads pointing to particular node are not obtained. Hence, to solve such problems, the threaded binary tree uses a node called head node. The head node will have an identical structure similar to the normal tree nodes. In such cases, if the tree is non-empty, then its left child will point at the root of the tree. Similarly, the left thread of node H and right thread of node G will point to its head node.

Exercise: Check Your Progress 6

Note: i) Use the space below for your answer.

Ex1: Create a binary tree for the algebraic expression ((T1* T2 + T3) + (T4 – T5) * T6) and represent the binary tree in array representation.

……..……

……..……

……...

4.10 Application of Binary Tree

Several important kinds of binary tree are the Huffman coding tree (which is used in data compression), the expression tree, the decision tree and the binary search tree (BST).

4.10.1 Huffman Coding Tree

In 1951, David A, Huffman constructed the Huffman Tree**.** Huffman codes are digital data compression codes which were devised by Prof. David A. Huffman. Huffman codes give good compression ratios. Even today, after 50 years, Huffman codes are very useful. Huffman compression is a compression technique in which there is no loss of information when the data is compressed i.e., after we decompress the data, the original information can be retrieved. Hence, it is named as 'lossless compression'. Lossless compression is desired in compressing text documents, bank records, and so on.

Data Encoding Schemes Fall Into Two Categories

1. *Fixed length encoding:* In fixed length encoding, all symbols are encoded using the same number of bits. An example of fixed length encoding is ASCII code which uses 7 bits to encode a total of 128 different symbols. The difficulty with fixed length codes is that the probability of occurrence of the symbols to be encoded is not considered. A symbol that occurs 1000 times is encoded with the same number of bits as a symbol which comes only 10 times. This disadvantage makes fixed length encoding inefficient for data compression.
2. *Variable length encoding:* Variable length encoding assigns less number of bits to symbols which occur more often and more number of bits to symbols whose frequency is less. The Huffman Encoding scheme falls in the category of variable length encoding i.e., code for the symbol depends on the frequency of occurrence of that symbol.

Algorithm for Constructing Huffman Tree

The following sequence of steps is to be followed to construct a Huffman tree:

1. Input all symbols along with their respective frequencies.
2. Create leaf nodes representing the scanned symbols.
3. Let S be a set containing all the nodes created in step 2

When there is only one node in S, then the following steps are followed:

1. Sort the nodes (symbols) in S with respect to their frequencies.
2. Create a new node to combine the two nodes with least frequencies.

3. Frequency of this new combined node will be equal to the sum of frequencies of nodes which were combined. This newly created combined node will be the parent of two nodes which were combined.
4. Replace the two nodes which were combined with the new combined node in S.
5. After the 4th step you will be left with only one node, which is the root of the Huffman tree, having frequency equal to the sum of all frequencies of all symbols. Thus, a tree is generated with leaf nodes containing the basic symbols whose code is to be found.
With the help of an example we will learn how to construct a Huffman tree. The Table 4.1 shows the frequency of occurrence of different symbols.

Table 4.36: Symbol Frequency Table.

Symbol	Frequency of Occurrence
A	24
B	12
C	10
D	8
E	8

Using the symbols and frequencies from the Table 4.36, we create the leaf nodes and then sort them. Symbols D and E have the least frequency, 8; these two nodes are combined to make a node DE having frequency 8+8=16. This new node DE is the parent node of the nodes D and E, and DE replaces D and E as shown in Figure 4.37.

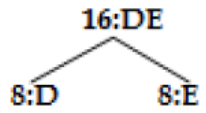


Figure 4.37: Step 1 for Constructing Huffman Tree.

Again we sort the nodes based on their frequency of occurrence. Now DE and C have the least frequencies i.e., 16 and 10 each. This time we combine DE and C to create a new node DEC having frequency 26. Nodes DE and C are replaced by their parent DEC as depicted in Figure 4.38.

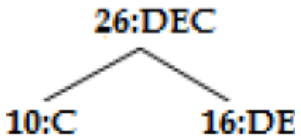


Figure 4.38: Step 2 for Constructing Huffman Tree.

Similarly, combine B with frequency 12 and DEC with frequency 26 to create BDEC. BDEC becomes the parent of B and DEC with frequency 38. At last only two nodes are left namely, BDEC and A. We again sort them and combine both to form ABDEC which has a frequency count of 62.

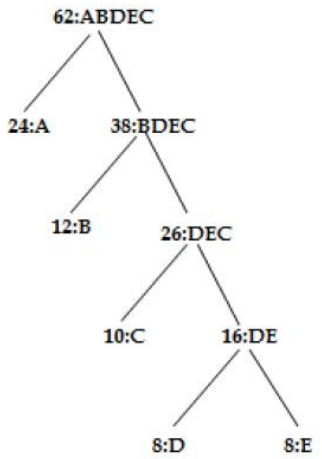


Figure 4.39: The Huffman Tree.

After making ABDEC parent of A and BDEC and replacing them with ABDEC, we have created the Huffman tree for the symbols in Table 4.1. Node ABDEC is the root of the tree. The Figure 4.38 shows the Huffman tree thus constructed.

Did You Know?

Claude Shannon was inventor of information theory to develop a similar code as Huffman code. Huffman avoided the major flaw of the suboptimal Shannon-Fano coding by building the tree from the bottom up instead of from the top down.

4.10.2 Expression Trees

The binary tree offers an alternative to the stack-based transformation and evaluation of arithmetic expressions. Expression trees for the expressions ((3 + 6) % 7) ^ 5 – 1 and 3 + 6 % 7 ^ 5 – 1 are shown below in Figure 4.40.

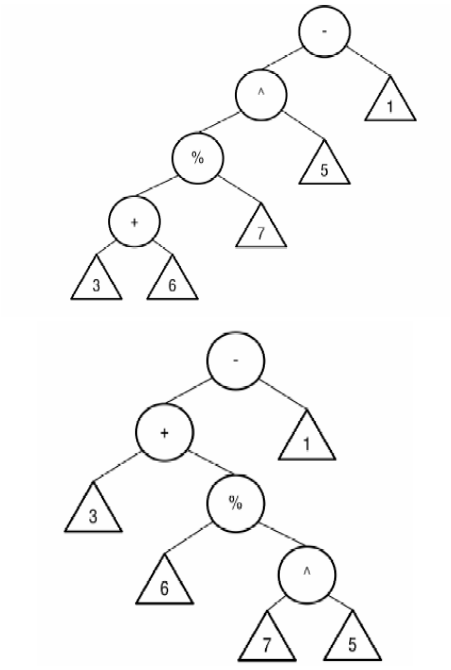


Figure 4.41: Two example expression trees; an expression tree's leaves are operands, its internal nodes operators.

An expression tree need not store parentheses because the correct order of operations inheres in the tree's structure. Preorder, inorder, and postorder traversals of an expression tree yield prefix, infix, and postfix expressions, respectively. Huffman coding trees and expression trees are examples of *full binary trees*. Any node of a full binary tree has either zero or two children. Some books reverse the definitions of full and complete binary trees presented here.

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: Construct a decision tree for finding the greatest number among a given set of numbers.

……..……

……..……

……...

4.11 B-Tree and Height Balanced Tree

4.11.1 B-Tree

A B-tree is a specialized multiway tree designed especially for use on disk. In a B-tree each node may contain a large number of keys. The number of subtrees of each node, then, may also be large. A B-tree is designed to branch out in this large number of directions and to contain a lot of keys in each node so that the height of the tree is relatively small. This means that only a small number of nodes must be read from disk to retrieve an item. The goal is to get fast access to the data, and with disk drives this means reading a very small number of records. Note that a large node size (with lots of keys in the node) also fits with the fact that with a disk drive one can usually read a fair amount of data at once.

A *multiway tree of order m* is an ordered tree where each node has at most m children. For each node, if k is the actual number of children in the node, then k – 1 is the number of keys in the node. If the keys and subtrees are arranged in the fashion of a search tree, then this is called a *multiway search tree of order m*. For example, the following is a multiway search tree of order . Note that the first row in each node shows the keys, while the second row shows the pointers to the child nodes. Of course, in any useful application there would be a record of data associated with each key, so that the first row in each node might be an array of records where each record contains a key and its associated data. Another approach would be to have the first row of each node contain an array of records where each record contains a key and a record number for the associated data record, which is found in another file. This last method is often used when the data records are large. The example software will use the first method.

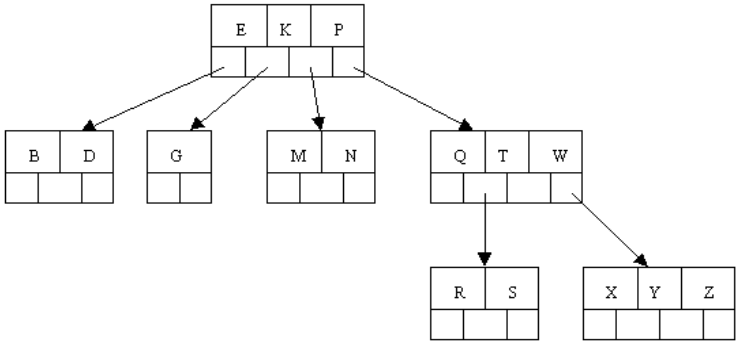


Figure 4.42: B-tree.

4.11.2 Height Balanced Tree

In order to prevent a tree unbalance, with height-balanced trees we associate a balance indicator with each node in the tree. This indicator will contain one of the three values which we denote as left(L), right(R), or balance(B), according to the given definitions.

Left: A node will be called left heavy if the longest path in its left subtree is one longer than the longest path of its right subtree.

Balance: A node will be called balanced if the longest paths in both of its subtrees are equal.

Right: A node will be called right heavy if the longest path in its right subtree is one longer than the longest path of its left subtree.

In a balanced tree each node must be in one of these three states. If there exists a node in a tree where this is not true, then such a tree is said to be unbalanced.

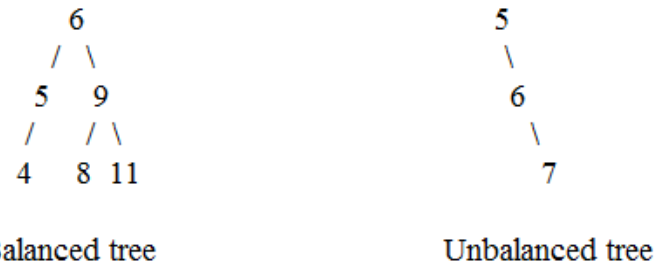


Figure 4.43: Balanced tree.

A new node is inserted at the leaf or terminal node level. The only nodes which can have their balance indicator changed by such an insertion are those which lie on a path between the root of the tree and the newly inserted leaf. The possible changes which can occur to a node on this path are as follows:

1. The node was either left or right heavy and has now become balanced.
2. The node was balanced and has now become left or right heavy.
3. The node was heavy and the new node has been inserted in the heavy subtree, thus creating an unbalanced subtree. Such a node is said to be a critical node.

If condition 1 applies to a current node, then the balance indicators of all ancestor nodes of this node remain unchanged, since the longest path in the subtree remains unchanged. When condition 2 applies to a current node, then the balance indicators of the ancestors will change. If condition 3 applies to a current node, then the tree has become unbalanced and this node has become critical.

In the case of rebalancing a tree when a critical node has been encountered, there are 2 broad cases which can arise, each of which can be further subdivided into 2 essentially similar subcases. A general representation of case 1 is given in the following figure, where the rectangles labeled T1, T2, and T3 represent trees and the node labeled NEW denotes the node being inserted.

The expression at the bottom of each rectangle denotes the maximum path length in that tree after insertions. For example in figure (a) since node X is critical, the node Y must have been balanced prior to insertion. This case covers the situation when Y has become heavy in the same direction that X was heavy. A concrete example of the second possibility for case 1 is exemplified in figure (c). The PATH and DIRECTION vectors are defined in the next algorithm.

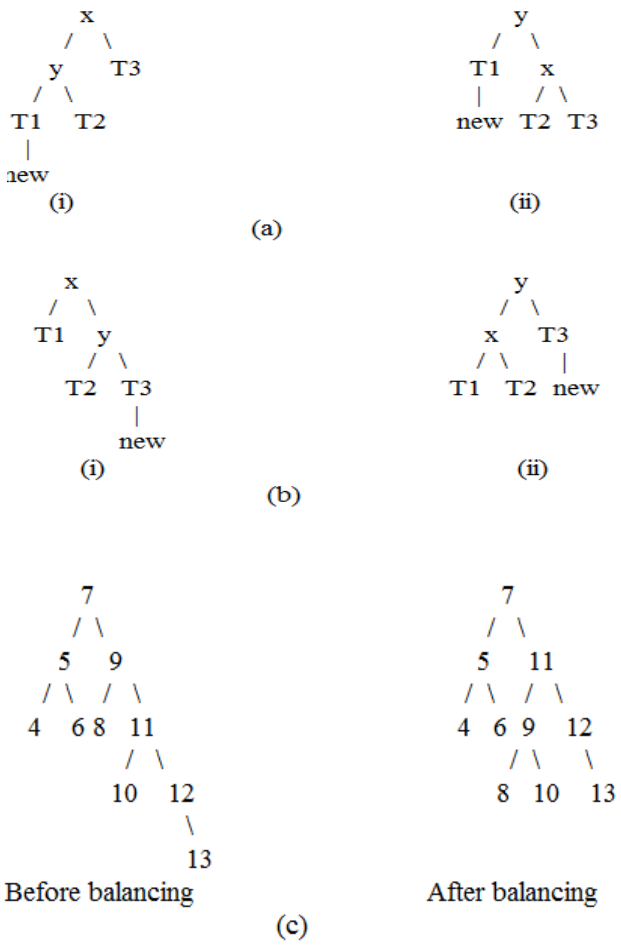
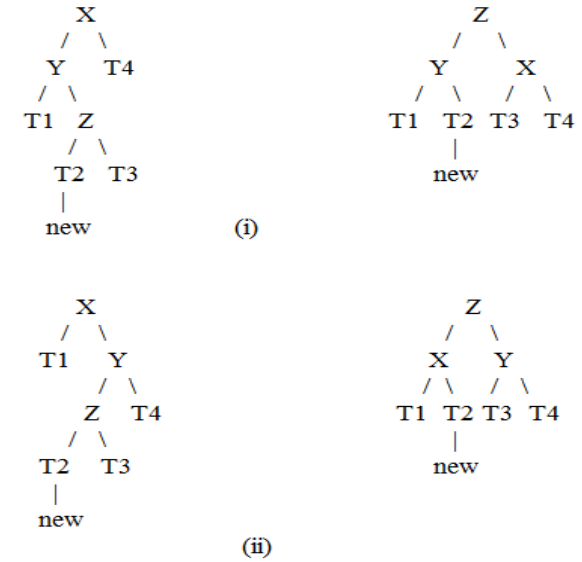
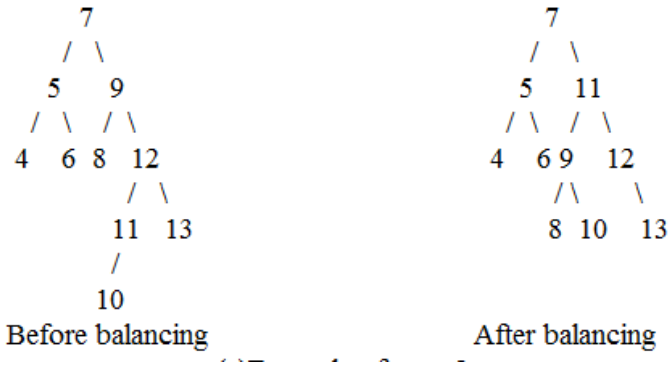


Figure 4.44: Case of rebalancing a tree.

The second case, which is given in figure (d), Y, becomes heavy in an opposite direction to that in which X was heavy. It is clear that node Z must have been balanced prior to insertion. A specific example of case 2b is given in figure (e). Again PATH and DIRECTION refer to vectors that are associated with the next algorithm.



(d) Case 2 for rebalancing a tree



(e)Example of case 2

Figure 4.45: Case of rebalancing a tree.

The node structure of a tree will consist of a left pointer (LPTR), a right pointer (RPTR), a key field (K), a balance indicator (BI), and an information field (DATA). The name of the node structure is NODE. A list head for the tree is assumed with its left pointer containing the address of the root of the actual tree. A general algorithm for inserting a node into a height balanced tree is as follows:-

1. If this is the first insertion then allocate a node, set its fields and exit
2. If the name is not already in the tree then attach the new node to the existing tree else write item already present and exit
3. Search for an unbalanced node
4. Adjust the balance indicators, if there is no critical node, exit
5. If the node was balanced and then becomes heavy or the node was heavy and becomes balanced then adjust the balance indicators and exit
6. Rebalance the tree and exit.

Lab Exercise

1. Write a C program to represent the linked list representation of binary tree into sequential representation.
2. Write a recursive function to add the first 'n' natural numbers. [Hint: (addition) = 1 + 2 + … + n]].

Case Study: Binary Tree Branches Out Big with Microsoft

New York-based Binary Tree started in 1993 helping businesses migrate to Lotus Notes. Ten years ago, its customers started requesting solutions for migrating to Microsoft. Driven by market demand and supported by a solid Microsoft product offering and partner ecosystem, Binary Tree evolved into a platform agnostic software company and developed new solutions for providing migrations to Microsoft. Today, with thousands of customers and millions of users migrated, Binary Trees decision has yielded impressive returns: More than half of its annual revenue comes from Microsoft platform migrations.

Situation: A Turning Tide of Customers When Binary Tree opened its doors in 1993, it helped companies migrate to Lotus Notes. By 1999, Binary Tree was noticing a new trend in the migration market – an increase in demand for migrations to Microsoft. Some of Binary Trees former customers who originally migrated to Lotus Notes were now pounding on Binary Trees doors to demand a complete overhaul to the Microsoft solution. "Those requests kept escalating until they reached a point where it was obvious that this was a big opportunity for us," said Steven Pivnik, CEO of Binary Tree. We have learned to listen to our customers and

there are a number of reasons why they want to migrate to the Microsoft platform. Many of them cite that the Microsoft platform is maturing substantially and that they are looking for reduced costs, better productivity, and lower costs of ownership.

4.12 Summary

- A tree structure is a way of presenting the hierarchical nature of a structure in a graphical form.
- The two ways to represent trees are linked representation and array representation.
- To evaluate an expression tree, we recursively evaluate the left and right sub-trees and then apply the operator at the root.
- The three types of graphs are directed graphs, undirected graphs, and mixed graphs.
- A binary tree is a finite set of data elements and each node contains a maximum of two branches.
- A binary tree is called a strictly binary tree when the non-terminal nodes have exactly two child nodes forming left and the right sub-trees.
- In a threaded binary tree, the pointers are represented as threads such that the threads points to the node in the binary tree for any operations.

4.13 Keywords

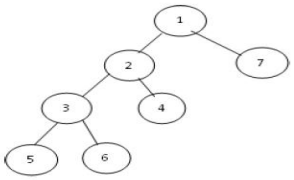
Converse inorder traversal: The tree obtained after the inorder traversal is represented in the original tree format.
Infinite loop: A series of instructions in a computer program which, on execution, result in a cyclic repetition of the same instructions.
Leaf nodes: The nodes without any successors.
Node predecessor: Node representing the parent node.
Node successor: Node representing the child node.
Ordered list: An ordered list is one which is maintained in some predefined order, such as alphabetical or numerical order.
Threads: It is the pointer linking the other nodes in the tree.
Traversing flag: Indicates if the node was visited during traversal.

4.14 Review Questions

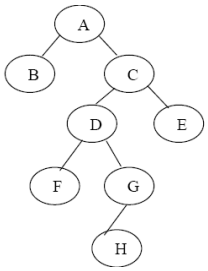
1. "Choosing the best data structure for a program is a challenging task". Discuss the sentence.
2. "Every 3-node in a 2-3 tree must have specific properties". Discuss with diagram and example.
3. What is a binary tree? Construct a tree with the given string notation (T1 (T2 (T3 (T4), T5, T6), T7 (T8, (T9 (T10))))).
4. Construct a tree structure for the following array representation.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
T1	T2	.	T3	.	.	.	T4	.	.	.	.	.	.	.	T5

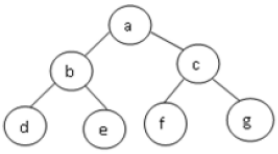
5. Write a procedure for linked list representation of binary tree. Consider the array representation provided in question 2 for constructing a linked representation.
6. Represent the following binary tree in an array.



7. "In threaded binary trees, threads are used instead of pointers." Justify with example.
8. In the binary tree given, delete node D and insert node I.



9. Perform an inorder traversal for the given binary tree.



10. "Binary search trees have more advantages when compared to other data structures". Justify the term.

Answers for Self Assessment Questions

1. (b)	2. (c)	3. (b)	4. (c)
5. (b)	6. (b)	7. (a)	8.(a)

4.15 Further Studies

1. http://www.i-programmer.info/babbages-bag/477-trees.html
2. http://people.cis.ksu.edu/~schmidt/300s05/Lectures/Week7b.html

5
Graph Structure

STRUCTURE

5.0 Objectives

5.1 Introduction

5.2 Undirected Graph

5.3 Directed Graph or Digraph

5.4 Adjacency Matrix

5.5 Graph Traversals

5.6 Shortest Path and all Pairs of Shortest Paths

5.7 Summary

5.8 Keywords

5.9 Review Questions

5.10 Further Studies

5.0 Objectives

After studying this chapter, you will be able to:

- Define the graphs
- Discuss the undirected and directed graph or digraph
- Explain the graph representation
- Discuss the breadth first traversal and depth first traversal
- Explain the adjacency matrix and adjacency list
- Explain the orthogonal representation of graph
- Discuss the adjacency multilist representation
- Discuss the graph traversals
- Explain the transitive closure

5.1 Introduction

In computer science, a graph is an abstract data structure that is meant to implement the graph concept from mathematics.

A graph data structure consists mainly of a finite (and possibly mutable) set of ordered pairs, called edges or arcs, of certain entities called nodes or vertices. As in mathematics, an edge (x, y) is said to point or go from x to y. The nodes may be part of the graph structure, or may be external entities represented by integer indices or references.

A graph data structure may also associate to each edge some edge value, such as a symbolic label or a numeric attribute (cost, capacity, length, etc.).

It is often useful to bound the running time of graph algorithms. Unlike most other computational problems, for a graph G = (V, E) there are two relevant parameters describing the size of the input the number |V| of vertices in the graph and the number |E| of edges in the graph. Inside asymptotic notation (and only there), it is common to use the symbols V and E, when someone really means |V| and |E|. We adopt this convention here to simplify asymptotic functions and make them easily readable. The symbols V and E are never used inside asymptotic notation with their literal meaning, so this abuse of notation does not risk ambiguity. For example O(E + VlogV) means O((E,V) |E| + |V| log |V| for a suitable metric of graphs. Another common convention-referring to the values |V| and |E| by the names n and m, respectively-sidesteps this ambiguity.

5.2 Undirected Graph

An undirected graph is a graph in which each edge has no direction. An undirected graph G = (V, E) consists of a finite set of vertices V and a set of edges E. It differs from a directed graph in that each edge in E is an unordered pair of vertices. If (v, w) is an undirected edge, then (v, w) = (w, v). Hereafter we refer to an undirected graph as simply a graph. Graphs are used in many different disciplines to model a symmetric relationship between objects. The objects are represented by the vertices of the graph and two objects are connected by an edge if the objects are related. In this chapter we present several data structures that can be used to represent graphs. We then present algorithms for three typical problems involving undirected graphs: constructing minimal spanning trees, biconnected components, and maximal matchings.

Much of the terminology for directed graphs is also applicable to undirected graphs. For example, vertices v and w are adjacent if (v, w) is an edge [or, equivalently, if (w, v) is an edge]. We say the edge (v, w) is incident upon vertices v and w. A path is a sequence of vertices $v_1, v_2, \ldots,$ vn such that (v_i, v_{i+1}) is an edge for 1 £ i < n. A path is simple if all vertices on the path are distinct, with the exception that v1 and vn may be the same. The length of the path is n-1, the number of edges along the path. We say the path $v_1, v_2, \ldots,$ vn connects v1 and v_n. A graph is connected if every pair of its vertices is connected. Let G = (V, E) be a graph with vertex set V and edge set E. A subgraph of G is a graph G' = (V', E') where

1. V' is a subset of V.
2. E' consists of edges (v, w) in E such that both v and w are in V'

If E' consists of all edges (v, w) in E, such that both v and w are in V', then G' is called an induced subgraph of G.

Example: In Figure 5.1 (a) graph G = (V, E) with V = {a, b, c, d} and E ={(a, b), (a, d), (b, c), (b, d), (c, d)}. In Figure 5.1(b) one of its induced subgraphs,the one defined by the set of vertices {a, b, c} and all the edges in Figure. 5.1(a) that are not incident upon vertex d.

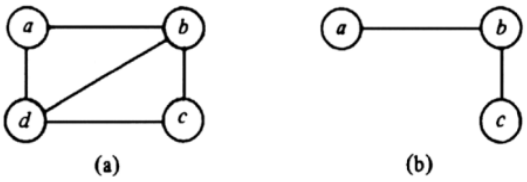


Figure 5.1: A graph and one of its subgraphs.

A connected component of a graph G is a maximal connected induced subgraph, that is, a connected induced subgraph that is not itself a proper subgraph of any other connected subgraph of G.

Example: Figure 5.1 is a connected graph. It has only one connected component, namely itself. Figure 5.2 is a graph with two connected components.

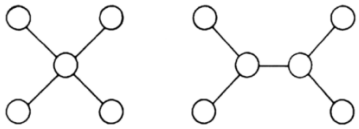


Figure 5.2: An unconnected graph.

A (simple) cycle in a graph is a (simple) path of length three or more that connects a vertex to itself. We do not consider paths of the form v (path of length 0), v, v (path of length 1), or v, w, v (path of length 2) to be cycles. A graph is cyclic if it contains at least one cycle. A connected, acyclic graph is sometimes called a free tree. Figure 5.2 shows a graph consisting of two connected components where each connected component is a free tree. A free tree can be made into an ordinary tree if we pick any vertex we wish as the root and orient each edge from the root.

1. Every free tree with n^3–1 vertices contains exactly n-1 edges.
2. If we add any edge to a free tree, we get a cycle.

We can prove (1) by induction on n, or what is equivalent, by an argument concerning the "smallest counterexample." Suppose G = (V, E) is a counterexample to (1) with the fewest vertices, say n vertices. Now n cannot be 1, because the only free tree on one vertex has zero edges, and (1) is satisfied. Therefore, n must be greater than 1.
Now claim that in the free tree there must be some vertex with exactly one incident edge. In proof, no vertex can have zero incident edges, or G would not be connected. Suppose every vertex has at least two edges incident. Then, start at some vertex v1, and follow any edge from v_1. At each step, leave a vertex by a different edge from the one used to enter it, thereby forming a path v_1, v_2, v_3, ...
Since there are only a finite number of vertices in V, all vertices on this path cannot be distinct; eventually, we find $v_i = v_j$ for some $i < j$. We cannot have i = j-1 because there are no loops from a vertex to itself, and we cannot have i = j-2 or else we entered and left vertex v_{i+1} on the same edge. Thus, i £ j-3, and we have a cycle v_i, v_{i+1}, . . . , $v_j = v_i$. Thus, we have contradicted the hypothesis that G had no vertex with only one edge incident, and therefore conclude that such a vertex v with edge (v, w) exists. Now consider the graph G' formed by deleting vertex v and edge (v, w) from G. G' cannot contradict (1), because if it did, it would be a smaller counterexample than G. Therefore, G' has n-1 vertices and n-2 edges. But G has one more edge and one more vertex than G', so G has n-1 edges, proving that G does indeed satisfy (1). Since there is no smallest counterexample to (1), we conclude there can be no counterexample at all, so (1) is true. Now easily prove statement (2), that adding an edge to a free tree forms a cycle. If not, the result of adding the edge to a free tree of n vertices would be a graph with n vertices and n edges. This graph would still be connected, and we supposed that adding the edge left the graph acyclic. Thus we would have a free tree whose vertex and edge count did not satisfy condition (1).

5.3 Directed Graph or Digraph

In problems arising in computer science, mathematics, engineering, and many other disciplines we often need to represent arbitrary relationships among data objects. Directed and undirected graphs are natural models of

such relationships. This chapter presents the basic data structures that can be used to represent directed graphs. Some basic algorithms for determining the connectivity of directed graphs and for finding shortest paths are also presented.

A directed graph (digraph for short) G consists of a set of vertices V and a set of arcs E. The vertices are also called nodes or points; the arcs could be called directed edges or directed lines. An arc is an ordered pair of vertices (v, w); v is called the tail and w the head of the arc. The arc (v, w) is often expressed by v –> w and drawn as.

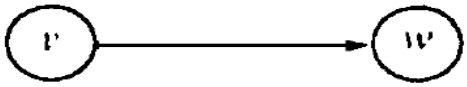


Figure 5.3: shows a digraph with four vertices and five arcs.

Notice that the "arrowhead" is at the vertex called the "head" and the tail of the arrow is at the vertex called the "tail." We say that arc v –> w is from v to w, and that w is adjacent to v.

Example:

The vertices of a digraph can be used to represent objects, and the arcs relationships between the objects. For example, the vertices might represent cities and the arcs airplane flights from one city to another. As another example, which we introduced in a digraph can be used to represent the flow of control in a computer program. The vertices represent basic blocks and the arcs possible transfers of flow of control.

A path in a digraph is a sequence of vertices v1, v2, . . . , vn, such that v_1 –> v_2, v_2 –> v_3, . . . , v_{n-1} –> v_n are arcs. This path is from vertex v_1 to vertex v_n, and passes through vertices v_2, v_3, . . . , v_{n-1}, and ends at vertex v_n. The length of a path is the number of arcs on the path, in this case, n-1. As a special case, a single vertex v by itself denotes a path of length zero from v to v. In Figure 5.4 the sequence 1, 2, 4 is a path of length 2 from vertex 1 to vertex 4.

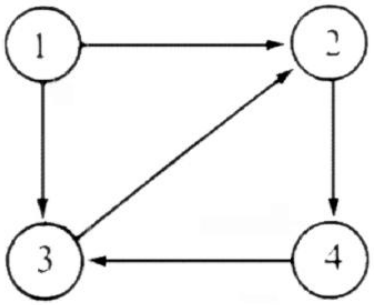


Figure 5.4: Directed graph.

A path is simple if all vertices on the path except possibly the first and last, are distinct. A simple cycle is a simple path of length at least one that begins and ends at the same vertex. In Figure 5.4, the path 3, 2, 4, 3 is a cycle of length three. In many applications it is useful to attach information to the vertices and arcs of a digraph. For this purpose we can use a labelled digraph, a digraph in which each arc and/or each vertex can have an associated label. A label can be a name, a cost, or a value of any given data type.

5.3.1 Representations for Directed Graphs

Several data structures can be used to represent a directed graph. The appropriate choice of data structure depends on the operations that will be applied to the vertices and arcs of the digraph. One common representation for a digraph G = (V, E) is the adjacency matrix. Suppose V = {1, 2 , . . . , n}. The adjacency matrix for G is an n x n matrix A of Booleans, where A[i, j] is true if and only if there is an arc from vertex i to

j. Often, we shall exhibit adjacency matrices with 1 for true and 0 for false; adjacency matrices may even be implemented that way. In the adjacency matrix representation the time required to access an element of an adjacency matrix is independent of the size of V and E. Thus the adjacency matrix representation is useful in those graph algorithms in which we frequently need to know whether a given arc is present.

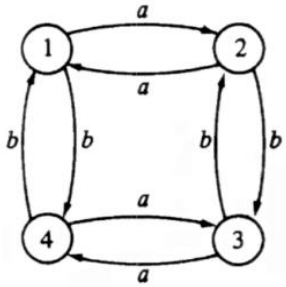


Figure 5.5: Transition digraph.

Closely related is the labelled adjacency matrix representation of a digraph, where A[i, j] is the label on the arc going from vertex i to vertex j. If there is no arc from i to j, then a value that cannot be a legitimate label must be used as the entry for A[i, j].

Did You Know?

The first textbook on graph theory was written by Dénes Kőnig, and published in 1936.

5.4 Adjacency Matrix

The adjacency matrix, sometimes also called the connection matrix, of a simple graph is a matrix with rows and columns labelled by graph vertices, with a 1 or 0 in position (v_i,v_j) according to whether vi and v_j are adjacent or not. For an undirected graph, the adjacency matrix is symmetric.

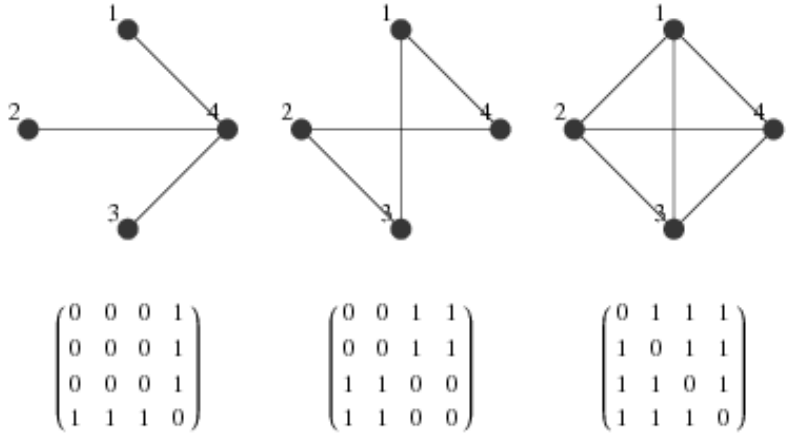


Figure 5.6: adjacency matrix.

Caution

For a simple graph with no self-loops, the adjacency matrix must have 0s on the diagonal.

Self Assessment Questions

1. If an edge has identical end points, it is called a…….

(a). degree (b). vertex (c). loop (d). path.

2. A graph is said to be…….if there is a path between any two of its nodes
(a). connected (b). complete (c). alanced (d). binary.

3. A graph g is……if every node u in G adjacent to every node v in G
(a). spanning (b). complete (c). balanced (d). binary.

4. The graph G is said to be……..if each edge in the graph is assigned a non negative numerical value called the weight or length of the edge
(a). complete (b). weighted (c). balanced (d). tree.

5. ……..is a common method used to store a graph.
(a). Stack (b). Adjacency matrix (c). Diagraph (d). List.

5.4.1 Adjacency List

The adjacency list representation of a graph consists of n lists one for each vertex v_i, , which gives the vertices to which vi is adjacent. The adjacency lists of a graph g may be computed using ToAdjacencyLists[g] in the mathematica package combinatorica. A graph may be constructed from adjacency lists using from adjacency lists

5.4.2 Adjacency Multilist Representation

In computer science, an adjacency multilist is a list associated with a graph data structure and used as a representation of the graph data structure. This is a modified version of the adjacency lists associated with graphs. In an adjacency list representation of a graph, we keep, for each vertex in the graph, a list of all other vertices which it has an edge to (that vertex's "adjacency list"). For example in one such representation we keep an array indexed by vertex numbers the array elements being pointers to linked lists of the neighbors of the corresponding vertices. An edge (v_i, v_j) will be represented by two entries: one on the list for vi and the other on the list for v_j. In adjacency multilists, adjacency lists are maintained as multilists, that is, as lists in which nodes are shared among several lists.
Let v_i and v_j be two vertices of an undirected graph. Then associated with the edge (v_i, v_j) joining v_i and v_j we define a node having the following structure.

M	v_i	v_j	LINK i for vi	LINK j for v_j

Here

1. M = One bit mark field to be used to indicate whether or not the edge has been examined.
2. v_i = Vertex in graph such that there is an edge joining v_i to v_j.
3. vj = Vertex in graph such that there is an edge joining v_i to v_j.
4. LINK i for v_i = Link to some other node representing an edge incident to v_i.
5. LINK j for v_i = Link to some other node reprenting an edge incident to v_j.

5.4.3 Orthogonal Representation of Graph

In orthogonal representations were used to prove that the complement of a tree has positive semi definite minimum rank at most 3. A complement of a tree can be constructed by adding one vertex at a time, with each new vertex adjacent to all but one of the prior vertices. In this section extend this technique to certain very dense graphs constructed by adding vertices adjacent to all but one or two prior vertices. These results will be

used in the next section to study complements of certain sparse graphs and the relationship between mr(G) and mr($\bar{G}$).

Theorm

Let Y = (V_Y ,E_Y) be a graph of order at least two such that there is an orthogonal representation in $\mathbb{R}^d, d \geq 3$ *satisfying*

$\vec{v} \notin \mathrm{Span}(\vec{u})$ for each pair of distinct vertices u, v in V_Y .

Let X be a graph that can be constructed by starting with Y and adding one vertex at a time, such that the newly added vertex is adjacent to all prior vertices except at most one vertex. Then there is d-dimensional orthogonal representation of X satisfying

(2); *in particular,* $\mathrm{mr}(X) \leq \mathrm{mr}_+(X) \leq d.$

Proof.

Let $V_Y = \{v_1, \ldots, v_k\}$. Let X be constructed from Y by adding vertices $v_{k+1}, \ldots, v_n$ such that for m > k, vm is adjacent to all but at most one of v1, . . . , vm−1. Assuming that an orthogonal representation of $X[v_1, \ldots, v_{m-1}]$ in Rd has been constructed satisfying (2), we show there is an orthogonal representation of $X[v_1, \ldots, v_m]$ in Rd satisfying (2). If v_m is adjacent to v1, . . . , v_{m-1} then choose as $\vec{v}_m$ any vector in

$$\mathbb{R}^d \text{ that is not in } \left(\bigcup_{i=1}^{m-1} \vec{v}_i^{\perp}\right) \cup \left(\bigcup_{i=1}^{m-1} \mathrm{Span}(\vec{v}_i)\right).$$

Otherwise, let v_s be the only vertex of $X[v_1, \ldots, v_{m-1}]$ not adjacent to v_m in $X[v_1, \ldots, v_m]$. We want to choose a vector $\vec{v}_m$ such that

$$\begin{aligned} \vec{v}_m &\in \vec{v}_s^{\perp} \\ \vec{v}_m &\notin \vec{v}_i^{\perp} \qquad \forall i \neq s, i < m \\ \vec{v}_m &\notin \mathrm{Span}(\vec{v}_i) \qquad \forall i < m. \end{aligned}$$

$$W = \vec{v}_s^{\perp} \text{ and subspaces}$$

$$\begin{aligned} A_i &= W \cap \vec{v}_i^{\perp} \qquad i \neq s, i < m \\ B_i &= W \cap \mathrm{Span}(\vec{v}_i) \qquad i \neq s, i < m \end{aligned}$$

We can conclude the desired vector exists, since clearly none of the subspaces A_i, B_i is equal to W. Thus we have constructed an orthogonal representation of X in R^d such that $\vec{u}$, *and* $\vec{v}$ are linearly independent for any distinct vertices u, v of X.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: What is Adjacency matrix?

……..……

……..……

……...

Ex2: What is Directed graph?

……..……

……..……

……...

5.5 Graph Traversals

5.5.1 Breath-First search

Another systematic way of visiting the vertices is called breadth-first search. The approach is called "breadth-first" because from each vertex v that we visit we search as broadly as possible by next visiting all the vertices adjacent to v. We can also apply this strategy of search to directed graphs.

As for depth-first search, we can build a spanning forest when we perform a breadth-first search. In this case, we consider edge (x, y) a tree edge if vertex y is first visited from vertex x in the inner loop of the search procedure BFS.

It turns out that for the breadth-first search of an undirected graph, every edge that is not a tree edge is a cross edge, that is, it connects two vertices neither of which is an ancestor of the other. The breadth-first search algorithm given in algorithm inserts the tree edges into a set T, which we assume is initially empty. Every entry in the array mark is assumed to be initialized to the value unvisited; algorithm works on one connected component. If the graph is not connected, BFS must be called on a vertex of each component. Note that in a breadth-first search we must mark a vertex visited before enqueuing it, to avoid placing it on the queue more than once.

Example: The breadth-first spanning tree for the graph G is shown in algorithm. We assume the search began at vertex a. As before, we have shown tree edges solid and other edges dashed. We have also drawn the tree with the root at the top and the children in the left-to-right order in which they were first visited.

Breadth-First Algorithm

The time complexity of breadth-first search is the same as that of depth-procedure first search.

```
bfs ( v );
{ bfs visits all vertices connected to v using breadth-first search }
var
Q: QUEUE of vertex;
x, y: vertex;
begin
mark[v] := visited;
ENQUEUE(v, Q);
while not EMPTY(Q) do begin
x := FRONT(Q);
DEQUEUE(Q);
for each vertex y adjacent to x do
if mark[y] = unvisited then begin
mark[y] := visited;
ENQUEUE(y, Q);
```

```
INSERT((x, y), T)
end
end
end; { bfs }
```

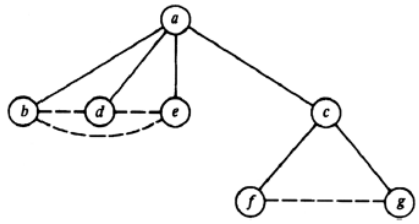


Figure 5.7: Breadth-first search of G.

Each vertex visited is placed in the queue once, so the body of the while loop is executed once for each vertex. Each edge (x, y) is examined twice, once from x and once from y. Thus, if a graph has n vertices and e edges, the running time of BFS is O(max(n, e)) if we use an adjacency list representation for the edges. Since e^3 n is typical, we shall usually refer to the running time of breadth-first search as O(e), just a we did for depth-first search. Depth-first search and breadth-first search can be used as frameworks around which to design efficient graph algorithms. For example, either method can be used to find the connected components of a graph, since the connected components are the trees of either spanning forest. We can test for cycles using breadth-first search in O(n) time, where n is the number of vertices, independent of the number of edges. As we discussed in any graph with n vertices and n or more edges must have a cycle. However, a graph could have n-1 or fewer edges and still have a cycle, if it had two or more connected components. One sure way to find the cycles is to build a breadth-first spanning forest. Then, every cross edge (v, w) must complete a simple cycle with the tree edges leading to v and w from their closest common ancestor, as shown in Figure 5.8.

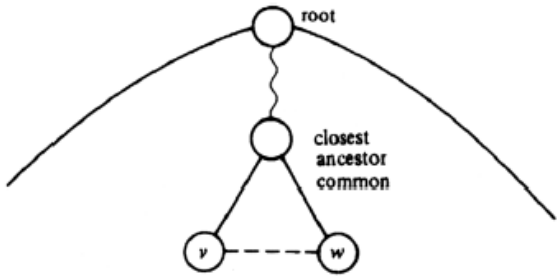


Figure 5.8: A cycle found by breadth-first search.

5.5.2 Depth-First Search

All the calls to DFS in the depth-first search of a graph with e arcs and n £ e vertices take O(e) time. To see why, observe that on no vertex is dfs called more than once, because as soon as we call dfs(v) we set mark[v] to visited at line (1), and we never call dfs on a vertex that previously had its mark set to visited. Thus, the total time spent at lines (2)-(3) going down the adjacency lists is proportional to the sum of the Shortest Path, all Pairs of Shortest Paths Suppose we have a labelled digraph that gives the flying time on certain routes connecting cities, and we wish to construct a table that gives the shortest time required to fly from any one city to any other. We now have an instance of the all pairs shortest paths (APSP) problem. To state the problem precisely, we are given a directed graph G = (V, E) in which each arc v –> w has a non-negative cost C[v, w]. The APSP lengths of those lists, that is, $O(e)$. Thus, assuming $n \leq e$, the total time spent on the depth- first

search of an entire graph is $O(e)$, which is, to within a constant factor, the time needed merely to "look at" each arc.

Alorithm Depth-First Search

```
procedure dfs ( v: vertex );
var
w: vertex;
begin
(1) mark[v]: = visited;
(2) for each vertex w on L[v] do
(3) if mark[w] = unvisited then
(4) dfs(w)
end; { dfs }
```

Example:

Assume the procedure dfs(v) is applied to the directed graph, with v = A. The algorithm marks A visited and selects vertex B from the adjacency list of vertex A. Since B is unvisited, the search continues by calling dfs(B). The algorithm now marks B visited and selects the first vertex from the adjacency list for vertex B. Depending on the order of the vertices on the adjacency list of B the search will go to C or D next.

Assuming that C appears ahead of D, dfs(C) is invoked. Vertex A is on the adjacency list of C. However, A is already visited at this point so the search remains at C.

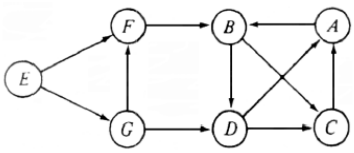


Figure 5.9: Directed graph.

Since all vertices on the adjacency list at C have now been exhausted, the search returns to B, from which the search proceeds to D. Vertices A and C on the adjacency list of D were already visited, so the search returns to B and then to A.

At this point the original call of dfs(A) is complete. However, the digraph has not been entirely searched; vertices E, F and G are still unvisited. To complete the search, we can call dfs(E).

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Write the algorithm of breath-first search and depth-first search.

……..……

……..……

……...

5.6 Shortest Path and all Pairs of Shortest Paths

Suppose we have a labelled digraph that gives the flying time on certain routes connecting cities, and we wish to construct a table that gives the shortest time required to fly from any one city to any other. We now have an

instance of the all pairs shortest paths (APSP) problem. To state the problem precisely, we are given a directed graph G = (V, E) in which each arc v –>w has a non-negative cost C[v, w].

The APSP problem is to find for each ordered pair of vertices (v, w) the smallest length of any path from v to w. We could solve this problem using Dijkstra's algorithm with each vertex in turn as the source. A more direct way of solving the problem is to use the following algorithm due to R. W. Floyd. For convenience, let us again assume the vertices in V are numbered 1, 2 , . . . , n. Floyd's algorithm uses an n x n matrix A in which to compute the lengths of the shortest paths. We initially set A[i, j] = C[i, j] for all i ¹ j. If there is no arc from i to j, we assume C[i, j] = ¥. Each diagonal element is set to 0. We then make n iterations over the A matrix. After the kth iteration, A[i, j] will have for its value the smallest length of any path from vertex i to vertex j that does not pass through a vertex numbered higher than k. That is to say, i and j, the end vertices on the path, may be any vertex, but any intermediate vertex on the path must be less than or equal to k.
In the kth iteration we use the following formula to compute A.

$$A_k[i, j] = \min \begin{cases} A_{k-1}[i, j] \\ A_{k-1}[i, k] + A_{k-1}[k, j] \end{cases}$$

The subscript k denotes the value of the A matrix after the kth iteration, and it should not be assumed that there are n different matrices. We shall eliminate these subscripts shortly. This formula has the simple interpretation shown in Figure.5.10.

To compute A_k[i, j] we compare A_{k-1}[i, j], the cost of going from i to j without going through k or any higher-numbered vertex, with A_{k-1}[i, k] + A_{k-1}[k, j], the cost of going first from i to k and then from k to j, without passing through a vertex numbered higher than k. If passing through vertex k produces a cheaper path than what we had for A_{k-1}[i, j], then we choose that cheaper cost for A_k[i, j].

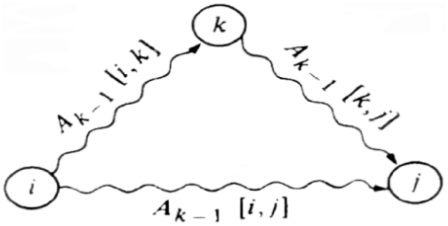


Figure 5.10: Including k among the vertices to go from i to j

Example: Consider the weighted digraph shown in Figure 5.10 The values of the a matrix initially and after the three iterations are shown in Figure 5.11.

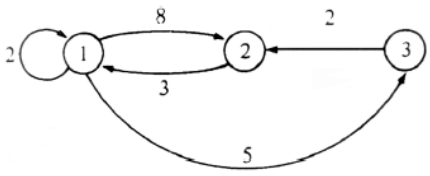


Figure 5.11: Weighted digraph.

Since A_k[i, k] = Ak-1[i, k] and A_k[k, j] = A_{k-1}[k, j], no entry with either subscript equal to k changes during the k^{th} iteration. Therefore, we can perform the computation with only one copy of the matrix. A program to perform this computation on n x n matrices The running time of this program is clearly $O(n_3)$, since the program is basically nothing more than a triply nested for-loop. To verify that this program works, it is easy to

prove by induction on k that after k passes through the triple for-loop, A[i, j] holds the length of the shortest path from vertex i to vertex j that does not pass through a vertex numbered higher than k.

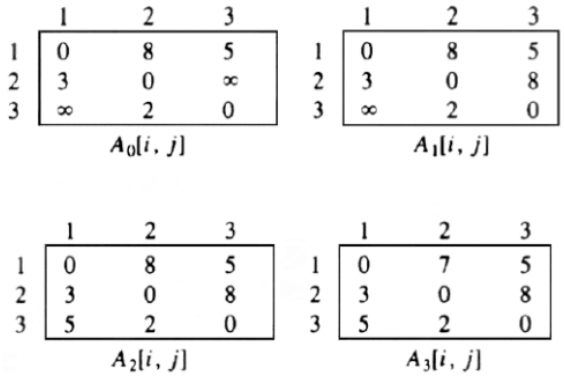


Figure 5.12: Values of successive a matrices.

Algorithm for Weighted digraph

```
procedure Floyd ( var A: array[1..n, 1..n] of real;
C: array[1..n, 1..n] of real );
{ Floyd computes shortest path matrix A given arc cost matrix C }
var
i, j, k: integer;
begin
for i := 1 to n do
for j := 1 to n do
A[i, j] := C[i, j];
for i:= 1 to n do
A[i, i] := 0;
for k:= 1 to n do
for i := 1 to n do
for j:= 1 to n do
if A[i, k] + A[k, j] < A
[i, j] then
A[i, j] := A[i, k] + A[k, j]
end; { Floyd }
```

Since the adjacency-matrix version of Dijkstra finds shortest paths from one vertex in O(n2) time, it, like Floyd's algorithm, can find all shortest paths in O(n3) time. The compiler, machine, and implementation details will determine the constants of proportionality. Experimentation and measurement are the easiest way to ascertain the best algorithm for the application at hand. If e, the number of edges, is very much less than n2, then despite the relatively low constant factor in the O(n3) running time of Floyd, we would expect the adjacency list version of Dijkstra, taking O(ne logn) time to solve the APSP, to be superior, at least for large sparse graphs.

5.6.1 Recovering the Paths

In many situations we may want to print out the cheapest path from one vertex to another. One way to accomplish this is to use another matrix P, where P[i, j] holds that vertex k that led Floyd to find the smallest

value of A[i, j]. If P[i, j]=0, then the shortest path from i to j is direct, following the arc from i to j. The modified version of Floyd in stores the appropriate intermediate vertices into P.

```
procedure shortest ( var A: array[1..n, 1..n] of real;
C: array[1..n, 1..n] of real; P:
array[1..n, 1..n] of integer );
{ shortest takes an n X n matrix C of arc costs and produces an n X n matrix A of lengths of shortest paths and an n X n matrix P giving a point in the "middle" of each shortest path }
var
i, j, k: integer;
begin
for i:= 1 to n do
for j := 1 to n do begin
A[i, j] := C[i, j];
P[i, j] := 0
end;
for i:= 1 to n do
A[i, i] := 0;
for k := 1 to n do
for i:= 1 to n do
for j:= 1 to n do
if A[i, k] + A[k, j] < A[i, j] then begin
A[i, j] := A[i, k] + A[k, j];
P[i, j] := k
end
end; { shortest }
```

To print out the intermediate vertices on the shortest path from vertex i to vertex j, we invoke the procedure path (i, j) where path is given in While on an arbitrary matrix P, path could loop forever, if P comes from shortest, we could not, say, have k on the shortest path from i to j and also have j on the shortest path from i to k. Note how our assumption of nonnegative weights is again crucial.

```
procedure path ( i, j: integer );
var
k: integer;
begin
k := P[i, j];
if k = 0 then
return;
path(i, k);
writeln(k);
path(k, j)
end; { path }
```

5.6.2 Transitive Closure

In some problems we may be interested in determining only whether there exists a path of length one or more from vertex i to vertex j. Floyd's algorithm can be specialized readily to this problem; the resulting algorithm, which predates Floyd's, is called Warshall's algorithm. Suppose our cost matrix C is just the adjacency matrix for the given digraph. That is, C[i, j] = 1 if there is an arc from i to j, and 0 otherwise. We wish to compute the

matrix A such that A[i, j] = 1 if there is a path of length one or more from i to j, and 0 otherwise. A is often called the transitive closure of the adjacency matrix.

Example:

Figure 5.13 shows the transitive closure for the adjacency matrix of the digraph.

The transitive closure can be computed using a procedure similar to Floyd by applying the following formula in the k^{th} pass over the Boolean A matrix.

$A_k[i, j] = A_{k-1}[i,j]$ or $(A_{k-1}[i, k]$ and $A_{k-1}[k, j])$

This formula states that there is a path from i to j not passing through a vertex numbered higher than k if

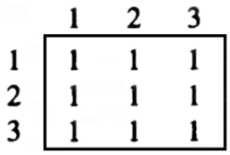


Figure 5.13: Transitive closure.

1. There is already a path from i to j not passing through a vertex numbered higher than k– 1 or
2. There is a path from i to k not passing through a vertex numbered higher than k – 1 and path from k to j not passing through a vertex numbered higher than k-1.

As before $A_k[i, k] = A_{k-1}[i, k]$ and $A_k[k, j] = A_{k-1}[k, j]$ so we can perform the computation with only one copy of the A matrix. The resulting Pascal program, named Warshall after its discoverer, is shown in algorithm.

Warshall's algorithm for transitive closure

```
procedure Warshall ( var A: array[1..n, 1..n] of boolean;
C: array[1..n, 1..n] of boolean );
{ Warshall makes A the transitive closure of C }
Var
i, j, k: integer;
begin
for i := 1 to n do
for j := 1 to n do
A[i, j] := C[i, j];
for k := 1 to n do
for i := 1 to n do
for j := 1 to n do
if A[i, j ] = false then A[i, j] := A[i, k] and A[k, j]
end; { Warshall }
```

5.6.3 Reflexive Transitive Closure

Reflexive-Transitive Closure (RTC) when used to persist hierarchical data, present many advantages over alternative persistence strategies such as path enumeration. Such advantages include, but are not limited to, referential integrity and a simple structure for data queries. RTC exponentially and present a concern with scaling both operational performance of the RTC and volume of the RTC data. Discovering these practical performance boundaries involves understanding the growth patterns of reflexive and transitive binary relations and observing sample data for large hierarchical models Reflexive-Transitive Closure (RTC) store binary relations that are both reflexive and transitive. RTC are well-suited for representing hierarchical data and, when implemented in relational data models, retain attractive qualities such as referential integrity and easy querying

Consider a set of data where elements are related to one another in a hierarchical structure, such as a family tree or corporate organizational structure. Each element is related to another via "parent-child" relationships. Choosing the family tree example, we know that a grandfather has one or more children who have one or more children, and so on.

Lab Exercise

1. Write a complete program for Dijkstra's algorithm using a partially ordered tree as a priority queue and linked adjacency lists.
2. Write a program to compute the transitive reduction of a digraph. What is the time complexity of your program?

Did You Know?

The graph theory written by Leonhard Euler on the Seven Bridges of Königsberg and published in 1736.

Case Study: Finding Strong Components- By performing two depth-first searches, we can test whether a directed graph is strongly connected, and if it is not, we can actually produce the subsets of vertices that are strongly connected to them. This can also be done in only one depth-first search, but the method used here is much simpler to understand.

First, a depth-first search is performed on the input graph G. The vertices of G are numbered by a post order traversal of the depth-first spanning forest, and then all edges in G are reversed, forming G_r. The graph Figure 1represents G_r for the graph G shown in Figure 2; the vertices are shown with their numbers.

The algorithm is completed by performing a depth-first search on G_r, always starting a new depth first search at the highest-numbered vertex. Thus, we begin the depth-first search of G_r at vertex G, which is numbered 10. This leads nowhere, so the next search is started at H. This call visits I and J. The next call starts at B and visits A, C, and F. The next calls after this are dfs(D) and finally dfs(E). The resulting depth-first spanning forest is shown in Figure3.

Each of the trees (this is easier to see if you completely ignore all non tree edges) in this depth-first spanning forest forms a strongly connected component. Thus, for our example, the strongly connected components are {G}, {H, I, J}, {B, A, C, F}, {D}, and {E}.

To see why this algorithm works, first note that if two vertices v and w are in the same strongly connected component, then there are paths from v to w and from w to v in the original graph G, and hence also in G_r.

Now, if two vertices v and w are not in the same depth-first spanning tree of G_r, clearly they cannot be in the same strongly connected component.

To prove that this algorithm works, we must show that if two vertices v and w are in the same depth-first spanning tree of G_r, there must be paths from v to w and from w to v. Equivalently, we can show that if x is the root of the depth-first spanning tree of G_r containing v, then there is a path from x to v and from v to x. Applying the same logic to w would then give a path from x to w and from w to x. These paths would imply paths from v to w and w to v (going through x).

Since v is a descendant of x in G_r's depth-first spanning tree, there is a path from x to v in G_r and thus a path from v to x in G. Furthermore, since x is the root, x has the higher postorder number from the first depth-first search. Therefore, during the first depth-first search, all the work processing v was completed before the work at x was completed. Since there is a path from v to x, it follows that v must be a descendant of x in the spanning tree for G otherwise v would finish after x. This implies a path from x to v in G and completes the proof.

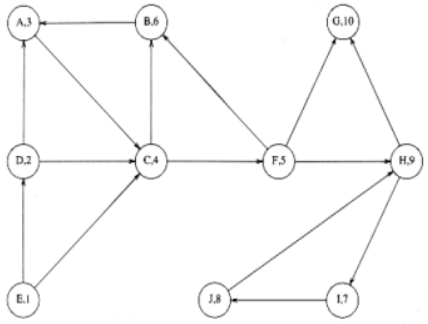


Figure 1: G_r numbered by postorder traversal of G

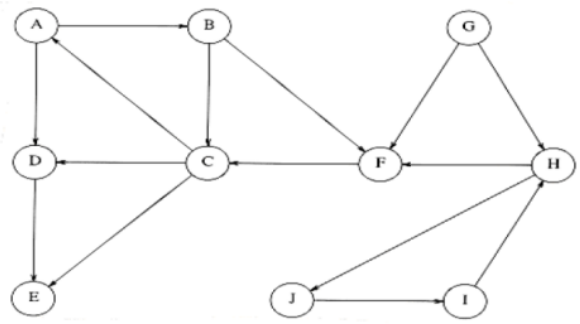


Figure 2: A directed graph

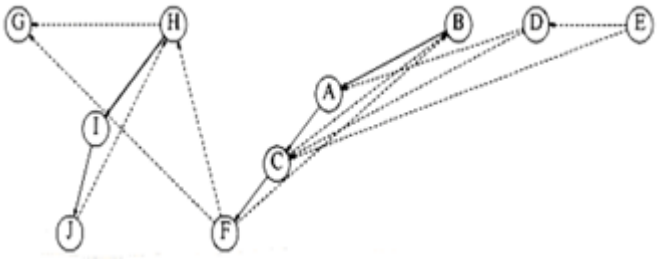


Figure 3: Depth-first search of G_r strong components are {G}, {H, I, J}, {B, A, C, F}, {D},{E}

Exercise: Check Your Progress 3
Note: i) Use the space below for your answer.
Ex1: Write the algorithm of BFS and DFS.?

……..……

……..……

……...

Ex2: What is transitive closer?

……..……

……..……

……...

5.7 Summary

- A path is simple if all vertices on the path, except possibly the first and last, are distinct.
- Adjacency matrix representation the time required to access an element of an adjacency matrix is independent of the size of V and E.
- Adjacency-matrix version of Dijkstra finds shortest paths from one vertex in $O(n_2)$ time, it, like Floyd's algorithm, all shortest paths in $O(n_3)$ time.
- The center of graph G is a vertex of minimum eccentricity. Thus, the centre of a digraph is a vertex that is closest to the vertex most distant from it.
- During a depth-first traversal of a directed graph, certain arcs, when traversed, lead to unvisited vertices. The arcs leading to new vertices are called tree arcs.

5.8 Keywords

Directed Graph (digraph for short): G consists of a set of vertices V and a set of arcs E. The vertices are also called nodes or points; the arcs could be called directed edges or directed lines.
Graph G: It is said to be complete if every node u in G is adjacent to every other node v in G.
Loops: An edge e is called a loop if it has identical endpoints, that is, if e=[u, u].
Mixed Graph: A mixed graph G contains both directed and undirected edges
Multiple Edges: Distinct edges e and e' are called-multiple edges if they connect the same endpoints, that is, if e=[u, v] and e= [u, v].

5.9 Review Questions

1. What is the graph representation?
2. Explain the undirected graph?
3. What are the digraph and representations for directed graphs?
4. Differentiate between adjacency matrix and adjacency list.
5. What is the adjacency multilist representation?
6. What is the orthogonal representation of graph?
7. Differentiate between breath-first search and depth-first search.
8. What are the all Pairs of Shortest Paths?
9. Write comparison between Floyd's and Dijkstra's algorithms.
10. What is the transitive and reflexive transitive closure?

Answers for Self Assessment Questions

1. (c) 2. (a) 3. (b) 4. (b) 5. (b)

5.10 Further Studies

1. http://en.wikibooks.org/wiki/Data_Structures/Graphs

6

Sorting and Searching

STRUCTURE

6.0 Objectives

6.1 Introduction

6.2 Bubble Sort

6.3 Selection Sort

6.4 Merge Sort

6.5 Quick Sort

6.6 Insertion Sort

6.7 Shell Sort

6.8 Address Calculation Sort

6.9 Radix Sort

6.10 Comparison of Sorting Methods

6.11 Hash Table

6.12 Linear Search (Sequential Search)

6.13 Binary Search

6.14 Searching an Ordered Table

6.15 Indexed Sequential Search

6.16 Interpolation Search

6.17 Summary

6.18 Keywords

6.19 Review Questions

6.20 Further Studies

6.0 Objectives

After studying this chapter, you will be able to:

- Define the bubble and selection sort
- Explain the merge and quick sort
- Explain the insertion and shell sort
- Discuss the address calculation and radix sort
- Explain the comparison of sorting methods

- Discuss the hash table and collision resolution techniques
- Explain the Linear Search (Sequential Search)
- Define the binary search
- Discuss the searching an ordered table

6.1 Introduction

Finding better algorithms to sort a given set of data is an ongoing problem in the field of computer science. Sorting is placing a given set of data in a particular order. Simple sorts place data in ascending or descending order. For discussion purposes, we will look at sorting data in ascending order. However, you may modify the code to sort the data in descending order by reversing the relational operators (i.e., change 'nums[j] < nums[j–1]' to 'nums[j] > nums[j–1]'). In this lesson we will analyze sorts of different efficiency, and discuss when and where they can be used. In order to simplify the explanation of certain algorithms, we will assume a swap( ) function exists that switches the values of two variables.

An example of such function for: int variables is void swap (int &item1,int &item2) //reference parameters, point directly to the storage location of the variables passed. Local copies are not made, and these values are saved after the function life span ends.

6.2 Bubble Sort

Bubble sort is a simple and well-known sorting algorithm. It is used in practice once in a blue moon and its main application is to make an introduction to the sorting algorithms. Bubble sort belongs to $O(n^2)$ sorting algorithms, which makes it quite inefficient for sorting large data volumes. Bubble sort is stable and adaptive.

Sequential Bubble Sort

```
for i ← 1 to length [A] do
for j ← length [A] downto i +1
do
        If A[A] < A[j–1] then
        Exchange A[j] ↔ A[j–1]
```

Implementation

```
void bubbleSort(int numbers[], int array_size)
{
  int i, j, temp;

  for (i = (array_size – 1); i >= 0; i– –)
  {
    for (j = 1; j <= i; j++)
    {
      if (numbers[j–1] > numbers[j])
      {
        temp = numbers[j–1];
        numbers[j–1] = numbers[j];
        numbers[j] = temp;
      }
```

```
    }
  }
}
```

Algorithm for Parallel Bubble Sort

1. For k = 0 to n–2
2. If k is even then
3. for i = 0 to (n/2)–1 do in parallel
4. If A[2i] > A[2i+1] then
5. Exchange A[2i] ↔ A[2i+1]
6. Else
7. for i = 0 to (n/2)–2 do in parallel
8. If A[2i+1] > A[2i+2] then
9. Exchange A[2i+1] ↔ A[2i+2]
10. Next k

Parallel Analysis

Steps 1-10 is a one big loop that is represented n–1 times. Therefore, the parallel time complexity is O(n). If the algorithm, odd-numbered steps need (n/2)–2 processors and even-numbered steps require (n/2) – 1 processors. Therefore, this needs O(n) processors.

6.3 Selection Sort

This is the simplest method of sorting. In this method, to sort the data in ascending order, the 0th element is compared with all the +ve elements. If the 0th element is found to be greater than T compared element then they are interchanged. So after the iteration the smallest element is placed at the 0th position. The same procedure is repeated for the 1st element and so on. This can be explained with the help of Figure 6.1.

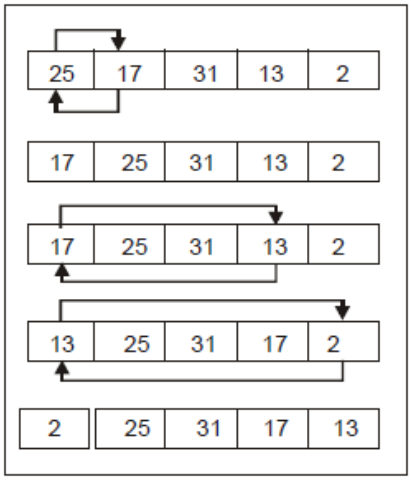


Figure 6.1: Selection Sort.

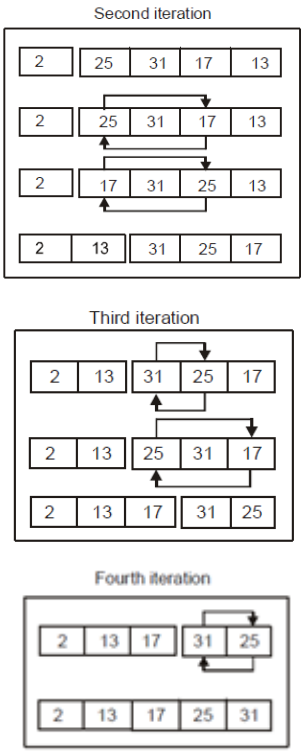


Figure 6.2: Selection Sort (Contd.).

Suppose an array are consists of 5 numbers. The selection sort algorithm works as follows:

1. In the first iteration the 0 th element 25 is compared with 15, element 17 and since 25 is greater than 17, they are interchanged.
2. Now the 0 th element 17 is compared with 2 nd element 31. But 17 being less than 31, hence they are not interchanged.
3. This process is repeated till 0 th element is compared with rest of the elements. During the comparison if 0 th element is found to be greater than the compared element, then they are interchanged, otherwise not.
4. At the end of the first iteration, the 0 th element holds the smallest number.
5. Now the second iteration starts with the l st element 25. The above process of comparison and swapping is repeated.
6. So if there are n elements, then after (n – 1) iterations the array is sorted.

Program

```
#include <stdio.h>
void main( ) //entry level of the program
{
int arr[5] = { 25, 17, 31, 13, 2 };//initialize the array
int i, j, temp;
for ( i = 0; i <= 3; i++ )//outer loop from 0 to 3
{
for ( j = i + 1 ; j <= 4 ; j++) //inner loop from 1 to 4
{
if (arr[i] > arr[j]) //check the condition for adjoining array
{
temp = arr[i];
```

```
arr[i] = arr[j]; //interchange the array[j] to array[i]
arr[j] = temp;
}
}
}
printf (“\n Array after sorting:\n”);
for ( i = 0; i <= 4; i++ )
printf (“%d\t”, arr[i] ) ; //print the sorted array
}
```

Algorithm of Program

Step 1: The main() method of program is called. // program start from here
Step 2: Initialization
Set arr[5]<-{ 25, 17, 31, 13, 2};
and var i, j, temp;
Step 3: loop:
Outer loop start: from 0 to 3 // for row
Inner loop start: from 1 to 4 // for column
Compare: if (arr[i] > arr[j])
Set temp <- arr[i]
Set arr[i] -> arr[j] // interchange the value of one position to other
Then set arr[j] -> temp
End of inner loop
End of outer loop
Step 4: Now print the sorted array through for loop from 0 to 4 times.
//End of algorithm

6.4 Merge Sort

This is another sorting technique having the same average-case and worst-case time complexities, but requiring an additional list of size n.

The technique that we use is the merging of the two sorted lists (see Figure 6.3) of size m and n to form a single sorted list of size (m + n). Given a list of size n to be sorted, instead of viewing it to be one single list of size n, we start by viewing it to be n lists each of size 1, and merge the first list with the second list to form a single sorted list of size 2. Similarly, we merge the third and the fourth lists to form a second single sorted list of size 2, and so on. This completes one pass. We then consider the first sorted list of size 2 and the second sorted list of size 2, and merge them to form a single sorted list of size 4.

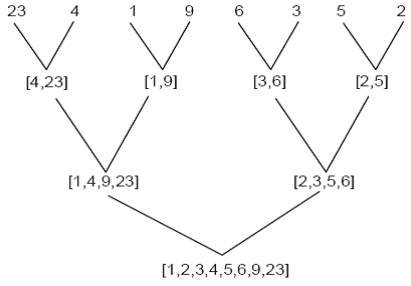


Figure 6.3: Merge Sort.

Similarly, we merge the third and the fourth sorted lists, each of size 2, to form the second single sorted list of size 4, and so on. This completes the second pass. In the third pass, we merge these adjacent sorted lists, each of size 4, to form sorted lists of size 8. We continue this process until we finally obtain a single sorted list of size n as shown next. To carry out this task, we require a function to merge the two sorted lists of size m and n to form a single sorted list of size (m + n). We also require a function to carry out one pass of the list to merge the adjacent sorted lists of the specified size. This is because we have to carry out repeated passes of the given list. In the first pass, we merge the adjacent lists of size 1. In the second pass, we merge the adjacent lists of size 2, and so on. Therefore, we will call this function by varying the size of the lists to be merged.

Program

```
#include <stdio.h>
#include <conio.h>
#define MAX_ARY 10
void merge sort (int x[], int end, int start); //function protype
int main (void) { // main method called with int return type
int ary[MAX_ARY]; //initialize of array
int j = 0; // assign the variable j with 0
printf("\n Enter the elements to be sorted: \n");

for(j=0; j<max_ary; j++) //input the elements by loop from 0 to max_ary
scanf("%d", &ary[j]);

/* array before mergesort */

printf("Before");

for(j = 0; j < MAX_ARY; j++)

printf("%d", ary[j]);

printf("\n");
merge_sort(ary, 0, MAX_ARY – 1); //method is called and assign the parameter and return void

/* array after mergesort */
printf("After Merge Sort:");
for(j = 0; j < MAX_ARY; j++)
printf(" %d", ary[j]); // print the merged array
printf("\n");
getch();
}
/* Method to implement Merge Sort*/
void merge_sort(int x[], int end, int start) {
int j = 0;
const int size = start – end + 1;
int mid = 0;
int mrg1 = 0;
int mrg2 = 0;
int executing[MAX_ARY];
```

```
if(end == start)
return;
mid = (end + start) / 2;
merge_sort(x, end, mid);
merge_sort(x, mid + 1, start);
for(j = 0; j < size; j++)
executing[j] = x[end + j];
mrg1 = 0;
mrg2 = mid – end + 1;
for(j = 0; j < size; j++) {
if(mrg2 <= start – end)
if(mrg1 <= mid – end)
if(executing[mrg1] > executing[mrg2])
x[j + end] = executing[mrg2++];
else
x[j + end] = executing[mrg1++];
else
x[j + end] = executing[mrg2++];
else
x[j + end] = executing[mrg1++];
}
}
```

Explanation

1. The merging of two sub lists, the first running from the index 0 to m, and the second running from the index (m + 1) to (n–1) requires no more than (n–l + 1) iterations.
2. So if l = 1, then no more than n iterations are required, where n is the size of the list to be sorted.
3. Therefore, if n is the size of the list to be sorted, every pass that a merge routine performs requires a time proportional to O(n), since the number of passes required to be performed is $\log_2 n$.
4. The time complexity of the algorithm is $O(n \log_2(n))$, for both average-case and worst case.
5. The merge sort requires an additional list of size n.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Write the merge sort algorithm using the C program.

……..……

……..……

……...

6.5 Quick Sort

Quick sort is one the fastest sorting algorithm used for sorting a list. A pivot point is chosen. Remaining elements are portioned or divided such that elements less than the pivot point are in left and those greater than the pivot are on the right. Now, the elements on the left and right can be recursively sorted by repeating the algorithm.

Program of Quick Sort

```
#include<stdio.h>
#include<conio.h>
/*global declaration*/
int a[20], n;
/*main function*/
void main()
{
int i; //initialize the variable
void qsort(int,int); //qsort() method is declared with parameters int and
void return type
clrscr(); //clear the screen
printf("Enter the array size");
scanf("%d", &n);
printf("\nEnter the elements to be sorted\n");
for(i=0; i<n; i++) //loop for the elements to be sorted from 0 to n
scanf("%d", &a[i]);
qsort(0, n–1); // method is called
printf("The sorted elements are\n");
for(i=0; i<n; i++) //loop for the elements that already sorted from 0 to n
printf("%d\t", a[i]);
getch();
}
```

/*quick sort function*/

```
void qsort(int left, int right)
{
int i, j, pivot, temp, k; // initialize the local variables
if(left<right) // check the condition for the left and right elements
{
i=left+1;
j=right;
pivot=left; // make the left elements pivot
for(;;)
{
while(a[pivot]>a[i]) //loop for row a[i]
i=i+1;
while(a[pivot]<a[j]) //loop for column a[j]
j=j–1;
if(i<j) // check the condition
{
temp=a[i];
a[i]=a[j]; //interchange the value
a[j]=temp;
for(k=0; k<n; k++)
printf("%d\t", a[k]);
printf("\n");
```

```
}
else
break;
}
temp=a[pivot];
a[pivot]=a[j]; //make the left element pivot
a[j]=temp;
for(k=0; k<n; k++)
printf("%d\t", a[k]); //print the elements a[k]
printf("\n");
qsort(left,j–1); // qsort() method is called
qsort(j+1,right);
}
return;
}
/*End of program
```

6.6 Insertion Sort

Insertion sort is implemented by inserting a particular element at the appropriate position. In this method, the first iteration starts with comparison of l st element with the 0 th element. In the second iteration 2 nd element is compared with the 0 th and l st element. Generally, in every iteration an element is compared with all elements before it. During comparison it is found that the element in question can be inserted at a suitable position then space is, related for it by shifting the other elements one position to the right and inserting the element at the suitable position. This procedure is repeated for all the elements in the array. Let us understand this with the help of Figure 6.4.

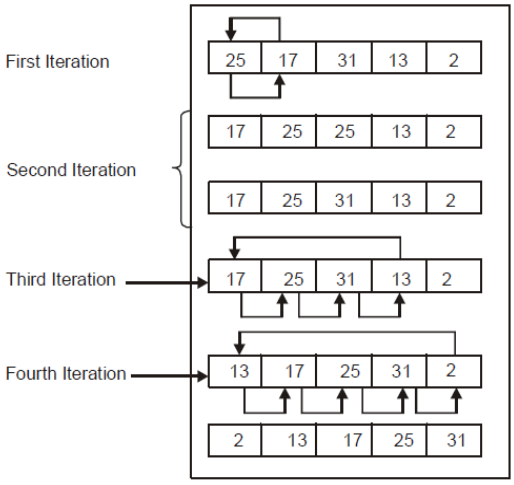


Figure 6.4: Insertion sort.

Following points explains the algorithm of insertion sort for a, array of 5 elements:

1. In the first iteration the lst element 17 is compared with 0 th element 25. Since 17 is smaller than 25, 17 is inserted on 0 th place. The 0 th element 25 is shifted one position to the right.

2. In the second iteration, the 2 nd element 31 and 0 th elements are compared. Since 31 is greater than 17, nothing is declared. Then the 2 nd element 31 is compared with the lst element, Again no action is taken as 25 is less than 31.
3. In the third iteration, the 3rd element 13 is compared with the 0 th element 17. Since, 13 is smaller than 17, 13 is inserted at the 0 th place in the array and all the element from 0 th till 2 nda position are shifted to right by one position.

4. In the fourth iteration the 4 th element 2 nd is compared with the 0 th element 13. Since, 2 is smaller than 13, the 4 th element is inserted at the 0 th place in the array and all the elements from 0 th till 3 rd are shifted right by one position. As a result, the array now becomes a sorted array.

Analysis of Insertion Sort

Because of the nested loops, each of which can take n iterations, insertion sort $O(n^2)$.
Furthermore, this bound is tight, because input in reverse order can actually achieve this bound. A precise calculation shows that the test at line 4 can be executed at most p times for each value of p. Summing over all p gives a total of:

$$\sum_{p=2}^{n} p = 2 + 3 + 4 + \ldots + n = \Theta(n^2)$$

On the other hand, if the input is pre-sorted, the running time is O(n), because the test in the inner for loop always fails immediately. Indeed, if the input is almost sorted, insertion sort will run quickly. Because of this wide variation, it is worth analyzing the average-case behaviour of this algorithm.

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
int A[20], N, Temp, i, j;
clrscr();
printf (“\n\n\t Enter The Number Of Terms... ”);
scanf (“%d”, &N);
printf (“\n\t Enter The Elements Of The Array...”);
for (i=0; i<N; i++)
{
gotoxy (25,11+i);
scanf (“\n\t\t%d”, &A[i]);
}
for (i=1; i<N; i++)
{
Temp = A[i];
j = i–1;
while (Temp<A[j] && j>=0)
{
A[j+1] = A[j];
j = j–1;
}
A[j+1] = Temp;
```

```
}
Printf (“\n\t The Ascending Order List is...:\n”);
for (i=0; i<N; i++)
printf (“\n\t\t\t %d”, A[i]);
getch();
}
```

6.7 Shell Sort

Shell sort is a multi-pass algorithm. Each pass is an insertion sort of the sequences consisting of every h th element for a fixed gap h also known as the increment. This is referred to as h-sorting.
Example:
A run of shell sort with gaps 5, 3 and 1 is shown as:
The first pass, 5-sorting, performs insertion sort on separate subarrays (a1, a6, a11), (a2, a7, a12), (a3, a8), (a4, a9), (a5, a10). For instance, it changes the subarray (a1, a6, a11) from (62, 17, 25) to (17, 25, 62). The next pass, 3-sorting, performs insertion sort on the subarrays (a1, a4, a7, a10), (a2, a5, a8, a11), (a3, a6, a9, a12). The last pass, 1-sorting, is an ordinary insertion sort of the entire array (a1,..., a12).
As the example illustrates, the subarrays that Shell sort operates on are initially short; later they are longer but almost ordered. In both cases insertion sort works efficiently.
Shell sort is unstable: it may change the relative order of elements with equal values. It has “natural” behaviour, in that it executes faster when the input is partially sorted.

Did You Know?
The Shell sort is named after its inventor, Donald Shell, was published the algorithm in 1959. Shell sort is a sequence of interleaved insertion sorts based on an increment sequence.

Exercise: Check Your Progress 2
Note: i) Use the space below for your answer.
Ex1: Write insertion sort?

……..……

……..……

……...

Ex2: What isQuick Sort?

……..……

……..……

……...

6.8 Address Calculation Sort

In this method, a function fn() is applied to each key. The result of this function determines into which of the several sub-files the record is to be placed. The function should have the property that x <= y, fn (x) <= fn (y).

Such a function is called order preserving. Thus all of the records in one sub-file will have keys that are less than or equal to the keys of the records in another sub-file. An item is placed into a sub-file in correct sequence by using any sorting method; simple insertion is often used. After all the items of the original file have been placed into sub-files, the sub-files may be concatenated to produce the sorted result.

Example:
An array,
25 57 48 37 12 92 86 33
Let us create ten sub-files, one for each of the ten possible first digits. Initially, each of these sub-files is empty. Consider an array of pointers f[10], where f[i] points to the first element in the file whose digit is i. After scanning the first element (i.e., 25), it is placed into the file header by f[2]. Each of the sub-files is maintained as a sorted linked list of the original array elements.

Self Assessment Questions

1. Sorting is not possible by using which of the following methods?
(a) Insertion (b) Selection (c) Exchange (d) Deletion

2. Which sorting algorithm is the best if the list is already in order?
(a). Quick sort (b). Merge sort (c). Insertion sort (d). Heap sort

3. The sort which inserts each elements A(K) into proper position in the previously sorted sub array A(1), ..., A(K–1)
(a). Insertion sort (b). Radix sort (c). Merge sort (d). Bubble sort

4. Which sorting algorithm is the best if the list is already in order?
(a). Quick sort (b). Merge sort (c). Insertion sort (d). Heap sort

6.9 Radix Sort

Radix sort is sometimes known as card sort, because it was used, until the advent of modern computers, to sort old-style punch cards.
If we have n integers in the range 1 to m (or 0 to m–1) 9, we can use this information to obtain a fast sort known as bucket sort. We keep an array called count, of size m, which is initialized to zero. Thus, count has m cells (or buckets), which are initially empty. When a_i is read, increment (by one) counts $[a_i]$. After all the input is read, scan the count array, printing out a representation of the sorted list. This algorithm takes O(m + n); the proof is left as an exercise. If m = (n), then bucket sort is O(n).
Radix sort is a generalization of this. The easiest way to see what happens is by example. Suppose we have 10 numbers, in the range 0 to 999, that we would like to sort. In general, this is n numbers in the range 0 to $n^p - 1$ for some constant p. Obviously, we cannot use bucket sort; there would be too many buckets. The trick is to use several passes of bucket sort. The natural algorithm would be to bucket-sort by the most significant digit (digit is taken to base n), then next most significant, and so on. That algorithm does not work, but if we perform bucket sorts by least significant digit first, then the algorithm works. Of course, more than one number could fall into the same bucket, and, unlike the original bucket sort, these numbers could be different, so we keep them in a list. Notice that all the numbers could have some digit in common, so if a simple array were used for the lists, then each array would have to be of size n, for a total space requirement of (n^2).

The example shows (see Figure 6.5) the action of radix sort on 10 numbers. The input is 64, 8, 216, 512, 27, 729, 0, 1, 343, 125 (the first ten cubes arranged randomly). The first step bucket sorts by the least significant digit. In this case the math is in base 10 (to make things simple), but do not assume this in general. So the list, sorted by least significant digit, is 0, 1, 512, 343, 64, 125, 216, 27, 8, 729. These are now sorted by the next least significant digit (the tens digit here). Pass 2 gives output 0, 1, 8, 512, 216, 125, 27, 729, 343, 64. This list is now sorted with respect to the two least significant digits. The final pass, bucket-sorts by most significant digit. The final list is 0, 1, 8, 27, 64, 125, 216, 343, 512, and 729. To see that the algorithm works, notice that the only possible failure would occur if two numbers came out of the same bucket in the wrong order. But the previous passes ensure that when several numbers enter a bucket, they enter in sorted order. The running time is O(p(n + b)) where p is the number of passes, n is the number of elements to sort, and b is the number of buckets. In our case, b = n.

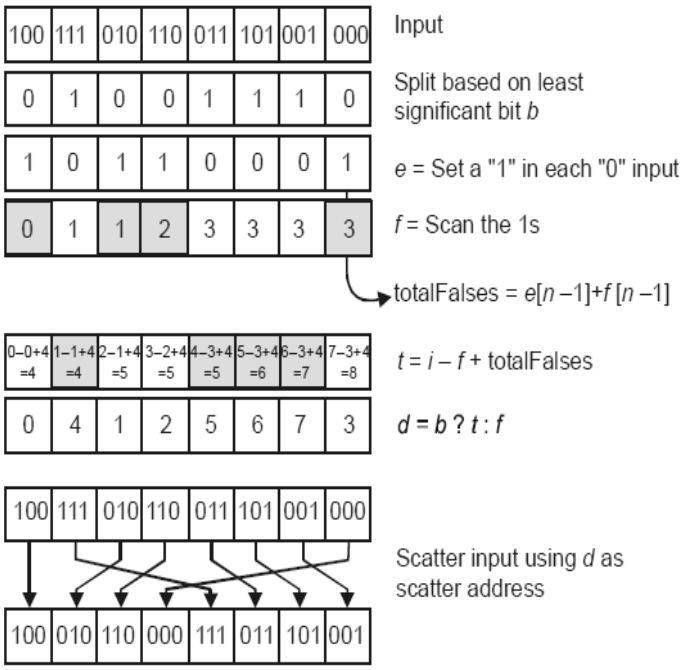


Figure 6.5: Radix Sort.

0 1 512 343 64 125 216 27 8 729
0 1 2 3 4 5 6 7 8 9
Buckets after first step of radix sort
8 729
1 216 27
0 512 125 343 64
---
0 1 2 3 4 5 6 7 8 9
Buckets after the second pass of radix sort
64
27
8
1
0 125 216 343 512 729
---
0 1 2 3 4 5 6 7 8 9

6.9.1 Buckets after the Last Pass of Radix Sort

As an example, we could sort all integers that are representable on a computer (32 bits) by radix sort, if we did three passes over a bucket size of 2. This algorithm would always be O(n) on this computer, but probably still not as efficient as some of the algorithms, because of the high constant involved (remember that a factor of log n is not all that high, and this algorithm would have the overhead of maintaining linked lists).

Program

```
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>
\* Intilization of variable and function*\
int check_order(unsigned int *, int);
void lsd_radix_sort(unsigned int *, int);
main(int argc, char *argv[]) {
int i, nvals = 10000;
unsigned int *array;
if(argc > 1)\\ checking condition
nvals = atoi(argv[1]);
array = malloc(nvals * sizeof(unsigned int));          \\assigning arrays size
for (i = 0; i < nvals; i++)\\ condition
array [i] = random();
lsd_radix_sort(array, nvals);
if(i = check_order(array, nvals))
printf ("%d misorderings\n", i);
else
printf ("array is in order\n");
}
int check_order(unsigned int *ip, int n)
{
int i, nrev = 0;
for (i = 1; i < n; i++)
if (ip[i–1] > ip[i])
nrev++;\\ incrementing the value
return (nrev);\\ returning the value }
```

6.10 Comparison of Sorting Methods

Methods for sorting have been intensively studied and optimized. Nevertheless, many poor algorithms (or poor versions of good algorithms) are still in circulation. The Max's friend David B. Ring have translated a variety of sorting routines into Visual Basic and compared their performance. The collection includes some of the faster algorithms known (e.g., QuickSort, RadixSort and Merge-Sort) and incorporates many optimizations known to increase speed and/or consistency. Methods for sorting have been intensively studied and optimized. Nevertheless, many poor algorithms (or poor versions of good algorithms) are still in circulation. The table 6.1 shows the description of different sorting methods.

Table 6.6: Table of Sorting Methods

Method	Worst Time	Average Time	# of comparison	# of swaps	Space	Stable	Note
Bubble Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^2)$	O(1)	Yes	Times are for best variant.
Selection Sort	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$	O(n)	O(1)	No	Outperform on small list of huge data set, when in case that writes are more expensive than reads.
Insertion Sort	$O(n^2)$		$O(n^2)$	O(n)	O(1)	Yes	Good for small data set.
Shell Sort						No	
Merge Sort	$\Theta(n \cdot \lg n)$	$\Theta(n \cdot \lg n)$	$O(n \cdot \lg n)$	N.A		Yes	Best choice for sorting a sequential access data set, like linked list or tape drive
Quick Sort	$O(n^2)$	$\Theta(n \cdot \lg n)$	$O(n \cdot \lg n)$	$O(n \cdot \lg n)$		No	The most practical general-purpose sorting method.
Heap Sort	$O(n \cdot \lg n)$	$O(n \cdot \lg n)$	$O(n \cdot \lg n)$	$O(n \cdot \lg n)$	O(1)	No	Always runs at $O(n \cdot \lg n)$ time
Counting Sort	O(n+k)	O(n+k)	N.A	N.A	O(2n+k)	Yes	Suitable when the keys have small range.
(LSB) Radix Sort	$O(n \cdot k)$	$O(n \cdot k)$	N.A	N.A	O(n)	Yes	Suitable when either keys are small, or with a lexicographic sequence.
Bucket Sort						Yes	Assume keys to be uniformly distributed.

6.11 Hash Table

A data object called a symbol table is required to be defined and implemented in many applications, such as compiler/assembler writing. A symbol table is nothing but a set of pairs (name, value), where value represents a collection of attributes associated with the name, and the collection of attributes depends on the program element identified by the name. For example, if a name x is used to identify an array in a program, and then the attributes associated with x are the number of dimensions, lower bound and upper bound of each dimension, and element type. Therefore, a symbol table can be thought of as a linear list of pairs (name, value), and we can use a list data object for realizing a symbol table. A symbol table is referred to or accessed frequently for adding a name, or for storing or retrieving the attributes of a name. Therefore, accessing efficiency is a prime concern when designing a symbol table. The most common method of implementing a symbol table is to use a hash table. The hash function operates on the name to be stored in the symbol table, or whose attributes are to be retrieved from the symbol table. If h is a hash function and x is a name, then h(x) gives the index of the table where x, along with its attributes, can be stored. If x is already stored in the table, then h(x) gives the index of the table where it is stored, in order to retrieve the attributes of x from the table. There are various methods of defining a hash function. One is the division method. In this method, we take the sum of the values of the characters, divide it by the size of the table, and take the remainder. This gives us an integer value lying in the range of 0 to (n−1), if the size of the table is n. Another method is the mid-square method. In this method, the identifier is first squared and then the appropriate number of bits from the middle of the square is used as the hash value. Since the middle bits of the square usually depend on all the characters

in the identifier, it is expected that different identifiers will result in different values. The number of middle bits that we select depends on the table size. Therefore, if r is the number of middle bits that we are using to form the hash value, then the table size will be 2^r. So when we use this method, the table size is required to be a power of 2.

A third method is folding, in which the identifier is partitioned into several parts, all but the last part being of the same length. These parts are then added together to obtain the hash value. To store the name or to add attributes of the name, we compute the hash value of the name, and place the name or attributes, as the case may be, at that place in the table whose index is the hash value of the name. To retrieve the attribute values of the name kept in the symbol table, we apply the hash function of the name to that index of the table where we get the attributes of the name. So we find that no comparisons are required to be done; the time required for the retrieval is independent of the table size. The retrieval is possible in a constant amount of time, which will be the time taken for computing the hash function. Therefore a hash table seems to be the best for realization of the symbol table, but there is one problem associated with the hashing, and that is collision. Hash collision occurs when the two identifiers are mapped into the same hash value. This happens because a hash function defines a mapping from a set of valid identifiers to the set of those integers that are used as indices of the table. Therefore we see that the domain of the mapping defined by the hash function is much larger than the range of the mapping, and hence the mapping is of a many-to-one nature. Therefore, when we implement a hash table, a suitable collision-handling mechanism is to be provided, which will be activated when there is a collision. Collision handling involves finding an alternative location for one of the two colliding symbols. For example, if x and y are the different identifiers and h(x = h(y)), x and y are the colliding symbols. If x is encountered before y, then the i^{th} entry of the table will be used for accommodating the symbol x, but later on when y comes, there is a hash collision. Therefore we have to find a suitable alternative location either for x or y. This means we can either accommodate y in that location, or we can move x to that location and place y in the i^{th} location of the table. Various methods are available to obtain an alternative location to handle the collision. They differ from each other in the way in which a search is made for an alternative location. The following are commonly used collision-handling techniques:

6.11.1 Collision Resolution Techniques

Linear Probing or Linear Open Addressing

In this method, if for an identifier x, h(x) = i, and if the i th location is already occupied, we search for a location close to the i th location by doing a linear search, starting from the (i+1) th location to accommodate x. This means we start from the (i+1) th location and do the linear search until we get an empty location; once we get an empty location we accommodate x there.

Rehashing

In rehashing we find an alternative empty location by modifying the hash function and applying the modified hash function to the colliding symbol. For example, if x is the symbol and h(x) = i, and if the i th location is already occupied, then we modify the hash function h to h_1, and find out $h_1(x)$, if $h_1(x) = j$. If the j th location is empty, then we accommodate x in the j th location. Otherwise, we once again modify h_1 to some h_2 and repeat the process until the collision is handled. Once the collision is handled, we revert to the original hash function before considering the next symbol.

Overflow chaining

Overflow chaining (see Figure 6.6) is a method of implementing a hash table in which the collisions are handled automatically. In this method, we use two tables: a symbol table to accommodate identifiers and their attributes, and a hash table, which is an array of pointers pointing to symbol table entries. Each symbol table

entry is made of three fields: the first for holding the identifier, the second for holding the attributes, and the third for holding the link or pointer that can be made to point to any symbol table entry.

The insertions into the symbol table are done as follows:
If x is the symbol to be inserted, it will be added to the next available entry of the symbol table. The hash value of x is then computed. If h(x) = i, then the i th hash table pointer is made to point to the symbol table entry in which x is stored, if the i th hash table pointer does not point to any symbol table entry. If the i th hash table pointer is already pointing to some symbol table entry, then the link field of the symbol table entry containing x is made to point to that symbol table entry to which the i th hash table pointer is pointing to, and the i th hash table pointer is made to point to the symbol entry containing x. This is equivalent to building a linked list on the i th index of the hash table. The retrieval of attributes is done as follows. If x is a symbol, then we obtain h(x), use this value as the index of the hash table, and traverse the list built on this index to get that entry which contains x. A typical hash table implemented using this technique is shown here. The symbols to b stored are x_1,y_1,z_1,x_2,y_2,z_2. The hash function that we use is h(symbol) = (value of first letter of the symbol) mod n, where n is the size of table.
if $h(x_1) = i$
$h(y_1) = j$
$h(z_1) = k$
then
$h(x_2) = i$
$h(y_2) = j$
$h(z_2) = k$

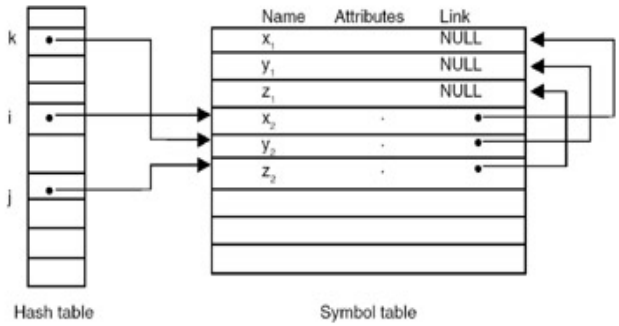


Figure 6.7: Hash table implementation using overflow chaining for collision handling.

Program

```
#include < stdio.h>
#include < stdlib.h>
#define HASHSIZE1000
#define MAXLINE1024
typedef struct tnode {
 char *data;
 struct tnode *next;
 } node;
void htable_init(node *hashtable);          // fire up hashtable
void htable_insert(node *hashtable, char *str);           // insert data into hashtable
void htable_resolve(node *hashtable, int loc, char *str); // resolve collisions in hashtable
void htable_display(node *hashtable);                  // display hashtable
int htable_delete(node *hashtable, char *str);           // delete an entry from hashtable
```

```
int htable_hash(char *str);                    // hash data for hashtable
int main(void) {
char line[MAXLINE];
node *hashtable;
hashtable = (node *)malloc(HASHSIZE * sizeof(node));
htable_init(hashtable);
while((fgets(line, MAXLINE, stdin)) != NULL)
 htable_insert(hashtable, line);
 htable_display(hashtable);
 return 0;
 }
/* fire up hashtable */
void htable_init(node *hashtable) {
 int i = 0;
 for(i = 0; i < HASHSIZE; i++)
 hashtable[i].data = NULL, hashtable[i].next = NULL;
 }
/* insert data into hashtable */
void htable_insert(node *hashtable, char *str) {
 int index = 0;
// determine hash function
 index = htable_hash(str);
 if(hashtable[index].data != NULL) {
 // collision occurs - resolve by chaining
 htable_resolve(hashtable, index, str);
 } else {
 hashtable[index].data = calloc(strlen(str) + 1, sizeof(char));
 strcpy(hashtable[index].data, str);
 }
}
/* hash data for hashtable */
int htable_hash(char *str) {
 int index = 0;
 char *tmp = NULL;
 tmp = calloc(strlen(str) + 1, sizeof(char));
 strcpy(tmp, str);
 while(*tmp) {
 index += *tmp;
 tmp++;
 }
index = index % HASHSIZE;
 return index;
 }
/* resolve collisions in hashtable */
void htable_resolve(node *hashtable, int loc, char *str) {
 node *tmp;
 tmp = hashtable + loc;
```

```
while(tmp->next != NULL)
 tmp = tmp->next;
  tmp->next = (node *)malloc(sizeof(node));
  tmp->next->data = calloc(strlen(str) + 1, sizeof(char));
  strcpy(tmp->next->data, str);
  tmp->next->next = NULL;
 }
/* display hashtable */
void htable_display(node *hashtable)
{
int i = 0;
 node *target;

for(i = 0; i < HASHSIZE; i++) {
  if(hashtable[i].data != NULL) {
  target = hashtable + i;
  while(target)
  /* printf("location: %d, data: %s", i, target->data), target = target->next; */
 printf("%s", target->data), target = target->next;
  } /* if */
 } /* for */
}
/* delete an entry from hashtable */
int htable_delete(node *hashtable, char *str) {
 node *bla;
node *blb;
char *tmp = NULL;
int index = 0;
index = htable_hash(str);
 /* no item at this location */
 if(hashtable[index].data == NULL)
 return 1;
/* only one item at this location */
 if(hashtable[index].next == NULL) {
  if(strcmp(hashtable[index].data, str) == 0) {
  /* item found */
  tmp = hashtable[index].data;
  hashtable[index].data = NULL;
  free(tmp);
 }
 } else
{
 /* There is a chaining case */
 bla = hashtable + index;
  /* linked list similar */
 while(bla->next != NULL)
{
```

```
  if(strcmp(bla->next->data, str) == 0) {
  blb = bla->next;
  if(bla->next->next)
   bla->next = bla->next->next;
   else
   bla->next = NULL;
   free(blb);
  } /* if */
  } /* while */
 } /* else */
return 0;
}
```

Exercise: Check Your Progress 3
Note: i) Use the space below for your answer.
Ex1: Write Shell sort?

……..……

……..……

……...

Ex2: What is Radix Sort?

……..……

……..……

……...

6.12 Linear Search (Sequential Search)

In computer science, linear search is a search algorithm, also known as sequential search, that is suitable for searching a set of data for a particular value. It operates by checking every element of a list until a match is found. Linear search runs in O(N). If the data are distributed randomly, on average N/2 comparisons will be needed. The best case is that the value is equal to the first element tested, in which case only 1 comparison is needed. The worst case is that the value is not in the list, in which case N comparisons are needed.The Figure 6.8 shows the linear search.

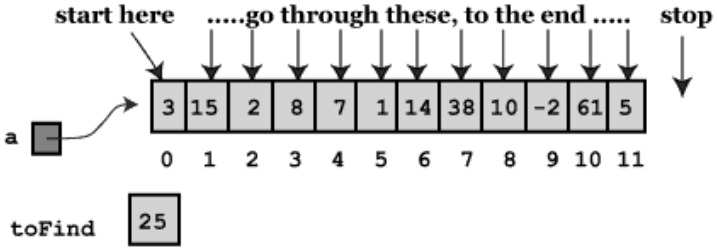


Figure 6.8: Linear Search

Program

```
#include <stdio.h>
#define MAX 10
void lsearch(int list[], int n, int element)
{
  int i, flag = 0;
  for(i=0;i<n;i++)
  if( list[i] == element)
  {
    printf("The element whose value is %d is present at position %d in list\n", element, i);
    flag =1;
    break;
  }
  if( flag == 0)
     printf("The element whose value is %d is not present in the list\n", element);
}
void readlist(int list[], int n)
{
  int i;
  printf("Enter the elements\n");
  for(i=0;i<n;i++)
     scanf("%d",&list[i]);

void printlist(int list[], int n)
{
  int i;
  printf("The elements of the list are: \n");
  for(i=0;i<n;i++)
     printf("%d\t", list[i]);
}
void main()
{
   int list[MAX], n, element;
  printf("Enter the number of elements in the list max = \n");
  scanf("%d", &n);
  readlist(list,n);
  printf("\nThe list before sorting is:\n");
  printlist(list,n);
  printf("\nEnter the element to be searched\n");
  scanf("%d",&element);
  lsearch(list, n, element);
}
```

Example

Input

Enter the number of elements in the list, max = 10

10

Enter the elements
23
1
45
67
90
100
432
15
77
55

Output

The list before sorting is:
The elements of the list are:
23 1 45 67 90 100 432 15 77 55
Enter the element to be searched
100
The element whose value is 100 is present at position 5 in list

Input

Enter the number of elements in the list max = 10
10
Enter the elements
23
1
45
67
90
101
23
56
44
22

Output

The list before sorting is:
The elements of the list are:
23 1 45 67 90 101 23 56 44 22
Enter the element to be searched
100
The element whose value is 100 is not present in the list

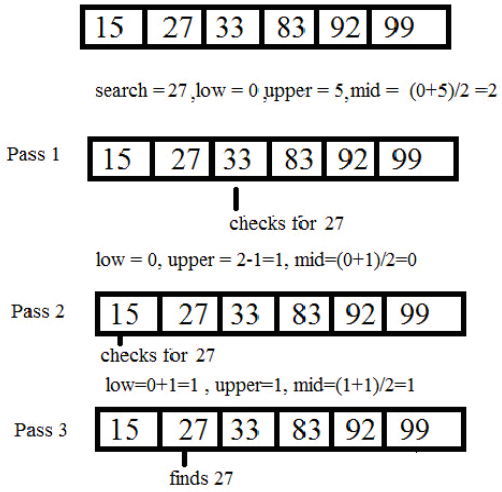
Explanation

1. In the best case, the search procedure terminates after one comparison only, whereas in the worst case, it will do n comparisons.
2. On average, it will do approximately n/2 comparisons, since the search time is proportional to the number of comparisons required to be performed.
3. The linear search requires an average time proportional to O(n) to search one element. Therefore to search n elements, it requires a time proportional to $O(n^2)$.

4. We conclude that this searching technique is preferable when the value of n is small. The reason for this is the difference between n and n^2 is small for smaller values of n.

6.13 Binary Search

The prerequisite for using binary search is that the list must be a sorted one. We compare the element to be searched with the element placed approximately in the middle of the list. If a match is found, the search terminates successfully. Otherwise, we continue the search for the key in a similar manner either in the upper half or the lower half. If the elements of the list are arranged in ascending order, and the key is less than the element in the middle of the list, the search is continued in the lower half. If the elements of the list are arranged in descending order, and the key is greater than the element in the middle of the list, the search is continued in the upper half of the list. The procedure for the binary search is given in the following program.



Program

```
#include <stdio.h>
#define MAX 10

void bsearch(int list[], int n, int element)
{
  int l, u, m, flag = 0;
  l = 0;
  u = n–1;
  while(l <= u)
  {
    m = (l+u)/2;
    if( list[m] == element)
    {
          printf(" The element whose value is %d is present at position %d in list\n",
              element,m);
          flag =1;
          break;
    }
    else
        if(list[m] < element)
            l = m+1;
```

```
        else
            u = m–1;
  }
  if( flag == 0)
  printf("The element whose value is %d is not present in the list\n", element);
}

void readlist(int list[], int n)
{
  int i;
  printf("Enter the elements\n");
  for(i=0;i<n;i++)
     scanf("%d", &list[i]);
}

void printlist(int list[], int n)
{
   int i;
  printf("The elements of the list are: \n");
  for(i=0; i<n; i++)
     printf("%d\t", list[i]);
}
void main()
{
  int list[MAX], n, element;
  printf("Enter the number of elements in the list max = 10\n");
  scanf("%d", &n);
  readlist(list, n);
  printf("\n The list before sorting is:\n");
  printlist(list, n);
  printf("\n Enter the element to be searched\n");
  scanf("%d", &element);
  bsearch(list, n, element);
}
```

Example

Input

```
Enter the number of elements in the list, max = 10
10
Enter the elements
34
2
1
789
99
45
66
```

33
22
11

Output

The elements of the list before sorting are:
34 2 1 789 99 45 66 33 22 11
1 2 3 4 5 6 7 8 9 10
Enter the element to be searched:
99
The element whose value is 99 is present at position 5 in the list

Input

Enter the number of elements in the list max = 10
10
Enter the elements
54
89
09
43
66
88
77
11
22
33

Output

The elements of the list before sorting are:
54 89 9 43 66 88 77 11 22 33
Enter the element to be searched
100
The element whose value is 100 is not present in the list.

In the binary search, the number of comparisons required to search one element in the list is no more than log2n, where n is the size of the list. Therefore, the binary search algorithm has a time complexity of O(n *(log2n.).)

6.14 Searching an Ordered Table

An ordered table can be searched using any of the following search methods.

Binary Search

Suppose we have to search for a word in the dictionary, we do not perform a linear search. Since a dictionary is sorted in alphabetical order, one of the things most people do by habit when using it is to flip the dictionary open toward the front, middle, or back depending on the location in the alphabet of the word being sought. If the page opened has words farther down in the alphabet, the search word cannot be on that page or anywhere in the dictionary to its right. It can only be to the left of the selected page. This process of elimination is behind the search technique called binary search. When searching for a specific key value among 2n records, at most n key comparisons are needed to determine whether it is present, as long as each comparison splits the remaining collection of records into two parts that differ in size by at most one. If

the set of records are of huge size and sorted using key values in ascending order, then the total time is considerably reduced by using this type of search. In all cases, the complexity of the algorithm is O(log n).

When we search the data which are stored in an arbitrary order in an array, we will not find the search key to be in a particular region in the array. However, in cases where the data is stored in a sorted order, there is a possibility that the search key is present in the middle element. If it is there, then it can be located by the running program which terminates the search. When the search key is not present in the middle element, then we can assume it to be greater than the key which is present in the middle element, and the middle element along with all the other elements present here, these are eliminated. Now, the only situation left is that it is above the middle element. Similarly, in cases where the search key is less than middle element's key then the middle element along with all the other elements present above it are eliminated. In any of these cases, more than half of the elements remain and we need to follow the same solution for these elements, hence elements 1 to [mid–1] or elements [mid+1] to n where the variable pointing to the middle element is called mid.

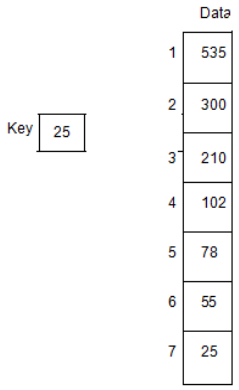


Figure 6.9: Array of integers sorted in descending order.

Consider Figure 6.9 for example. If mid is 5 in this case, then the search key value, 25, is not greater than data [mid], 78. Hence, if at all, it is present in the data, then it must be in between positions mid+1 and n, i.e., between 6 and 9. Let us take 7 as the middle element of these, the new mid value is 15. Key = [mid], 25. Thus, we are able to find the middle key.

Lab Exercise

1. Write a program that implements insertion sort algorithm for a linked list of integers.
2. Write a program that sorts the elements of a two-dimensional array row wise and column wise.

6.15 Indexed Sequential Search

An index file can be used to effectively overcome the above mentioned problem, and to speed up the key search as well. The simplest indexing structure is the single-level one: a file whose records are pair's key-pointer, where the pointer is the position in the data file of the record with the given key. Only a subset of data records, evenly spaced along the data file, are indexed, so to mark intervals of data records. A key search then proceeds as follows: the search key is compared with the index ones to find the highest index key preceding the search one, and a linear search is performed from the record the index key points onward, until the search key is matched or until the record pointed by the next index entry is reached. In spite of the double file access (index + data) needed by this kind of search, the decrease in access time with respect to a sequential file is significant. Consider, for example, the case of simple linear search on a file with 1,000 records. With the sequential organization, an average of 500 key comparisons is necessary (assuming uniformly distributed

search key among the data ones). However, using and evenly spaced index with 100 entries, the number of comparisons is reduced to 50 in the index file plus 50 in the data file: a 5:1 reduction in the number of operations.

This scheme can obviously be hierarchically extended: an index is a sequential file in itself, amenable to be indexed in turn by a second-level index, and so on, thus exploiting more and more the hierarchical decomposition of the searches to decrease the access time. Obviously, if the layering of indexes is pushed too far, a point is reached when the advantages of indexing are hampered by the increased storage costs and by the index access times as well.

Program for Indexed sequential search

```
#include <stdio.h>
#define MAX 24
void createIndex(int index[],int isize,int arr[],int asize)
{
	int i,j;
	for(i=0, j=0; i<asize; i+=8, j++)
	{
		index[j]= arr[i];
	}
	index[j] = arr[assize–1];
}
int indexSeqSearch(int val, int index[], int isize, int arr[], int asize)
{
	int i=0, j=0, pos=0;
	int high=0, low=0;
	if(val > index[isize–1] && val < index[0])
		return –1;

	while(i<isize)
	{
		if(val == index[i])
		{
			pos = 8 * i;
			return pos;
		}
		if(val < index[i])
		{
			low = 8 * (i–1);
			high = 8 * i;
			break;
		}
		else
		{
			low = 8 * i;
			high = 8 * (i+1);
		}
		i++;
```

```
    }
    while(low < high)
    {
        if(val == arr[low])
            return low;
        else
            low++;
    }
    return –1;
}
int main()
{
    int arr[MAX]={8,14,26,38,72,115,306,
321,329,387,409,512,540,567,583,592,602,611,618,741,798,811,814,876};
    int index[(MAX/8)+1]={0};
    createIndex(&index[0],(MAX/8)+1,&arr[0],MAX);
    int opt=0, pos=0;
    while(opt < MAX)
    {
        pos = indexSeqSearch(arr[opt], &index[0], (MAX/8)+1, &arr[0], MAX);
        if( pos != –1)
        {
            printf("\n%d found at position %d", arr[opt],pos);
        }
        else
            printf("\n%d not found", arr[opt]);

        opt++;
    }

    return 0;
}
```

Exercise: Check Your Progress 4
Note: i) Use the space below for your answer.
Ex1: Write lenear search?

……..……

……..……

……...

6.16 Interpolation Search

Interpolation search (sometimes referred to as extrapolation search) is an algorithm for searching for a given key value in an indexed array that has been ordered by the values of the key. It parallels how humans search through a telephone book for a particular name, the key value by which the book's entries are ordered. In each

search step it calculates where in the remaining search space the sought item might be, based on the key values at the bounds of the search space and the value of the sought key, usually via a linear interpolation. The key value actually found at this estimated position is then compared to the key value being sought. If it is not equal, then depending on the comparison, the remaining search space is reduced to the part before or after the estimated position. This method will only work if calculations on the size of differences between key values are sensible. By comparison, the binary search always chooses the middle of the remaining search space, discarding one half or the other, again depending on the comparison between the key value found at the estimated position and the key value sought. The remaining search space is reduced to the part before or after the estimated position. The linear search uses equality only as it compares elements one-by-one from the start, ignoring any sorting.

On average the interpolation search makes about log(log(n)) comparisons (if the elements are uniformly distributed), where n is the number of elements to be searched. In the worst case (for instance where the numerical values of the keys increase exponentially) it can make up to O(n) comparisons.

In interpolation-sequential search, interpolation is used to find an item near the one being searched for, and then linear search is used to find the exact item.

Program

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 5
int interpolationsearch(int a[], int low, int high, int x)
{
        int mid;
          while(low<=high)
        {
            mid=low+(high–low)*((x–a[low])/(a[high]–a[low]));
            if(x==a[mid])
                return mid+1;
            if(x<a[mid])
                high=mid–1;
            else
                low=mid+1;
        }
     return –1;
}
int main()
{
    int arr[MAX];
    int i, n;
    int val, pos;
    printf("\n Enter total elements (n < %d) :",MAX);
    scanf("%d", &n);
    printf("Enter %d Elements :",n);
    for(i=0; i<n; i++)
                scanf("%d", &arr[i]);

    printf("\n LIST:");
```

```
        for(i=0; i<n; i++)
        printf("%d\t",arr[i]);
        printf("\n Search For :");
        scanf("%d", &val);
        pos=interpolationsearch(&arr[0],0,n,val);
        if(pos== –1)
           printf("\n Element %d not found\n", val);
        else
           printf("\n Element %d found at position %d\n", val, pos);
        return 0;
}
```

Self Assessment Questions

5. A sorting algorithm is stable if…….
(a). its time complexity is constant irrespective of the nature of input.
(b). preserves the original order of records with equal keys.
(c). its space complexity is constant irrespective of the nature of input.
(d). it sorts any volume of data in a constant time.

6. Hashing collision resolution techniques are
(a). Huffman coding, linear hashing (b). Bucket addressing, Huffman coding
(c). Chaining, Huffman coding (d). Chaining, Bucket addressing.

7. The running time of the following sorting algorithm depends on whether the Partitioning is balanced or unbalanced
(a). Insertion sort (b). Selection sort (c). Quick sort (d). Merge sort.

8. The average case of quick sort has order
(a). $O(n^2)$ (b). O(n) (c). O(n log n) (d). O(log n)

Exercise: Check Your Progress 5

Case Study: Starting and Searching: Search History- Google Maps search history allows you to quickly access your past search terms or queries without having to type them in again. Depending on your phone, Maps provides search history stored both on your phone and in your Google Account with Web History. Learn how to manage search history stored in your Phone Google Maps stores your search history locally on your phone by default. Reset Google Maps from the menu to restore it to the state it was in when first installed, clearing the following locally stored info if applicable: search history, favorites, cookies, preferences, map and satellite image cache, and layers.

Android-powered devices: From your home screen, go to 'Menu' > 'Settings' > 'Applications' > 'Manage applications' > 'Maps' or 'Google Maps', and then choosing to 'Clear data' or 'Factory reset'. If you updated Maps from Market, you may need to uninstall the update before you can clear data.

BlackBerry: From the main menu, select 'Help' > 'Reset Google Maps'.

Symbian S60: Select 'Options' > 'Tools' > 'Reset Google Maps'.

Windows Mobile: Select 'Menu' > 'Options' > 'Reset Google Maps'.

Other phones: Look for the 'Reset Google Maps' option in the Help menu.

6.17 Summary

- Insertion sort is implemented by inserting a particular element at the appropriate position.
- Merge sort is a technique having the same average case and worst case time complexities but requiring an additional list of size of n.
- Radix sort is sometimes known as card sort because it was used, until the advent of the modern computers, to sort old-style punch cards.
- Hashing is a method of directly computing the index of the table by using suitable mathematical function called a hash function.
- In rehashing we find an alternative empty location by modifying the hash function to the colliding symbol.

6.18 Keywords

Card Sort: It is called Radix sort.
Hashing: A data object called a symbol table is required to be defined and implemented in many applications, such as compiler/assembler writing.
Insertion Sort: Insertion sort is implemented by inserting a particular element at the appropriate position.
Merge Sort: This is another sorting technique having the same average-case and worst-case time complexities, but requiring an additional list of size n.
Radix Sort: Radix sort is sometimes known as card sort, because it was used, until the advent of modern computers, to sort old-style punch cards.

6.19 Review Questions

1. What is sorting?
2. What are the different types of sorting techniques in C language?
3. How many types of sorting algorithm?
4. What is the difference between sorting and searching in a database?
5. What is sequential search? What is the average number of comparisons in a sequential search?
6. What is binary search?
7. What is the quick sort? Explain with example.
8. What are the collision resolution techniques?
9. What are the ordered table and interpolation searching?
10. Write a C program to search the number 23 in given numbers using Binary search.

12, 34, 23, 35, 67, 86, 78, 90, 123

Answers for Self Assessment Questions

1. (a)	2. (c)	3. (a)	4. (c)
5. (b)	6. (d)	7. (c)	8. (c)

6.20 Furhter Studies

1. Sorting and Searching (Data Structures and Algorithms) by Kurt Mehlhorn
2. https://www.cs.auckland.ac.nz/~jmor159/PLDS210/niemann/s_man.pdf

Operating System Principle

(MCA/PGDCA - 105)



Jaipur National University
Directorate of Distance Education

Established by Government of Rajasthan
Approved by UGC under Sec 2(f) of UGC ACT 1956
(**Recognised by Joint Committee of UGC-AICTE-DEC, Govt. of India**)

1

Operating System

STRUCTURE

1.0 Objectives

1.1 Introduction

1.2 Operating System

1.3 Types of Operating System

1.4 System Components and its Services

1.5 System Calls

1.6 System Programs

1.7 Structure, Design and, Implementation

1.8 Operating System Generation

1.9 Summary

1.10 Keywords

1.11 Review Questions

1.12 Further Studies

1.0 Objectives

After studying this chapter, you will be able to:

- Discuss the history of operating systems
- Define Operating system
- Define the types of operating system
- Discuses the system components and its services
- Example the system calls
- Understand the system programs

1.1 Introduction

Modern general-purpose computers, including personal computers and mainframes, have an operating system to run other programs, such as application software. Examples of operating systems for personal computers include Microsoft Windows, Mac OS (and Darwin), UNIX, and Linux The lowest level of any operating system is its kernel. This is the first layer of software loaded into memory when a system boots or starts up. The kernel provides access to various common core services to all

other system and application programs. These services include, but are not limited to: disk access, memory management, task scheduling, and access to other hardware devices.

As well as the kernel, an operating system is often distributed with tools for programs to display and manage a graphical user interface (although Windows and the Macintosh have these tools built into the operating system), as well as utility programs for tasks such as managing files and configuring the operating system. They are also often distributed with application software that does not relate directly to the operating system's core function, but which the operating system distributor finds advantageous to supply with the operating system.

The delineation between the operating system and application software is not precise, and is occasionally subject to controversy. From commercial or legal points of view, the delineation can depend on the contexts of the interests involved. For example, one of the key questions in the United States v. Microsoft antitrust trial was whether Microsoft's Web browser was part of its operating system, or whether it was a separable piece of application software.

Like the term "operating system" itself, the question of what exactly should form the "kernel" is subject to some controversy, with debates over whether things like file systems should be included in the kernel. Various camps advocate microkernels, monolithic kernels, and so on. Operating systems are used on most, but not all, computer systems. The simplest computers, including the smallest embedded systems and many of the first computers did not have operating systems. Instead, they relied on the application programs to manage the minimal hardware themselves, perhaps with the aid of libraries developed for the purpose. Commercially-supplied operating systems are present on virtually all modern devices described as computers, from personal computers to mainframes, as well as mobile computers such as PDAs and mobile phones.

Did You Know?

The Mac OS of 1984 was the first system that could really deliver computer power to all users.

1.2 Operating System

An Operating System is a software program or set of programs that mediate access between physical devices (such as a keyboard, mouse, monitor, disk drive or network connection) and application programs (such as a word processor, World-Wide Web browser or electronic mail client).

Some characteristics of an Operating System are:

- Whether multiple programs can run on it simultaneously: *multi-tasking*
- Whether it can take advantage of multiple processors: *multi-processing*
- Whether multiple users can run programs on it simultaneously: *multi-user*
- Whether it can reliably prevent application programs from directly accessing hardware devices: *protected*
- Whether it has built-in support for graphics.
- Whether it has built-in support for networks.

Some popular Operating System's are:

- UNIX: Multi-tasking, multi-processing, multi-user, protected, with built-in support for networking but not graphics.
- Windows NT: Multi-tasking, multi-processing, single-user, protected, with built-in support for networking and graphics.
- Windows 95/98: Multi-tasking, multi-processing, single-user, unprotected, with built-in support for networking and graphics.

- Windows 3.x: Single-tasking, single-processing, single-user, unprotected, with built-in support for graphics but not networking.
- DOS: Single-tasking, single-processing, single-user, unprotected with no built-in support for graphics or networking.
- NetWare: Multi-tasking, multi-processing, single-user, unprotected, with built-in support for networking but not graphics.

1.3 Types of Operating System

Following are few of the important types of operating system which are most commonly used.

Batch operating system

The users of batch operating system do not interact with the computer directly. Each user prepares his job on an off-line device like punch cards and submits it to the computer operator. To speed up processing, jobs with similar needs are batched together and run as a group. Thus, the programmers left their programs with the operator. The operator then sorts programs into batches with similar requirements.

The problems with Batch Systems are following.

- Lack of interaction between the user and job.
- CPU is often idle, because the speeds of the mechanical I/O devices are slower than CPU.
- Difficult to provide the desired priority.

Time-sharing operating systems

Time sharing is a technique which enables many people, located at various terminals, to use a particular computer system at the same time. Time-sharing or multitasking is a logical extension of multiprogramming. Processor's time which is shared among multiple users simultaneously is termed as time-sharing. The main difference between Multi programmed Batch Systems and Time-Sharing Systems is that in case of multi programmed batch systems, objective is to maximize processor use, whereas in Time-Sharing Systems objective is to minimize response time. Multiple jobs are executed by the CPU by switching between them, but the switches occur so frequently. Thus, the user can receive an immediate response. For example, in a transaction processing, processor execute each user program in a short burst or quantum of computation. That is if n users are present, each user can get time quantum. When the user submits the command, the response time is in few seconds at most. Operating system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time. Computer systems that were designed primarily as batch systems have been modified to time-sharing systems. Advantages of Timesharing operating systems are following:

- Provide advantage of quick response.
- Avoids duplication of software.
- Reduces CPU idle time. Disadvantages of Timesharing operating systems are following.
- Problem of reliability.
- Question of security and integrity of user programs and data.
- Problem of data communication.

Distributed operating System

Distributed systems use multiple central processors to serve multiple real time application and multiple users. Data processing jobs are distributed among the processors accordingly to which one can perform each job most efficiently. The processors communicate with one another through various communication lines (such as high-speed buses or telephone lines). These are referred as loosely

coupled systems or distributed systems. Processors in a distributed system may vary in size and function. These processors are referred as sites, nodes, and computers and so on.

The advantages of distributed systems are following:

- With resource sharing facility user at one site may be able to use the resources available at another.
- Speedup the exchange of data with one another via electronic mail.
- If one site fails in a distributed system, the remaining sites can potentially continue operating.
- Better service to the customers.
- Reduction of the load on the host computer.
- Reduction of delays in data processing.

Network operating System

Network Operating System runs on a server and provides server the capability to manage data, users, groups, security, applications, and other networking functions. The primary purpose of the network operating system is to allow shared file and printer access among multiple computers in a network, typically a local area network (LAN), a private network or to other networks. Examples of network operating systems are Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD.

The advantages of network operating systems are following:

- Centralized servers are highly stable.
- Security is server managed.
- Upgrades to new technologies and hardware can be easily integrated into the system.
- Remote access to servers is possible from different locations and types of systems.

The disadvantages of network operating systems are following.

- High cost of buying and running a server.
- Dependency on a central location for most operations.
- Regular maintenance and updates are required.

Real Time operating System

Real time system is defines as a data processing system in which the time interval required to process and respond to inputs is so small that it controls the environment. Real time processing is always on line whereas on line system need not be real time. The time taken by the system to respond to an input and display of required updated information is termed as response time. So in this method response time is very less as compared to the online processing. Real-time systems are used when there are rigid time requirements on the operation of a processor or the flow of data and real-time systems can be used as a control device in a dedicated application. Real-time operating system has well-defined, fixed time constraints otherwise system will fail. For example Scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, and home-appliance controllers, Air traffic control system etc. There are two types of real-time operating systems.

Hard real-time systems

Hard real-time systems guarantee that critical tasks complete on time. In hard real-time systems secondary storage is limited or missing with data stored in ROM. In these systems virtual memory is almost never found.

Soft real-time systems

Soft real time systems are less restrictive. Critical real-time task gets priority over other tasks and retains the priority until it completes. Soft real-time systems have limited utility than hard real-time systems. For example Multimedia, virtual reality, Advanced Scientific Projects like undersea exploration and planetary rovers etc.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Prepare a list of system requirements before installing an operating system.

……..……

……..……

……...

1.4 System Components and its Services

A computer system has two basic parts:

- Hardware
- Software

The equipment associated with a computer system is the hardware. Computer hardware is responsible for performing four basic functions: input, processing, output, and storage. Let us go back to the basic definition of a computer. A computer is an electronic device that is programmed to accept data (input), process it into useful information (output), and store it for future use (storage). The processing function is under the control of a set of instructions (software); we will explore this later. The four primary components of a computer system are:

Table 1

Input	Input devices send data and instructions to the central processing unit.
Output	Central processing unit executes computer instructions. Memory holds data and programs temporarily.
Processing	Output devices make processed data available to the user.
Secondary Storage	Secondary storage devices store data and programs permanently.

1.4.1 Hardware

In order to function properly, a computer system must have all four types of hardware: input, processing, output, and storage.



Figure 1

In this example, the mouse and keyboard are the input devices and the monitor and speakers are output devices. The processor is contained inside the tower unit and the storage devices are the hard drive, CD-ROM drive and the diskette drive. Let us explore each of the devices in detail.

Input devices accept data in a form that the computer can utilize. Also, the input devices send the data or instructions to the processing unit to be processed into useful information. There are many examples of input devices, but the most commonly used input devices are shown below:



Figure 2



Figure 3

The input device feeds data, raw unprocessed facts, to the processing unit. The role of the processing unit or central processing unit is to use a stored program to manipulate the input data into the information required. In looking at the computer system below, the Central Processing Unit (CPU) is not exactly visible. The CPU is found inside the tall, vertical unit, called a tower, located just to the right of the monitor.



Figure 4

The CPU is the brain of the computer. The CPU consists of electronic circuits that interpret and execute instructions; it communicates with the input, output, and storage devices. The CPU, with the help of memory, executes instructions in the repetition of machine cycles.
A machine cycle consists of four steps:

1. The control unit *fetches* an instruction and data associated with it from memory.
2. The control unit *decodes* the instruction.
3. The arithmetic/logic unit *executes* the instruction.
4. The arithmetic/logic unit stores the *result* in memory.

The first two instructions are called instruction time, I-time. Steps 3 and 4 are called execution time, E-time. The speed of computer is measured in megahertz, MHz
A MHz is a million machine cycles per second. A personal computer listed at 500 MHz has a processor capable of handling 500 million machine cycles per second. Another measure of speed is gigahertz (GHZ), a billion machine cycles per second. A third measure of speed is a megaflop, which stands for one million floating-point operations per second. It measures the ability of the computer to perform complex mathematical operations.

Memory, or primary storage, works with the CPU to hold instructions and data in order to be processed. Memory keeps the instructions and data for whatever programs you happen to be using at the moment. Memory is the first place data and instructions are placed after being input; processed information is placed in memory to be returned to an output device. It is very important to know that memory can hold data only temporarily because it requires a continuous flow of electrical current. If current is interrupted, data is lost. Memory is in the form of a semiconductor or silicon chip and is contained inside the computer.



Figure 5

There are two types of memory: ROM and RAM. ROM is read only memory. It contains programs and data that are permanently recorded when the computer is manufactured. It is read and used by the processor, but cannot be altered by the user. RAM is random access memory. The user can access data in RAM memory randomly. RAM can be erased or written over at will by the computer program or the computer user. The amount of RAM has increased dramatically in recent years.

Memory is measured in bytes. A byte is usually made up of 8 bits and represents one character—a letter, digit, or symbol. The number of bytes that can be held is a measure of the memory and storage capacity. Bytes are usually measured in groups of kilobytes, megabytes, gigabytes, and terabytes. The following chart defines each term.

Table 2

Kilobyte	KB	Roughly 1,000 bytes
Megabyte	MB	Roughly 1,000,000 bytes
Gigabyte	GB	Roughly 1,000,000,000 bytes
Terabyte	TB	Roughly 1,000,000,000 bytes

Memory is usually measured in Megabytes; a typical personal computer will have 64MB or more. Storage is usually measured in Gigabytes.

Since we have said that memory is in the form of chips and must maintain a constant flow of electricity, there must be a more permanent form of storage that does not depend on a constant flow of electricity. That form of storage is called secondary or auxiliary storage. The benefits of secondary storage are large space capacity, reliability, convenience and economy.

Magnetic disk storage is a very popular type of secondary storage—the floppy disk drive is an external disk drive, while a hard disk drive is an internal disk drive. The floppy disk drive is usually a 3 ½``drive and uses a diskette made of flexible Mylar and coated with iron oxide, a substance that can be magnetized. A diskette records data as magnetized spots on the tracks of its surface. A floppy disk can hold 1.44 MBs, or a 'Zip' drive can hold 100 MBs. A hard disk, an internal disk, is a metal platter coated with magnetic oxide that can be magnetized to represent data. Hard disks come in a variety of sizes and can be assembled into a disk pack. Hard disks for personal computers are 3 ½`` disks in sealed modules. A hard disk is capable of holding a great deal more than floppy disks. Hard disks for personal computers are measured in gigabytes. (Remember, a gigabyte is roughly a thousand megabytes or a thousand floppy disks.)
While the size or data capacity of a hard drive is very important, the speed of accessing that data is equally as important. Files on hard drives can be accessed significantly faster and more conveniently than floppy drives.



Figure 6

The ever-demanding need for storage has required even better storage capacity than that of magnetic disks. Optical disk technology meets that need. Included in the list of this type of technology is the optical disk, the CD-ROM or DVD-ROM. The CD-ROM, compact disk read-only memory can hold up to 660 MBs per disk or the equivalent of more than 400 standard 3 ½`` diskettes. The new storage technology that outpaces all others is called DVD-ROM, digital versatile disk. The DVD has a 4.7 GB capacity, which is about seven times that of the CD-ROM.
A backup system is way of storing data in more than one location. Magnetic tape is usually used for this purpose. Magnetic tape is an inexpensive type of storage; it looks like the tape used in audiocassettes. Finally, the last component of a computer system is the output device. An output device displays the processed information to the user. The two most popular forms of output devices are the printer and the monitor. The monitor produces output that is temporary—the output is lost when it is rewritten or erased or when power is lost. Monitor output is called softcopy. The printer displays output in a permanent manner; it is called hardcopy. Other types of output devices include voice output and music output devices.

Caution
In order to protect the data on your hard drive, you should have a backup system.

Did You Know?
The first known publication of the term mouse as a pointing device is in Bill English's 1965 publication "Computer-Aided Display Control."

1.4.2 Software

As important as hardware devices may be, they are useless without the instructions that control them. These instructions used to control hardware and accomplish tasks are called software. Software falls into two broad categories— applications and systems software.

Applications software allows you to perform a particular task or solve a specific problem. A word processor is the most widely used example of applications software; it can be used to create a letter or memo or anything else you need to type. Other examples include games, spreadsheets, tax preparation programs, typing tutor, etc. Applications software can be purchased in stores and is called packaged or commercial software. In other words, it is prewritten. However, there may be situations that require a specific type of software that is not available. It would then be necessary to design and write a program; this software is called custom software. Most often, personal computers utilize packaged software.

When packaged software is purchased, it will come with written instructions for installation and use. These instructions are documentation. Packaged software can be purchased, or in some cases, it is available for no cost. Freeware is software considered to be in the public domain, and it may be used or altered without fee or restriction. Another form of somewhat free software is shareware. The author of shareware hopes you will make a voluntary contribution for using the product.

The most important applications software categories included in office suites are described in the table below:

Table 3: Applications software categories

Software Category	Function
Word Processor	Provides the tools for entering and revising text, adding graphical elements, formatting and printing documents.
Spreadsheets	Provides the tools for working with numbers and allows you to create and edit electronic spreadsheets in managing and analyzing information.
Database Management	Provides the tools for management of a collection of interrelated facts. Data can be stored, updated, manipulated, retrieved, and reported in a variety of ways.
Presentation Graphics	Provides the tools for creating graphics that represent data in a visual, easily understood format.
Communication Software	Provides the tools for connecting one computer with another to enable sending and receiving information and sharing files and resources.
Internet Browser	Provides access to the Internet through a service provider by using a graphical interface.

As important as applications software may be, it is not able to directly communicate with hardware devices. Another type of software is required operating systems software.

Operating Systems software is the set of programs that lies between applications software and the hardware devices. Think of the cross section of an onion (see Figure 7). The inner core of the onion represents the hardware devices, and the applications software represents the outside layer. The middle layer is the operating systems software. The instructions must be passed from the outer layer through the middle layer before the reaching the inner layer.

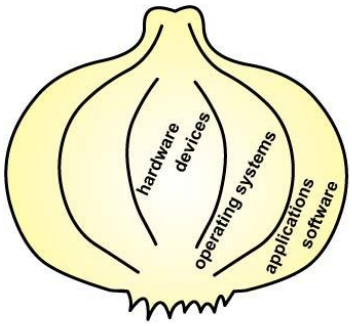


Figure .7: The onion example.

All computers, regardless of size, require the operating systems software. As soon as your personal computer is turned on, the operating systems software is loaded into RAM in order to use your computer devices and other software. A few short years ago, personal computers used an operating system call MS-DOS, Microsoft Disk Operating System. This was a command-driven program in which you needed to know command names and syntax. The need for a more user-friendly system brought about Microsoft Windows operating systems software. Icons or pictures, requiring no knowledge of spelling or syntax, drive Windows operating systems software. Windows is a GUI, graphical user interface. A GUI uses graphic symbols, icons, in its interface. Further, Windows allows you to multitask, which means that you may use more than one program at the same time. The newest version of Windows is Windows 8.
Here are the expected features of Windows 8:

Did You Know?
The first laser disk, demonstrated in 1972, was the Laservision 12-inch video disk.

Exercise: Check Your Progress 2
Note: I) Use the space below for your answer.
Ex1: Search the latest System Components available in market.

……..……

……..……

……...

1.5 System Calls

The interface between the operating system and the user programs is defined by the set of "extended instructions" that the operating system provides. These extended instructions are known as system calls. System calls provide an interface between the process and the operating system. System calls allow user-level processes to request some services from the operating system which process itself is not allowed to do. In handling the trap, the operating system will enter in the kernel mode, where it

has access to privileged instructions, and can perform the desired service on the behalf of user-level process. It is because of the critical nature of operations that the operating system itself does them every time they are needed. For example, for I/O a process involves a system call telling the operating system to read or write particular area and this request is satisfied by the operating system. System programs provide basic functioning to users so that they do not need to write their own environment for program development (editors, compilers) and program execution (shells). In some sense, they are bundles of useful system calls

Types of System Call

The main types of system calls are as follows:

Process Control: These types of system calls are used to control the processes. Some examples are end, abort, load, execute, create process, terminate process etc.

File Management: These types of system calls are used to manage files. Some examples are Create file, delete file, open, close, read, write etc.

Device Management: These types of system calls are used to manage devices. Some examples are Request device, release device, read, write, get device attributes etc.

Information Maintenance: These types of system calls are used to maintain information. Some examples are Get time or date, set time or date, get system data, set system data etc.

Communications: These types of system calls are used to for communication. Some examples are Create, delete communication connection, send, receive messages etc.

1.6 System Programs

System can be viewed as a collection of system programs. Most system supplies a large collection of system programs to solve common problems and provide a convenient environment for program development and execution. These system programs can be divided into following categories:

(a) File manipulation: These programs create, delete, copy, rename, print, dump and manipulate files and directories.

(b) Status information: Some programs need the date, time, available memory, disk space, number of users etc and then format the information and print it on the terminal or file or on some output device.

(c) File modification: A number of text editors are provided to create the file and manipulate their contents.

(d) Programming language support: A number of compilers, assemblers, and interpreters are provided for common programming languages such as C, Basic etc. with the operating systems.

(e) Program loading and execution: With operating system loaders and linkers are provided which are required to load and execute the program after their compilation. There are different types of loaders such as absolute loader, reloadable loader, overlay loaders, and linkage editors etc.

(f) Application programs: In addition to above a number of common programs provided with operating system are database systems, compilers-compiler, statistical analysis package, text formatters etc.

The most important system program for an operating system is its command interpreter. It is the program which reads and interprets the commands given by the user. This program is also known as control card interpreter or command line interpreter or the console command processor (in CP/M) or the shell (In Unix). Its function is simple: get the next command and execute it. The commands given to the command interpreter are implemented in two ways. In one approach the command interpreter itself contains the code to execute the command. So the numbers of commands that can be given

determine the size of the command interpreter. An alternative approach implements all commands by special system programs. So the command interpreter merely uses the command to identify a file to be loaded into memory and executed. Thus a command delete X would search for a file called delete, load it into the memory and pass it the parameter X. In this approach new commands can be easily added to the system by creating new files of the proper name. The command interpreter, which can now be quite small, need not be changed in order to add new commands.

Self Assessment Questions

1. Which of the command is used to remove files?

(a). erase (b). rm
(c). delete (d). Dm

2. The commonly used UNIX commands like date, Is, cat, etc. are stored in

(a). /dev directory (b). /bin and /usr/bin directories
(c). /UNIX directory (d). /tmp directory

3. When a computer is first turned on or restarted, a special type of absolute loader called .....is executed

(a). Compile and Go loader (b). Boot loader
(c). Bootstrap loader (d). Relating loader

4. Which of the following Operating systems is better for implementing a Client-Server network

(a). MS DOS (b). Windows 95
(c). Windows 98 (d). Windows 2000

1.7 Structure, Design and, Implementation

OS Structure

The computer architecture of a computing system defines its attributes as seen by the programs that are executed in that system, that is, the conceptual structure and functional behaviour of the machine hardware. Then, the computer architect defines the *functions* to be executed in the hardware and the *protocol* to be used by the software in order to exploit such functions. Note that the architecture has nothing to do with the organization of the data flow, the logical design, the physical design, and the performance of any particular implementation in the hardware.

Hence by Architecture we mean the order in which certain hardware Processes are carried out by the OS and has nothing to do with the logical software flow of the Computer.

Simple View

An Operating System is the layer between the hardware and software, as in Figure 8.

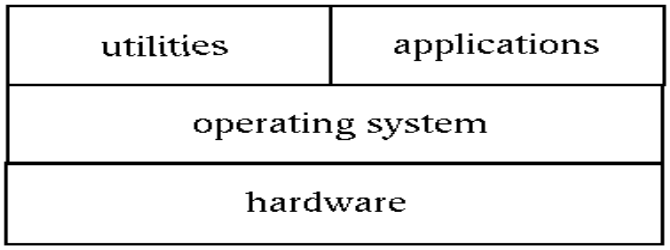


Figure 8: Operating System is the layer between the hardware and software.

An Operating System is responsible for the following functions

- Device management using device drivers
- Process management using processes and threads
- Inter-process communication
- Memory management
- File systems

In addition, all operating systems come with a set of standard utilities. The utilities allow common tasks to be performed such as

- being able to start and stop processes
- being able to organise the set of available applications
- organise files into sets such as directories
- view files and sets of files
- edit files
- rename, copy, delete files
- communicate between processes

Kernel

The *kernel* of an operating system is the part responsible for all other operations. When a computer boots up, it goes through some initialisation functions, such as checking memory. It then loads the kernel and switches control to it. The kernel then starts up all the processes needed to communicate with the user and the rest of the environment (e.g. the LAN)

The kernel is always loaded into memory, and kernel functions always run, handling processes, memory, files and devices.

The traditional structure of a kernel is a *layered* system, such as UNIX. In this, all layers are part of the kernel, and each layer can talk to only a few other layers. Application programs and utilities live above the kernel.

The UNIX kernel (see Figure 9)

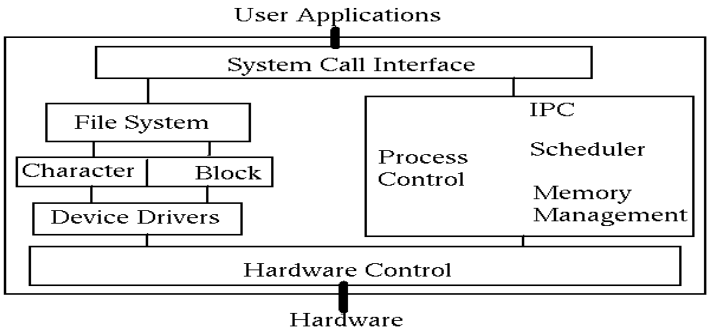


Figure 9: UNIX kernel looks.

Most of the Operating Systems being built now use instead a *micro kernel*, which minimises the size of the kernel. Many traditional services are made into user level services. Communication being services is often by an explicit *message passing* mechanism.

The major micro-kernel Operating System is Mach. Many others use the concepts of Mach. (see Figure 1.0)

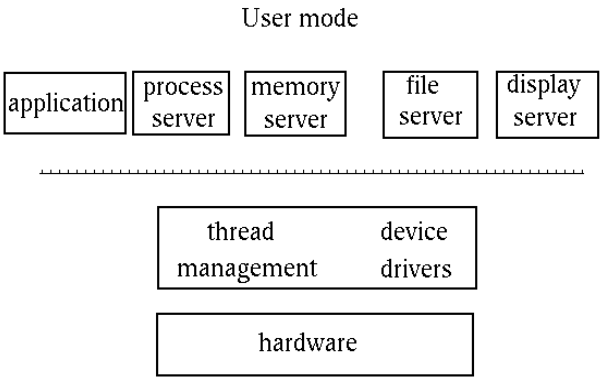


Figure 10: Micro-kernel Operating System.

Some systems, such as Windows NT use a mixed approach (see Figure 1.10)

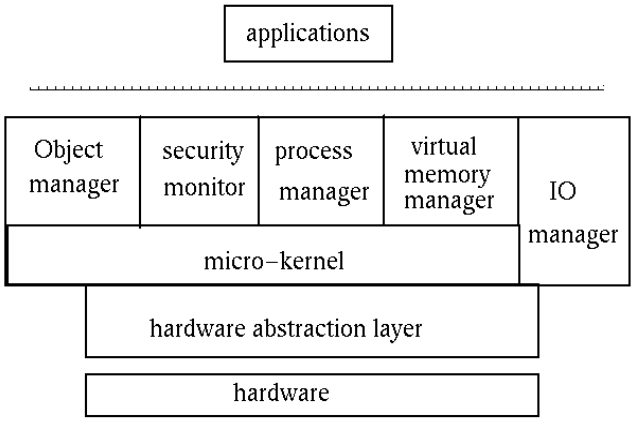


Figure 1.10: Windows NT structure.

1.8 Operating System Generation

Operating systems have evolved through a number of distinct phases or generations which corresponds roughly to the decades:

The 1940's - First Generations

The earliest electronic digital computers had no operating systems. Machines of the time were so primitive that programs were often entered one bit at time on rows of mechanical switches (plug boards). Programming languages were unknown (not even assembly languages).

The 1950's - Second Generation

By the early 1950's, the routine had improved somewhat with the introduction of punch cards. The General Motors Research Laboratories implemented the first operating systems in early 1950's for their IBM 701. The system of the 50's generally ran one job at a time. These were called single-stream batch processing systems because programs and data were submitted in groups or batches.

The 1960's - Third Generation

The systems of the 1960's were also batch processing systems, but they were able to take better advantage of the computer's resources by running several jobs at once. So operating systems

designers developed the concept of multiprogramming in which several jobs are in main memory at once; a processor is switched from job to job as needed to keep several jobs advancing while keeping the peripheral devices in use. For example, on the system with no multiprogramming, when the current job paused to wait for other I/O operation to complete, the CPU simply sat idle until the I/O finished. The solution for this problem that evolved was to partition memory into several pieces, with a different job in each partition. While one job was waiting for I/O to complete, another job could be using the CPU.

Fourth Generation

With the development of LSI (Large Scale Integration) circuits, chips, operating system entered in the system entered in the personal computer and the workstation age. Microprocessor technology evolved to the point that it becomes possible to build desktop computers as powerful as the mainframes of the 1970s. Two operating systems have dominated the personal computer scene: MS-DOS, written by Microsoft, Inc. for the IBM PC and other machines using the Intel 8088 CPU and its successors, and UNIX, which is dominant on the large personal computers using the Motorola 6899 CPU family.

Exercise: Check Your Progress 3

Case Study-History of the Android OS

Android seems to be getting everywhere, from cell phones to tablet pc's and even TV sets. But how did the potentially Apple beating OS come into being?

Google, who buy up many small businesses, brought a small company back in 2005 called Android Inc. The little known company developed software for mobile phones. When the mighty iPhone was launched in June 2007, bringing touch screens to the world there were rumours that Google wanted in on the mobile phone business, to go in tandem with it is mobile search features. A new Google handset was rumoured, and it was said that Google was courting the major manufacturers and carriers. The phone would implement many Google Labs' ideas and Google maps and Gmail.

It was then revealed that Google had been in cahoots with other manufacturers had been developing a new open-source OS for mobile devices which was the beginning of the Open Handset Alliance (OHA). This group included LG, HTC, T-Mobile, Samsung, Motorola, Intel, and Qualcomm amongst others. They all got together to develop an open standard for mobile devices based on Linux.

The OS has been available since Oct 2008 for developers as open-source software.

The mobile OS was first introduced to buyers in late 2008 with the HTC Dream (also called T-Mobile G1), and started getting really popular with the launch of HTC Hero in July 2009.

Many started calling the new OS a serious rival to iPhone, and by February 2010 there were 60,000 plus Android phones shipping daily. The App Market now boasts over 30,000 apps to download. Still has a way behind the iPhone, but app creators now bring out apps for both systems.

Android has had many updates from v1.0. They released v1.5 which added camcorder functionality and home screen widgets. Then v1.6 came along with voice search and an improved Android App Market. Larger screens came with v2.0 and satnav. Then came v2.1 which brought extra home screens and animated wallpapers amongst other features.

Currently there are many manufacturers developing and releasing Android hardware, from smart phones, tablet PC's, e-book readers, not forgetting Google TV.

Back in March 2010 Google has partnered with Sony and Intel to create a TV platform powered by the Android OS. It is mission to merge the TV and the Internet together bringing it into the home using a new generation of televisions, blu-ray players and set-top boxes. Google have also partnered with Dish Network to integrate Google TV within their DVR's.

Questions

1. How the Android operating system first introduced to buyers?
2. Write the benefits of Android operating system.

1.9 Summary

- Kernel is a program that constitutes the central core of a computer operating system.
- Operating system (OS) is a software program that manages the hardware and software resources of a computer.
- System call is the mechanism used by an application program to request service from the operating system.
- An operating system that utilizes multitasking is one that allows more than one program to run simultaneously.
- Symmetric multi processing (SMP) involves a multiprocessor computer architecture where two or more identical processors connect to a single shared main memory.

1.10 Keywords

Asymmetric multi processing: Asymmetric hardware systems commonly dedicated individual processors to specific tasks.

MISD multi processing: Multiple Instructions, Single Data is a type of parallel computing architecture where many functional units perform different operations on the same data.

Real time operating system (RTOS): Real-time operating systems are used to control machinery, scientific instruments and industrial systems such as embedded systems.

SIMD multi processing: In a single instruction stream, multiple data stream computer one processor handles a stream of instructions, each one of which can perform calculations in parallel on multiple data locations.

UNIX-like: UNIX-like family is a diverse group of operating systems, with several major subcategories including System V, BSD, and Linux.

1.11 Review Questions

1. What does an operating system do?
2. What are the three main purposes of an operating system?
3. List the four steps needed to run a program on a completely dedicated machine.
4. What are the main differences between operating systems for mainframe computers and PCs?
5. How are network computers different from traditional personal computers?
6. What are the tradeoffs inherent in handheld computers?
7. What are the advantages and disadvantages of using the same system call interface for manipulating both files and devices?

8. What is the purpose of the command interpreter? Why is it usually separate from the kernel? Would it be possible for the user to develop a new command interpreter using the system-call interface provided by the operating system?
9. Why is the separation of mechanism and policy desirable?
10. In what ways is the modular kernel approach similar to the layered approach? In what ways does it differ from the layered approach?

Answers for Self Assessment Questions

1.(b) 2. (b) 3. (c) 4.(d)

1.12 Further Studies

1. Operating Systems : Internals and Design Principles 6 Edition William Stallings
2. Operating System Principles by Silberschatz, Galvin and Gagne

2

Process

STRUCTURE

2.0 Objectives

2.1 Introduction

2.2 Process Concept

2.3 Description and Control

2.4 Process State Model

2.5 Process Description-PCB

2.6 Threads

2.7 Threads in Linux

2.8 Summary

2.9 Keywords

2.10 Review Questions

2.11 Further Studies

2.0 Objectives

After studying this chapter, you will be able to:

- Define process
- Describes the process state model
- Explain the description and control of process
- Describes the PCB, process control
- Understand the threads
- Explain the threads in LINUX

2.1 Introduction

A process is a program that is running on your computer. This can be anything from a small background task, such as a spell-checker or system events handler to a full-blown application like Internet Explorer or Microsoft Word. All processes are composed of one or more threads.

Since most operating systems have many background tasks running, your computer is likely to have many more processes running than actual programs. For example, you may only have three programs running, but there may be twenty active processes. You can view active processes in Windows by

opening the Task Manager (press Ctrl-Alt-Delete and click Task Manager). On a Mac, you can see active processes by opening Activity Monitor (in the Applications→Utilities folder).
The term "process" can also be used as a verb, which means to perform a series of operations on a set of data. For example, your computer's CPU processes information sent to it by various programs.

2.2 Process Concept

A process is sequential program in execution. A process defines the fundamental unit of computation for the computer. Components of process are :

1. Object Program
2. Data
3. Resources
4. Status of the process execution.

Object program i.e. code to be executed. Data is used for executing the program. While executing the program, it may require some resources. Last component is used for verifying the status of the process execution. A process can run to completion only when all requested resources have been allocated to the process. Two or more processes could be executing the same program, each using their own data and resources.

Processes and Programs:
Process is a dynamic entity that is a program in execution. A process is a sequence of information executions. Process exists in a limited span of time. Two or more processes could be executing the same program, each using their own data and resources.
Program is a static entity made up of program statement. Program contains the instructions. A program exists at single place in space and continues to exist. A program does not perform the action by itself.

Did You Know?
Process is one of the fundamental concepts in modern operating systems. It was first introduced by the designers of Multicast in the 1960s.

2.2.1 Process State

- When process executes, it changes state. Process state is defined as the current activity of the process. Fig. 2.1 shows the general form of the process state transition diagram. Process state contains five states. Each process is in one of the states. The states are listed below.
    1. New
    2. Ready
    3. Running
    4. Waiting
    5. Terminated(exist)

1. New: A process that just been created.
2. Ready: Ready processes are waiting to have the processor allocated to them by the operating system so that they can run.
3. Running: The process that is currently being executed. A running process possesses all the resources needed for its execution, including the processor.

4. Waiting: A process that cannot execute until some event occurs such as the completion of an I/O operation. The running process may become suspended by invoking an I/O module.
5. Terminated: A process that has been released from the pool of executable processes by the operating system.

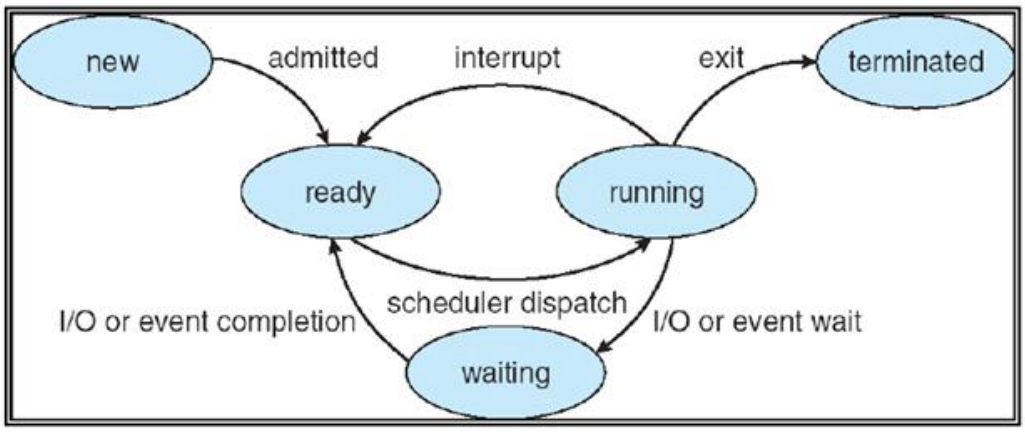


Figure 2.1 Diagrams for Process State

- Whenever processes changes state, the operating system reacts by placing the process PCB in the list that corresponds to its new state. Only one process can be running on any processor at any instant and many processes may be ready and waiting state.

Suspended Processes

Characteristics of suspend process

1. Suspended process is not immediately available for execution.
2. The process may or may not be waiting on an event.
3. For preventing the execution, process is suspended by OS, parent process, process itself and an agent.
4. Process may not be removed from the suspended state until the agent orders the removal. Swapping is used to move all of a process from main memory to disk. When all the process by putting it in the suspended state and transferring it to disk.

Reasons for process suspension

1. Swapping: OS needs to release required main memory to bring in a process that is ready to execute.
2. Timing: Process may be suspended while waiting for the next time interval.
3. Interactive user request: Process may be suspended for debugging purpose by user.
4. Parent process request: To modify the suspended process or to coordinate the activity of various descendants.

Process Control Block (PCB):

Each process contains the process control block (PCB). PCB is the data structure used by the operating system. Operating system groups all information that needs about particular process. Fig. 2.2 shows the process control block.

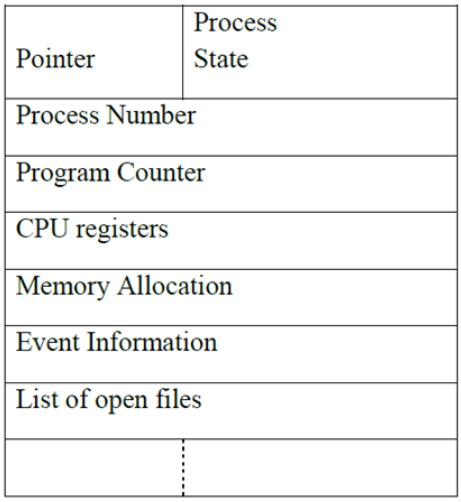


Figure 2.2 Process Control Block

1. **Pointer:** Pointer points to another process control block. Pointer is used for maintaining the scheduling list.
2. **Process State:** Process state may be new, ready, running, waiting and so on.
3. **Program Counter:** It indicates the address of the next instruction to be executed for this process.
4. **Event information:** For a process in the blocked state this field contains information concerning the event for which the process is waiting.
5. **CPU register:** It indicates general purpose register, stack pointers, index registers and accumulator's etc. number of register and type of register totally depends upon the computer architecture.

1. Memory Management Information: This information may include the value of base and limit register. This information is useful for deal locating the memory when the process terminates.
2. Accounting Information: This information includes the amount of CPU and real time used time limits, job or process numbers, account numbers etc.

- Process control block also includes the information about CPU scheduling, I/O resource management, file management information, priority and so on. The PCB simply serves as the repository for any information that may vary from process to process.
- When a process is created, hardware registers and flags are set to the values provided by the loader or linker. Whenever that process is suspended, the contents of the processor register are usually saved on the stack and the pointer to the related stack frame is stored in the PCB. In this way, the hardware state can be restored when the process is scheduled to run again.

2.3 Description and Control

Process Control is the active changing of the process based on the results of process monitoring. Once the process monitoring tools have detected an out-of-control situation, the person responsible for the process makes a change to bring the process back into control.

2.3.1 Process Description

Operating systems are considered as a manager of the underneath various hardware resources; however this management is not the extreme goal, but a way in which the processes may access the resources reasonably. Thus we can think of the operating system as an entity that manages the use of system resources by processes, which may be illustrated in Figure 2.3, where a solid arrow from a process to a resource entity indicates the former occupies the latter, while a dotted one means a process needs the corresponding resource but does not occupy it anyway. Figure 2.3 actually shows a snapshot of the resource allocation.

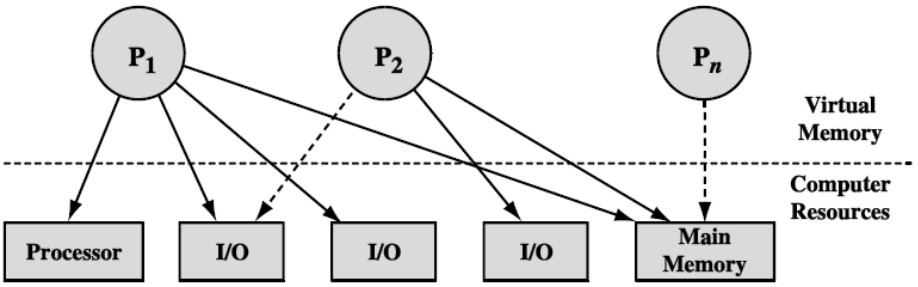


Figure 2.3: Processes and resources.

An operating system, as a service provider, naturally also imposes control on processes that consume the services. That is both processes and resources are under the operating system's control. To enable this control, some facilities must be provided, which is control tables we are covering.

Control Structures

The operating system constructs and maintains tables of information about each entity that it is managing. Figure 2.4 illustrates that four different types of tables are maintained by the operating system:

• *Memory tables:* are used to keep track of both main memory and secondary memory. Part of main memory is reserved for use by the operating system; the remainder is available for use by processes.

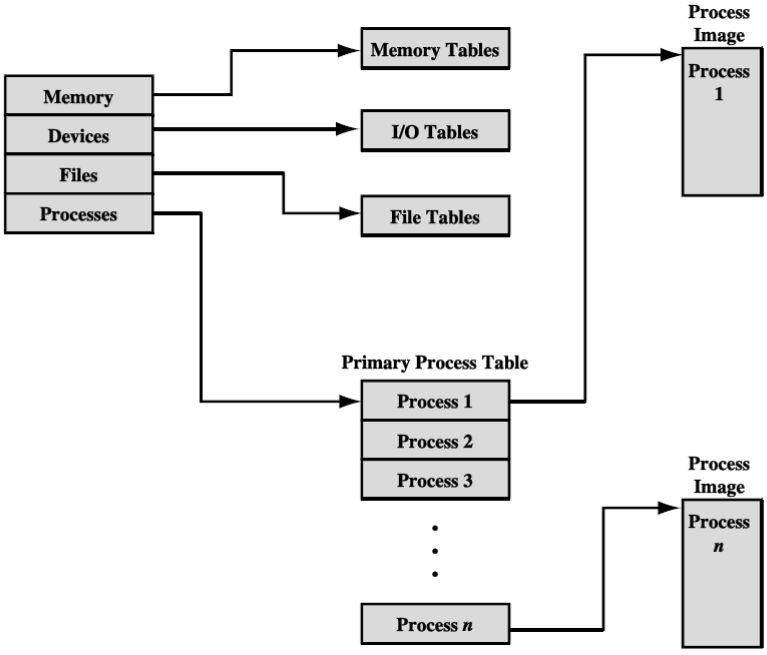


Figure 2.4: Operating system control tables.

Processes may also resize on secondary memory due to the utilization of swapping mechanism. The memory tables generally include the following information:

- The allocation of main memory to processes
- The allocation of secondary memory to processes
- Protection attributes of blocks of main or virtual memory

• *I/O tables:* are used by the operating system to manage I/O devices. They should record:
  - The availability of each particular device
  - The status of I/O operations relating to each device and the location in main memory being used as the source or destination of the I/O transfer.

• *File tables:* provides information about
  - the existence of files
  - their location on secondary memory
  - their current status and attributes

• *Process tables:* contains what the operating system must know to manage and control processes, including:

Process location

First a program statically consists of a set of instructions that manipulate data, thus the operating system needs to allocate space for its code and data. In addition, the dynamic execution of a program requires a stack that is used to keep track of procedure calls and parameter passing between procedures.

Finally, each process has associated with it a number of attributes that are used by the operating system for process control. Typically the operating system needs to maintain a structure called process control block (PCB) containing these attributes. The collection of the code, data, stack, and attributes is referred to as process image.

The location of a process image depends on how the memory management is implemented. Most modern operating systems use a memory management scheme in which a process image consists of a set of blocks that need not be stored contiguously. The blocks may be variable length (usually called segments), or fixed length (called pages), or a combination. This scheme allows the operating system to bring in only a portion of any particular process. Therefore process tables must show the location of each segment and/or page of each process image. Figure 2.5 depicts a primary process table with one entry for each process.

Process attributes

Process attributes are stored in PCB. Different systems organize this information in different ways, so here we examine only what type of information should be included as attributes, instead of how. The typical elements of a PCB are:

Process identification

All operating systems need to assign a unique identifier to each process in the systems so as to refer to them conveniently. The identifier may be numeric, and for example simply the index of the corresponding entry in the primary process table; otherwise, a mapping must be available to allow the operating system to locate the right entry based on the identifier.

Almost every process is created by another process, its parent process, on behalf of a user. Their identifiers should also be present in the PCB of the child process.

Process state information

It consists of the contents of processor registers. Typically the context includes user-visible registers, control and status registers, and stacks pointers.

Process control information

This may include scheduling and state information, data structuring, process privileges, and resource ownerships. For example in Figure 2.6, the various queues that we discussed in the preceding class could be implemented as linked lists of PCBs with the help of the data structuring fields.

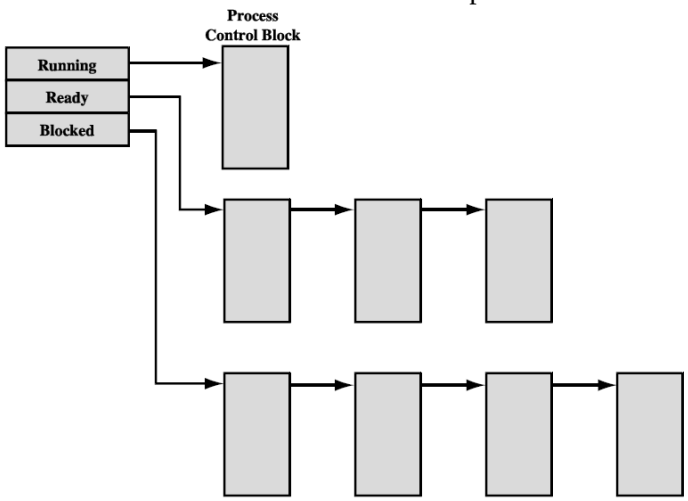


Figure 2.5: Process list structures

The role of the PCB

The PCB is the most important structure in an operating system in the sense that each PCB contains all of the information about a process and the set of PCBs defines the state of the operating system. Its unique position in the operating system thus brings up an important design issue. Within the operating system, a number of modules will need access to PCB. If the direct access is allowed, then a bug in some module may damage PCBs, which could cause the inability of the operating system to manage the processes. In addition, a design change in the structure or semantics of the PCB could affect a number of modules in the system. A solution to it is setting up an interface for visiting PCBs, which regulates all access and thus separates PCBs from the other parts of the operating system. The strategy is similar to the object-orientation method, where local variables are invisible and cannot be visited directly, and public methods make up the access interface.

Caution

When a process is interrupted, the context of the processor must be saved so that it can be restored when the process resumes execution.

2.3.2 Process Control

Models of execution

An operating system are in nature software, consisting of a set of instructions, however it plays a unique role in the computer system since the operating system can do some privileged operations, such as accessing PCBs, while user programs cannot.

To meet this requirement, most processors support at least two modes of execution: the less-privileged one or user mode, and the more-privileged one or kernel mode (also called system mode). The operating system runs in the kernel mode and thus may use certain instructions that can only be

executed in this mode. In this mode, the software has complete control of the processor registers and memory. User programs run in the user mode, and may not read or alter a control register, or access certain regions of memory.

With multiple modes, there are two questions to be answered:

- How does the processor know in which mode it is to be executed? Typically there is a bit in PSW that indicates the mode of execution.
- How is the mode changed? This is done by executing an instruction that changes the mode, for example the CHM (Change Mode) instruction on the VAX1 machines. When the user makes a system service call or when an interrupt transfers control to a system routine, the routine executes CHM to enter the kernel mode and executes it again to enter the user mode before returning control to the user process.

Process creation

Before we have roughly discussed what happens when a process is created, now it is clearer what is involved in the period after we know the control structures for processes. First, a unique identifier is assigned to the new process and a new entry is added to the primary process table. Second, the space for all elements of a process image is allocated, and then the PCB is initialized. The values for some fields of the PCB structure are already known, e.g. the ID of the parent process, program counter, and system stack pointers, while others fields will use default values, e.g. the process state initialized to Ready or Ready/Suspend, or simply be filled with zero. Finally the PCB will be linked into some data structures and other structures may be created for billing and performance assessment purposes.

Process switching

When to switching processes

It is mentioned before that multiprogramming systems perform process switching when the currently running process has to wait for the completion of I/O operation and thus keep the processor busy for higher efficiency. Upon the arrival of interrupt signal, the operating system will move all the blocked processes waiting for the signal to the Ready state, and the decide whether to resume execution of the process currently in the Running state or to pre-empt that process for a higher-priority Ready process. This relates to process scheduling, what we will talk about next week.

Change of process state

When a process switching occurs, the following steps will be taken:

1. Save the context of the processor, including PC and other registers.
2. Update the related fields in the PCB, e.g. the state of the process.
3. Move the PCB to the appropriate queue.
4. Select another process for execution.
5. Update the PCB of the process selected.
6. Load the values of PC and other registers from the PCB, and thus restore the context of the processor to the one when this selected process was last switched out.

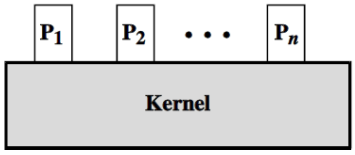
Although interrupts usually cause process switching, they do not necessarily do so.

In some operating systems, the occurrence of an interrupt may simply involve a mode switching. That is after the interrupt handler has executed the currently running process will resume execution. In this case, all that is necessary is to save the processor state information when the interrupt occurs and restore that information when control is returned to the program that was in progress. The advantage of the mode switch is little overhead involved.

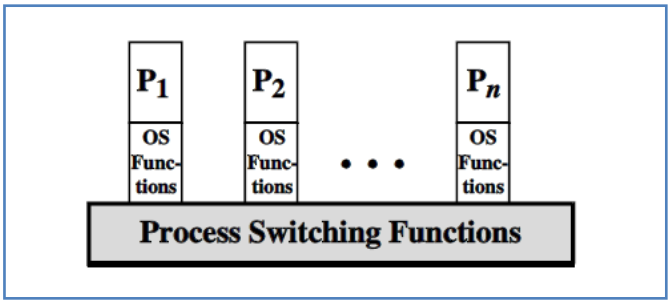
Execution of the operating system

We know programs run in a computer system as processes, but what about the operating system itself? Does it also take the form of process? Actually with different design approaches of an operating system, the answers also vary. Figure 2.8 shows three popular options.

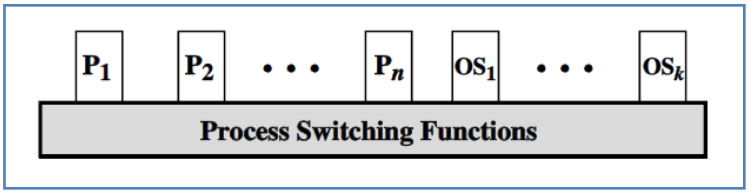
Non-process kernel



(a) Separate kernel.



(b) OS functions execute as separate processes.



(c) OS functions execute as separate processes.

Figure 2.6: Relationship between operating system and user processes.

Execution within user processes

An alternative is to execute virtually all operating system software in the context of a user process. That is that the operating system is primarily a collection of routines that the user calls to perform various functions, executed within the environment of the user's process. This is shown in Figure 2.6 (b). The advantage of this approach is that when an interrupt, trap, or supervisor call occurs, only a mode switching happens. Upon completion of its work, depending on system design, the operating system may or may not continue to execute the current process; in the latter case, a process switching occurs

Process-based operation system

Another alternative, illustrated in Figure 2.6 (c), is to implement the operating system as a collection of system processes. Although the kernel still executes in a kernel mode, major kernel functions are organized as separate processes.

This approach has several advantages:

- It encourages the use of a modular operating system with clean interfaces between the modules.
- Some noncritical operating system functions are conveniently implemented as separate processes.

- The form of processes makes it possible to distribute different parts of the operating system in a multiprocessor environment, improving performance.

Process control refers to the control of one or more system parameters, such as temperature, flow rate or position. While most systems are a continual process, such as maintaining a temperature, other processes may be a sequence of actions, for example, the assembly of a product.

Control systems can be very simple or very complex. For control of the process, an input (such as a set point control or switch) is required into the controller. Based on the input, the controller will drive an actuator to cause the desired effect on the process. Examples of actuators are sheaters for temperature, pumps for flow, and servos for positioning.

Did You Know?

Statistical process control was pioneered by Walter A. Shewhart in the early 1920s.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Define Suspend Process.

……..……

……..……

……...

Ex2: Control Structure.

……..……

……..……

……...

2.4 Process State Model

The above two-state model, to some extent, reflects the behaviours of processes; however the real situation is much more complex. We cannot simply deal with all the processes that are not running in the same way. For example, in a time sharing system, a process may lose control of the processor just because of timeout, i.e. its time slot is used up. Although it moves to the Not Running state, it is always ready to run right away. But some processes stopped running to wait for the completion of I/O operation. Before the interrupt signal notifies the completion, it cannot run and make any progress, and have to wait. Thus it is not proper to use a single state and a single for all the not-running processes.

Naturally, the Not Running state is to be split into two states: Ready and Blocked. Thus if the creation period and termination period of a process are also considered states, we will obtain a five-state model as depicted in Figure 2.7.

During the lifespan of a process, its execution status may be in one of four states: (associated with each state is usually a queue on which the process resides)

- *Executing:* the process is currently running and has control of a CPU
- *Waiting:* the process is currently able to run, but must wait until a CPU becomes available
- *Blocked:* the process is currently waiting on I/O, either for input to arrive or output to be sent.

- *Suspended:* the process is currently able to run, but for some reason the OS has not placed the process on the ready queue.
- *Ready:* the process is in memory, will execute given CPU time.

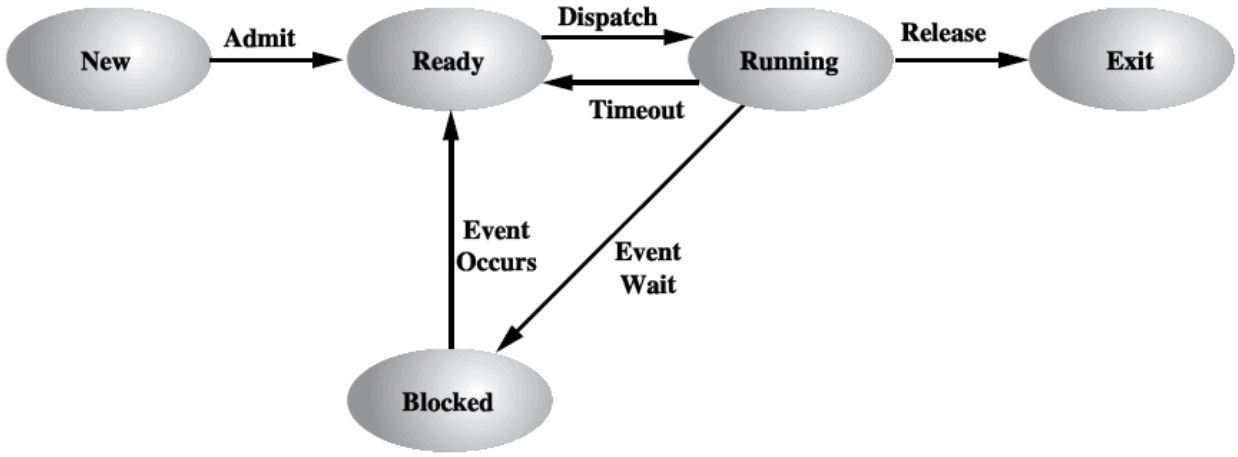
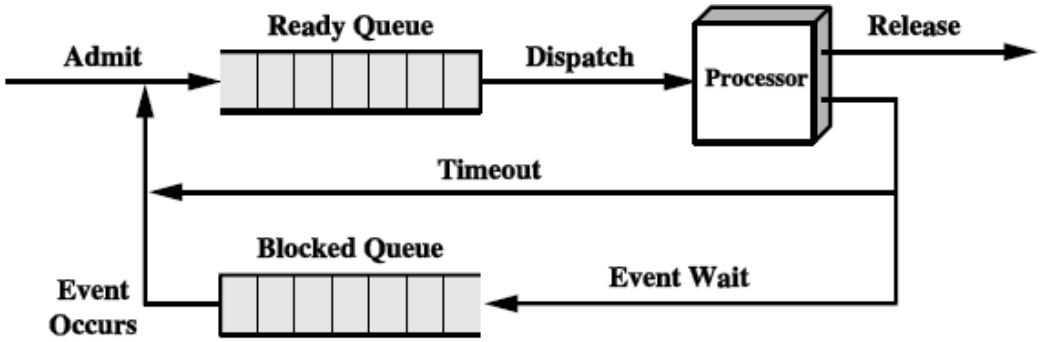


Figure 2.7: Five-State process model.

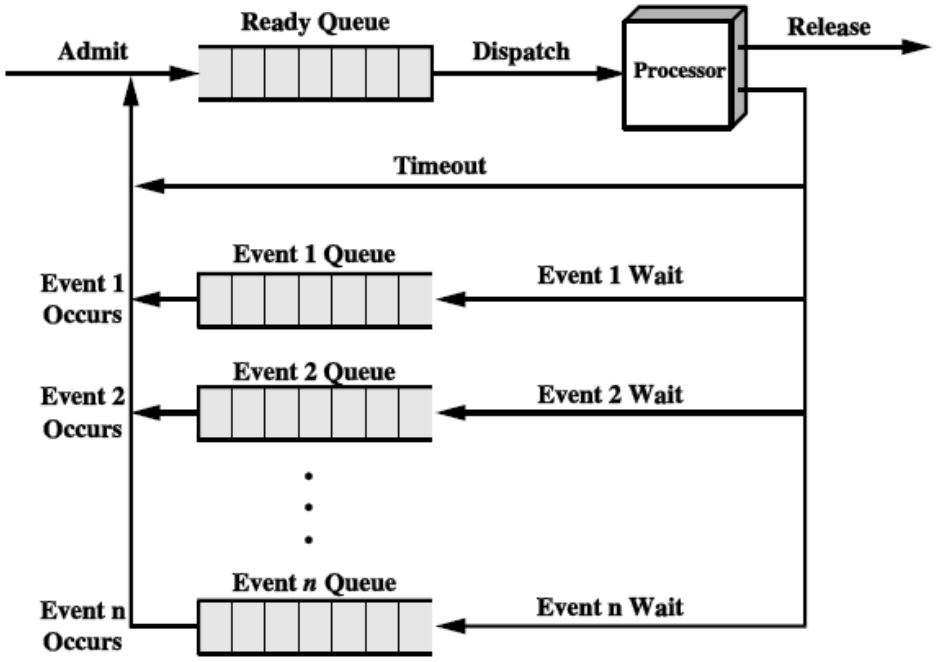
Processes in the Blocked state cannot execute until some event occurs, such as the completion of I/O operation, while a ready process is always prepared to execute when given the opportunity.
Correspondingly, the queuing diagram in Figure 2.7 (b) may be extended to reflect this five-state model. Figure 2.7 (a) differentiates the blocked processes and the ones that may be dispatched again immediately by giving two paths from the Running state to the Ready state. An additional queue is set up for blocked processes. It means that when an event occurs, the dispatcher will go all the way through the queue for those processes waiting for that event. In some cases, there may be hundreds or even more processes in that queue, therefore it would be more efficient to have a number of queues, one for each event. Thus Figure 2.8 (b) is obtained.

2.4.1 Process Swapping

The five-state model provides a systematic way of modelling the behaviour of processes. Many operation systems are indeed constructed using this model.
However there are still problems. Recall that the reason for introducing the process concept and process state transition model is that there is a huge gap between the speed of CPU and I/O devices.



(a) Single blocked queue.



(b) Multiple blocked queue.

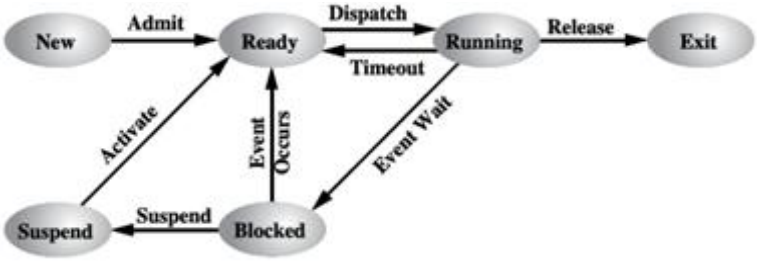
Figure 2.8: Queuing model.

When a process has to wait for the completion of I/O operation and thus gives up the processor, dispatching control to another process may avoid the idle of the processor. But this arrangement does not entirely solve the problem. The processor may be so much faster than I/O that all of the processes in memory are waiting for I/O. Thus even with multiprogramming, a processor could be idle for a long time.

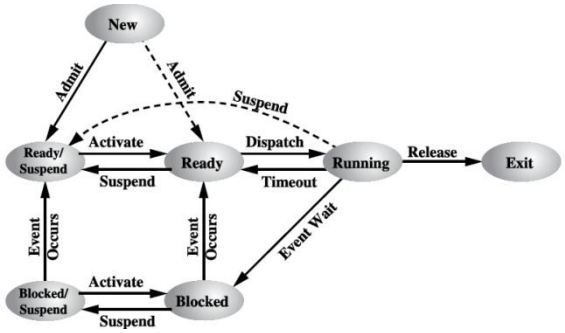
Why not expand main memory so that more processes may be accommodated? It is possible, but cannot be a once-for-all solution since more memory means higher cost and with more memory the average size of programs is also likely to increase.

A real solution is swapping, which involves moving part or all of a blocked process from main memory to disk. Thus memory space may be freed for the system to bring in new processes to run. With swapping, a new state, Suspend, must be added to the process behaviour model, as depicted in Figure 2.9 (a)

However a single Suspend state is still not enough, since the system needs to distinguish the suspended processes that remain blocked and those that are though suspended and residing in pended and residing in secondary memory but are available for execution as soon as they are loaded into main memory.



(a) With one suspend state.



(b) With two suspend state.

Figure 2.9: Process State Transition Diagram with Suspend States.

Accordingly, two Suspend states, Blocked/Suspend and Ready/Suspend, are introduced in Figure 2.9 (b). A process may move from Blocked/Suspend to Ready/Suspend when the event for which it has been waiting happens. All processes in either state may be brought back into main memory.

Did You Know?

The concept of debt-for-nature swaps was first conceived by Thomas Lovejoy of the World Wildlife Fund in 1984 as an opportunity to deal with the problems of developing-nation indebtedness and its consequent deleterious effect on the environment.

Self Assessment Questions

1. PCB stands for

(a) Program Control Block (b) Process Control Block
(c) Process Communication Block (d) None of the above.

2. FIFO scheduling is……..

(a) Pre-emptive Scheduling (b) Non Pre-emptive Scheduling
(c) Deadline Scheduling (d) Fair share scheduling.

3. Switching the CPU to another Process requires to save state of the old process and loading new process state is called as……..

(a) Process blocking (b) context switch
(c) Time sharing (d) none of the above.

4. Which directory implementation is used in most Operating System?

(a) Single level directory structure (b) Two level directory structure
(c) Tree directory structure (d) acyclic directory structure.

5. The state of a process after it encounters an I/O instruction is……..

(a) ready (b) blocked/Waiting
(c) Idle (d) running.

2.5 Process Description-PCB

Each process is represented in the operating system by a process control block (PCB)-also called a task control block. A PCB is shown in Figure 2.10. It contains many pieces of information associated with a specific process, including these:

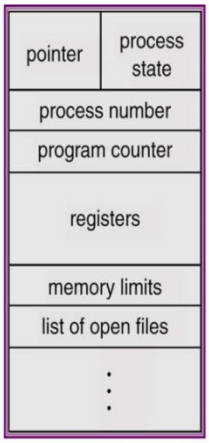


Figure 2.10: Process control block (PCB).

Process state: The state may be new, ready, running, and waiting, halted, and so on.
Program counter: The counter indicates the address of the next instruction to be executed for this process.
CPU registers: The registers vary in number and type, depending on the computer architecture. They include accumulators, index registers, stack pointers, and general-purpose registers, plus any condition-code information.
Along with the program counter, this state information must be saved when an interrupt occurs, to allow the process to be continued correctly afterward (Figure 2.11).
CPU-scheduling information: This information includes a process priority, pointers to scheduling queues, and any other scheduling parameters. Memory-management information: This information may include such information as the value of the base and limit registers, the page tables, or the segment tables, depending on the memory system used by the operating system.
Memory-management information: This information may include such information as the value of the base and limit registers, the page tables, or the segment tables, depending on the memory system used by the operating system.

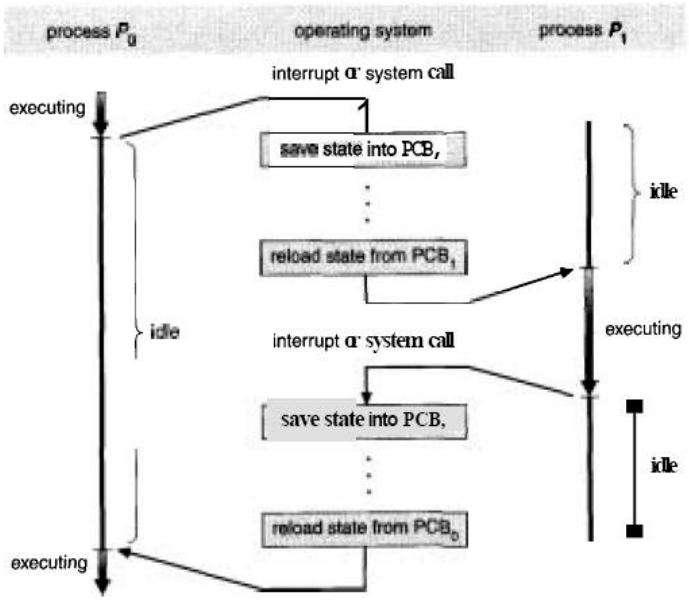


Figure 2.11: Diagram showing CPU switch from process to process.

Accounting information: This information includes the amount of CPU and real time used, time limits, account numbers, job or process numbers, and so on.

Status information: The information includes the list of I/O devices allocated to this process, a list of open files, and so on. The PCB simply serves as the repository for any information that may vary from process to process.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Process Swapping.

……..……

……..……

……...

Ex2: Process Description in PCB.

……..……

……..……

……...

2.6 Threads

A thread is the smallest unit of processing that can be performed in an OS. In most modern operating systems, a thread exists within a process - that is, a single process may contain multiple threads.

2.6.1 Processes Vs Threads

As we mentioned earlier that in many respect threads operate in the same way as that of processes. Some of the similarities and differences are:

Similarities

- Like processes threads share CPU and only one thread active (running) at a time.
- Like processes, threads within processes, threads within processes execute sequentially.
- Like processes, thread can create children.
- And like process, if one thread is blocked, another thread can run.

Differences

- Unlike processes, threads are not independent of one another.
- Unlike processes, all threads can access every address in the task.
- Unlike processes, threads are design to assist one other. Note that processes might or might not assist one another because processes may originate from different users.

Why Threads?

Following are some reasons why we use threads in designing operating systems.

1. A process with multiple threads makes a great server for example printer server.
2. Because threads can share common data, they do not need to use undercrosss communication.
3. Because of the very nature, threads can take advantage of multiprocessors.

2.7 Threads in Linux

Threads are “light weight processes” (LWPs). The idea is a process has five fundamental parts: code (“text”), data (VM), and stack, file I/O, and signal tables. “Heavy-weight processes” (HWPs) have a significant amount of overhead when switching: all the tables have to be flushed from the processor for each task switch. Also, the only way to achieve shared information between HWPs is through pipes and “shared memory”. If a HWP spawns a child HWP using fork(), the only part that is shared is the text.
Threads reduce overhead by sharing fundamental parts. By sharing these parts, switching happens much more frequently and efficiently. Also, sharing information is not so “difficult” anymore: everything can be shared. There are two types of threads: user-space and kernel-space.

2.7.1 User-Space Threads

User-space avoids the kernel and manages the tables itself. Often this is called “cooperative multitasking” where the task defines a set of routines that get “switched to” by manipulating the stack pointer. Typically each thread “gives-up” the CPU by calling an explicit switch, sending a signal or doing an operation that involves the switcher. Also, a timer signal can force switches. User threads typically can switch faster than kernel threads.

Disadvantages: User-space threads have a problem that a single thread can monopolize the time slice thus starving the other threads within the task. Also, it has no way of taking advantage of SMPs (Symmetric Multiprocessor systems, e.g. dual-/quad-Pentiums). Lastly, when a thread becomes I/O blocked, all other threads within the task lose the time slice as well.
Solutions/work around. Some user-thread libraries have addressed these problems with several work-around. First time slice monopolization can be controlled with an external monitor that uses its own clock tick. Second, some SMPs can support user-space multithreading by firing up tasks on specified CPUs then starting the threads from there [this form of SMP threading seems tenuous, at best]. Third, some libraries solve the I/O blocking problem with special wrappers over system calls, or the task can be written for non-blocking I/O.

2.7.2 Kernel-Space Threads

Kernel-space threads often are implemented in the kernel using several tables (each task gets a table of threads). In this case, the kernel schedules each thread within the time slice of each process. There is a little more overhead with mode switching from user->kernel-> user and loading of larger contexts, but initial performance measures indicate a negligible increase in time.

Advantages: Since the clock tick will determine the switching times, a task is less likely to hog the time slice from the other threads within the task. Also I/O blocking is not a problem. Lastly, if properly coded, the process automatically can take advantage of SMPs and will run incrementally faster with each added CPU.

2.7.3 Combination

Some implementations support both user- and kernel-space threads. This gives the advantages of each to the running task. However, since Linux’s kernel-space threads nearly perform as well as user-space, the only advantage of using user-threads would be the cooperative multitasking.
Several problems with threads originate from a classic view and its intrinsic concurrency complexity.

2.7.4 Classic View

In many other multithreaded OSs, threads are not processes merely parts of a parent task. Therefore, the question of "what happens if a thread calls fork() or (worse) if a thread execve()'s some external program" becomes problematic: the **whole task** could be replaced. The POSIX 1c standard defines a thread calling fork() to duplicate only the calling thread in the new process; and an execve() from a thread would stop all threads of that process.

Having two different implementations and schedulers for processes is a flaw that has perpetuated from implementation to implementation. In fact, some multitasking OSs have opted **not** to support threads due to these problems (not to mention the effort needed to make the kernel and libraries 100% re-entrant). For example, the POSIX-compliant Windows NT does not to support threads (Windows NT *does* support threads but they are not POSIX compliant).

2.7.5 Concurrency Complexity

Most people have a hard enough time understanding tasks, never mind "chopped up tasks" or threads. The first problem while programming is answering the question: "What *can* be threaded in my app?" That, in it, can be very laborious.

Another problem is locking. All the nightmares about sharing, locking, deadlock, race conditions, etc. come vividly alive in threads. Processes do not usually have to deal with this, since most shared data is passed through pipes. Now, threads can share file handles, pipes, variables, signals, etc. Trying to test and duplicate error conditions can cause more gray hair than a wayward child.

Exercise: Check Your Progress 3

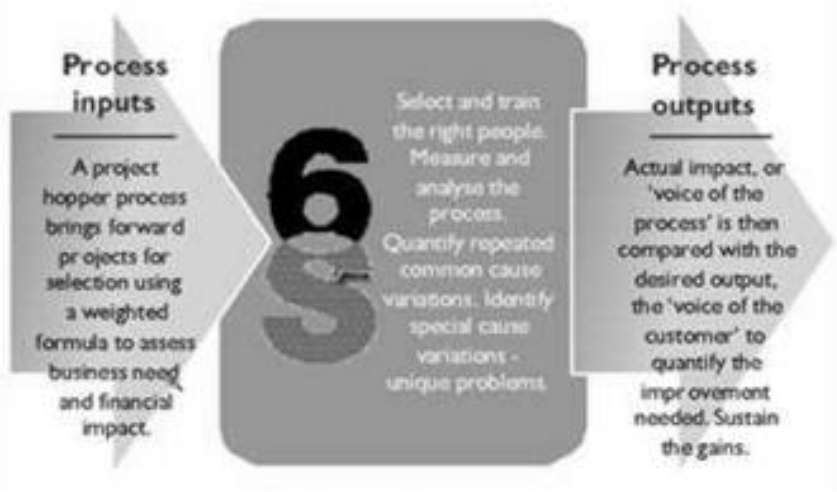
Case Study-Cummins Operating System business model

The focus for any business model is a vision that shows a clear idea of where an organization is going. The company's vision is "making people's lives better by unleashing the Power of Cummins." This vision is then translated into objectives such as:

- Motivating people to act like owners working together
- Exceeding customer expectations by always being first to market with the best products
- Partnering with customers to make sure that they succeed
- Demanding that everything Cummins does leads to a cleaner, healthier and safer environment
- Creating wealth for all stakeholders.

Achieving this vision is based on a single business operating system known as COS - Cummins Operating System, utilized throughout the company. This incorporates 10 major practices, as illustrated, with Six Sigma as the primary process improvement method. COS practices understand the requirements of both internal and external customers. Having understood these requirements Cummins can then compare its desired and current performance.

Six Sigma is not simply a quality initiative and nor is it just about solving problems. It is about making major process improvements in key areas of the business where a gap exists between desired and current levels of performance. This starting point provides an opportunity for Cummins to look carefully at every input into its business processes.



Six Sigma therefore helps Cummins to understand the key processes in helping it to deliver results; and provides an appreciation that the business outputs are determined by the processes that the inputs go through to become outputs. Using Six Sigma to improve business processes for selected projects thus enables positive gains to be made in key areas of the business.

Question

1. Explain the Cummins operating system business model.
2. What do you understand by Six Sigma?

2.8 Summary

- Processes access all operating system resources through a well-defined system call interface with the operating system.
- Depending on the operating system (OS), a process may be made up of multiple threads of execution that execute instructions concurrently.
- A process consists of a set of instructions, which may include ones related to I/O operations.
- Process Control is the active changing of the process based on the results of process monitoring.
- Operating systems are considered as a manager of the underneath various hardware resources; however this management is not the extreme goal, but a way in which the processes may access the resources reasonably.
- The five-state model provides a systematic way of modelling the behaviour of processes. Many operation systems are indeed constructed using this model.
- A proxy server satisfying the requests for a number of computers on a LAN would be benefited by a multi-threaded process.
- Accounting information includes the amount of CPU and real time used, time limits, account numbers, job or process numbers, and so on.

2.9 Keywords

Buffering: A buffer is a temporary storage location for data while the data is being transferred.

CPU Registers: **The central processing unit (CPU)** contains a number of memory locations which are individually addressable and reserved for specific purpose. These memory locations are called **registers.**

Process Control Block (PCB): The PCB is a certain store that allows the operating systems to locate key information about a process.

Process Counter: Program instructions uniquely identified by their program counters (PCs) provide a convenient and accurate means of recording the context of program execution and PC-based prediction techniques have been widely used for performance optimizations at the architectural level.

Process Management: The operating system manages many kinds of activities ranging from user programs to system programs like printer spooler, name servers, file server etc. Each of these activities is encapsulated in a process.

Process State: The process state consist of everything necessary to resume the process execution if it is somehow put aside temporarily.

Synchronization: In computer science, especially parallel computing, synchronization means the coordination of simultaneous threads or processes to complete a task in order to get correct runtime order and avoid unexpected race conditions.

Thread: A thread is a single sequence stream within in a process. Because threads have some of the properties of processes, they are sometimes called lightweight processes. In a process, threads allow multiple executions of streams.

2.10 Review Questions

1. What is a process? Explain in brief.
2. What about process states?
3. What is a process control block?
4. How do processes synchronize their activity?
5. What are the benefits and the detriments of each of the following? Consider both the systems and the programmer's levels.
    a. Direct and indirect communication.
    b. Symmetric and asymmetric communication.
    c. Automatic and explicit buffering.
    d. Send by copy and send by reference.
6. Describe the actions taken by a kernel to switch context between processes.
7 Describe the actions used in Buffering in the processes.
8. Explain the threads in LINUX.
9. Describes the process states in operating system.
10. What are the threads? Explain in brief.

Answers for Self Assessment

1.(b) 2.(b) 3. (b) 4.(c) 5. (b)

2.11 Further Studies

1. http://my.safaribooksonline.com/book/information-technology-and-software-development/9788131733097/operating-system/ch07lev1sec5
2. Operating System Principles by Silberschatz, Galvin and Gagne

3

Process Scheduling

STRUCTURE

3.0 Objectives

3.1 Introduction

3.2 Types of Scheduler

3.3 Scheduling Criteria

3.4 Uniprocessor Scheduling

3.5 Multiprocessor Scheduling

3.6 Algorithm Evaluation

3.7 Process Scheduling in Linux

3.8 Summary

3.9 Keywords

3.10 Review Questions

3.11 Further Studies

3.0 Objectives

After studying this chapter, you will be able to:

- Understand the types of scheduler
- Explain the scheduling criteria
- Discuss about uniprocessor scheduling
- Discuss about multiprocessor scheduling
- Understand the algorithm evaluation
- Explain the process scheduling in Linux

3.1 Introduction

Back in the old days of batch systems with input in the form of card images on a magnetic tape, the scheduling algorithm was simple: just run the next jobs on the tape. With timesharing systems, the scheduling algorithm became more complex, because there were generally multiple users waiting for service. There may be one or more batch streams as well (e.g., at an insurance company, for processing claims). On a personal computer you might think there would be only one active process. After all, a user entering a document on a word processor is unlikely to be simultaneously compiling a program in the background. However, there are often background jobs, such as electronic mail

daemons sending or receiving e-mail. You might also think that computers have gotten so much faster over the years that the CPU is rarely a scarce resource any more. However, new applications tend to demand more resources. Processing digital photographs or watching real time video is examples.

3.2 Types of Scheduler

Operating systems may feature up to 3 distinct types of schedulers: a long-term scheduler (also known as an admission scheduler or high-level scheduler), a mid-term or medium-term scheduler and a short-term scheduler (also known as a dispatcher). The names suggest the relative frequency with which these functions are performed. Figure 1 shows the possible traversal paths of jobs and programs through the components and queues, depicted by rectangles, of a computer system. The primary places of action of the three types of schedulers are marked with down-arrows. As shown in Figure 1, a submitted batch job joins the batch queue while waiting to be processed by the long-term scheduler. Once scheduled for execution, processes spawned by the batch job enter the ready queue to await processor allocation by the short-term scheduler. After becoming suspended, the running process may be removed from memory and swapped out to secondary storage. Such processes are subsequently admitted to main memory by the medium-term scheduler in order to be considered for execution by the short-term scheduler.

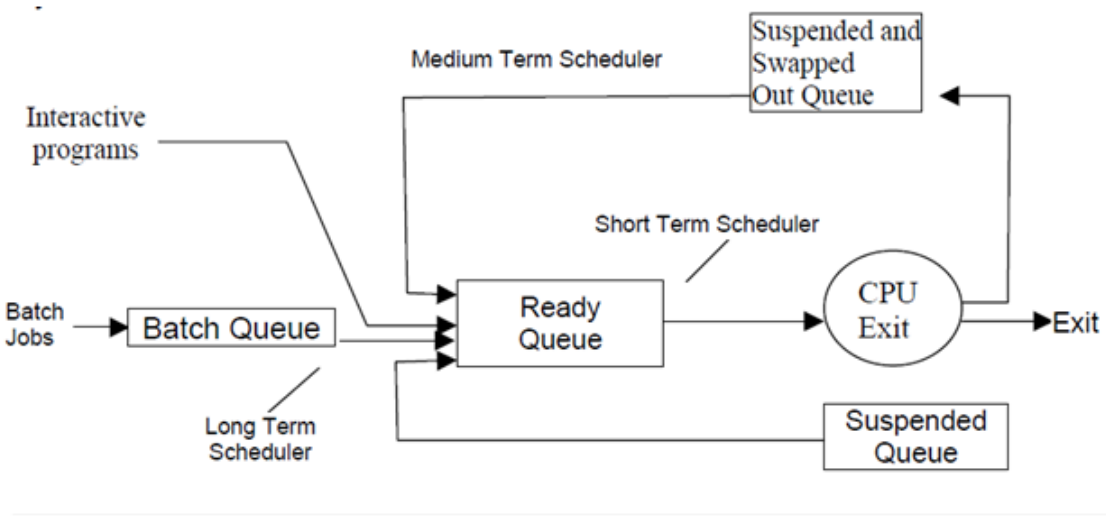


Figure 1: Process Scheduler

3.2.1 Long-term Scheduler

The long-term, or admission, scheduler decides which jobs or processes are to be admitted to the ready queue; that is, when an attempt is made to execute a program, its admission to the set of currently executing processes is either authorized or delayed by the long-term scheduler. Thus, this scheduler dictates what processes are to run on a system and the degree of concurrency to be supported at any one time - i.e.: whether a high or low amount of processes are to be executed concurrently, and how the split between IO intensive and CPU intensive processes is to be handled. Typically for a desktop computer, there is no long-term scheduler as such, and processes are admitted to the system automatically. However this type of scheduling is very important for a real-time operating system, as the system's ability to meet process deadlines may be compromised by the slowdowns and contention resulting from the admission of more processes than the system can safely handle.

3.2.2 Mid-term Scheduler

The mid-term scheduler, present in all systems with virtual memory, temporarily removes processes from main memory and places them on secondary memory (such as a disk drive) or vice versa. This is commonly referred to as "wapping out" or "swapping in" (also incorrectly as"paging out" or"paging in"). The mid-term scheduler may decide to swap out a process which has not been active for some time, or a process which has a low priority, or a process which is page faulting frequently, or a process which is taking up a large amount of memory in order to free up main memory for other processes, swapping the process back in later when more memory is available, or when the process has been unblocked and is no longer waiting for a resource.

3.2.3 Short-term Scheduler

The short-term scheduler (also known as the dispatcher) decides which of the ready, in-memory processes are to be executed (allocated a CPU) next following a clock interrupt, an IO interrupt, an operating system call or another form of signal. Thus the short-term scheduler makes scheduling decisions much more frequently than the long-term or mid-term schedulers - a scheduling decision will at a minimum have to be made after every time slice and these are very short. This scheduler can be pre-emptive, implying that it is capable of forcibly removing processes from a CPU when it decides to allocate that CPU to another process, or non-pre-emptive, in which case the scheduler is unable to "force" processes of the CPU.

Did You Know?

In the 1940s, the earliest electronic digital systems had no operating systems. Electronic systems of this time were so primitive compared to those of today that instructions were often entered into the system one bit at a time on rows of mechanical switches or by jumper wires on plug boards.

3.3 Scheduling Criteria

Different CPU scheduling algorithms have different properties, and the choice of a particular algorithm may favour one class of processes over another. In choosing which algorithm to use in a particular situation, we must consider the properties of the various algorithms. Many criteria have been suggested for comparing CPU scheduling algorithms. Which characteristics are used for comparison can make a substantial difference in which algorithm is judged to be best.

The criteria include the following:

- *CPU Utilization*: We want to keep the CPU as busy as possible.
- *Throughput:* If the CPU is busy executing processes, then work is being done. One measure of work is the number of processes that are completed per time unit, called throughput. For long processes, this rate may be one process per hour; for short transactions, it may be 10 processes per second.
- *Turnaround time*: From the point of view of a particular process, the important criterion is how long it takes to execute that process. The interval from the time of submission of a process to the time of completion is the turnaround time. Turnaround time is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.
- *Waiting time*: The CPU scheduling algorithm does not affect the amount of the time during which a process executes or does I/O; it affects only the amount of time that a process spends waiting in the ready queue. Waiting time is the sums of periods spend waiting in the ready queue.

- *Response time:* In an interactive system, turnaround time may not be the best criterion. Often, a process can produce some output fairly early and can continue computing new results while previous results are being output to the user. Thus, another measure is the time from the submission of a request until the first response is produced. This measure, called response time, is the time it takes to start responding, not the time it takes to output the response. The turnaround time is generally limited by the speed of the output device.

3.4 Uniprocessor Scheduling

As we know, multiprogramming is used to improve the efficiency of the micro-processor. That is when the currently running process is blocked due to waiting for an I/O event; another process may be scheduled to run. At this moment, the issue of scheduling is involved, deciding which process should be executed by the processor, because usually more than one process may be ready for being scheduled.

3.4.1 Types of Processor Scheduling

There are three different types of scheduling:

Long-term scheduling: Long-term scheduling is performed to decide if a new process is to be created and be added to the pool of processes. Long-term scheduling controls the degree of multiprogramming. The more processes that are created, the smaller is the percentage of time that each process can be executed. Thus the long term scheduler may limit the degree of multiprogramming to provide satisfactory service to the current set of processes. Whenever a process terminates, or the fraction of time that the processor is idle exceeds a certain threshold, the long-term scheduler may be invoked.

What process to be created is another issue that long-term scheduling needs to deal with? The decision may be made on a first-come-first-served basis or it can be a tool to manage system performance. For example, if the information is available, the scheduler may attempt to keep a mix of processor-bound and I/O-bound processes. A processor bound process is one that mainly performs computational work and occasionally uses I/O devices, while an I/O-bound process is one that uses I/O devices more than the microprocessor.

Medium-term scheduling: Medium-term scheduling is a part of the swapping function. It decides if a process should be loaded into the main memory either completely or partially so as to be available for execution.

Short-term scheduling: Short-term scheduling is the most common use of the term scheduling, i.e. deciding which ready process to execute next. The short-term scheduler, also known as the dispatcher, is invoked whenever an event occurs that may lead to the suspension of the current process or that may provide an opportunity to pre-empt a currently running process in favour of another.

Examples of such events include:
– Clock interrupts
– I/O interrupts
– Operating system calls
– Signals

Combined with the state transition of processes, the relationships among the above three types of scheduling may be illustrated in Figure 3.1.

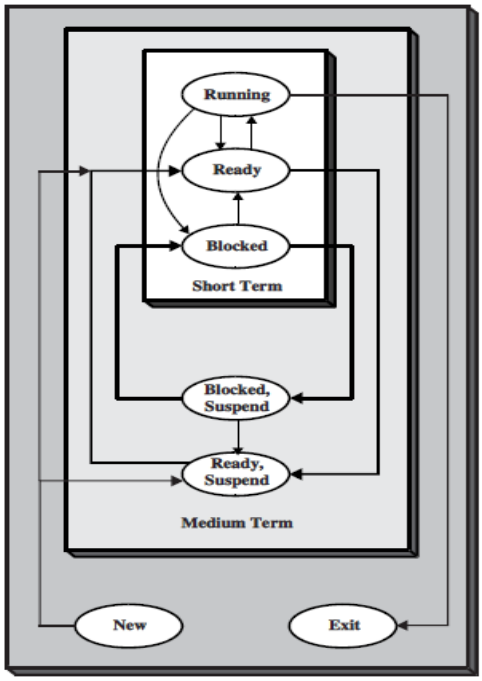


Figure 2: **Levels of scheduling**

Figure 2: associates the three levels of scheduling with different storage facilities, the outmost storage devices accommodating file systems for executable programs, the virtual memory space for blocked processes, and main memory for processes that may be executed right away.

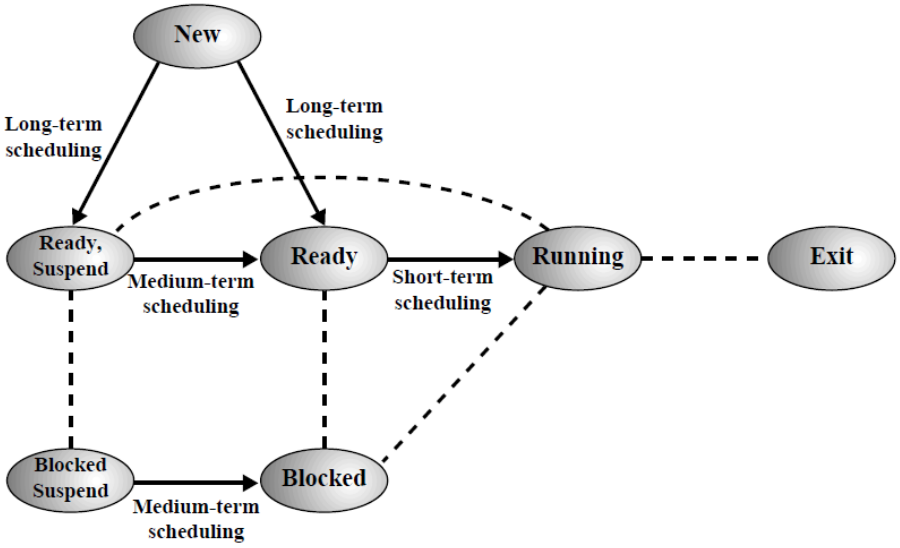


Figure 3: **Scheduling and Process State Transitions**

3.4.2 Pre-emption

Another issue relating to scheduling is whether a running process could be pre-empted or not. There are two categories:

Non pre-emptive: In this case, a running process continues to execute until (a) it terminates or (b) blocks itself to wait for I/O or to request some operating system service.

Pre-emptive: The currently running process may be interrupted and moved to the Ready state by the operating system. The pre-emption may possibly be made due to the arrival of a new process, or the occurrence of an interrupt that places a blocked process in the READY state.
Pre-emptive policies incur greater overhead than non pre-emptive ones but may be preferred since they prevent some processes from monopolizing the processor for a long time.

3.4.3 Priority

In many systems, a process is assigned a priority and the scheduler will always choose a process of higher priority over one of lower priority. Figure 4 depicts the revised process queue model with the consideration of priority.

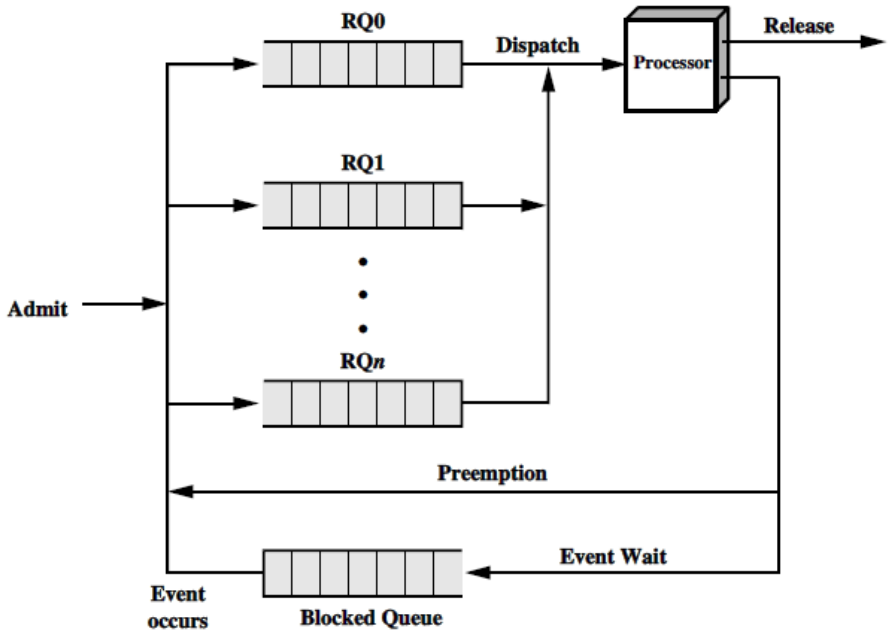


Figure 4: Priority queuing

3.4.4 Scheduling Algorithms

We move on to explore various scheduling policies. To compare the performance of these different strategies, we will use the following process set as an example.
The Table 1 shows when the processes arrive respectively and how much time they need to get completed.

Table 1: The processes arrive respectively

Process	Arrival Time	Service Time (T_s)
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

If we use turnaround time to measure the performance of various algorithms, we may obtain Table 2, which also includes the so-called normalized turnaround time, which is the ratio of turnaround time to service time. It indicates the relative delay experienced by a process. The minimum possible value for this ratio is of course 1.0; increasing values correspond to a decreasing level of service.

Table 2: **A performance comparison of scheduling policies**

							Mean
FCFS	Finish Time	3	9	13	18	20	
	Turnaround Time (T_r)	3	7	9	12	12	8.60
	T_r/T_s	1.00	1.17	2.25	2.40	6.00	2.56
RR $q = 1$	Finish Time	4	18	17	20	15	
	Turnaround Time (T_r)	4	16	13	14	7	10.80
	T_r/T_s	1.33	2.67	3.25	2.80	3.50	2.71
RR $q = 4$	Finish Time	3	17	11	20	15	
	Turnaround Time (T_r)	3	15	7	14	11	10.00
	T_r/T_s	1.00	2.5	1.75	2.80	5.50	2.71
SPN	Finish Time	3	9	15	20	11	
	Turnaround Time (T_r)	3	7	11	14	3	7.60
	T_r/T_s	1.00	1.17	2.75	2.80	1.50	1.84
SRT	Finish Time	3	15	8	20	10	
	Turnaround Time (T_r)	3	13	4	14	2	7.20
	T_r/T_s	1.00	2.17	1.00	2.80	1.00	1.59
HRRN	Finish Time	3	9	13	20	15	
	Turnaround Time (T_r)	3	7	9	14	7	7.20
	T_r/T_s	1.00	1.17	2.25	2.80	3.5	2.14
FB $q = 1$	Finish Time	4	20	16	19	11	
	Turnaround Time (T_r)	4	18	12	13	3	10.00
	T_r/T_s	1.33	3.00	3.00	2.60	1.5	2.29
FB $q = 2^i$	Finish Time	4	17	18	20	14	
	Turnaround Time (T_r)	4	15	14	14	6	10.60
	T_r/T_s	1.33	2.50	3.50	2.80	3.00	2.63

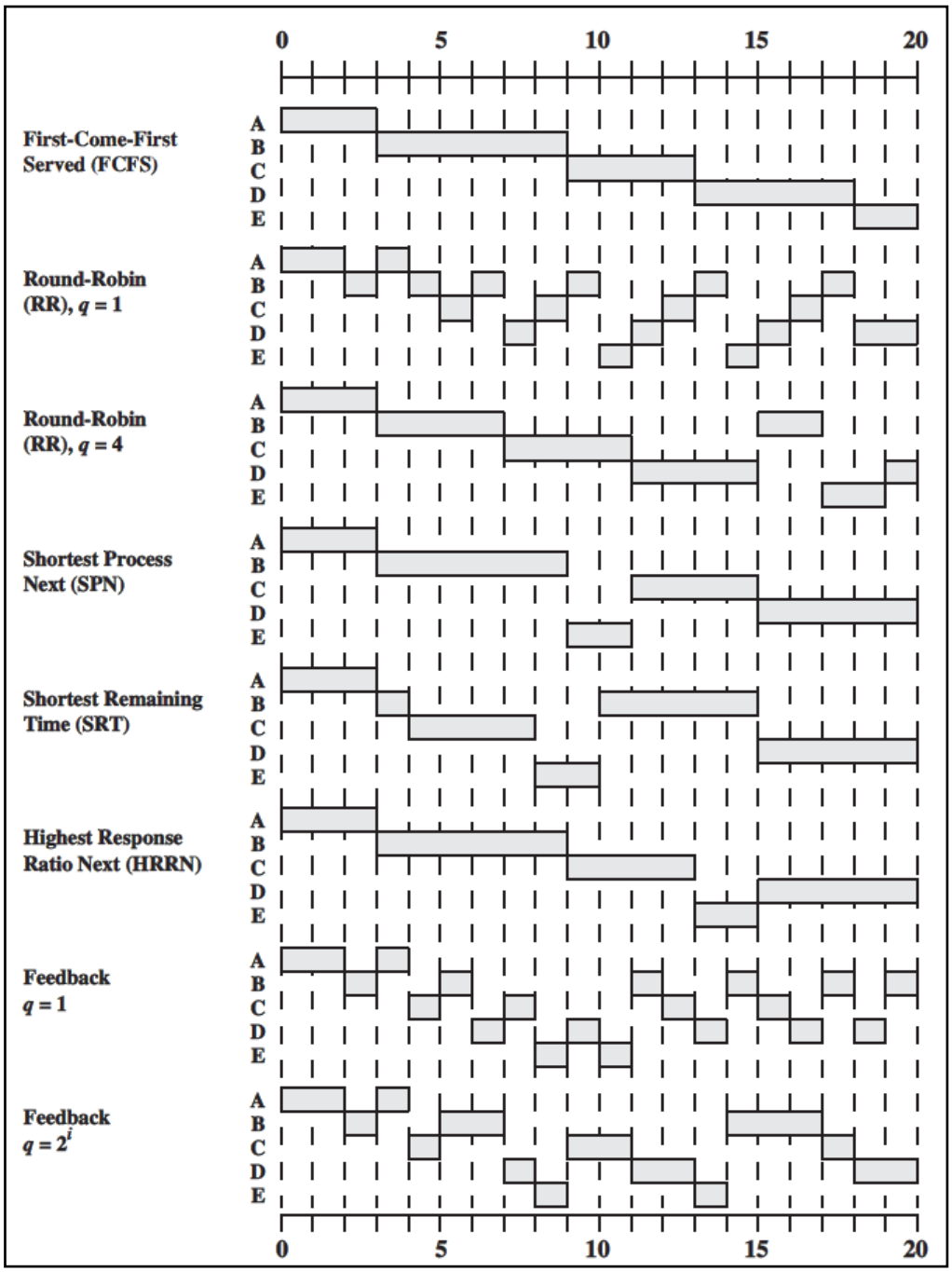


Figure 5: A comparison of scheduling policies

First-Come-First-Served

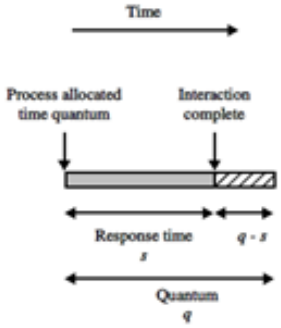
First-come-first-served (FCFS) is the simplest scheduling policy, also known as FIFO. With this policy, when a process becomes ready, it joins the ready queue and when the currently running process finishes, the process at the head of the ready queue, which has waited there for the longest time, is selected for execution.

As we can see from Table 2, though process E requires only 2 units of service time, it has to wait until all the preceding processes complete. When we line up for checking out books in the library, we of course are unwilling to be behind some guy who will borrow a hundred books. So FCFS performs much better for long processes and is pretty unfair for short ones.

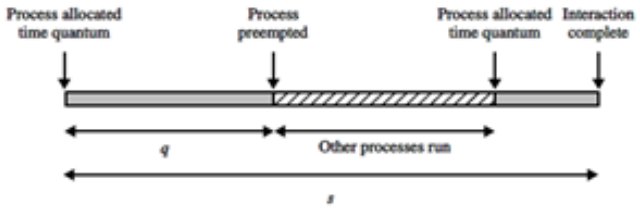
Thus a system with FCFS tends to favour processor-bound processes over I/O-bound ones, since the latter, though requiring relative light use of the processor has to wait a long time for the completion of processes before it in the ready queue before it gets dispatched for a short time and is blocked for I/O operation. The same thing may repeatedly happen during the execution of the I/O-bound process. Noticeably, while the process is waiting, the I/O devices that are supposed to be used will be idle, thus leading to the inefficiency of I/O devices.
Hence FCFS is not an attractive alternative on its own, but it may be combined with a priority scheme to provide an effective scheduler.

Round Robin

A straightforward way to reduce the suffering of short processes is to use a time-sharing scheme, called round robin, with which the operating system assigns short time slices to each process and if the slices allocated for a process are not enough for it to complete, then the process has to wait until its time slice comes again. With the help of a clock, whenever a clock interrupt occurs, the operating system will check if the time slice for the current process ends.
If yes, then another process will be scheduled and allocated a time slice. With round robin, the key design issue is the length of the time slice or quantum. If the slice is short, then short processes tend to get chance to run early. However very short time slices should be avoided, since there is overhead involved in handling the clock interrupt and performing the scheduling. In principle, a time slice should be slightly greater than the time required for a typical interaction. Figure 6 illustrates the effect the decision has on response time. Figure 4 and Table 2 show the results for our example using time quanta q of 1 and 4 time units respectively. The round robin policy is generally more effective than FCFS, however the I/O-bound processes are still to some extent treated unfairly, because these processes are very likely to be blocked before they use up a complete time quantum, while the processor-bound processes generally make great use of time slices.



(a) Time quantum greater than typical interaction



(b) Time quantum less than typical interaction

Figure 6: Effect of size of pre-emption time quantum.

An improvement to round robin is that when a blocked process is waked up, it is put into an auxiliary ready queue instead of the regular one. The processes in the auxiliary queue have higher priority than those in the latter, though they are allowed to run up to the length of time slices they are supposed to use. This approach, called virtual round robin (VRR), is illustrated in Figure 7. Experiment has shown that VRR is indeed superior to round robin in terms of fairness.

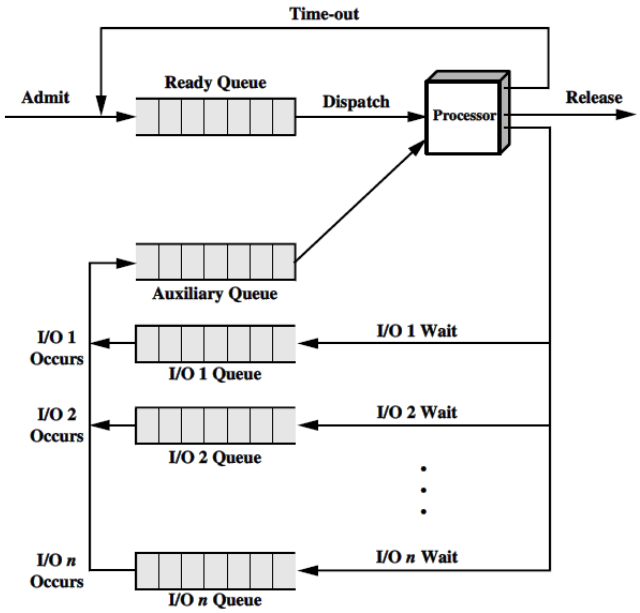


Figure 7: **Queuing diagram for virtual round robin scheduler.**

Shortest Process Next

Another policy avoiding long processes in favour in FCFS is a nonpreemptive one, called shortest process next (SPN), in which the process with the shortest expected execution time is selected next. However one difficulty with the SPN is that we need to know in advance the required processing time of each process. Again, it is impossible for us to know the future, though in batch environment, the programmers may possibly specify the quantum of the processing time of each process. We will not touch the details here, which are included in the text. The SPN policy is in favour of shorter processes, which however may lead to starvation of longer processes, if shorter ones come in constantly. On the other hand, it is not suitable for a time-sharing or transaction processing environment because of the lack of pre-emption.

Shortest Remaining Time

The shortest remaining time (SRT) policy is a pre-emptive version of SPN. Whenever a new process is brought in, the scheduler will run to perform the selection function to see if the new process has shorter remaining time than the currently running one. If yes, the current process is pre-empted and the new one is scheduled. Compared with SPN, the SRT policy's pre-emption makes it possible to run a shorter new process. On the other hand, unlike round robin, the SRT policy has less overhead since it does not need to respond to timer interrupts and May simply let the current process run into completion unless there comes in a new short process.

Highest Response Ratio Next

As we mentioned above, the normalized turnaround time may be used to measure the performance of scheduling algorithms. The highest response ratio next (HRRN) policy is proposed to minimize the

average value of the normalized turnaround time over all processes. For each process in the process pool, we first compute the following ratio:

$$R = \frac{(w + s)}{s}$$

Where w is the time since the process was created and s is the expected service time. Then whenever the current process is blocked or completes, the process with the greatest value will be scheduled to run.

Since a smaller denominator (smaller expected service time) in the fraction results in a greater value, shorter processes are favoured. And again as SRT and SPN, the expected service time needs to be estimated based on the history.

Feedback

Another approach to favour shorter processes is to penalize processes that have been running longer, thus avoiding predicting the expected processing time.

To do so, a dynamic priority mechanism is used. As Figure 3.7 shows, the ready process queue is split into several queues, each with a different priority. When a process first enters the system, it is put in the queue RQ0, which has the highest priority. After its first execution and when it becomes READY again, it is placed in RQ1, and so on and so forth. The feedback approach is also pre-emptive. Like round robin, a specific quantum of time is allocated to each scheduled process. When the time is out, the current process is pre-empted and another process is chosen from the queues on the highest-priority-first basis. The processes in the same queue follow a FCFS policy. Figure 4 and Table 2 shows our example in the case when the quantum of time is one unit of time.

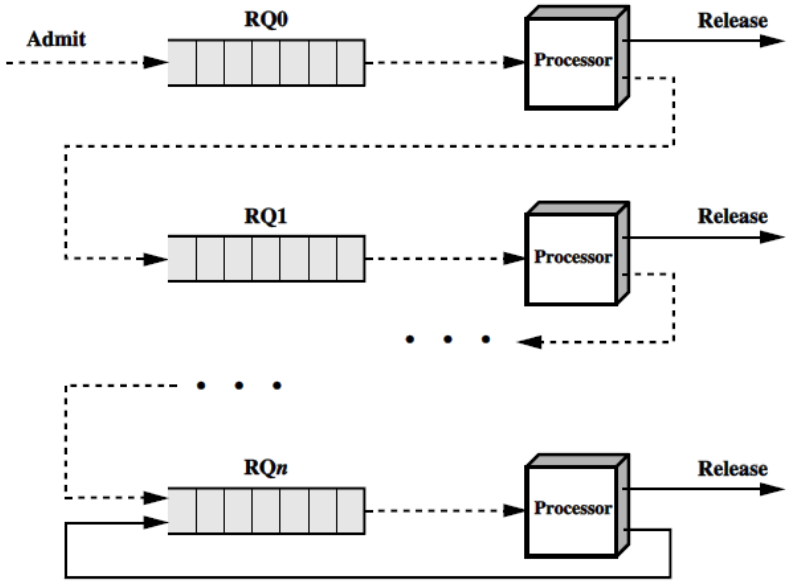


Figure 8: **Feedback scheduling**

Obviously, shorter processes are favoured over longer ones since the latter tend to gradually drift downward and cannot get a chance to run for a long time until there are no processes of higher priority. Starvation is also possible in the feedback policy if new processes come in frequently.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Types of Scheduler.

……..……

……..……

……...

Ex2: Identify Processor Scheduling.

……..……

……..……

……...

3.5 Multiprocessor Scheduling

On a uniprocessor, scheduling is one dimensional. The only question that must be answered (repeatedly) is: "Which process should be run next?" On a multiprocessor, scheduling is two dimensional. The scheduler has to decide which process to run and which CPU to run it on. This extra dimension greatly complicates scheduling on multiprocessors.

Another complicating factor is that in some systems, all the processes are unrelated whereas in others they come in groups. An example of the former situation is a timesharing system in which independent users start up independent processes. The processes are unrelated and each one can be scheduled without regard to the other ones. An example of the latter situation occurs regularly in program development environments. Large systems often consist of some number of header files containing macros, type definitions, and variable declarations that are used by the actual code files. The program make is commonly used to manage development. When make is invoked, it starts the compilation of only those code files that must be recompiled on account of changes to the header or code files. Object files that are still valid are not regenerated. The original version of make did its work sequentially, but newer versions designed for multiprocessors can start up all the compilations at once. If 10 compilations are needed, it does not make sense to schedule 9 of them quickly and leave the last one until much later since the user will not perceive the work as completed until the last one finishes. In this case it makes sense to regard the processes as a group and to take that into account when scheduling them.

Caution

While multiprocessor scheduling when a header file is changed, all the code files that include it must be recompiled.

3.5.1 Timesharing

Let us first address the case of scheduling independent processes; later we will consider how to schedule related processes. The simplest scheduling algorithm for dealing with unrelated processes (or threads) is to have a single system wide data structure for ready processes, possibly just a list, but more likely a set of lists for processes at different priorities as depicted in Figure 9 (a). Here the 16 CPUs are all currently busy, and a prioritized set of 14 processes are waiting to run. The first CPU to finish its current work (or have its process block) is CPU 4, which then locks the scheduling queues and selects the highest priority process, A, as shown in Figure 9 (b). Next, CPU 12 goes idle and chooses process B, as illustrated in Figure 9 (c). As long as the processes are completely unrelated, doing scheduling this way is a reasonable choice.

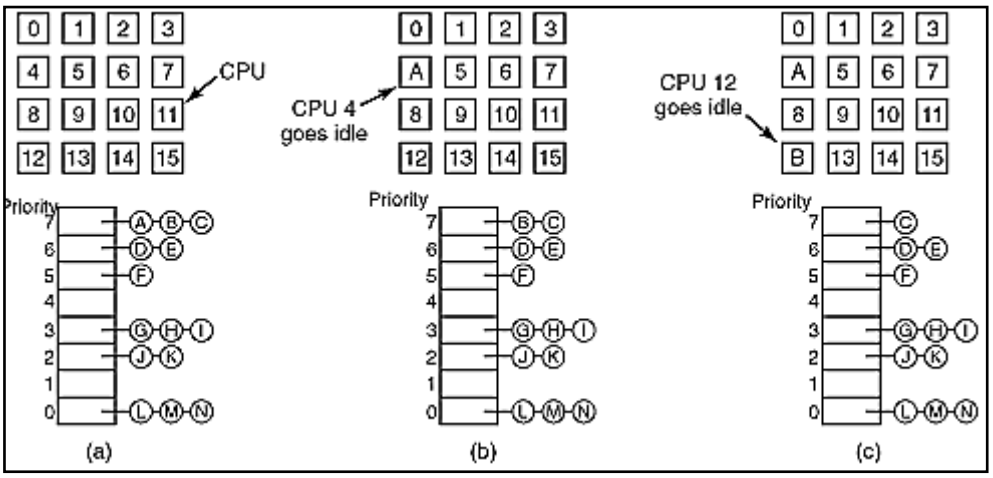


Figure 9: **Using a single data structure for scheduling a multiprocessor.**

Having a single scheduling data structure used by all CPUs timeshares the CPUs, much as they would be in a uniprocessor system. It also provides automatic load balancing because it can never happen that one CPU is idle while others are overloaded. Two disadvantages of this approach are the potential contention for the scheduling data structure as the numbers of CPUs grows and the usual overhead in doing a context switch when a process blocks for I/O.

It is also possible that a context switch happens when a process' quantum expires. On a multiprocessor, that has certain properties not present on a uniprocessor. Suppose that the process holds a spin lock, not unusual on multiprocessors, as discussed above. Other CPUs waiting on the spin lock just waste their time spinning until that process is scheduled again and releases the lock. On a uniprocessor, spin locks are rarely used so if a process is suspended while it holds a mutex, and another process starts and tries to acquire the mutex, it will be immediately blocked, so little time is wasted.

To get around this anomaly, some systems use smart scheduling, in which a process acquiring a spin lock sets a process-wide flag to show that it currently has a spin lock. When it releases the lock, it clears the flag. The scheduler then does not stop a process holding a spin lock, but instead gives it a little more time to complete its critical region and release the lock. Another issue that plays a role in scheduling is the fact that while all CPUs are equal, some CPUs are more equal. In particular, when process A has run for a long time on CPU k, CPU k's cache will be full of A's blocks. If A gets to run again soon, it may perform better if it is run on CPU k, because k's cache may still contain some of A's blocks. Having cache blocks preloaded will increase the cache hit rate and thus the process' speed? In addition, the TLB may also contain the right pages, reducing TLB faults. Some multiprocessors take this effect into account and use what is called affinity scheduling. The basic idea here is to make a serious effort to have a process run on the same CPU it ran on last time. One way to create this affinity is to use a two-level scheduling algorithm. When a process is created, it is assigned to a CPU, for example based on which one has the smallest load at that moment. This assignment of processes to CPUs is the top level of the algorithm. As a result, each CPU acquires its own collection of processes.

3.5.2 Space Sharing

The other general approach to multiprocessor scheduling can be used when processes are related to one another in some way. Earlier we mentioned the example of parallel make as one case. It also often occurs that a single process creates multiple threads that work together. For our purposes, a job consisting of multiple related processes or a process consisting of multiple kernel threads are

essentially the same thing. We will refer to the schedulable entities as threads here, but the material holds for processes as well. Scheduling multiple threads at the same time across multiple CPUs is called space sharing. The simplest space sharing algorithm works like this. Assume that an entire group of related threads is created at once. At the time it is created, the scheduler checks to see if there are as many free CPUs as there are threads. If there are, each thread is given its own dedicated (i.e., non multi-programmed) CPU and they all start. If there are not enough CPUs, none of the threads are started until enough CPUs are available. Each thread holds onto its CPU until it terminates, at which time the CPU is put back into the pool of available CPUs. If a thread blocks on I/O, it continues to hold the CPU, which is simply idle until the thread wakes up. When the next batch of threads appears, the same algorithm is applied.

At any instant of time, the set of CPUs is statically partitioned into some number of partitions, each one running the threads of one process. In Figure 10, we have partitions of sizes 4, 6, 8, and 12 CPUs, with 2 CPUs unassigned, for example. As time goes on, the number and size of the partitions will change as processes come and go.

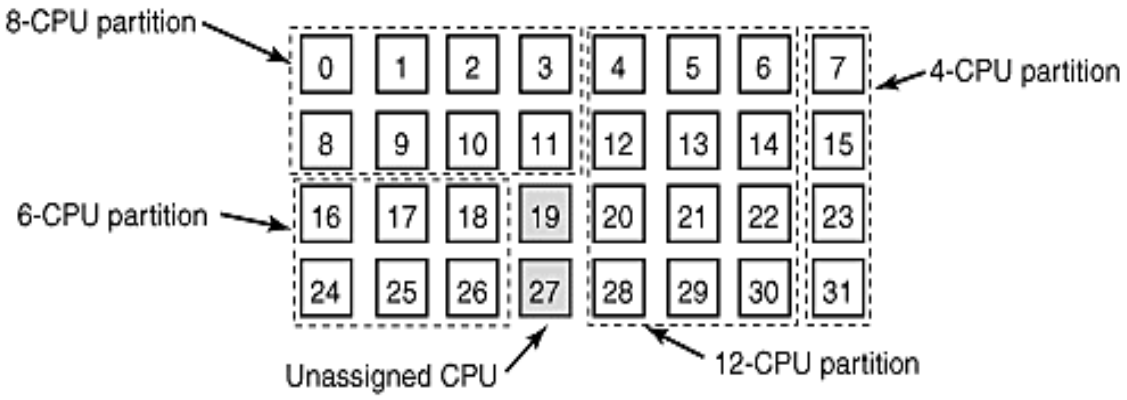


Figure 10: **A set of 32 CPUs split into four partitions, with two CPUs available.**

Periodically, scheduling decisions have to be made. In Uniprocessor systems, shortest job first is a well-known algorithm for batch scheduling. The analogous algorithm for a multiprocessor is to choose the process needing the smallest number of CPU cycles that is the process who's CPU-count X run-time is the smallest of the candidates. However, in practice, this information is rarely available, so the algorithm is hard to carry out.

3.5.3 Gang Scheduling

A clear advantage of space sharing is the elimination of multiprogramming, which eliminates the context switching overhead. However, an equally clear disadvantage is the time wasted when a CPU blocks and has nothing at all to do until it becomes ready again. Consequently, people have looked for algorithms that attempt to schedule in both time and space together, especially for processes that create multiple threads, which usually need to communicate with one another.

To see the kind of problem that can occur when the threads of a process (or processes of a job) are independently scheduled, consider a system with threads A_0 and A_1 belonging to process A and threads B_0 and B_1 belonging to process B. threads A_0 and B_0 are timeshared on CPU 0; threads A_1 and B_1 are timeshared on CPU 1. Threads A_0 and A_1 need to communicate often. The communication pattern is that A_0 sends A_1 a message, with A_1 then sending back a reply to A_0, followed by another such sequence. Suppose that luck has it that A_0 and B_1 start first, as shown in Figure: 11.

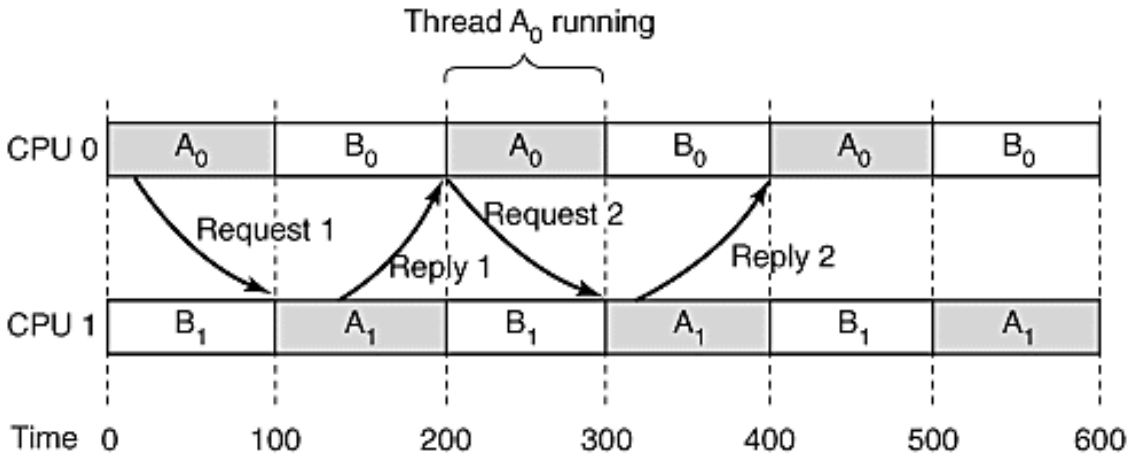


Figure: 11 Communication between two threads belonging to process A that are running out of phase.

In time slice 0, A_0 sends A_1 a request, but A_1 does not get it until it runs in time slice 1 starting at 100 msec. It sends the reply immediately, but A_0 does not get the reply until it runs again at 200 msec. The net result is one request-reply sequence every 200 msec.

The solution to this problem is gang scheduling, which is an outgrowth of co-scheduling. Gang scheduling has three parts:

- Groups of related threads are scheduled as a unit, a gang.
- All members of a gang run simultaneously, on different timeshared CPUs.
- All gang members start and end their time slices together.

The trick that makes gang scheduling work is that all CPUs are scheduled synchronously. This means that time is divided into discrete quanta as we had in Figure: 3.10. At the start of each new quantum, all the CPUs are rescheduled, with a new thread being started on each one. At the start of the following quantum, another scheduling event happens. In between, no scheduling is done. If a thread blocks, its CPU stays idle until the end of the quantum.

	CPU 0	1	2	3	4	5
Time slot 0	A_0	A_1	A_2	A_3	A_4	A_5
1	B_0	B_1	B_2	C_0	C_1	C_2
2	D_0	D_1	D_2	D_3	D_4	E_0
3	E_1	E_2	E_3	E_4	E_5	E_6
4	A_0	A_1	A_2	A_3	A_4	A_5
5	B_0	B_1	B_2	C_0	C_1	C_2
6	D_0	D_1	D_2	D_3	D_4	E_0
7	E_1	E_2	E_3	E_4	E_5	E_6

Figure: 12 Gang scheduling.

The idea of gang scheduling is to have all the threads of a process run together, so that if one of them sends a request to another one, it will get the message almost immediately and be able to reply almost

immediately. In Figure: 12, since all the A threads are running together, during one quantum, they may send and receive a very large number of messages in one quantum, thus eliminating the problem of Figure: 9.

Self Assessment Question

1. The long-term scheduler is also known as.....scheduler.
(a). ready (b). Admission (c). CPU (d). Job

2. The mid-term scheduler temporarily removes processes from main memory.
(a). True (b). False

3. …….is the sums of periods spend waiting in the ready queue.
(a). Turnaround time (b). Response time
(c). Waiting time (d). None of these

4. One measure of work is the number of processes that are completed per time unit, called.....
(a). Efficiency (b). State (c). Waiting time (d). Throughput

5. The dispatcher always chooses a process in the queue with lowest priority to execute.
(a). True (b). False

3.6 Algorithm Evaluation

The first step in determining which algorithm (and what parameter settings within that algorithm) is optimal for a particular operating environment is to determine what criteria are to be used, what goals are to be targeted, and what constraints if any must be applied. For example, one might want to "maximize CPU utilization, subject to a maximum response time of 1 second":

3.6.1 Deterministic Modelling

One type of analytic evaluation is deterministic modeling. This method takes a particular predetermined workload and defines the performance of each algorithm for that workload.

3.6.2 Queuing Models

We can examine the distribution of CPU burst and I/O wait times. We can apply a formula and compute one of three things.

- Average length of the queue
- Average waiting time of the queue
- Average arrival rate

3.6.3 Simulation

To get a more accurate evaluation of scheduling algorithms, we can use simulations. Simulations involve programming a model of the computer system.

3.6.4 Implementation

The only real way to test an operating system is to write the code and run it. However, this approach is very expensive.

Caution
Be aware that a pre-empted process is not suspended, since it remains in the TASK_RUNNING state; it simply no longer uses the CPU.

Did You Know?
The defining component of Linux is the Linux kernel, an operating system kernel first released October 5, 1991 by Linus Torvalds.

3.7 Process Scheduling in Linux

Like any time-sharing system, Linux achieves the magical effect of an apparent simultaneous execution of multiple processes by switching from one process to another in a very short time frame.

3.7.1 Scheduling Policy

The scheduling algorithm of traditional UNIX operating systems must fulfil several conflicting objectives: fast process response time, good throughput for background jobs, avoidance of process starvation, and reconciliation of the needs of low- and high-priority processes, and so on. The set of rules used to determine when and how selecting a new process to run is called scheduling policy.
Linux scheduling is based on the time-sharing technique, Timing Measurements: several processes are allowed to run "concurrently," which means that the CPU time is roughly divided into "slices," one for each run able process. Of course, a single processor can run only one process at any given instant. If a currently running process is not terminated when its time slice or quantum expires, a process switch may take place. Time-sharing relies on timer interrupts and is thus transparent to processes. No additional code needs to be inserted in the programs in order to ensure CPU time-sharing. The scheduling policy is also based on ranking processes according to their priority. Complicated algorithms are sometimes used to derive the current priority of a process, but the end result is the same: each process is associated with a value that denotes how appropriate it is to be assigned to the CPU.
An alternative classification distinguishes three classes of processes:

Interactive processes

These interact constantly with their users, and therefore spend a lot of time waiting for key presses and mouse operations. When input is received, the process must be woken up quickly, or the user will find the system to be unresponsive. Typically, the average delay must fall between 50 and 150 ms. the variance of such delay must also be bounded, or the user will find the system to be erratic. Typical interactive programs are command shells, text editors, and graphical applications.

Batch processes

These do not need user interaction, and hence they often run in the background. Since such processes do not need to be very responsive, they are often penalized by the scheduler. Typical batch programs are programming language compilers, database search engines, and scientific computations.

Real-time processes

These have very strong scheduling requirements. Such processes should never be blocked by lower-priority processes, they should have a short response time and, most important, such response time should have a minimum variance. Typical real-time programs are video and sound applications, robot controllers, and programs that collect data from physical sensors.
The two classifications we just offered are somewhat independent. For instance, a batch process can be either I/O-bound (e.g., a database server) or CPU-bound (e.g., an image-rendering program). While in Linux real-time programs are explicitly recognized as such by the scheduling algorithm,

there is no way to distinguish between interactive and batch programs. In order to offer a good response time to interactive applications, Linux (like all UNIX kernels) implicitly favours I/O-bound processes over CPU-bound ones.

3.7.2 Process Pre-emption

Linux processes are pre-emptive. If a process enters the TASK_RUNNING state, the kernel checks whether its dynamic priority is greater than the priority of the currently running process. If it is, the execution of current is interrupted and the scheduler is invoked to select another process to run (usually the process that just became runnable). Of course, a process may also be pre-empted when its time quantum expires. When this occurs, the *need_resched* field of the current process is set, so the scheduler is invoked when the timer interrupt handler terminates.

Some real-time operating systems feature pre-emptive kernels, which mean that a process running in Kernel Mode can be interrupted after any instruction, just as it can in User Mode. The Linux kernel is not pre-emptive, which means that a process can be pre-empted only while running in User Mode; non pre-emptive kernel design is much simpler, since most synchronization problems involving the kernel data structures are easily avoided.

Exercise: Check Your Progress 2

Case Study-Process Scheduling Lead Generation for SAP Partner

The Client

Our client is an international provider of process automation, job scheduling, and enterprise report management solutions that help businesses become more productive. Their solutions enable global organisations to maximise the value of their IT infrastructure and information resources to meet today's business challenges.

The Campaign

SAP Central Process Scheduling (CPS), co-developed by our client and SAP, is the only enterprise job scheduling solution validated, sold and supported by SAP. It is a core component in the SAP NetWeaver platform and enhances SAP business applications including SAP Financial Closing cockpit, SAP BI, and SAP Solution Manager.

As part of our client's formal partner alliance with SAP Martrain were commissioned by SAP's Alliance Director to help drive new revenues for CPS by identifying new business opportunities in the SAP customer base. Working closely with both partners and their Account Managers Martrain designed a one to one marketing campaign.

After creating a thumbnail sketch of each account individually tailored letters and emails were sent to carefully selected decision makers within the target accounts. Follow up calls, again with individually tailored messages, built a business case for the solution to be considered.

Once firm interest had been established the dialogue could then be developed by the Partner or Account Manager directly.

The Results

Given the ongoing success of Martrain's approach the campaign was twice extended to cover a total of 88 target accounts and generate 29 leads.

Questions

1. Write brief discussion of the campaign of the lead generation for sap partner.
2. Explain the concept of central process scheduling.

3.8 Summary

- Real-time processes should never be blocked by lower-priority processes, they should have a short response time and, most important, such response time should have a minimum variance.
- The mid-term scheduler temporarily removes processes from main memory and places them on secondary memory (such as a disk drive) or vice versa.
- Dispatcher is invoked whenever an event occurs that may lead to the suspension of the current process or that may provide an opportunity to pre-empt a currently running process in favour of another.
- The scheduling algorithm of traditional UNIX operating systems must fulfil several conflicting objectives: fast process response time, good throughput for background jobs.
- The scheduling policy is also based on ranking processes according to their priority.

3.9 Keywords

Nonpreemptive: In this case, a running process continues to execute until (a) it terminates or (b) blocks itself to wait for I/O or to request some operating system service.

Pre-emptive: The currently running process may be interrupted and moved to the Ready state by the operating system.

Throughput: One measure of work is the number of processes that are completed per time unit, called throughput.

Turnaround time: Turnaround time is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.

Waiting time: Waiting time is the sums of periods spend waiting in the ready queue.

3.10 Review Questions

1. Define the scheduler and its types.
2. Discuss the scheduling criteria.
3. Explain the term uniprocessor scheduling.
4. How many types of processor scheduling? Explain.
5. What is the preemption? Explain.
6. Define the scheduling algorithms.
7. Explain about multiprocessor scheduling.
8. What is the space sharing?
9. Explain the algorithm evaluation.
10. Define the process scheduling in Linux.

Answers of Self Assessment Questions

1. (b) 2. (a) 3. (c) 4. (d) 5. (b)

3.11 Review Questions

1. *Scheduling: Theory, Algorithms, and Systems by Michael Pinedo*
2. Operating System Principles by Silberschatz, Galvin and Gagne

4

W Process Synchronization and Deadlocks

STRUCTURE

4.0 Objectives

4.1 Introduction

4.2 The Critical-section Problem

4.3 Synchronization Hardware

4.4 Classical Problems of Synchronization

4.5 Critical Regions

4.6 Deadlocks System Model

4.7 Deadlock Prevention

4.8 Avoidance and Detection

4.9 Recovery from Deadlocks

4.10 Combined Approach to Deadlock Handling

4.11Summary

4.12 Keywords

4.13 Review Questions

4.14 Further Studies

4.0 Objectives

After studying this chapter, you will be able to:

- Explain the critical-section problem
- Discuss the synchronization hardware
- Explain the classical problems of synchronization
- Define the critical regions
- Explain the deadlocks-system model

4.1 Introduction

Among the problems that need to be addressed by computer scientists in order for sophisticated operating systems to be built are deadlock and process synchronization. Deadlock occurs when two or

more processes (programs in execution) request the same resources and are allocated them in such a way that a circular chain of processes is formed, where each process is waiting for a resource held by the next process in the chain. As a result, no process can continue; they are deadlocked. An operating system can handle this situation with various prevention or detection and recovery techniques. For example, resources might be numbered 1, 2, 3, and so on. If they must be requested by each process in this order, it is impossible for a circular chain of deadlocked processes to develop. Another approach is simply to allow deadlocks to occur, detect them by examining non active processes and the resources they are holding, and break any deadlock by aborting one of the processes in the chain and releasing its resources.

Process synchronization is required when one process must wait for another to complete some operation before proceeding. For example, one process (called a writer) may be writing data to a certain main memory area, while another process (a reader) may be reading data from that area and sending it to the printer. The reader and writer must be synchronized so that the writer does not overwrite existing data with new data until the reader has processed it. Similarly, the reader should not start to read until data has actually been written to the area. Various synchronization techniques have been developed. In one method, the operating system provides special commands that allow one process to signal to the second when it begins and completes its operations, so that the second knows when it may start. In another approach, shared data, along with the code to read or write them, are encapsulated in a protected program module. The operating system then enforces rules of mutual exclusion, which allow only one reader or writer at a time to access the module. Process synchronization may also be supported by an interprocess communication facility, a feature of the operating system that allows processes to send messages to one another.

4.2 The Critical-section Problem

If a data item is shared by number threads, race conditions could occur if the shared item is not protected properly. The easiest protection mechanism is a lock. In general, if a set of data items must be protected so that at any time there is no more than one thread can have access to it, we can associate the set of data items with a lock. The use of locks is actually very easy. For every thread, before it accesses the set of data items, it acquires the lock. Once the lock is successfully acquired, the thread becomes the owner of that lock and the lock is locked. Then, the owner can access the protected items. After this, the owner must release the lock and the lock becomes unlocked. It is possible that while the owner is accessing one of the protected data items and another thread comes of course, this second thread must acquire that lock. However, since the lock is locked, this request is unsuccessful and the requesting thread will be suspended and queued at the lock. When the lock is released by its owner, one of the waiting threads will be allowed to continue and locks the lock.

This mechanism can be seen in everyday life. For example, on an airplane, before use its lavatory, check to see if it is locked. If it is, join the waiting line; otherwise, enter and lock the door. Once the door is locked, are protected from the intrusion of anybody else (i.e., mutual exclusion). When exit, unlock the door so that one of those waiting can enter. There could be more than one waiting persons, and who will be allowed to enter and lock the door depends on some queuing policy (e.g., first-in-first-out). But, a good programmer should not make any assumption about this policy. Before entering a critical section, a thread acquires a lock. If it is successful, this thread enters the critical section and the lock is locked. As a result, all subsequent acquiring requests will be queued until the lock is unlocked. In this way, the owner of the lock (i.e., the thread that successfully acquired the lock) is the only thread that can execution the instructions of the indicated critical section. At the end of the execution of the instructions in a critical section, the owner releases the lock, and, at this point,

the lock is unlocked, allowing the next thread to enter this critical section. Therefore, mutual exclusion is guaranteed. Because of this, a lock is also usually referred to as a mutex for mutual exclusion in see Figure 4.1.

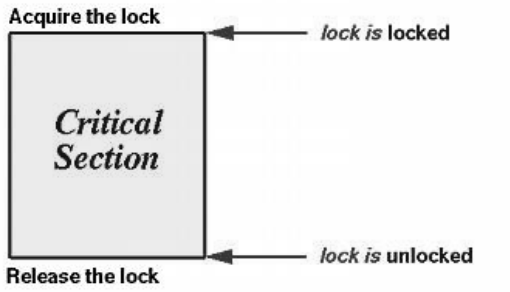


Figure 4.1: Critical section A.

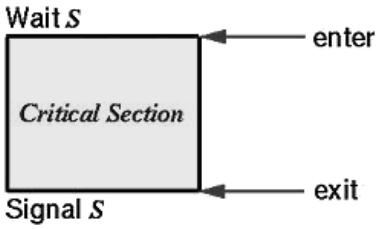


Figure 4.2: Critical section B.

4.3 Synchronization Hardware

Many systems provide hardware support for critical section code

- Uniprocessors – could disable interrupts
  - Currently running code would execute without preemption
  - Generally too inefficient on multiprocessor systems
  - Operating systems using this not broadly scalable
  - Interrupt sent to all processors
- Modern machines provide special atomic hardware instructions
  - Atomic = non-interruptable
  - Either test memory word or set value
  - Or swap contents of two memory words

Did You Know?

The Semaphores were invented by the late Edsger Dijkstra.

4.4 Classical Problems of Synchronization

Some classical problems of synchronization, such as shared buffer problem a fairly simple to solve using reliable synchronization facilities such as Dijkstra's semaphore. Other problems that programmers must solve are more complex; in fact, some of them are just downright tricky to solve correctly such that there is never a problem with starvation, deadlock or data inconsistency. Fortunately, there are a lot of really smart people who have already tackled these problems and we have known solutions to them. These problems are ones that frequently arise, so discuss them and their solutions in fairly generic terms and refer to them as the classical synchronization problems. One of the biggest challenges that the programmer must solve is to correctly identify their problem as

an instance of one of the classic problems. It may require thinking about the problem or framing it in a less than obvious way, so that a known solution may be used. The advantage to using a using known solution is assurance that it is correct. Some instances of deadlock can occur only in rare conditions are difficult to detect, so for any programming task using synchronization, it is best to reference the algorithm used to a known solution to one of the classic problems.

- Shared Buffer
- Bounded Buffer
- Readers and Writers
- Dinning Philosophers

Shared Buffer Problem

An example of data inconsistency that can occur because of a race condition is what can happen with a shared bank account. Mohan adds money to an account and poor student withdraws from the account.

When either accesses the account, they execute a critical section consisting of three steps (see Figure 4.3).

- Read account balance
- Update the balance
- Write the new balance to the account

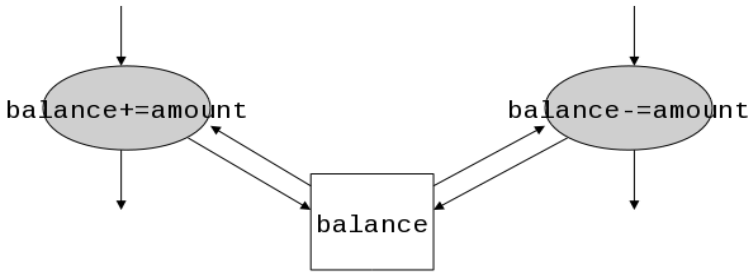
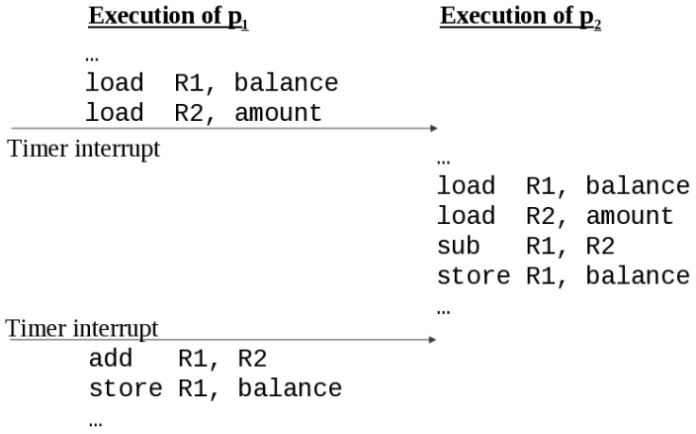


Figure 4.3: Shared Buffer Problem.

One day, Mohan checks the balance and seeing that it is 4,500 decides to add 2,250 to the account. Unfortunately, access to the account is not locked, so just then poor students withdraws 900 from the account and the new balance is recorded as 3,600. After adding the 2,250 Mohan records the balance as 6,750 rather than 5,850 as it should be.

The nature of the problem is clearer when we examine the assembly language code for such an operation:



The root of the problem stems from a context switch occurring in the middle of the execution of the critical section.
We only need to enforce mutual exclusion with a single lock or semaphore to correctly solve this problem. In python, a lock object from the threading module is a semaphore where s == 1. That is, only one thread is allowed to hold the lock at a time (mutual exclusion).

Bounded Buffer
This problem is also called the producers and consumers problem. A finite supply of containers is available. Producers take an empty container and fill it with a product. Consumers take a full container, consume the product and leave an empty container. The main complexity of this problem is that we must maintain the count for both the number of empty and full containers that are available.

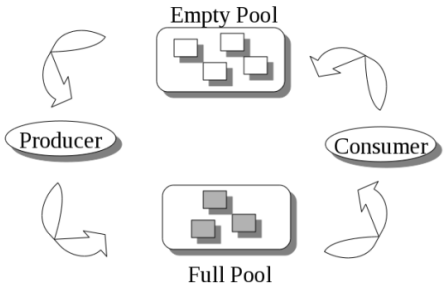


Figure 4.4: Bounded Buffer.

Producers produce a product and consumers consume the product, but both use of one of the containers each time. Here is a solution to the bounded buffer problem with N containers using a monitor:

Readers and Writers
If one notebook exists where writers may write information to, only one writer may write at a time. Confusion may arise if a reader is trying read at the same as a writer is writing. Since readers only look at the data, but do not modify the data, we can allow more than one reader to read at the same time. The main complexity with this problems stems from allowing more than one reader to access the data at the same time.



Figure 4.5: Readers and Writers.

Writers have mutual exclusion, but multiple readers at the same time are allowed. Here is a basic python class that implements a solution to readers and writers using simple locks

Dinning Philosophers

Five philosophers (the tasks) spend their time thinking and eating. They eat at a round table with five individual seats. To eat, each philosopher needs two forks (the resources). There are five forks on the table, one to the left and one to the right of each seat. When a philosopher cannot grab both forks, he sits and waits. Eating takes random time, and then the philosopher puts the forks down and leaves the dining room. After spending some random time thinking about the nature of the universe, he again becomes hungry, and the circle repeats itself.

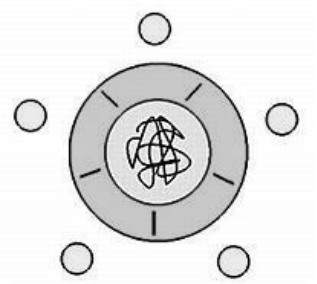


Figure 4.6: Dinning Philosophers.

A Philosopher needs both the fork on his left hand on his right to eat. The forks are shared with the neighbours on either side.

It can be observed that a straightforward solution, when forks are implemented by semaphores, is exposed to deadlock. There exist two deadlock states when all five philosophers are sitting at the table holding one for each. One deadlock state is when each philosopher has grabbed the fork left of him, and another is when each has the fork on his right.

Here is a basic monitor based solution:

import threading

Did You Know?

In 1984, K. Mani Chandy and J. Misra proposed a different solution to the dining philosophers problem to allow for arbitrary agents (numbered P1, ..., Pn) to contend for an arbitrary number of resources, unlike Dijkstra's solution.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: How to Handle Critical-section Problem?

……..……

……..……

……...

Ex2: Semaphore?

……..……

……..……

……...

4.5 Critical Regions

1. Motivation: The time dependent errors can be easily generated when semaphores are used to solve the critical section problem. To overcome this difficulty a new language construct, the critical region was introduced.
2. Definition and notation

A variable v of type T, which is to be shared among many processes, can be declared:

VAR v: SHARED T;

The variable v can be accessed only inside a region statement of the following form:

REGION v DO S;

This construct means that while statement S is being executed, no other process can access the variable v. Thus, if the two statements, are executed concurrently in distinct sequential processes, the result will be equivalent to the sequential execution S1 followed by S2, or S2 followed by S1

REGION v DO S1;
REGION v DO S2;

To illustrate this construct, consider the frames class defined in abstract data type. Since mutual exclusion is required when accessing the array free, we need to declare it as a shared array.

VAR free: SHARED ARRAY [l..n] OF Boolean;

The acquire procedure must be rewritten as follows;

```
PROCEDURE ENTRY acquire (MAR index: integer);
BEGIN
REGION free DO
FOR index := 1 TO n DO
IF free[index] THEN
BEGIN
free[index]:= false;
exit;
END;
index := 1;
END;
```

The critical-region construct guards against some simple errors associated with the semaphore solution to the critical section problem which may be made by a programmer.

3. Compiler implementations of the critical region construct. For each declaration VAR v: SHARED T; the compiler generates a semaphore v-mutex initialized to 1. For each statement, REGION v DO S; the compiler generates the following code:

```
p(v-mutex);
S;
V(v-mutex);
```

Critical region may also be nested. In this case, however, deadlocks may result.
Example of deadlock)

VAR x, y: SHARED T;

```
PARBEGIN
Q: REGION x DO REGION y DO S1;
R: REGION y DO REGION x DO S2;
PAREND;
```

4.5.1 Conditional Critical Regions

The critical region construct can be effectively used to solve the critical section problem. It cannot, however, be used to solve some general synchronization problems. For this reason the conditional critical region was introduced. The major difference between the critical region and the conditional critical region constructs is in the region statement, which now has the form:

```
REGION v WHEN B DO S;
```

Where B is a Boolean expression, regions referring to the same shared variable exclude each other in time. Now, however, when a process enters the critical section region. The Boolean expression B is evaluated if the expression is true, statement S is executed. If it is false, then process relinquishes the mutual exclusion and is delayed until B becomes lame and no other process is in the region associated with v.

Example for bounded buffer problem

```
VAR buffer: SHARED RECORD
pool: ARRAY [0..n-l] OF item;
count, in. out: integer;
END;
```

The producer process inserts a new item nextp in buffer by executing

```
REGION buffer WHEN count < n DO
BEGIN
pool[in] := nextp;
in := in + i MOD n;
count := count + 1;
END;
```

The counter process removes an item from the shared buffer and puts it in nextc by executing:

```
REGION buffer WHEN count > 0 DO
BEGIN
nextc := pool[out];
out := out +1 MOD n;
count := count - 1;
END;
```

However, the class concept alone cannot guarantee that such sequences will be observed.

- A process might operate on the file without first gaining access permission to it
- A process might never release the file once it has been granted access to it
- A process might attempt to release a file that it never required
- A process might request the same file twice

4.5.2 Monitors

- A monitor is a collection of procedures, variables, and data structures grouped together.

- Processes can call the monitor procedures but cannot access the internal data structures.
- Only one process at a time may be active in a monitor.
- Active in a monitor means in ready queue or CPU with the program counter somewhere in a monitor method.
- A monitor is a language construct.
- Compare this with semaphores, which are usually an OS construct.
- The compiler usually enforces mutual exclusion.
- Condition variables allow for blocking and unblocking
    - cv.wait() blocks a process.
      The process is said to be waiting for (or waiting on) the condition variable cv.
    - cv.signal() (also called cv.notify) unblocks a process waiting for the condition variable cv.

When this occurs, we need to still require that only one process is active in the monitor. This can be done in several ways:

- On some systems the old process (the one executing the signal) leaves the monitor and the new one enters
- On some systems the signal must be the last statement executed inside the monitor
- On some systems the old process will block until the monitor is available again
- On some systems the new process (the one unblocked by the signal) will remain blocked until the monitor is available again.

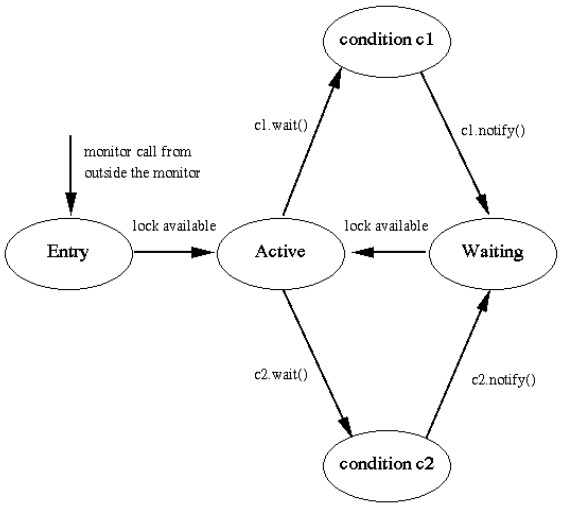


Figure 4.7: The relative priority of these queues determines the operation of the monitor implementation.

Exercise: Check Your Progress 2
Note: i) Use the space below for your answer.
Ex1: Draw the diagram of operation the monitor implementation.

……..……

……..……

……...

4.6 Deadlocks System Model

There are four necessary and sufficient conditions for deadlock:

1. Resources must be used in mutual exclusion.
2. Resources cannot be pre-empted.
3. Processes hold resources while waiting to acquire others.
4. There exists a circular chain of processes, each of which is holding a resource while waiting for a resource held by another process in the chain.

Holt characterizes deadlock with a directed (wait-for) graph. In Figures 4.8 and 4.9, processes P1 and P2 are in deadlock.

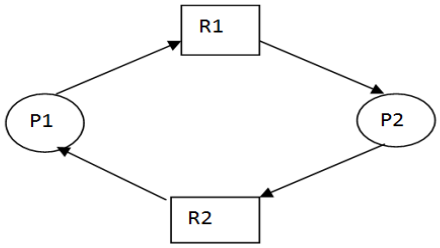


Figure 4.8: Deadlock with identifiable resource elements.

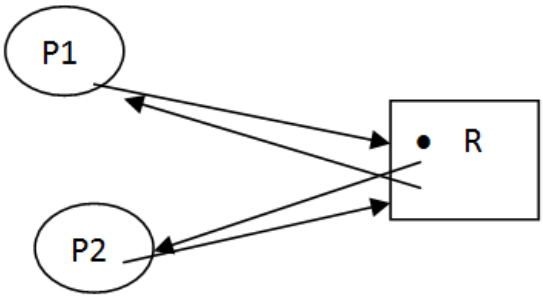


Figure 4.9: Deadlock with fungible resources.

- Detection schemes find the circular chain. At least one victim is chosen; its partial service is rolled back and its resources freed, in an attempt to break the deadlock and allow the other processes to complete. The victim(s) can be restarted later, without alteration, but its rolled back service must be repeated.
- Holt provides an example of a single process in deadlock: process revenge is infinitely suspended by a wait for an event that never occurs. Similarly, states that a kernel process is in deadlock if it is permanently blocked waiting for a resource that will never become available. In accordance with the halting problem, however, this type of dead state is impossible, in general, to prevent or even detect with current technology. A deadlock cannot occur with only one process. Deadlock is an anomaly of traffic control (i.e., competition synchronization) and contains at least two competing processes.
- Holt also considers deadlock as a circular wait with consumable resources, such as messages, that can be dynamically created and destroyed. Message deadlocks, however, are controllable neither by serial execution nor by most of the previously.
- Researchers in the field of distributed databases typically limit their applications to requests for data locks, and their treatments are consistent with deadlock material. Additional restrictions are

required to obtain a global knowledge of a deadlock, however, and fungible resources (basically OR-requests) are generally not included in detection algorithms.

- Some additional remarks will be useful. Processes in deadlock may be suspended in an inactive wait or looping endlessly in a busy wait. They may be holding resource buffers, "soft" resources, which are typically fungible. In an attempt to differentiate between different types of dead states, some of the literature has to use the term resource deadlock. In the remainder of this paper, we accede to this terminology, although dead states with resources are obviously not always resource deadlocks.

Table 4.1: A Classification of dead states

	Circular dead states	General dead states (Supersets of corresponding circular dead states)
Caused by cooperation mechanisms	Communication deadlocks	Communication dead states
Caused by competition mechanisms	Scheduling deadlocks	Scheduling dead states
Caused by interleaved execution of competing requests whose service is restricted by cooperation mechanisms.	Interleaved deadlocks (Supersets of resource deadlocks)	Interleaved dead states

Circular Dead States

Deadlock has been defined as any infinite circular waiting state in the literature of communication and database systems. This simplification also occurs in several operating systems, even when the rest of the section on deadlock clearly is about resource deadlock. We illustrate three distinct classes of circular infinite waits, with only resource deadlock, a subset of the third class, having all of the characteristics.

Communication Deadlock

A dead state can occur with a circular chain of incorrectly written processes that wait for a message from another in the chain before each sends the awaited message. For example, two processes will never complete if they are programmed to wait for a (consumable) resource instance before they create the resource instance requested by the other. The producer/consumer example previously cited also contains errors in the cooperation mechanism, preventing the sending of the awaited signal. These communication deadlocks are not caused by interleaved code execution, nor can they be prevented by resource pre-allocation, serial execution, or maintenance of safe states. Typically, after some period of time, a circular wait is detected or assumed and the processes are aborted.

Scheduling Deadlock

A circular chain of four cars infinitely waiting behind stop signs at each of the four corners of an intersection has been called a resource deadlock. A dead state with two trains approaching on different tracks where each gives precedence to the other at a crossroads has also been incorrectly labelled. These cars and trains are not waiting for resources held by others; the scheduler has failed to assign a single process the priority required for accessing the resource. Recovery does not entail roll back and the resultant repetition of service; it is sufficient to dynamically modify the scheduling

algorithm at the present point of service, perhaps by waiving one vehicle through. These scheduling deadlocks, caused by incomplete competition mechanisms, do not require any alteration of processes.

Interleaved Deadlock

Interleaved deadlock is caused by a combination of competition and cooperation mechanisms, such that only the competition mechanisms are incorrect. Requests from at least two processes compete for the same resource elements and the scheduler interleaves their execution, allocating a resource to each. Each process is restricted from releasing the held resource while waiting for an event to occur at another process. In resource deadlock, the scheduler continually assigns highest priority to the earlier requests (resources cannot be pre-empted), blocking later (active or suspended) competing requests. Each high priority request waits for the service of its process's blocked request(s). A circular chain exists of processes that are waiting for resources held by other processes in the chain. Resource deadlock can be controlled by mechanisms of either the processes or of the scheduler. For example, processes can agree to access resources in a linear order or the scheduler can maintain a safe state. When resource deadlock is rare, control systems may avoid the overhead of prevention algorithms. Detection schemes find the circular wait. After roll back, one or more processes are restarted for repeated service, without correction, in the hope that the interleaved execution that caused the dead state does not recur.

Active and Inactive Dead States

In some of the computer science literature, a deadlock is defined as a state containing processes that can make no more forward progress based on the transitions defined in the protocol. Alternatively, some researchers label deadlock as such inactive states and live lock as active ones. There exist three classes of these dead states, corresponding to each of the circular waiting states.

Communication Dead States

Communication dead states are caused by cooperation mechanisms. Such dead states may be active, such as a kernel process that is infinitely looping, or inactive, such as infinite suspensions with wait macro facilities. An inactive dead state can also occur if a process outputs an incorrect value at a resource; all processes whose service is dependent upon that value will be aborted and never complete service. Or, if a resource is flawed, perhaps containing a defective coprocessor, then a process may be reconceived continuously and still never make progress, assuming no other computer is available. Similarly, if a resource is represented by a flawed server, a client may not receive service. In addition, processes that access forbidden resources should be doomed to failure. We claim that all of these processes enter into a dead state as defined in this section unless correction occurs. Communication dead states may be impossible to prevent. They are generally the most expensive dead states to correct, requiring that processes be reconceived by their users.

Scheduling Dead States

- Scheduling dead states are caused by incorrect conflict resolution, independent of user assigned cooperation mechanisms. For example, a process may be suspended on a queue from which it is never retrieved. In addition, if a scheduling policy requires pre-allocation of resources, two malevolent or poorly controlled dining philosophers can alternate acquiring chopsticks and keeping the philosopher between them in an inactive dead state. Alternatively, two processes may repeatedly access a resource, collide, and back off, perhaps following a roll back policy to recover from resource deadlock, causing an accordion effect. Circular routing is also an active scheduling dead state.

- This class of anomalies is not caused by interleaved execution of process's requests. For correction, waits can be detected and the scheduling policy adjusted without loss of service. Typically, older processes are buffered at their present point of service and their priority increased by their waiting time (a policy called aging). A linear order of processes resolves conflict for processes with the same access time. Sometimes, insertion of a random delay, as in Ethernet, is a probabilistic handler. Other adjustments may be necessary to the scheduling policy, but not to the processes themselves.

Interleaved Dead States

A combination of competition and cooperation mechanisms can cause dead states if the scheduling mechanism allows interleaved access to pre-emptable instead of non-preemptable resources. At least two requests from at least two processes compete; service of the competing requests is interleaved. In the lost update problem, two inputs are serviced at the same resource at the same time, each waiting for its output to perform an update. The scheduler places an additional wait restriction on simultaneous input requests from different users so that they cannot complete service as long as any of them have not had their outputs serviced. The first output writes to the resource, beginning its service, and waits for its input. The second output writes to the resource and conflicts with the first output. The later write is assigned a higher priority and overwrites the earlier one. The first output and its matching input are cancelled, entering into an inactive dead state. The second output and its input now complete service since there are no more outstanding outputs. Inconsistent retrieval has similar characteristics. This is not true for all outputs, for example, in broadcasting systems using C-Aloha.

Two processes may continuously alternate in moving an access mechanism to seek the location for disk input. The first process's seek performs its update, but must wait until its input is serviced. Its execution is interleaved with a second process. The second process's seek is given higher priority, pre-empting the first process's seek request, but is interrupted before issuing its read. The first process next detects the overwrite and reissues it is seek, cancelling the second process's seek, with this cycle continuing and no input completed. Note that resources are not being held; it would be possible for a third process to be scheduled in between these two and complete service, even though the first two processes repeatedly pre-empt each other's seek.

Resource pre-allocation and serial execution is effective prevention schemes for interleaved dead states. Database systems, for example, either pre-allocate data or else seek serially equivalent schedules.

Unacceptable Waits

Some of the literature defines deadlock as a state in which processes wait forever, and thus never complete service. We expand study to include unacceptable waits, states in which some processes do not complete service either by the end of time or by a shorter time restriction. (It does not concern us if a wait is unbounded, such as in the protocols of stop signs or slotted Aloha, only that the wait is beyond a limit imposed by the user or the resource system.)

Communication Unacceptable Waits

In communication dead states, users assign cooperation mechanisms for progress that can never occur. In communication unacceptable waits, cooperation restrictions prevent progress within a necessary time interval. For example, a process may have to wait to enter its critical section until an independent process has been serviced in the critical section. If it is possible that the second process will not enter the critical section within a required time period, correction is required. In many cases,

cooperation mechanisms that delay service are not incorrect; they ensure proper usage of the resource.

Scheduling Unacceptable Waits

In scheduling dead states, a scheduler assigns competition mechanisms that prevent the progress of processes. In scheduling unacceptable waits, the scheduler assigns priorities to conflicting processes that delay service of some or all of the processes within a given time constraint. For example, batch processes may be repeatedly given low priorities if interactive requests keep arriving. Device drivers may be delayed by higher priority real-time processes and lose data. For processes with the same urgency, scheduling waits (such as wall times or clock interrupts for inactive waits, or time quanta or hop counters for active waits) may signal the scheduler to employ aging mechanisms. On the other hand, a hard real-time process, which is aborted if it does not complete service within its time constraint, requires process-based priority assignments. In most cases, priority assignments are not incorrect; they are necessary to meet service requirements. If process arrival rate approaches or exceeds the maximum service rate of a system for a significant interval of time. We learn from queuing theory that waits become unacceptable using any scheduling discipline. Such a wait, called instability, is controllable chiefly by throttling mechanisms, an extreme form of aging, which often cause unacceptable delays in the user buffer.

Interleaved Unacceptable Waits

In interleaved dead states, each process that is holding resources cannot complete service based on a wait for another's service. If resources are locked, the scheduler always assigns newly arriving requests a lower priority, guaranteeing that processes currently being serviced will not be pre-empted. Then interleaved execution can cause resource deadlock. If resources are not locked, interleaved execution may cause the repeated pre-emption of processes currently being serviced and an interleaved dead state. In interleaved unacceptable waits, processes also execute requests in an interleaved order, with some requests unable to complete service because of their process's delayed requests. For example, in fragmentation-based congestion collapse in networks, a packet is broken into fragments and then must be reassembled before it is forwarded. Buffering or servicing of some fragments causes other fragments from each packet to be discarded, perhaps because of time-outs or overflowing buffers. In reassembly deadlock, the buffers of some nodes are filled with partial messages that are waiting for packets that cannot be accepted, causing congestion in the surrounding nodes.

The Model and Resource Deadlock

The identified two waiting relationships in resource deadlock: Follow within, a cooperation mechanism, and blocked by, a mutual exclusion mechanism. In the wait-for graph of Figure 4.10, two processes in resource deadlock compete for two resource units. The arrow within each process circle signifies a Follow within relationship and the composite arrows between two processes signify a blocked by relationship. Although we have repeated the requirement in that resource deadlock requires non-preemptable resources, interleaved deadlock can also occur if a process not contained in the circular wait is allowed to pre-empt the blocking request and complete service, as long as the resource is then reallocated to the pre-empted request(s). Resource deadlock with fungible resources requires this relationship between all processes with a request that holds a unit of a requested resource.

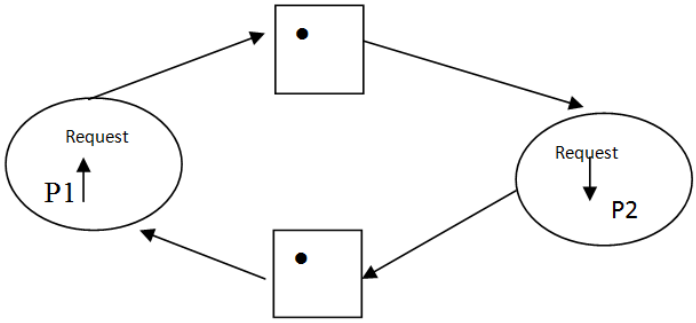


Figure 4.10: Modified wait-for graph for resource deadlock.

The producer-consumer problem of the introduction contains two processes, each with a follow within wait, one with a blocked by wait, and the other with a contingency containing a request of the other process. This anomaly is not a resource deadlock. Confusion occurs because the same construct has been used to implement both competition and cooperation synchronization.
A dead state that appears to be ambiguous. In priority deadlock, a system pre-allocates resource buffers for message reassembly. All buffers at a receiving router have been reserved for low-priority messages. High-priority packets are scheduled to all of the bandwidth, but are discarded by the receiving router because of the lack of buffers. The sending router keeps timing out and resending the high-priority packets, so that low-priority packets cannot reach their pre-allocated buffers and complete their reassembly. A circular dead state occurs due to interleaved service of packets that are assigned resources needed by others in the circular chain. We see that the prevention schemes of (total) resource pre-allocation as well as resource pre-emption are applicable. Yet, are there inconsistencies? Can the scheduler break the dead state without roll back by simply raising the priority of low-priority packets? In addition, it appears that bandwidth is pre-empted, violating one of the preconditions. On further examination, we see that bandwidth is repeatedly reassigned to the high-priority packets. In addition, high-priority packets are indeed rolled back each time they are refused buffer space at the receiving node. The deadlock can be broken only following roll back. This resource deadlock indeed satisfies our definition. Low-priority requests are blocked by high-priority requests for bandwidth, while high-priority requests are blocked by low-priority requests for buffer space. Messages cannot complete service since each of the blocking packets has been assigned a contingency containing a blocked packet of its message.

4.6.1 Characterization

Deadlock occurs if the following four conditions take place simultaneously in a system:

Mutual Exclusion: At least one resource must be held in a non-sharable mode. It means that only one process at a time can use the resource. If another process requests the resource, the requesting process must be delayed until the resource has been released.
Hold and Wait: A process must be holding at least one resource and waiting, to acquire additional resources that are currently being held by other processes.
No Pre-emption: Resources cannot be pre-empted. A resource can be released only voluntarily by the process holding it after that process has completed its task.

Circular Wait: A set $\{P_0\ P_1, P_2 ... P_n\}$ of waiting processes must exist such that P_1 is waiting for a resource that is held by P_1 is waiting for a resource that is held by P_2, ..., P_{n-1} is waiting for a resource that is held by P_n and P_n is waiting for a resource that is held by P_0.
All four conditions must hold for a deadlock to occur.

Caution
Processes in communication deadlock must be reconceived by their users before they can complete service.

Self Assessment Question

1. Several processes compete for a finite set of resources in a multi-programmed environment.
(a). True (b). False

2. A process Pi that is waiting for some currently unavailable resource is said to be.....
(a). Operating system (b). blocked
(c). Both(a) and (b) (d). None of these.

3. A condition where at least one of the resources is non-sharable in a system is called.....
(a). Mutual Exclusion (b). hold and wait
(c). circular wait (d). None of the above

4. Shared resources always involve in deadlocks.
(a). True (b). False

5. A deadlock..... algorithm requires each process to make known in advance the maximum number of resources of each type that it may need.
(a). Banker's (b). avoidance
(c). resource request (d). None of these.

4.7 Deadlock Prevention

Deadlock prevention works by preventing one of the four Coffman conditions from occurring.

1. Removing the mutual exclusion condition means that no process will have exclusive access to a resource. This proves impossible for resources that cannot be spooled. But even with spooled resources, deadlock could still occur. Algorithms that avoid mutual exclusion are called non-blocking synchronization algorithms.
2. The hold and wait or resource holding conditions may be removed by requiring processes to request all the resources they will need before starting up (or before embarking upon a particular set of operations). This advance knowledge is frequently difficult to satisfy and, in any case, is an inefficient use of resources. Another way is to require processes to request resources only when it has none. Thus, first they must release all their currently resources before requesting all the resources they will need from scratch. This too is often impractical. It is so because resource may be allocated and remain unused for long periods. Also, a process requiring a popular resource may have to wait indefinitely as such a process may be always allocated to some process, resulting in resource starvation.

3. The no pre-emption condition may also be difficult or impossible to avoid as a process has to be able to have a resource for a certain amount of time, or the processing outcome may be inconsistent or thrashing may occur. However, inability to enforce pre-emption may interfere with a priority algorithm. Pre-emption of a “locked out” resource generally implies a rollback, and is to be avoided, since it is very costly in overhead. Algorithms that allow pre-emption include lock-free and wait-free algorithms and optimistic concurrency control.
4. The final condition is the circular wait condition. Approaches that avoid circular waits include disabling interrupts during critical sections and using a hierarchy to determine a partial ordering of resources. If no obvious hierarchy exists, even the memory address of resources has been used to determine ordering and resources are requested in the increasing order of the enumeration.

Exercise: Check Your Progress 3
Note: i) Use the space below for your answer.
Ex1: Monitor?

……..……

……..……

……...

Ex2: Identify Model and Resource Deadlock.

……..……

……..……

……...

4.8 Avoidance and Detection

4.8.1 Avoidance

Deadlock can be avoided if certain information about processes is available to the operating system before allocation of resources, such as which resources a process will consume in its lifetime. For every resource request, the system sees if granting the request will mean that the system will enter an unsafe state, meaning a state that could result in deadlock. The system then only grants requests that will lead to safe states. In order for the system to be able to determine whether the next state will be safe or unsafe, it must know in advance at any time:

- Resources currently available
- Resources currently allocated to each process
- Resources that will be required and released by these processes in the future

It is possible for a process to be in an unsafe state but for this to not result in a deadlock. The notion of safe/unsafe states only refers to the ability of the system to enter a deadlock state or not. For example, if a process requests A which would result in an unsafe state, but releases B which would prevent circular wait, then the state is unsafe but the system is not in deadlock. One known algorithm that is used for deadlock avoidance is the Banker's algorithm, which requires resource usage limit to

be known in advance. However, for many systems it is impossible to know in advance what every process will request. This means that deadlock avoidance is often impossible.
Two other algorithms are wait/die and wound/wait, each of which uses a symmetry-breaking technique. In both these algorithms there exists an older process (O) and a younger process (Y). Process age can be determined by a timestamp at process creation time. Smaller timestamps are older processes, while larger timestamps represent younger processes.

	Wait/Die	Wound/Wait
O needs a resource held by Y	O waits	Y dies
Y needs a resource held by O	Y dies	Y waits

Banker's Algorithm
The algorithm was developed in the design process for the operating system and originally. The name is by analogy with the way that bankers account for liquidity constraints**.**

Algorithm
The Banker's algorithm is run by the operating system whenever a process requests resources. The algorithm avoids deadlock by denying or postponing the request if it determines that accepting the request could put the system in an unsafe state (one where deadlock could occur). When a new process enters a system, it must declare the maximum number of instances of each resource type that may not exceed the total number of resources in the system. Also, when a process gets all its requested resources it must return them in a finite amount of time.
Resources
For the Banker's algorithm to work, it needs to know three things:

- How much of each resource each process could possibly request
- How much of each resource each process is currently holding
- How much of each resource the system currently has available

Resources may be allocated to a process only if it satisfies the following conditions:

- request ≤ max, else set error condition as process has crossed maximum claim made by it.
- request ≤ available, else process waits until resources are available.

Some of the resources that are tracked in real systems are memory, semaphores and interface access.
The Banker's algorithm derives its name from the fact that this algorithm could be used in a banking system to ensure that the bank does not run out of resources, because the bank would never allocate its money in such a way that it can no longer satisfy the needs of all its customers. By using the Banker's algorithm, the bank ensures that when customers request money the bank never leaves a safe state. If the customer's request does not cause the bank to leave a safe state, the cash will be allocated otherwise the customer must wait until some other customer deposits enough.
Basic data structures to be maintained to implement the Banker's algorithm:
Let n be the number of processes in the system and m be the number of resource types. Then we need the following data structures:

- Available: A vector of length m indicates the number of available resources of each type. If Available[j] = k, there are k instances of resource type Rj available.
- Max: An n×m matrix defines the maximum demand of each process. If Max[i,j] = k, then Pi may request at most k instances of resource type Rj.

- Allocation: An n×m matrix defines the number of resources of each type currently allocated to each process. If Allocation [i,j] = k, then process Pi is currently allocated k instance of resource type Rj.
- Need: An n×m matrix indicates the remaining resource need of each process. If Need[i,j] = k, then Pi may need k more instances of resource type Rj to complete task.
  Note: Need = Max - Allocation.

4.8.2 Detection

The deadlock detection, deadlocks are allowed to occur. Then the state of the system is examined to detect that a deadlock has occurred and subsequently it is corrected. An algorithm is employed that tracks resource allocation and process states, it rolls back and restarts one or more of the processes in order to remove the detected deadlock. Detecting a deadlock that has already occurred is easily possible since the resources that each process has locked and/or currently requested are known to the resource scheduler of the operating system. This approach is simpler than deadlock avoidance or deadlock prevention. It is so because predicting a deadlock before it happens is difficult, as it is generally an undecidable problem, which itself results in a halting problem. However, in specific environments, using specific means of locking resources, deadlock detection may be decidable. In the general case, it is not possible to distinguish between algorithms that are merely waiting for a very unlikely set of circumstances to occur and algorithms that will never finish because of deadlock. Deadlock detection techniques include, but are not limited to model checking. This approach constructs a finite state-model on which it performs a progress analysis and finds all possible terminal sets in the model. These then each represent a deadlock.

After a deadlock is determined, it can be corrected by using one of the following methods:

Process Termination: One or more process involved in the deadlock may be aborted. We can choose to abort all processes involved in the deadlock. This ensures that deadlock is resolved with certainty and speed. But the expense is high as partial computations will be lost. Or, we can choose to abort one process at a time until the deadlock is resolved. This approach has high overheads because after each abortion an algorithm must detect if the system is still in deadlock. Several factors must be considered while choosing a candidate for termination, such as priority and age of the process.

Resource Pre-emption: Resource allocated to various processes may be successively pre-empted and allocated to other processes until deadlock is broken.

4.9 Recovery from Deadlocks

We can recover from a deadlock by using two approaches:

- Kill the processes that release all resources for killed process.
- Take away resources.

rocess Termination:

When recovering from a deadlock by process termination, we have two approaches.

- We can terminate all processes involved in a deadlock
- We can terminate them one by one until the deadlock disappears

Killing all processes is costly since some processes may have been doing something important for a long time and will need to be re-executed again. Killing one process at a time until deadlock is resolved is also costly. We must run deadlock detection algorithm every time we terminate a process to make sure we got rid of the deadlock. Some priority must be considered when terminating processes because we do not want to kill an important process when less important processes are available. Priority might also include things like. How many resources are being held by that process

- How long has it executed
- How long it has to go before it completes
- How many processes will have to be terminated
- How many resources it needs to complete its job etc?

Resource Pre-emption:

This approach takes resources from waiting processes and gives them to other processes. The victim process cannot continue regularly. We have a choice of how to handle it. We can either terminate that process or roll it back to some previous state so that it can request the resources again.

We have to consider the following factors for applying resource pre-emption:

Selection of the Victim: The most important decision is the selection of the victim process to be pre-empted. It will be determined by considering the number of resources held by that process, the amount of time consumed by the process etc.

Rollback: When a resource is pre-empted from a process, it cannot continue its normal execution. We should rollback that process to a safe state so that it can be restarted later.

Starvation: It is possible that the same process is selected as victim process for pre-emption. If this happens, the process will not continue and a starvation situation will occur. We should ensure that no process should be selected as a victim more than a limited number of times.

If the system has resource pre-emption, a deadlock cannot occur. The type of resource pre-emption we are talking about is abnormal pre-emption. It only occurs when a deadlock detection mechanism detect a deadlock.

4.10 Combined Approach to Deadlock Handling

One-basic strategy for handling deadlocks is to ensure violation of at least one of the three conditions necessary for deadlock (exclusive control, hold-wait, and no pre-emption). This method is usually referred to as deadlock prevention, unless it is primary aim is to avoid deadlock by using information about the processes future intentions regarding resource requirements. A totally strategy interrogates the process/resource relationships from time to time in order to identify the existence of a deadlock. This latter method presumes that the system can subsequently do something about the problem.

Detection Techniques: These techniques assume that all resource requests will be granted eventually. A periodically invoked algorithm examines current resource allocations and outstanding requests to determine if any processes or resources are deadlocked. If a deadlock is discovered, the system must recover as gracefully as possible by pre-empting resources from acted processes until the deadlock is broken. Detection-scheme overhead includes not only the run-time cost of the algorithm but also the potential losses inherent in pre-empting resources. Since no action takes place until a deadlock actually occurs, re-sources may be held idle by blocked processes for long periods of time. Sometimes, using detection principles effectively is difficult for example, when pre-emption of resources such as tape drives might acceptable overhead. Nevertheless, detection techniques have some advantages since the schemes are invoked intermittently and only essential pre-emption's need be performed. In the database context, detection methods rely on the management system to abort, roll back to a previous checkpoint, and restart at least one process to break the deadlock. Here, the problem of rollback and recovery assumes great importance from the viewpoint of maintaining database consistency.

Prevention Mechanisms: Prevention is the process of constraining system users so that requests leading to deadlock never occur. Most proposals for prevention require each process to specify all needed resources before transactions begin. Deadlocks can be prevented in several ways, including requesting all resources at once, pre-empting resources held, and ordering resources.

The simplest way of preventing deadlock is to outlaw concurrency, but this leads to very poor resource utilization and is not consistent with current system design philosophies. Another method requires that all resources be acquired before processing starts. Such a scheme is inefficient, since resources held may be idle for prolonged periods, but works well for processes which perform a single burst of activity, such as input/output drivers, since the resource can be released immediately after each use. For processes with requirements, the method can be impractical. In a database environment, it may be impossible for a data-driven process to specify and acquire all needed resources before beginning execution. In any case, the scheme discriminates heavily against data-driven processes where relationships in the data indicate what future resources are required for processing. Certain other prevention methods require a blocked process to release resources requested by an active process. For example, when a process needs more main memory than is currently available, it becomes blocked. Subsequently the process is swapped to secondary storage by pre-empting its memory for use by an active process. The blocked process is swapped back only when the entire, larger quantity of memory is available. For some peculiar situations in database systems, this use of pre-emption to prevent deadlocks is subject to cyclic restart, in which two or more processes loop by continually blocking, aborting, and restarting each other.

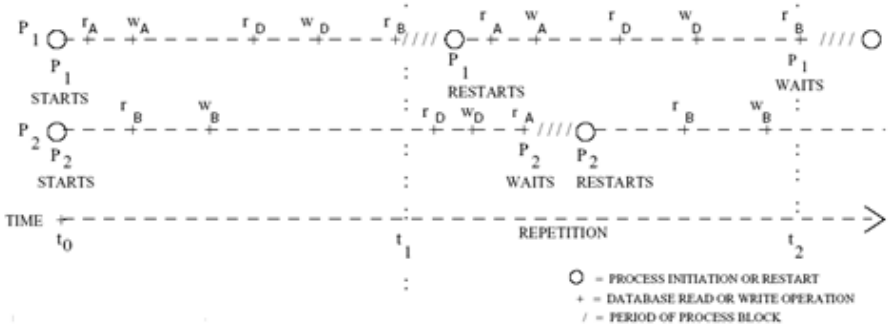


Figure 4.11: Cyclic restart.

Self Assessment Question

6. .....algorithm is used where resources have multiple instances.

(a). Resource Allocation Graph (b). Bankers

(c). resource request (d). None of the above.

7. If the system does not ensure that a deadlock cannot be prevented or a deadlock cannot be avoided, then a deadlock may occur.

(a). True (b). False

8. .....is a variant of the resource allocation graph, which can be used to detect deadlocks in the system that has resources, all of which have only single instances.

(a). Wait-for Graph (b). Prevention

(c). avoidance (d). None of these.

9.....are the basic approaches to handle deadlocks.

(a). Prevention (b). avoidance

(c). detection (d). All of these.

10 Codes that reference one or more variables in a.....fashion while any of those variables is possibly being altered by another thread.

(a). Read-update-write (b). write-update-read
(c). Both (a) and (b) (d). None of these.

4.11 Summary

- The use of locks is actually very easy. For every thread, before it accesses the set of data items.
- It is successful, this thread enters the critical section and the lock is locked.
- The key to preventing trouble involving shared storage is find some way to prohibit more than one process from reading and writing the shared data simultaneously.
- The process synchronization there is a thumb rule which states that production of information is done by the producer and the consumption is done by the consumer processes.
- Monitors are implemented by using queues to keep track of the processes attempting to become active monitor.

4.12 Keywords

Bounded Waiting: The exists a limit as to how many other processes can get into their critical sections after a process requests entry into their critical section and before that request is granted.
Circular Wait: The circular chain of waiting, in which each process is waiting for a resource held by the next process in the chain.
Deadlock: A condition that occurs when two processes are each waiting for the other to complete before proceeding.
Mutual Exclusion: The no more than one process can execute in its critical section at one time.
Operating System: The most important program that runs on a computer. Every general-purpose computer must have an operating system to run other programs.

4.13 Review Questions

1. What are the critical-section problems?
2. Explain the synchronization hardware and semaphores.
3. What are the advantages and disadvantages of semaphore?
4. What are the classical problems of synchronization?
5. Define the critical regions and monitors.
6. What are the deadlocks-system models? And classification of dead states.
7. What are the characteristics of deadlock?
8. Discuss the deadlock prevention, avoidance and detection.
9. What is the recovery from deadlocks?
10. Explain the combined approach to deadlock handling.

Answers for Self Assessment Questions

1. (a)	2. (b)	3. (a)	4. (b)	5. (b)
6. (b)	7. (a)	8. (a)	9. (d)	10. (a)

4.14 Review Questions

1. http://codex.cs.yale.edu/avi/os-book/OSE1/slide-dir/PDF-dir/ch6.pdf
2. Operating System Principles by Silberschatz, Galvin and Gagne

5

Security and Protection

STRUCTURE

5.0 Objective

5.1 Introduction

5.2 Protection

5.3 Domain of Protection

5.4 Access Matrix

5.5 Implementation of Access Matrix

5.6 Revocation of Access Rights

5.7 Language-Based Protection

5.8 Security

5.9 Authentication

5.10 One-Time Passwords

5.11 Program Threats

5.12 System Threats

5.13 Threat Monitoring

5.14 Encryption

5.15 Summary

5.16 Keywords

5.17 Review Questions

5.18 Further Studies

5.0 Objectives

After studying this chapter, you will be able to:

- Discuss protection of a system
- Discuss the domain of protection
- Define the implementation of access matrix

5.1 Introduction

OS Security revolves around the appropriate protection of four elements. Confidentiality prevents or minimizes unauthorized access and disclosure of data and information. Integrity makes sure that the

data being worked with is actually the correct data. Availability is the property of a system or system resource being accessible and usable upon demand by an authorized system entity, according to performance specification for the system. Authenticity makes possible that a computer system be able to verify the identity of a user.

5.2 Protection

5.2.1 Goals of Protection

As computer systems have become more sophisticated and pervasive in their applications, the need to protect their integrity has also grown. Protection was originally conceived as an adjunct to multiprogramming operating systems, so that untrustworthy users might safely share a common logical name space, such as a directory of files, or share a common physical name space, such as memory. Modern protection concepts have evolved to increase the reliability of any complex system that makes use of shared resources.

There are several reasons for providing protection. Most obvious is the need to prevent mischievous, intentional violation of an access restriction by a user. Of more general importance, however, is the need to ensure that each program component active in a system uses system resources only in ways consistent with the stated policies for the uses of these resources. This requirement is an absolute one for a reliable system.

Protection can improve reliability by detecting latent errors at the interfaces between component subsystems. Early detection of interface errors can often prevent contamination of a healthy subsystem by a subsystem that is malfunctioning.

An unprotected resource cannot defend against use (or misuse) by an unauthorized or incompetent user. A protection-oriented system provides means to distinguish between authorized and unauthorized usage.

5.2.2 Principles of Protection

- The ***principle of least privilege*** dictates that programs, users, and systems be given just enough privileges to perform their tasks.
- This ensures that failures do the least amount of harm and allow the least of harm to be done.
- For example, if a program needs special privileges to perform a task, it is better to make it a SGID program with group ownership of “network” or “backup” or some other pseudo group, rather than SUID with root ownership. This limits the amount of damage that can occur if something goes wrong.
- Typically each user is given their own account, and has only enough privilege to modify their own files.
- The root account should not be used for normal day to day activities - The System Administrator should also have an ordinary account, and reserve use of the root account for only those tasks which need the root privileges

5.2.3 Protection Mechanisms

The concept of multiprogramming introduces the sharing resources among users. This sharing involves Memory, I/O devices, Programs and Data. The ability to share these resources introduces the need for protection. An OS may offer protection along the following Spectrum:

No Protection: This is appropriate when sensitive procedures are being run at separate times.

Isolation: This approach implies that each process operates separately from other processes, with no sharing. Each process has its own address space, files, and other objects

Share all or Share nothing: In this method, the owner of an object declares it to be public or private, in the other words, only the owner's processes may access the object.

Share via access limitation: The OS checks the permissibility of each access by a specific user to specific object; the OS therefore acts as a guard between users and objects, ensuring that only authorized accesses occur.

Share via dynamic capabilities: This extends the concept of access control to allow dynamic creation of sharing rights for objects.

Limit use of an object: This form of protection limits not just access to an object but the use to which that object may be put.

A given OS may provide different degree of protection for different objects, users and applications The OS needs to balance the need to allow sharing, with the need to protect the resources of individual users.

Protection of Memory

In a multiprogramming environment, protection of main memory is essential. The concern here is not just security but the correct functioning of the various processes that are active. The separation of the memory space of various processes is easily accomplished with a virtual-memory scheme. Segmentation or Paging, or two in combination, provides an effective tools of managing main memory. If complete isolation is sought, then the OS must simply ensure that each segment or page accessible only by the process to which it is assigned. This is accomplished by requiring that there be no duplicate entries in page and/or segment tables. If sharing is to be allowed then the same segment or page may appear in more than one table. Segmentation specially lends itself to the implementation of protection and sharing policies. Because each segment table entry includes a length as well as a base address. A program cannot access a main memory location beyond the limit of a segment. To achieve sharing, it is possible for a segment to be referenced in the segment tables of more than on process. In the paging system, the page structure of the programs and data is not visible to the programmer.

The measures taken to control access in a data processing systems fall into two categories:

- User-Oriented
- Data-Oriented

User-Oriented Access Control

User control of access is sometimes referred to as Authentication. The most common technique for user access control on a shared system or server is the user log, which requires ID and Password. User access control in distributed environment can be either centralized or decentralized in a centralized approach network provides a log on service, determining who is allowed to use the network and to whom the user is allowed to connect. Decentralized user access control treats the network as a transport communication link, and the destination host carries out the usual log on procedure. In many networks, two levels of access control may be used.

Data-Oriented Access Control

Following successful log on, the user has been granted access to one or set of hosts and applications. At this time we need Data access control. In this regard real world operating system protection models fall basically into one of two types:

1. Mandatory access controls (MAC)
2. Discretionary access controls (DAC)

In computer security passive resources are called *objects* and active entities that utilize the resources are called *subjects*. Typical objects include: files, directories, memory, printers … And typical subjects include: users, processes… The roles depend on situation: For example, a Process can

request access to some resource (act as a subject) and later be a target of access request (act as an object)

Mandatory access controls (MAC)

In Mandatory access controls, also called multilevel access control, Objects (information) are classified on hierarchical levels of security sensitivity (typically, top secrets, secret, confidential) Subjects (Users) are assigned their security clearance. Access of a subject to an object is granted or denied depending on the relation between the clearance of the subject and the security classification of the object. Lattice model and Bell-LaPadula model are based on MAC

Discretionary access controls (DAC)

Each object has its unique owner. The owner exercises its discretion over the assignment of access permissions. Lampson introduced the access matrix model for DAC. The core of this model is a matrix whose rows are indexed by subjects and columns by objects.

In real systems, however, access control matrices are not very practical, because the matrix is usually sparse and there is a lot of redundancy and new subjects and objects can be added or removed easily, but the centralized matrix could become a bottleneck. The matrix may be decomposed by columns, yielding Access Control List (ACL). Thus for each object, an ACL details users and their permitted access rights. ACL may contain a default or public entry. Decomposition by rows yields capability tickets. A capability ticket specifies authorized objects and operations for a user. Each user has a number of tickets and may be authorized to lend or give them to others. Because tickets may be dispersed around the system, they present a greater security problem than ACL. To accomplish this problem, OS hold all tickets on behalf of the users. These tickets would have to be held in a region of memory inaccessible to users.

5.2.4 Protection Based on an OS Mode

Most processor supports at least two modes of operations: 1- User mode 2- Kernel mode. The reason for using two modes should be clear. It is necessary to protect the OS and the key OS table such as process control blocks, from interference by user programs in kernel mode. This level of control is not necessary for user mode.

File Sharing

Multi user systems almost always require that files can be shared among a number of users. We have several access rights such as Reading, Appending, Updating, Changing protection… these Access rights can be provided to different classes of users such as Specific user (with user ID), user groups and all users. When access is granted to more than one user to append or update a file The OS must enforce discipline. A brute-force approach is to allow a user to lock the entire file when it is to be updated.

Operating Systems Vulnerabilities

Vulnerabilities will probably always exist in large and complex software systems. At least with today's software methods, techniques, and tools, it seems to be impossible to completely eliminate all flaws. Operating systems are examples of software components that are large, very complex, and vulnerable. At the same time, these components play an important role in the achievement of overall system security, since many protection mechanisms and facilities, such as authentication and access control, are provided by the operating system. Vulnerability is defined as a place at which the probability of a breach exceeds a predefined threshold.

Observations on common operating system vulnerabilities indicate that similar vulnerabilities have been grouped together in a heuristic manner rather than according to some formal model. A collection of five common security problems has been identified and will be further described below. These are:

- Improper input validation

- Weak cryptographic algorithms
- Weak authentication protocols
- Insecure bootstrapping
- Configuration mistakes

Caution
Inflexibility of a protection system can prevent to secure a system or data.

5.3 Domain of Protection

A computer can be viewed as a collection of processes and objects (both HW & SW). The need to know principle states that a process should only have access to those objects it needs to accomplish its task, and furthermore only in the modes for which it needs access and only during the time frame when it needs access. The modes available for a particular object may depend upon its type.

5.3.1 Domain Structure

- A ***protection domain*** (see Figure 5.1) specifies the resources that a process may access.
- Each domain defines a set of objects and the types of operations that may be invoked on each object.
- An ***access right*** is the ability to execute an operation on an object.
- A domain is defined as a set of < object, { access right set } > pairs, as shown below. Note that some domains may be disjoint while others overlap.

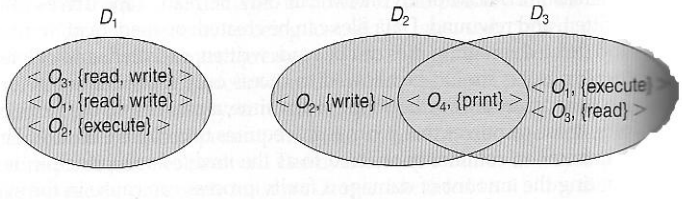


Figure 5.1: System with three protection domains.

- The association between a process and a domain may be *static* or *dynamic*.
  - If the association is static, then the need-to-know principle requires a way of changing the contents of the domain dynamically.
  - If the association is dynamic, then there needs to be a mechanism for ***domain switching.***
- Domains may be realized in different fashions - as users, or as processes, or as procedures. E.g. if each user corresponds to a domain, then that domain defines the access of that user, and changing domains involves changing user ID.

5.3.2 An Example: UNIX

- UNIX associates domains with users.
- Certain programs operate with the SUID bit set, which effectively changes the user ID, and therefore the access domain, while the program is running. (and similarly for the SGID bit.) Unfortunately this has some potential for abuse.
- An alternative used on some systems is to place privileged programs in special directories, so that they attain the identity of the directory owner when they run. This prevents crackers from placing SUID programs in random directories around the system.

- Yet another alternative is to not allow the changing of ID at all. Instead, special privileged daemons are launched at boot time, and user processes send messages to these daemons when they need special tasks performed.

5.3.3 An Example: MULTICS

- The MULTICS system uses a complex system of rings, each corresponding to a different protection domain, as shown in Figure 5.2:

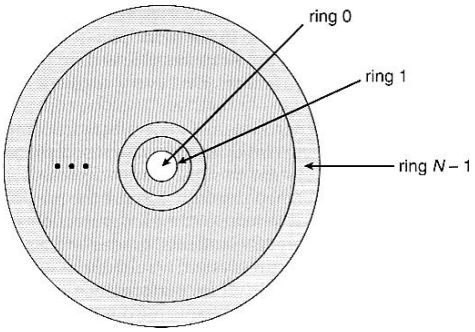


Figure 5.2: MULTICS ring structure.

- Rings are numbered from 0 to 7, with outer rings having a subset of the privileges of the inner rings.
- Each file is a memory segment, and each segment description includes an entry that indicates the ring number associated with that segment, as well as read, writes, and executes privileges.
- Each process runs in a ring, according to the *current-ring-number*, a counter associated with each process.
- A process operating in one ring can only access segments associated with higher (farther out) rings, and then only according to the access bits. Processes cannot access segments associated with lower rings.
- Domain switching is achieved by a process in one ring calling upon a process operating in a lower ring, which is controlled by several factors stored with each segment descriptor:
  - An ***access bracket***, defined by integers b1 <= b2.
  - A ***limit*** b3 > b2
  - A ***list of gates,*** identifying the entry points at which the segments may be called.
- If a process operating in ring i calls a segment whose bracket is such that b1 <= i <= b2, then the call succeeds and the process remains in ring i.
- Otherwise a trap to the OS occurs, and is handled as follows:
  - If i < b1, then the call is allowed, because we are transferring to a procedure with fewer privileges. However if any of the parameters being passed are of segments below b1, then they must be copied to an area accessible by the called procedure.
  - If i > b2, then the call is allowed only if i <= b3 and the call is directed to one of the entries on the list of gates.
- Overall this approach is more complex and less efficient than other protection schemes.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: In a ring-protection system, level 0 has the greatest access to objects and level *n* (greater than zero) has fewer access rights. The access rights of a program at a particular level in the ring structure

are considered as a set of capabilities. What is the relationship between the capabilities of a domain at level *j* and a domain at level *i* to an object (for / > *)?

……..……

……..……

……...

5.4 Access Matrix

The model of protection that we have been discussing can be viewed as an ***access matrix,*** in which columns represent different system resources and rows represent different protection domains. Entries within the matrix indicate (see Figure 5.3) what access that domain has to that resource.

domain \ object	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

Figure 5.3: Access matrix.

Domain switching can be easily supported under this model, simply by providing "switch" access to other domains:

domain \ object	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch
D_3		read	execute					
D_4	read write		read write		switch			

Figure 5.4: Access matrix of Figure 5.3 with domain as objects.

The ability to copy rights is denoted by an asterisk, indicating that processes in that domain have the right to copy that access within the same column, i.e. for the same object. There are two important variations:

- If the asterisk is removed from the original access right, then the right is transferred, rather than being copied. This may be termed a transfer right as opposed to a copy right.
- If only the right and not the asterisk is copied, then the access right is added to the new domain, but it may not be propagated further. That is the new domain does not also receive the right to copy the access. This may be termed a limited copy right, as shown in Figure 5.5:

object / domain	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute		

(a)

object / domain	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute	read	

(b)

Figure 5.5: Access matrix with copy rights.

The ***owner*** right adds the privilege of adding new rights or removing existing ones:

object / domain	F_1	F_2	F_3
D_1	owner execute		write
D_2		read* owner	read* owner write
D_3	execute		

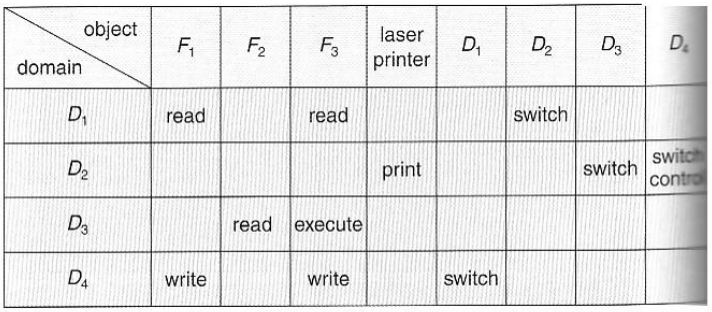
(a)

object / domain	F_1	F_2	F_3
D_1	owner execute		write
D_2		owner read* write*	read* owner write
D_3		write	write

(b)

Figure 5.6: Access matrix with owner rights.

- Copy and owner rights (see Figure 5.6) only allow the modification of rights within a column. The addition of ***control rights***, which only apply to domain objects, allow a process operating in one domain to affect the rights available in other domains. For example in the Figure 5.7, a process operating in domain D2 has the right to control any of the rights in domain D4.



domain \ object	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch control
D_3		read	execute					
D_4	write		write		switch			

Figure 5.7: Modified access matrix of Figure 5.4.

5.5 Implementation of Access Matrix

5.5.1 Global Table

The simplest approach is one big global table with < domain, object, rights > entries. Unfortunately this table is very large (even if sparse) and so cannot be kept in memory (without invoking virtual memory techniques.) There is also no good way to specify groupings - If everyone has access to some resource, then it still needs a separate entry for every domain.

5.5.2 Access Lists for Objects

- Each column of the table can be kept as a list of the access rights for that particular object, discarding blank entries.
- For efficiency a separate list of default access rights can also be kept, and checked first.

5.5.3 Capability Lists for Domains

In a similar fashion, each row of the table can be kept as a list of the capabilities of that domain.

Capability lists are associated with each domain, but not directly accessible by the domain or any user process.

Capability lists are themselves protected resources, distinguished from other data in one of two ways:

- A ***tag,*** possibly hardware implemented, distinguishing this special type of data. (other types may be floats, pointers, Booleans, etc.)
- The address space for a program may be split into multiple segments, at least one of which is inaccessible by the program itself, and used by the operating system for maintaining the process's access right capability list.

5.5.4 A Lock-Key Mechanism

- Each resource has a list of unique bit patterns, termed locks.
- Each domain has its own list of unique bit patterns, termed keys.
- Access is granted if one of the domain's keys fits one of the resource's locks.
- Again, a process is not allowed to modify its own keys.

5.5.5 Comparison

- Each of the methods here has certain advantages or disadvantages, depending on the particular situation and task at hand.

- Many systems employ some combination of the listed methods.

Self Assessment Questions

1. ……..prevents or minimizes unauthorized access and disclosure of data and information.
(a) Integrity (b) Assurance (c) Confidentiality (d) None of these.

2. …… makes sure that the data being worked with is actually the correct data.
(a) Integrity (b) Assurance (c) Confidentiality (d) None of these.

3. ……..is the property of a system or system resource being accessible and usable upon demand by an authorized system entity, according to performance specification for the system. (a) Integrity (b) Assurance (c) Availability (d) None of these

4. Authenticity makes possible that a computer system be able to verify the identity of a user.
(a) True (b) False

5.6 Revocation of Access Rights

In a dynamic protection system, we may sometimes need to revoke access rights to objects shared by different users. Various questions about revocation may arise:

- Immediate versus delayed: Does revocation occur immediately, or is it delayed? If revocation is delayed, can we find out when it will take place?
- Selective versus general: When an access right to an object is revoked, does it affect all the users who have an access right to that object, or can we specify a select group of users whose access rights should be revoked?
- Partial versus total: Can a subset of the rights associated with an object be revoked, or must we revoke all access rights for this object?
- Temporary versus permanent: Can access be revoked permanently (that is, the revoked access right will never again be available), or can access be revoked and later be obtained again?

With an access list scheme revocation is easy, immediate, and can be selective, general, partial, total, temporary, or permanent, as desired.

With capabilities lists the problem is more complicated, because access rights are distributed throughout the system. A few schemes that have been developed include:

- Reacquisition - Capabilities are periodically revoked from each domain, which must then re-acquire them.
- Back-pointers - A list of pointers is maintained from each object to each capability which is held for that object.
- Indirection - Capabilities point to an entry in a global table rather than to the object. Access rights can be revoked by changing or invalidating the table entry, which may affect multiple processes, which must then re-acquire access rights to continue.
- Keys - A unique bit pattern is associated with each capability when created, which can be neither inspected nor modified by the process.
  - A master key is associated with each object.
  - When a capability is created, its key is set to the object's master key.
  - As long as the capability's key matches the object's key, then the capabilities remain valid.

- The object master key can be changed with the set-key command, thereby invalidating all current capabilities.
- More flexibility can be added to this scheme by implementing a **list** of keys for each object, possibly in a global table.

5.6.1 Capability-Based Systems

An Example: Hydra

- Hydra is a capability-based system that includes both system-defined rights and user-defined rights. The interpretation of user-defined rights is up to the specific user programs, but the OS provides support for protecting access to those rights, whatever they may be
- Operations on objects are defined procedurally, and those procedures are themselves protected objects, accessed indirectly through capabilities.
- The names of user-defined procedures must be identified to the protection system if it is to deal with user-defined rights.
- When an object is created, the names of operations defined on that object become ***auxiliary rights,*** described in a capability for an ***instance*** of the type. For a process to act on an object, the capabilities it holds for that object must contain the name of the operation being invoked. This allows access to be controlled on an instance-by-instance and process-by-process basis.
- Hydra also allows ***rights amplification,*** in which a process is deemed to be ***trustworthy,*** and thereby allowed to act on any object corresponding to its parameters.
- Programmers can make direct use of the Hydra protection system, using suitable libraries which are documented in appropriate reference manuals.

5.7 Language-Based Protection

As systems have developed, protection systems have become more powerful, and also more specific and specialized.

To refine protection even further requires putting protection capabilities into the hands of individual programmers, so that protection policies can be implemented on the application level, i.e. to protect resources in ways that are known to the specific applications but not to the more general operating system.

5.7.1 Compiler-Based Enforcement

In a compiler-based approach to protection enforcement, programmers directly specify the protection needed for different resources at the time the resources are declared.

This approach has several advantages:

1. Protection needs are simply declared, as opposed to a complex series of procedure calls.
2. Protection requirements can be stated independently of the support provided by a particular OS.
3. The means of enforcement need not be provided directly by the developer.
4. Declarative notation is natural, because access privileges are closely related to the concept of data types.

Regardless of the means of implementation, compiler-based protection relies upon the underlying protection mechanisms provided by the underlying OS, such as the Cambridge CAP or Hydra systems. Even if the underlying OS does not provide advanced protection mechanisms, the compiler can still offer some protection, such as treating memory accesses differently in code versus data segments. (e.g. code segments cannot be modified, data segments cannot be executed.)

There are several areas in which compiler-based protection can be compared to kernel-enforced protection:

- ***Security*.** Security provided by the kernel offers better protection than that provided by a compiler. The security of the compiler-based enforcement is dependent upon the integrity of the compiler itself, as well as requiring that files not be modified after they are compiled. The kernel is in a better position to protect itself from modification, as well as protecting access to specific files. Where hardware support of individual memory accesses is available, the protection is stronger still.
- ***Flexibility.*** A kernel-based protection system is not as flexible to provide the specific protection needed by an individual programmer, though it may provide support which the programmer may make use of. Compilers are more easily changed and updated when necessary to change the protection services offered or their implementation.
- ***Efficiency.*** The most efficient protection mechanism is one supported by hardware and microcode. Insofar as software based protection is concerned, compiler-based systems have the advantage that many checks can be made off-line, at compile time, rather that during execution.

The concept of incorporating protection mechanisms into programming languages is in its infancy, and still remains to be fully developed. However the general goal is to provide mechanisms for three functions:

- Distributing capabilities safely and efficiently among customer processes. In particular a user process should only be able to access resources for which it was issued capabilities.
- Specifying the ***type*** of operations a process may execute on a resource, such as reading or writing.
- Specifying the ***order*** in which operations are performed on the resource, such as opening before reading.

5.7.2 Protection in Java

Java was designed from the very beginning to operate in a distributed environment, where code would be executed from a variety of trusted and untrusted sources. As a result the Java Virtual Machine, JVM incorporates many protection mechanisms. When a Java program runs, it loads up classes dynamically, in response to requests to instantiates objects of particular types. These classes may come from a variety of different sources, some trusted and some not, which requires that the protection mechanism be implemented at the resolution of individual classes, something not supported by the basic operating system.

When a request is made to access a restricted resource in Java, (e.g. open a local file), some process on the current ***call stack*** must specifically assert a privilege to perform the operation. In essence this method ***assumes responsibility*** for the restricted access. Naturally the method must be part of a class which resides in a protection domain that includes the capability for the requested operation. This approach is termed ***stack inspection,*** and works like this:

- When a caller may not be trusted, a method executes an access request within a do Privileged ( ) block, which is noted on the calling stack.
- When access to a protected resource is requested, check Permissions ( ) inspects the call stack to see if a method has asserted the privilege to access the protected resource.
  - If a suitable do privileged block is encountered on the stack before a domain in which the privilege is disallowed, then the request is granted.
  - If a domain in which the request is disallowed is encountered first, then the access is denied and an Access Control Exception is thrown.
  - If neither is encountered, then the response is implementation dependent.

protection domain:	untrusted applet	URL loader	networking
socket permission:	none	*.lucent.com:80, connect	any

class:	gui:
	...
	get(url);
	open(addr);
	...
		get(URL u):
		...
		doPrivileged {
		open('proxy.lucent.com:80');
		}
		<request u from proxy>
		...
			open(Addr a):
			...
			checkPermission
			(a, connect);
			connect (a);
			...

Figure 5.8: Stack inspection.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Consider a computer system in which "computer games" can be played by students only between 10 P.M. and 6 A.M., by faculty members between 5 P.M. and 8 A.M. and by the computer centre staff at all times. Suggest a scheme for implementing this policy efficiently.

……..……

……..……

……...

5.8 Security

Security, on the other hand, requires not only an adequate protection system, but also consideration of the external environment within which the system operates. Internal protection is not useful if the operator's console is exposed to unauthorized personnel, or if files (stored, for example, on tapes and disks) can simply be removed from the computer system and taken to a system that has no protection. These security problems are essentially management, rather than operating-system, problems.

The information stored in the system (both data and code), as well as the physical resources of the computer system, need to be protected from unauthorized access, malicious destruction or alteration, and accidental introduction of inconsistency. In this chapter, we examine the ways in which information may be misused or intentionally made inconsistent. We then present mechanism to guard against such occurrences.

5.8.1 The Security Problem

We have already discussed various mechanisms that the operating system can provide (with appropriate aid from the hardware) that allow users to protect their resources (usually programs and data). These mechanisms work well as long as the users do not try to circumvent the intended use of and access to these resources. Unfortunately, this situation is seldom realized. When it is not, security comes into play. We say that a system is *secure* if its resources are used and accessed as intended under all circumstances. Unfortunately, it is not generally possible to achieve total security. Nonetheless, mechanisms must be available to make security breaches a rare occurrence, rather than the norm. Security violations (misuse) of the system can be categorized as being either intentional (malicious) or accidental. It is easier to protect against accidental misuse than to protect against malicious misuse. Among the forms of malicious access are the following:

- Unauthorized reading of data (theft of information)
- Unauthorized modification of data
- Unauthorized destruction of data

Absolute protection of the system from malicious abuse is not possible, but the cost to the perpetrator can be made sufficiently high to deter most, if not all, attempts to access, without proper authority, the information residing in the system.

To protect the system, we must take security measures at two levels:
- Physical: The site or sites containing the computer systems must be physically secured against armed or surreptitious entry by intruders.
- Human: Users must be screened carefully so that the chance of authorizing a user who then gives access to an intruder (in exchange for a bribe, for example) is reduced.

Did You Know?
The first system that had a login command and requested a **password** was introduced in 1961 in the Massachusetts Institute of Technology.

5.9 Authentication

A major security problem for operating systems is the authentication problem. The protection system depends on an ability to identify the programs and processes that are executing. This ability, in turn, eventually rests on our power to identify each user of the system. A user normally identifies him. How do we determine whether a user's identity is authentic? Generally, authentication is based on one or more of three items: user possession (a key or card), user knowledge (a user identifier and password), and a user attribute (fingerprint, retina pattern, or signature).

5.9.1 Passwords

The most common approach to authenticating a user identity is the use of user passwords. When the user identifies herself by user ID or account name, she is asked for a password. If the user-supplied password matches the password stored in the system, the system assumes that the user is legitimate.
Passwords are often used to protect objects in the computer system, in the absence of more complete protection schemes. They can be considered a special case of either keys or capabilities. For instance, a password could be associated with each resource (such as a file). Whenever a request is made to use the resource, the password must be given. If the password is correct, access is granted. Different password may be associated with different access rights.
For example, different passwords may be used for each of reading, appending, and updating *a* file.

5.9.2 Password Vulnerabilities

Although there are problems associated with their use, passwords are nevertheless extremely common, because they are easy to understand and use. The problems with passwords are related to the difficulty of keeping a password secret. Passwords can be compromised by being guessed, accidentally exposed, or illegally transferred from an authorized user to an unauthorized one, as we show next.

There are two common ways to guess a password. One is for the intruder (either human or program) to know the user or to have information about the user. All too frequently, people use obvious

information (such as the names of their cats or spouses) as their passwords. The other way is to use brute force; trying all possible combinations of letters, numbers, and punctuation until the password is found. Short passwords do not leave enough choices to. Prevent their being guessed by repeated trials. For example, a four-decimal password provides only 10,000 variations. On the average, guessing 5000 times would produce a correct hit. If a program could be written that would try a password every 1 millisecond, it would then take only about 5 seconds to guess a four-digit password. Longer passwords are less susceptible to being guessed by enumeration and systems that differentiate between uppercase and lowercase letters and that allow use of numbers and all punctuation characters in passwords, make the task of guessing the password much more difficult. Of course, users must take advantage of the large password space and must not, for example, use only lowercase letters.

5.9.3 Encrypted Passwords

One problem with all these approaches is the difficulty of keeping the password secret. The UNIX system uses encryption to avoid the necessity of keeping its password list secret. Each user has a password. The system contains a function that is extremely difficult (the designers hope impossible) to invert, but is simple to compute. That is, given a value *x*, it is easy to compute the function value *f(x)*. Given a function value *f(x)*, however, it is impossible to compute *x*.

This function is used to encode all passwords. Only the encoded passwords are stored. When a user presents a password, it is encoded and compared against the stored encoded password. Even if the stored encoded password is seen, it cannot be decoded, so the password cannot be determined. Thus, the password file does not need to be kept secret. The function *f(j)* is typically an *encryption* algorithm that has been designed and tested rigorously.

The flaw in this method is that the system no longer has control over the passwords. Although the passwords are encrypted, anyone with a copy of the password file can run fast encryption routines against it, encrypting each word in a dictionary, for instance, and comparing the results against file passwords.

Caution

Security at both levels must be maintained if operating-system security is to be ensured. A weakness at a high level of security (physical or human) allows circumvention of strict low-level (operating-system) security measures. Ignoring any level can occur a problem in system.

5.10 One-Time Passwords

To avoid the problems of password sniffing and shoulder surfing, a system could use a set of paired passwords. When a session begins, the system randomly selects and presents one part of a password pair; the user must supply the other part. In this system, the user is challenged and must respond with the correct answer to that challenge.

This approach can be generalized to the use of an algorithm as a password. The algorithm might be an integer function, for example. The system selects a random integer and presents it to the user. The user applies the function and replies with the correct result. The system also applies the function. If the two results match, access is allowed.

Suppose that there were a password method that was not susceptible to exposure. For example, a user could type in a password, and any entity intercepting that password and trying to reuse it would fail. Such a system does exist; it involves the use of algorithmic passwords. In this variation, system and

the user share a secret. The secret is never transmitted over a medium that allows exposure. Rather, the secret is used as input to the function, along with a shared seed. A seed is a random number or alphanumeric sequence.

The seed is the authentication challenge from the computer. The secret and the seed are used as input to the function *f(secret, seed).* The result of this function is transmitted as the password to the computer. Because the computer also knows the secret and the seed, it can perform the same computation. If the results match, the user is authenticated. The next time that the user needs to be authenticated, another seed is generated and the same steps ensue. This time, the password is different.

5.11 Program Threats

In an environment where a program written by one user may be used by, another user, there is an opportunity for misuse, which may result in unexpected behaviour. Now we will describe two common methods by which such behaviour may occur: Trojan horses and trap doors.

5.11.1 Trojan Horse

Many systems have mechanisms for allowing programs written by users to be executed by other users. If these programs are executed in a domain that provides the access rights of the executing user, they may misuse these rights.

Inside a text-editor program, for example, there may be code to search the file to be edited for certain keywords. If any are found, the entire file may be copied to a special area accessible to the creator of the text editor. A code segment that misuses its environment is called a Trojan horse. The Trojan-horse problem is exacerbated by long search paths (such as are common on UNIX systems). The search path lists the set of directories to search when an ambiguous program name is given. The path is searched for a file of that name and the file is executed. All the directories in the search path must be secure, or a Trojan horse could be slipped into the user's path and executed accidentally.
For instance, consider the use of the "." character in a search path. The "." tells the shell to include the current directory in the search. Thus, if a user has "." in her search path, has set her current directory to a friend's directory, and enters the name of a normal system command, the command may be executed from the friend's directory instead. The program would run within the user's domain, allowing including deleting the user's files, for instance.

5.11.2 Trap Door

The designer of a program or system might leave a hole in the software that only she is capable of using. This type of security breach was shown in the movie "War Games." For instance, the code might check for a specific user identifier or password, and might circumvent normal security procedures. There have been cases of programmers being arrested for embezzling from banks by including rounding errors in their code, and having the occasional half-cent credited to their accounts. This account crediting can add up to a large amount of money, considering the number of transactions that a large bank executes!
A clever trap door could be included in a compiler. The compiler could generate standard object code as well as a trap door, regardless of the source code being compiled. This activity is particularly nefarious, since a search of the source code of the program will not reveal any problems. Only the source code of the compiler would contain the information. Trap doors pose a difficult problem

because, to detect them, we have to analyze all the source code for all components of a system. Given that software systems may consist of millions of lines of code, this analysis is not done frequently.

5.12 System Threats

Most operating systems provide a means for processes to spawn other processes. In such an environment, it is possible to create a situation where operating-system resources and user files are misused. The two most common methods for achieving this misuse are worms and viruses.

5.12.1 Worms

A worm is a process that uses the *spawn* mechanism to clobber system performance.
The worm spawns copies of itself, using up system resources and perhaps locking out system use by all other processes. On computer networks, worms are particularly potent, since they may reproduce themselves among systems and thus shutdown the entire network. Such an event occurred in 1988 to UNIX systems on the worldwide Internet network, causing millions of dollars of lost system and programmer time. The Internet links thousands of government, academic, research, and industrial computers internationally, and serves as the infrastructure for electronic exchange of scientific information. At the close of the workday on November 2, 1988, Robert Tappan Morris, Jr., a first-year Cornell graduate student, unleashed a worm program on one or more hosts connected to the Internet. Targeting Sun Microsystems' Sun 3 workstations and VAX computers "running variants of Version 4 BSD UNIX, the worm quickly spread over great distances; within a few hours of its release, it had consumed system resources to the point of bringing down the infected machines.
Although Robert Morris designed the self-replicating program, for rapid reproduction and distribution, some of the features of the UNIX networking environment provided the means to propagate the worm throughout the system.

It is likely that Morris chose for initial infection an Internet host left open; for and accessible to outside users. From there, the worm program exploited flaws in the UNIX operating system's security routines and took advantage of UNIX utilities that simplify resource sharing in local-area networks to gain unauthorized access to thousands of other connected sites. Morris' methods of attack are outlined next.

5.12.2 Viruses

Another form of computer attack is a virus. Like worms, viruses are designed to spread into other programs and can wreak havoc in a system, including modifying or destroying files and causing system crashes and program malfunctions.
Whereas a worm is structured as a complete, standalone program, a virus is a fragment of code embedded in a legitimate program. Viruses are a major problem for computer users, especially users of microcomputer systems.
Multiuser computers, generally, are not prone to viruses because the executable programs are protected from writing by the operating system. Even if a virus does infect a program, its powers are limited because other aspects of the system are protected. Single-user systems have no such protections and, as a result, a virus has free run. —
Viruses are usually spread by users downloading viral programs from public bulletin boards or exchanging floppy disks containing an infection. A case from February 1992 involving two Cornell University students provides an illustration. The students had developed three Macintosh game programs with an embedded virus that they distributed to worldwide software archives via the

Internet. The virus was discovered when a mathematics professor in Wales downloaded the games, and antivirus programs on his system alerted him to an infection. Some 200 other users had also downloaded the games. Although the virus was not designed to destroy data, it could spread to application files and cause such problems as long delays and program malfunctions.

Exercise: Check Your Progress 3

Note: i) Use the space below for your answer.

Ex1: Discuss a means by which managers of systems connected to the Internet could have designed their systems to limit or eliminate the damage done by the worm. What are the drawbacks of making the change that you suggest?

……..……

……..……

……...

Caution

Be aware while using Internet, mainly viruses enters through accessing Internet. Some viruses can harm the software as well as hardware.

5.13 Threat Monitoring

The security of a system can be improved by two management techniques. One is threat monitoring: The system can check for suspicious patterns of activity in an attempt to detect a security violation. A common example of this scheme is a time-sharing system that counts the number of incorrect passwords given when a user is trying to log in. More than a few incorrect attempts may signal an attempt to guess a password.

Another common technique is an audit log. An audit log simply records the time, user, and type of all accesses to an object. After security has been violated, the audit log can be used to determine how and when the problem occurred and perhaps what amount of damage was done. This information can be useful, both for recovery from the violation and, possibly, in the development of tighter security measures to prevent future problems. Unfortunately, logs can become large, and logging uses system resources that are then unavailable to the users. Rather than log system activities, we can scan the system periodically for security holes. These scans can be done when the computer has relatively little traffic, so they will have less effect than logging.

The problem is how can trusted computers be connected safely to an untrustworthy network? One solution is the use of a firewall to separate trusted and untrusted systems. A firewall is a computer or router that sits between the trusted and the untrusted. It limits network access between the two security domains, and monitors and logs all connections. For instance, web servers use the http protocol to communicate with web browsers. A firewall therefore may need to allow http to pass. The Morris Internet worm used the finger protocol to break into computers, so finger would not be allowed to pass. In fact, a firewall can separate a network into multiple domains. A common implementation has the Internet as the untrusted domain; a semi trusted and semi secure network, called the demilitarized zone (DMZ), as another domain; and a company's computers as a third domain (see Figure 5.10). Connections are allowed from the Internet to the DMZ computers and from the company computers to the Internet, but are not allowed from the Internet or DMZ computers to the company computers.

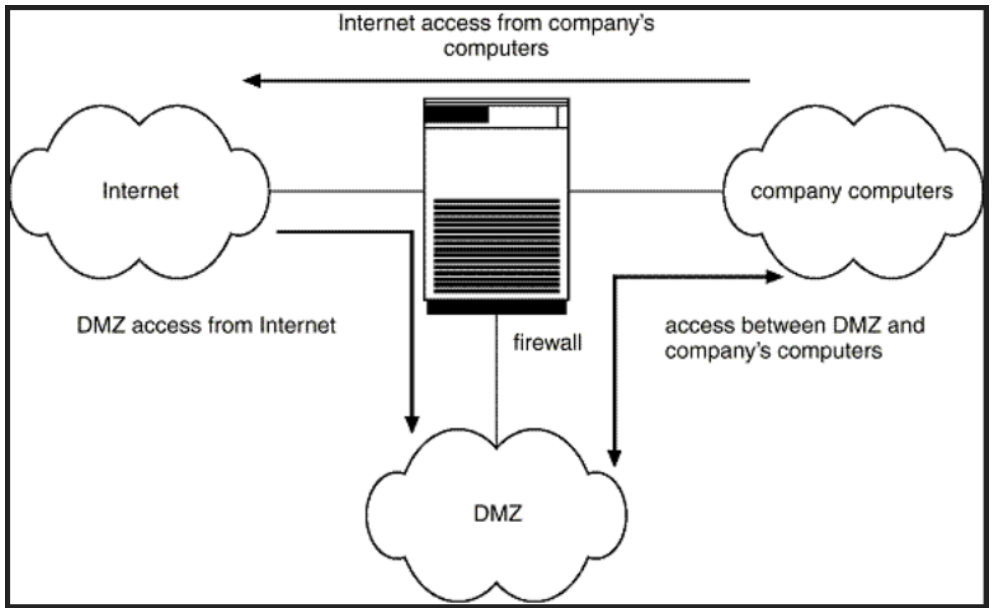


Figure 5.10: Network security through domain separation via firewall.

Optionally, controlled communications may be allowed between the DMZ and one or more company computers. For instance, a web server on the DMZ may need to query a database server on the corporate network. In this manner, all access is contained, and any DMZ systems that are broken into based on the protocols allowed through the firewall still are unable to access the company computers.

5.14 Encryption

The various provisions that an operating system may make for authorization may not offer sufficient protection for highly sensitive data. Moreover, as computer networks gain popularity, more sensitive (classified) information is being transmitted over channels where eavesdropping and message interception are possible. To keep such sensitive information secure, we need mechanisms to allow a user to protect data that are transferred over the network.

Encryption is one common method of protecting information transmitted over unreliable links. The basic mechanism works as follows:

1. The information (text) is *encrypted* (encoded) from its initial readable form (called *clear text),* to an internal form (called *cipher text).* This internal text form, although readable, does not make any sense.
2. The cipher text can be stored in a readable file, or transmitted over unprotected channels.
3. To make sense of the cipher text, the receiver must *decrypt* (decode) it back into clear text.

Did You Know?

The earliest known use of cryptography is found in non-standard hieroglyphs carved into monuments from the Old Kingdom of Egypt circa 1900 BC is known as classical encryption.

Self Assessment Questions

5. A protection domain specifies the resources that a process may access.

(a) True (b) False

6. The ……..system uses a complex system of rings, each corresponding to a different protection domain.
(a) MULTICS (b) UNIX (c) OS/2 (d) None of these.

7. Security provided by the kernel offers better protection than that provided by a compiler.
(a) True (b) False

8. A major security problem for operating systems is the authentication problem.
(a) True (b) False

Exercise: Check Your Progress 4

Case Study-Managing Information Resources: Ensuring Data Security

This case is about a new threat to data security in organizations called pod slurping. Pod slurping refers to the stealing of data from the organizations' network with portable data storage devices like ipods, digital cameras, pen drives and smart phones. Addressing this threat posed tough challenges to organizations as it is the employees of the organization who could be stealing the data. While data security experts were grappling with measures to discourage data theft by employees using such portable devices, the organizations were on the horns of a dilemma. Banning popular gadgets such as iPods could alienate the employees, and in many cases it was practically impossible to put a blanket ban on portable storage devices as they were not only ubiquitous but also very useful.

Experts felt that organizations had to be balanced in their efforts in addressing this problem as too much monitoring or high-handedness on the organization's part might hurt the morale of the employees, breed disloyalty, and lower productivity. Moreover, this threat was expected to intensify in the future as these portable devices were getting more sophisticated and acquiring wireless capabilities.

The Problem

Data theft, one of the major security issues facing the companies, could lead to heavy financial and economic loss to the organizations, according to experts.

Apart from the more prevalent forms of data theft like online hacking of the organization's networks and stealing of hard copies of the files of the organization, the companies were now waking up to yet another 'physical' way of data theft with innocent looking IT gadgets like iPods, digital cameras, MP3 players, and smart phones. The expenses incurred on preventing the theft of data were slowly taking a major part of the IT budget for many organizations.

Motives for Slurping

Employees could easily use their advantage of access to the internal resources of the company to steal vital data. An employee could engage in pod slurping for various reasons, according to experts. The main motives for stealing the data were: monetary gain, malicious intent, and curiosity. Corporate data was profitable in many ways; patented formulae, tender details, plans for future expansion were some bits of information which could be very useful to competitors and fetch a good sum of money in the market. Dishonest employees could steal data and sell it to their competitors.

A New HR Dilemma

The possible threats posed by these portable devices were also becoming a HR dilemma for the organizations. Insider information was the biggest threat to the safety of the data resources of the organization.

Corporate insiders could easily evade the ring of security.
Statistics showed that internal security breaches were growing faster than the external security breaches; and it constituted almost half of the total security breaches in the organizations.

A Serious Threat or Just Hype?
While the debate continued on the dangers of pod slurping, there were some who argued that it was more of hype than a real threat. Though there was every possibility of the data getting stolen and used for malicious purposes, some experts opined that the threat had been over-publicized in the media. Vladimir Chrenavsky (Chrenavsky), CEO of Advanced Force, a maker of security software, remarked that the threat was "overblown".

Questions
1. What is the pod slurping?
2. How the data security can improve in an organization?

5.15 Summary

- Computer systems contain many objects. These objects need to be protected from misuse. Objects may be hardware (such as memory CPU time, or I/O devices) or software (such as files, programs, and abstract data types).
- The access matrix is sparse. It is normally implemented either as access lists associated with each object, or as capability lists associated with each domain.
- Real systems are much more limited, and tend to provide protection for only files.
- UNIX is representative, providing read, write, and execution protection separately for the owner, group, and general public for each file.
- Protection is an internal problem. Security must consider both the computer system and the environment (people, buildings, businesses, valuable objects, and threats) within which the system is used.
- The data stored in the computer system must be protected from unauthorized access, malicious destruction or alteration, and accidental introduction of inconsistency.
- The various authorization provisions in a computer system may not confer sufficient protection for highly sensitive data.

5.16 Keywords

Access Matrix: The access matrix is a general model of protection. The access matrix provides a mechanism for protection without imposing a particular protection policy on the system or its users.
Access Right: An access right is permission to perform an operation on an object.
Domain: A domain is a set of access rights. Processes execute in domains and may use any of the access rights in the domain to access and manipulate objects.
Encryption: Encryption is the conversion of data into a form, called a ciphertext that cannot be easily understood by unauthorized people.
Virus: A virus is a fragment of code embedded in a legitimate program. Viruses are a major problem for computer users, especially users of microcomputer systems.
Worm: A worm is a process that uses the spawn mechanism to clobber system performance. The worm spawns copies of itself, using up system resources and perhaps locking out system use by all other processes.

5.17 Review Questions

1. What are the goals and principles of protection?
2. What are the protection mechanism and OS mode protection?
3. What are the main differences between capability lists and access lists?
4. Explain the domain of the protection.
5. What is the access matrix? How it is implemented?
6. What are the revocations of access rights?
7. What is the language based protection? Explain with example.
8. What is security? Explain the security problem. What is authentication?
9. Write the short notes:
   a. One time password
   b. Program threats
   c. System threats
   d. Threat monitoring
10. What are two advantages of encrypting data stored in the computer system?

Answers for Self Assessment Questions

1. (c)	2. (a)	3. (c)	4. (a)
5. (a)	6. (a)	7. (a)	8. (a)

5.18 Further Studies

1. https://www.cs.columbia.edu/~smb/classes/s06-4118/l25.pdf
2. Operating System Principles by Silberschatz, Galvin and Gagne

6

Memory Management

STRUCTURE

6.0 Objectives

6.1 Introduction

6.2 Concept of Memory Management

6.3 Memory Management Requirements

6.4 Address Space

6.5 Linking and Loading

6.6 Swapping

6.7 Memory Partitioning

6.8 Paging

6.9 Segmentation

6.10 Summary

6.11 Keywords

6.12 Review Questions

6.13 Further Studies

6.0 Objectives

After studying this chapter, you will be able to:

- Understand memory management
- Explain the memory management requirements
- Describes the address space
- Explain the linking and loading
- Define swapping
- Describes the memory partitioning
- Understand the paging
- Explain the segmentation

6.1 Introduction

Memory management is the act of managing computer memory. In its simpler forms, this involves providing ways to allocate portions of memory to programs at their request, and freeing it for reuse when no longer needed. The management of main memory is critical to the computer system.
Virtual memory systems separate the memory addresses used by a process from actual physical addresses, allowing separation of processes and increasing the effectively available amount of RAM using disk swapping. The quality of the virtual memory manager can have a big impact on overall system performance.
Garbage collection is the automated allocation and deal-location of computer memory resources for a program. This is generally implemented at the programming language level and is in opposition to manual memory management, the explicit allocation and deal-location of computer memory resources. Region-based memory management is an efficient variant of explicit memory management that can deal-locate large groups of objects simultaneously.

6.2 Concept of Memory Management

The main memory is usually located on chips inside the system unit. There is having two kinds of memory:

1. Random-access memory (RAM) and
2. Read-only memory (ROM).

The instructions that the computer gets and the data the processes are kept in RAM during computer works. RAM is not a permanent storage place for information. It is active only when computer is on. If computer is switched off, the information is deleted from RAM. ROM is the memory where information is only read. When the computer is turned a off, information is not deleted from ROM. Writing information to ROM is made by vendor. The size of the main memory is measured in megabytes.
The external memory is disk. Unlike information stored in RAM, information stored on a disk is not deleted when the computer turned off. Information stored on the disks is moved in is and out of RAM. The amount of space on a disk is measured in give bytes.

There are two kinds of disks:
Hard disk and floppy disk.
The main memory and the floppy disk have less storage capacity than the hard disk. The hard disk can write and read information to and from the main memory much faster than a floppy disk. The access speed of main memory is also much faster than a hard disk.
Disk-based systems provide the primary on-line storage of information, both program and data. Most programs, like assemblers, sort routines, editors, formatters, and so on, are stored on a disk until loaded into memory, and then use the disk as both the source and destination of the information for their processing. Hence, the proper management of disk storage is of central importance to a computer system.
In a larger sense, however, disks systems are only one of many possible storage systems registers, caches, main memory, disks, and tapes and so on. Each storage system provides the basic functions of storing a piece of data, holding the data until it is retrieved at a later time. The main differences between the various storage systems are their speed, cost, size, and volatility. We can see a wide variety of storage in a computer system, which can be organized in a hierarchy (see Figure 6.1) According to their speed or their cost. The higher levels are expensive, but very fast. As we move down the hierarchy, the cost per bit decreases, while the access time increases and the amount of storage at each level increases. This is reasonable, since if a given storage system was both fatter and

cheaper than another, with other properties being the same, there would be no reason to use the slower, more expensive memory.

The design of a computer memory system attempts to balance these factors: using only as much expensive memory as absolutely necessary while trying to provide as much cheap memory as possible. Demand paging is an example of such a system design: a limited amount of expensive, fast main memory is used to provide fast access to a much larger virtual memory which is actually stored on cheaper, slower secondary memory. Main memory can be viewed as a fast cache for secondary memory.

Caching is an important principle of computer systems, both in hardware and software.

Information is normally stored in some storage system (like main memory). As it is used, it is copied into a faster storage system (the cache) on a temporary basis. When a particular piece of information is needed, we first check if it is in the cache. If it is, we use the information directly from the cache. If not, we use the information from the main storage system, putting a copy in the cache in the hope that it will be needed again.

Since caches have limited size, cache management is an important design problem. Careful selection of cache size and a replacement policy can mean that 80 to 90% of all accesses are in the cache, resulting in very high performance.

Extending this view, internal programmable registers, such as index registers and accumulators, are a high-speed cache for main memory. The programmer (or compiler) implements the register allocation and replacement algorithms to decide what information to keep in registers, and what to keep in main memory. The movement of information between levels of a storage hierarchy may be either explicit or implicit.

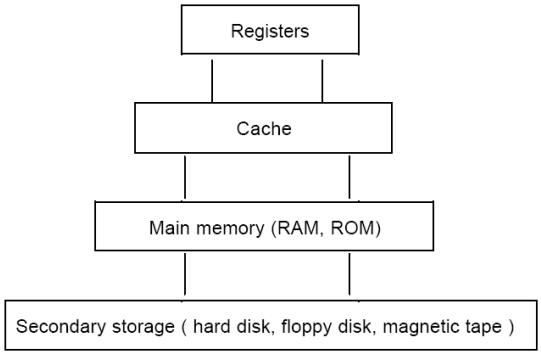


Figure 6.1: Storage hierarchy.

A program's machine language code must be in the computer's main memory in order to execute. Assuring that at least the portion of code to be executed is in memory when a processor is assigned to a process is the job of the memory manager of the operating system. This task is complicated by two other aspects as modern computing systems.

The first is multiprogramming. From its definition, we know that multiprogramming mean that several (at least two) processes can be active within the system during any particular time interval. But these multiple active processes result from various jobs entering and leaving the system in an unpredictable manner. Pieces, or blocks, of memory are allocated to these processes when they enter the system, and are subsequently freed when the process leaves the system. Therefore, at any given moment, the computer's memory, viewed as a whole, consists of a part of blocks, some allocated to processes active at that moment, and others free and available to a new process which may, at any time, enter the system.

In general, then, programs designed to execute in this multiprogramming environment must be compiled so that they can execute from any block of storage available at the time of the program's execution. Such program is called reloadable programs, and the idea of placing them into any currently available block of storage is called relocation.

The second aspect of modern computing systems affecting memory management is the need to allow the programmer to use a range of program addresses, which may be larger, perhaps significantly larger than the range of memory locations actually available. That is, we want to provide the programmer with a virtual memory, with characteristics (especially size) different from actual memory, and provide it in a way that is invisible to the programmer. This is accomplished by extending the actual memory with secondary memory such as disk. Providing an efficiently operating virtual memory is another task for the memory management facility.

Physical memory is divided into fixed size blocks called frames. Logical memory is also divided into blocks of the same, fixed size called pages. When a program is to be executed, its pages are loaded into any available memory frames from the disk. The disk is also divided into fixed size, which is the same size as the memory frames.

The logical address space is the set of addresses seen by the user (or user process)

The physical address space is the set of addresses in physical memory (RAM). The two address spaces need not be the same size and usually are not in most modern systems. In systems with "virtual memory", the logical address space is typically much larger than the physical address space. (see Figure 6.2)

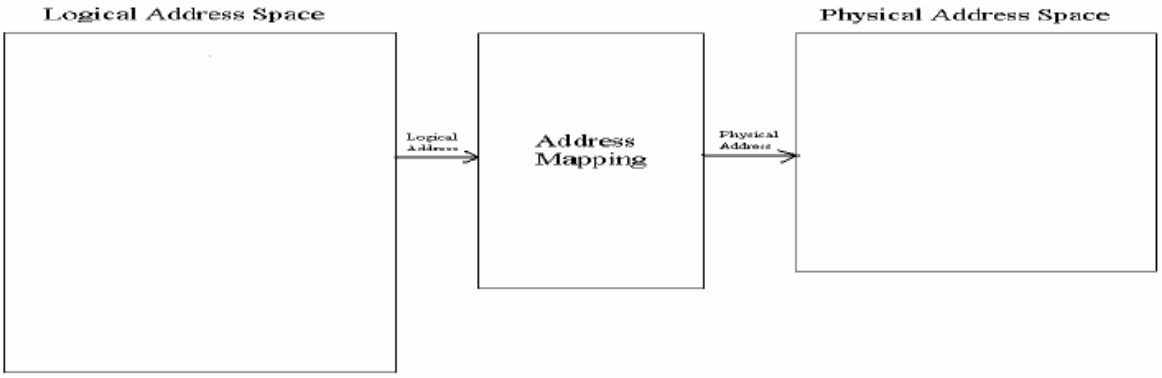


Figure 6.2: Logical and physical address space.

Caution

The code and data for a process must be in RAM before it could be run.

6.3 Memory Management Requirements

Memory management systems on multi-tasking operating systems usually deal with the following issues:

6.3.1 Relocation

- In systems with virtual memory, programs in memory must be able to reside in different parts of the memory at different times. This is because when the program is swapped back into memory after being swapped out for a while it cannot always be placed in the same location. The virtual memory management unit must also deal with concurrency. Memory management in the operating system should therefore be able to relocate programs in memory and handle memory references

and addresses in the code of the program so that they always point to the right location in memory.

6.3.2 Protection and Sharing

Processes should not be able to reference the memory for another process without permission. This is called memory protection, and prevents malicious or malfunctioning code in one program from interfering with the operation of other running programs.

Even though the memory for different processes is normally protected from each other, different processes sometimes need to be able to share information and therefore access the same part of memory. Shared memory is one of the fastest techniques for Inter-process communication.

6.3.3 Logical organization

Programs are often organized in modules. Some of these modules could be shared between different programs, some are read only and some contain data that can be modified. The memory management is responsible for handling this logical organization that is different from the physical linear address space. One way to arrange this organization is segmentation.

6.3.4 Physical Organization

Memory is usually classed by access rate as with primary storage and secondary storage. Memory management systems handle moving information between these two levels of memory.

6.3.5 Two-Level Memory Organization

We have discussed memory hierarchy which includes both fast expensive memories and slow but cheap storage devices. Typically two levels of storage are present, main memory and disks. The former is fast, but usually cannot provide enough space for concurrent processes.

In this case, hard disks are used for the operating system to swap out inactive processes, which may later be swapped back into the main memory.

Self Assessment Question

1. There is having two kinds of memory random-access memory (RAM) and read-only memory (ROM).

(a). True (b). False.

2. .....is not a permanent storage place for information

(a). RAM (b). ROM

(c). DRAM (d). None of these

3. The main memory and the floppy disk have less storage capacity than the.....

(a). Floppy disk (b). hard disk

(c). Both (a) and (b) (d). None of these

4. The access speed of .....is also much faster than a hard disk.

(a). main memory (b). read only memory

(c). random access memory (d). None of these

6.4 Address Space

All 32-bit applications have a 4-gigabyte (GB) process address space (32-bit addresses can map a maximum of 4 GB of memory). Microsoft Windows operating systems provide applications with access to 2 GB of process address space, specifically known as user mode virtual address space. All threads owned by an application share the same user mode virtual address space. The remaining 2 GB are reserved for the operating system (also known as kernel mode address space). All operating system editions starting with Windows 2000 Server, including Windows Server 2003, have a boot.ini switch that can provide applications with access to 3 GB of process address space, limiting the kernel mode address space to 1 GB.

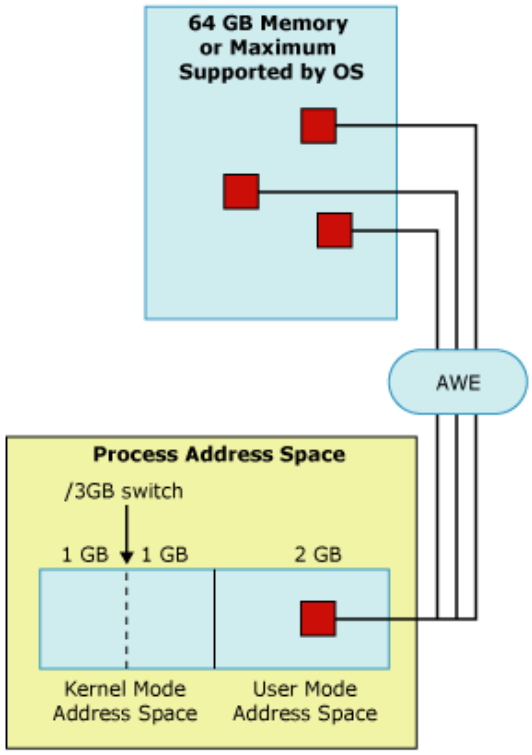


Figure 6.3: Storage areas in an address space.

This feature will be removed in the next version of Microsoft SQL Server. Do not use this feature in new development work, and modify applications that currently use this feature as soon as possible.

Did You Know?

If there is more than 16 GB of physical memory available on a computer, the operating system needs 2 GB of process address space for system purposes and therefore can support only a 2 GB user mode address space. In order for AWE to use the memory range above 16 GB, be sure that the /3 GB parameter is not in the boot.ini file. If it is, the operating system cannot address any memory above 16 GB.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Recruitment of Memory Management?

……..……

……..……

……...

Ex2: Identify Address Spacing.

……..……

……..……

……...

6.5 Linking and Loading

The loading is performed in the following ways:

6.5.1 Absolute Loading

An absolute loader copies a load module into memory. It makes no adjustments to the logical addresses assigned by the translator. It assumes that the translator or the programmer assigned the correct absolute addresses. With this approach, a load module must always occupy the same set of memory locations. This approach is virtually never used.

6.5.2 Re-locatable Loading

In this loading, the ***re-locatable loader*** adds an offset to each logical address in the program as it copies a load module into memory. This type of loader requires the translator to generate relative addresses. It is also necessary for the program to be loaded into consecutive memory locations. The physical addresses are found by adding M to each relative address. M is the start address of the load module. In this case, addresses are bound at load time. It is more flexible approach than compile time binding. The program can be loaded into a separate set of locations each time it runs.

6.5.3 Dynamic Loading (Dynamic Relocation)

Re-locatable loading does not support swapping. Once a process has been loaded and relocated, its addresses are absolute. A swapped-out process could only be swapped back to the memory locations it started in. This is too restrictive. Dynamic loading or dynamic relocation defers the process of determining absolute addresses until the address is used. The loader places the load module into memory without adjusting relative addresses. A special register is loaded with the start address of the load module. Hardware automatically adds the register contents to a relative address each time the address is used. In this case, address binding is at run-time. Relocation is done each time the address is used. It must be handled in hardware.

6.5.4 Linking

A ***linker*** joins several object modules into a single load module. Input to the linker is a set of object modules. Each module has been translated with relative addresses, relative to 0 as the start address of the module.

References from one module to another (function calls, data references, etc.) are still symbolic:
A dynamic linker postpones some of the linkage functions until run time. Dynamic linking can be done at load-time or at run time. Load-time dynamic linking prepares a reloadable load module in normal way. It leaves some external references unresolved. These references are usually to system utilities or language libraries. At load time, system copies of the target modules can be linked. Utilities can be changed without forcing programmers to re-link existing load modules. A shared code can be linked to more than one program. Run-time dynamic linking goes a step farther. Some modules will not be linked in until they are actually called.

Caution
The loader must generate a single contiguous module in which all the external references have been resolved.

6.5.5. Overlaying

Overlaying is a technique that is used to execute a process even if the memory is not enough. The programmer can define two or more overlays. These overlays can execute in the memory independently. The operating system can swap overlays and manage the memory. The disadvantage of overlying is that it requires extensive involvement of programmer. The programme has to identify and define overlays efficiently. It is a very difficult task.
Following the creation of a high level language (HLL) source program, there are processing before we can get a process as shown in Figure 6.4.

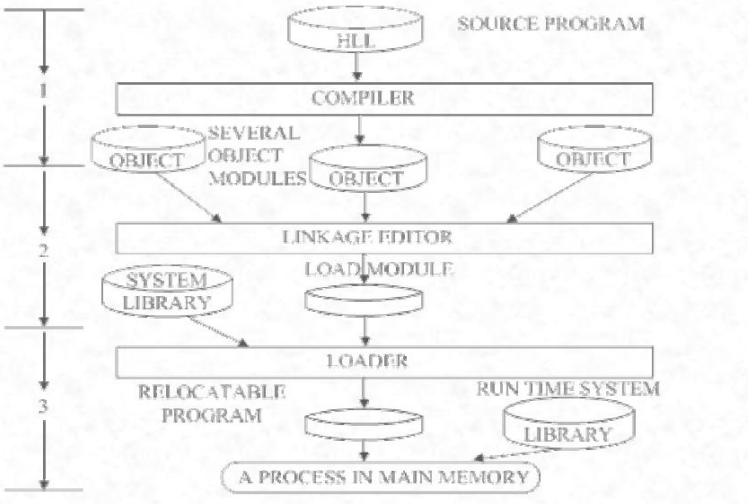


Figure 6.4: A process in main memory.

6.6 Swapping

Swapping is exchanging data between the hard disk and the RAM
The goal of the virtual memory technique is to make an application think that it has more memory than actually exists. If you read the recommended question then you know that the virtual memory manager (VMM) creates a file on the hard disk called a swap file. Basically, the swap file (also known as a paging file) allows the application to store any extra data that can't be stored in the RAM – because the RAM has limited memory. Keep in mind that an application program can only use the data when it's actually in the RAM. Data can be stored in the paging file on the hard disk, but it is not usable until that data is brought into the RAM. Together, the data being stored on the hard disk combined with the data being stored in the RAM comprise the entire data set needed by the application program. So, the way virtual memory works is that whenever a piece of data needed by an application program cannot be found in the RAM, then the program knows that the data must be in the paging file on the hard disk.

The Purpose of Swapping
So, we can say that the purpose of swapping, or paging, is to access data being stored in hard disk and to bring it into the RAM so that it can be used by the application program. Remember that swapping is only necessary when that data is not already in the RAM.

Excessive Swapping Causes Thrashing
Excessive use of swapping is called thrashing and is undesirable because it lowers overall system performance, mainly because hard drives are far slower than RAM.

Did You Know?
1801 First memory device invented by Joseph-Marie Jaquard, in the form of perforated cards for a programmable loom.

Exercise: Check Your Progress 2
Note: i) Use the space below for your answer.
Ex1: Show the working process of swapping with help of block diagram.

……..……

……..……

……...

6.7 Memory Partitioning

Allocating a single contiguous section of memory to each process is the most primitive method of memory management, usually called partitioning. It has been used in some now-obsolete operating systems. Each single section is called a partition.

6.7.1 Fixed Partitioning

The simplest partitioning method is dividing memory into several fixed-sized partitions in advance, called fixed partitioning. Each partition may contain exactly one process. As Figure 6.5 (a) shows, a 64M memory is divided into 8 equal-size partitions with 8 megabytes each. Any process whose size is less than or equal to the partition size can be loaded into any partition available. And when no partition is available and a new process is to be loaded, a process already residing in main memory may be selected to be swapped out to free a partition. So the placement algorithm for this method is very simple.

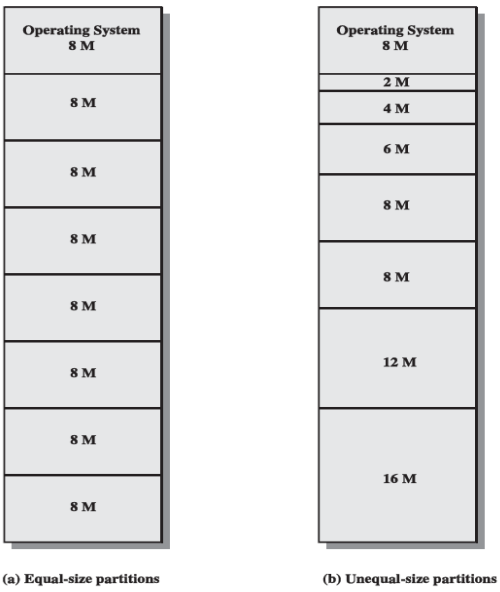


Figure 6.5: Example of fixed partitioning of a 64-Megabyte memory.

However two disadvantages are present with it:

- A program that is too big to be held in a partition needs some special design, called overlay, which brings heavy burden on programmers. With overlay, a process consists of several portions with each being mapped to the same location of the partition, and at any time, only one portion may reside in the partition. When another portion is referenced, the current portion will be switched out.
- A program may be much smaller than a partition, thus space left in the partition will be wasted, which is referred to as internal fragmentation.

6.7.2 Dynamic Partitioning

To overcome difficulties with fixed partitioning, partitioning may be done dynamically, called dynamic partitioning. With it, the main memory portion for user applications is initially a single contiguous block. When a new process is created, the exact amount of memory space is allocated to the process. Similarly when no enough space is available, a process may be swapped out temporarily to release space for a new process. The way how the dynamic partitioning works is illustrated in Figure 6.6.

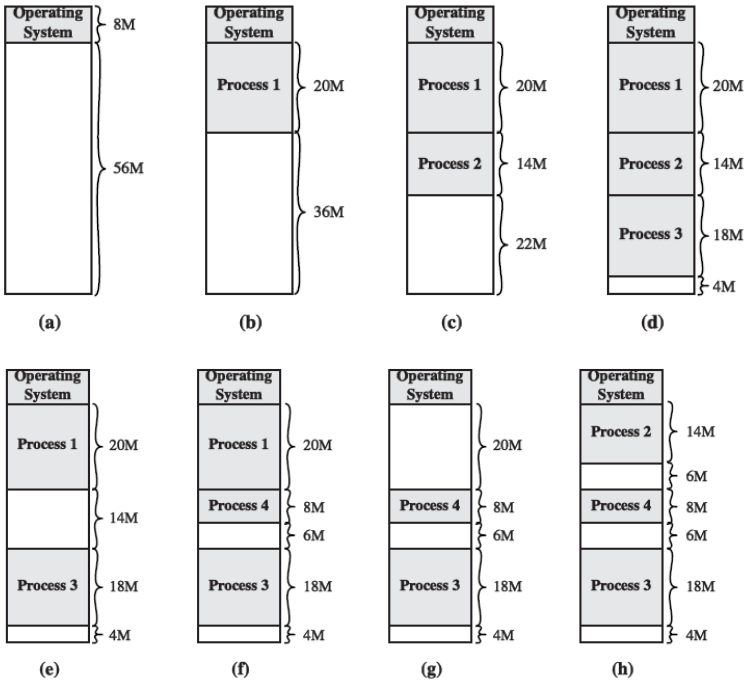


Figure 6.6: The effect of dynamic partitioning.

6.7.3 Address Translation

Figure 6.7 shows the address translation procedure with dynamic partitioning, where the processor provides hardware support for address translation, protection, and relocation.

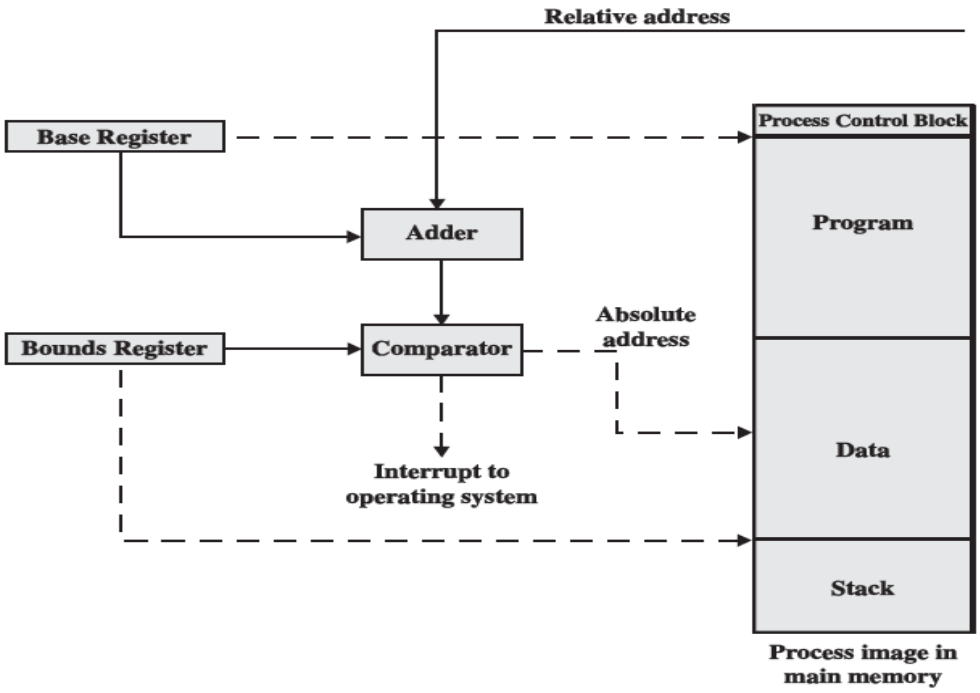


Figure 6.7: Address translation with dynamic partitioning.

The base register holds the entry point of the program, and may be added to a relative address to generate an absolute address. The bounds register indicates the ending location of the program, which is used to compare with each physical address generated.

If the later is within bounds, then the execution may proceed; otherwise, an interrupt is generated, indicating illegal access to memory. The relocation can be easily supported with this mechanism with the new starting address and ending address assigned respectively to the base register and the bounds register.

Placement algorithm

Different strategies may be taken as to how space is allocated to processes:

• ***First fit****:* Allocate the first hole that is big enough. Searching may start either at the beginning of the set of holes or where the previous first-fit search ended.

• ***Best fit****:* Allocate the smallest hole that is big enough. The entire list of holes must be searched unless it is sorted by size. This strategy produces the smallest leftover hole.

• ***Worst fit****:* Allocate the largest hole. In contrast, this strategy aims to produce the largest leftover hole, which may be big enough to hold another process.

Experiments have shown that both first fit and best fit are better than worst fit in terms of decreasing time and storage utilization.

Exercise: Check Your Progress 3

Note: i) Use the space below for your answer.

Ex1: Write about Linking and Loading.

……..……

……..……

……...

Ex2: Write a short note about Swapping.

……..……

……..……

……...

6.8 Paging

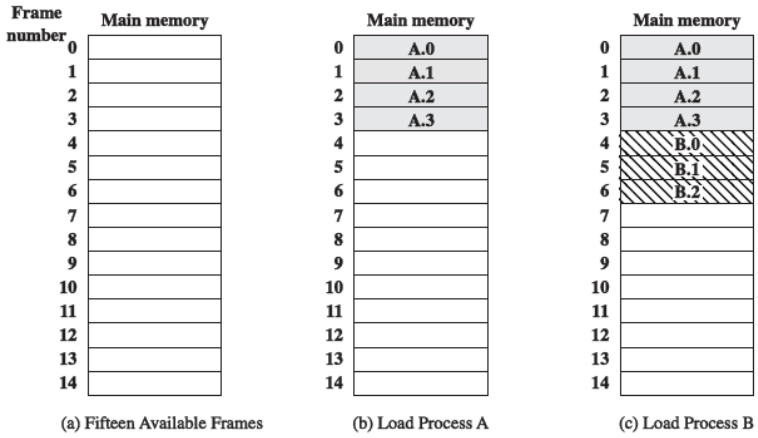
Paging is a memory-management scheme that permits the physical-address space of a process to be non-contiguous. Paging avoids the considerable problem of fitting the varying-sized memory chunks onto the backing store, from which most of the previous memory-management schemes suffered. When some code fragments or data residing in main memory need to be swapped out, space must be found on the backing store. The fragmentation problems discussed in connection with main memory are also prevalent with backing store, except that access is much slower, so compaction is impossible. Because of its advantages over the previous methods, paging in its various forms is commonly used in most operating systems.

6.8.1 Pages and Frames

Both fixed-sized and variable-sized partitions are inefficient due to the problem of either internal or external fragmentation. The fundamental reason of the inefficiency is that each program is allocated a monolithic section of memory. To permit the process image in the main memory to be non-contiguous, we break the main memory into fixed-sized blocks called frames and break the process image into blocks of the same size called pages. When a process is to be executed, its pages are loaded into any available frames, as illustrated in Figure 6.13. The frames allocated to a single process may be contiguous or not.

6.8.2 Page Table and Address Translation

To transform the logical addresses into physical ones, a simple base address register will no longer suffice. Instead, a page table is needed. As Figure 6.8, each entry in a page table associates a page with a frame. The number of the frame holding page *i* is available in item *i* of the page table.



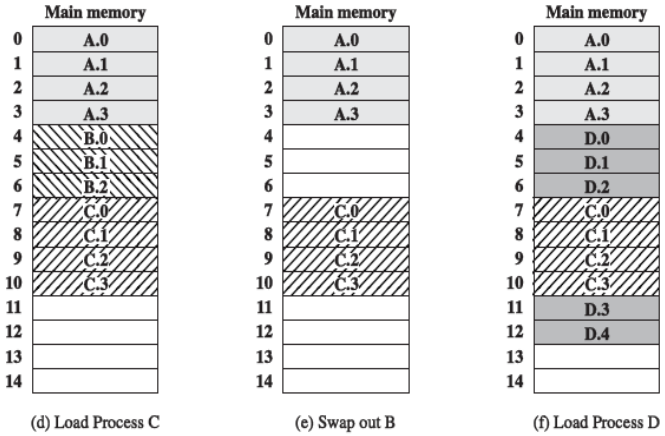


Figure 6.8: Assignment of process pages to free frames.

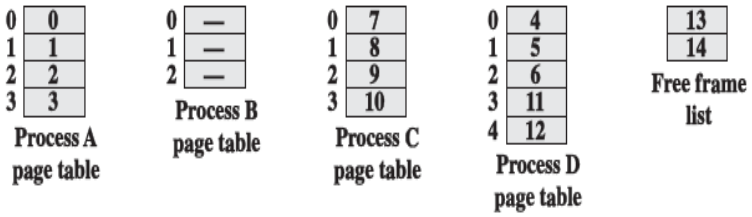


Figure 6.9: Page tables for processes.

Accordingly, a list of free frames is maintained by the operating system.

To facilitate the address translation, the frame or page size is always a power of 2 so that a relative address in the logical address space can be easily split into two parts:

The page number and the offset address. For example, suppose 16-bit addresses are used, and the page size is 1K = 1024 bytes. Thus 10 bits are needed as an offset in a page, leaving 6 bits for the page number. As Figure 6.10 (b) shows, the relative address 1502, which in binary form is 0000010111011110, responds to an offset of 478 (0111011110) on page 1 (000001). Using a page size that is a power of 2 thus makes it easier not only for a compiler/linker to generate logical addresses prepared for this paging scheme, but also for the hardware to support the translation.

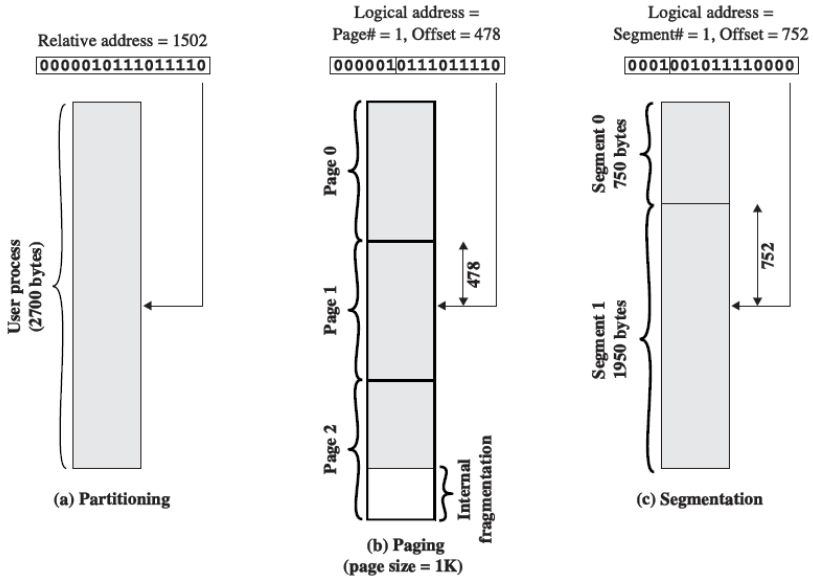
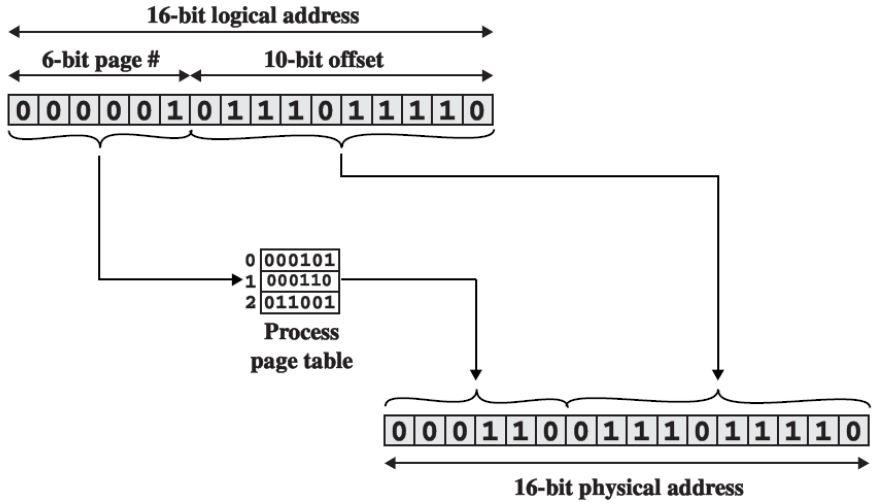


Figure 6.10: Logical addresses.

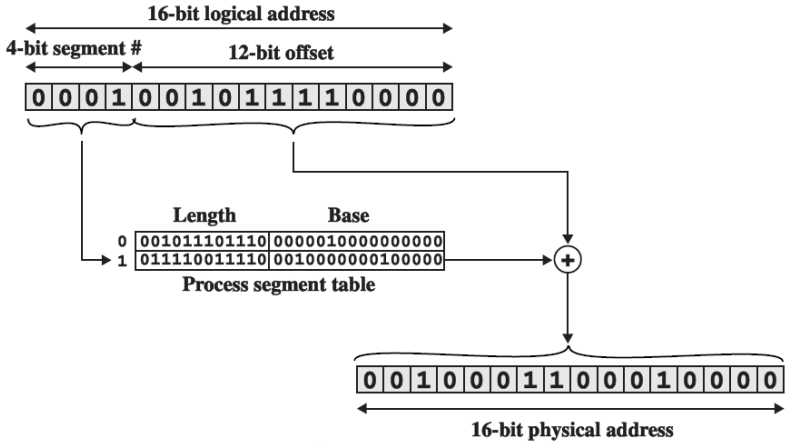
Consider an address of $n+m$ bits, where the leftmost n bits specifies the page number and the rightmost m bits are the offset address, then as Figure 6.11 (a) illustrates, the following steps are needed for address translation:

1. Extract the leftmost n bits of the logical address as the page number.
2. Use the page number as an index into the process page table to find the corresponding frame number, k.
3. The starting physical address of the frame is $k \times 2$m, and the physical address of the referenced byte is that number plus the offset, which may be easily constructed by appending the offset to the frame number.

Paging makes sharing easier by simply associating the frames holding the shared code with the pages in each process involved.



(a) Paging.



(b) Segmentation.

Figure 6.11: Address translation with paging and segmentation.

Traditionally, support for paging has been handled by hardware. However, recent designs have implemented paging by closely integrating the hardware and operating system, especially on 64-bit microprocessors. (See Figure 6.12)

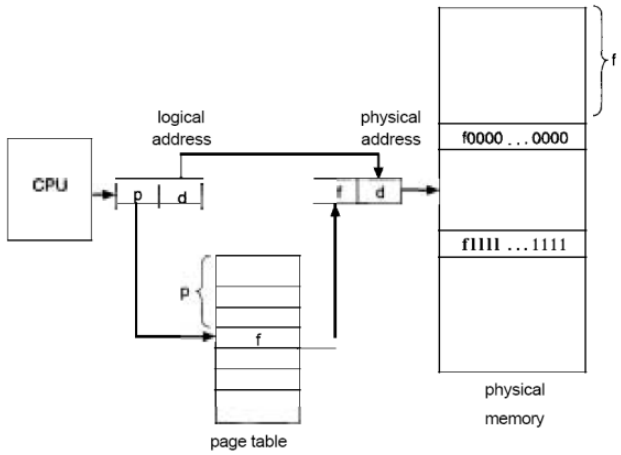


Figure 6.12: Paging hardware.

6.8.3 Shared Pages

Another advantage of paging is the possibility of sharing common code. This consideration is particularly important in a time-sharing environment. Consider a system that supports 40 users, each of whom executes a text editor. If the text editor consists of 150 KB of code and 50 KB of data space, we would need 8,000 KB to support the 40 users. If the code is re-entrant code, here we see a three-page editor-each page of size 50 KB; the large page size is used to simplify the figure-being shared among three processes. Each process has its own data page. Re-entrant code (or pure code) is non-self-modifying code. If the code is re-entrant, then it never changes during execution. Thus, two or more processes can execute the same code at the same time. Each process has its own copy of registers and data storage to hold the data for the process execution. The data for two different processes will, of course, vary for each process.

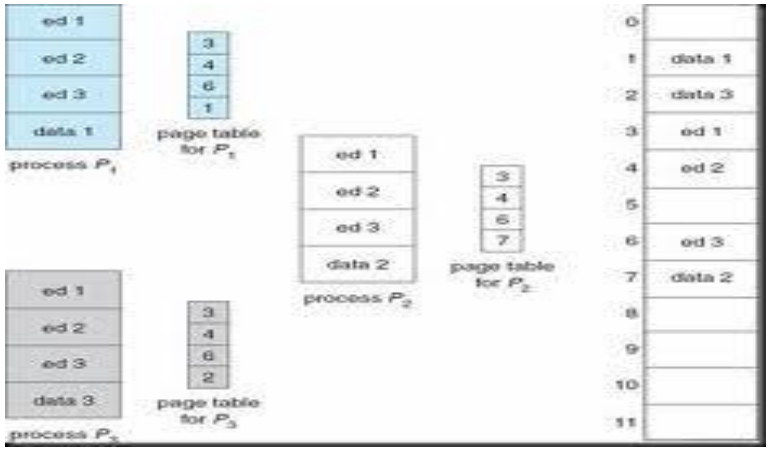


Figure 6.13: Sharing of code in a paging environment.

Self Assessment Question

5. .....is the act of managing computer memory

(a). File management (b). Memory management

(c). Operating system (d). None of these

6. .....if it is known in advance that a program will reside at a specific location of main memory

(a). Compiler time (b). Run time
(c). Both (a).and (b) (d). None of these

7. Once a process has been.....its addresses are absolute
(a). loaded (b). relocated
(c). Both (a) and (b) (d). None of these

8.....or dynamic relocation defers the process of determining absolute addresses until the address is used.
(a). memory management (b). dynamic loading
(c). file management (d). None of these

6.9 Segmentation

Segmentation avoids internal fragmentations which are present in both fixed partitioning and paging, but like dynamic partitioning, it suffers from external fragmentation. However the problem is not that serious because a process may be broken into a number of smaller pieces and the resulting external holes will be much smaller.

Different from paging, which is invisible to the programmer and the compiler, segmentation, is usually visible, which is actually based on the programmers' logical view of programs. Typically, the programmers will assign programs and data to different segments, thus leading to a major advantage of segmentation that the protection and sharing may be easily supported. With segmentation, the logical addresses and physical addresses do not have a simple relationship any more like with partitioning and paging. Each logical address is explicitly expressed by a two tuple: segment-number, offset. The operating system maintains a segment table for each process and a list of free blocks of main memory. As each segment table entry would have to give the starting address in main memory of the corresponding segment, as well as the length of the segment to assure that invalid addresses are not used. Paging and segmentation nowadays have been combined to eliminate both external and internal fragmentation, and facilitate programmers' control on programs. An example for this is the architecture of the Intel 80x86. An important aspect of memory management that became unavoidable with paging is the separation of the user's view of memory and the actual physical memory. The user's view of memory is not the same as the actual physical memory. The user's view is mapped onto physical memory. The mapping allows differentiation between logical memory and physical memory.

Exercise: Check Your Progress 4

Case Study-Fine Soft Studio Memory Manager

Fine Soft Studio developed a most efficient Memory Manager in the world. Applications that used in our Memory Manager, Shows striking improvement of runtime performance, especially on multiprocessor platforms.

Business Description

Existing hardware is constantly improving. Multiprocessors and CPUs with more than one core becoming usual things. From new applications are expected to use efficiently the possibilities of coming hardware. One of the most important system components is Memory Manager - the part of the system that manages program memory so that the program can run most efficiently. Modern programs have object-oriented, multithreaded architectures which put a lot of pressure on memory manager. For web applications hit processing time is measured in milliseconds, so it is very important to have

optimized each component of the system. About 20% of CPU time is spent for managing memory. The application memory manager must keep it up with minimum of CPU overhead, and the total memory used. But existing memory managers have not cope with this goals adequately especially when running on multi CPU ore multi-core modern hardware. Existing memory managers are not able to get most of computer hardware and substantially limit hardware scalability. When we add new processor, or additional core to CPU programs slow down. Instead of expected linear increase runtime performance decreases because memory managers have memory locking problems. If two threads request a memory at the same time, the memory manager must enqueue the requests before processing them. One thread must wait for another thread's request to be processed. This serialization step is a large performance barrier, resulting in scaling problems. The challenge was to build own memory manager which ensures increasing of application runtime performance when new CPU is added and which must be memory space efficient and very fast.

Test application

Application creates a specified number of threads. Each thread executes same actions (emulating memory manager real usage): randomly selects pointer from an array and allocates block of memory if pointer is not assigned or frees memory if pointer is assigned. Size of the memory block is random within defined margins. Application calculates total number of memory block allocations and releases in millions per sec. Resulting value is the maximum value achieved during specified period (higher is better). Test system configuration: CPU - Inter Core 2 Quad (4 cores), OS - Windows Server 2003

Testing Results Analysis

As you may see from the table above our Memory Manager shows dramatic performance increase even for single threaded mode and perfect scalability for multithreaded mode on multiprocessor/multi core system. The closest competitor MS NET 2.0 shows 2 times less performance even for single threaded mode and much less scalability. Other memory managers show even performance degradation when running in multithreaded mode.

Testing Results on 16 Cores system

As a Partner in Intel Software Partner Program we got access to latest hardware platforms and tested our Memory Manager on 16 Cores system (4 CPU Intel Xeon, 4 Cores each). Performance increase of our application running on such system reached unbelievable 15.5 times!

Question

1. What are the solution and key benefits of fine soft studio memory manager?
2. Explain the business description fine soft studio memory manager.

6.10 Summary

- Segmentation memory management scheme is used to divide a program into a number of smaller blocks, called segments.
- With segmentation, the logical addresses and physical addresses do not have a simple relationship any more like with partitioning and paging.
- Logical memory is broken into blocks of the same size called pages.
- An important aspect of paging is the clear separation between the user’s view of memory and the actual physical memory.
- The simplest partitioning method is dividing memory into several fixed-sized partitions in advance, called fixed partitioning.

- The address translation procedure with dynamic partitioning, where the processor provides hardware support for address translation, protection, and relocation.
- Swapping is a general term for exchanging blocks of program code or data between main and secondary memory.
- The range of virtual addresses that the operating system assigns to a user or separately running program is called an address space.

6.11 Keywords

First fit*:* Allocate the first hole that is big enough. Searching may start either at the beginning of the set of holes or where the previous first-fit search ended.
Memory Management: It is about sharing memory so that the largest number of processes can run in the most efficient way.
Paging: It is a memory-management scheme that permits the physical-address space of a process to be non-contiguous
Segmentation: It avoids internal fragmentations which are present in both fixed partitioning and paging, but like dynamic partitioning, it suffers from external fragmentation.
Swapping: It is a general term for exchanging blocks of program code or data between main and secondary memory.

6.12 Review Questions

1. What do you mean by memory management?
2. Explain the memory management requirements.
3. What is the two-level memory organization?
4. Describes the address space.
5. Explain the linking and loading.
6. Define swapping.
7. Describes the memory partitioning.
8. Write short notes on:
   (a) Fixed Partitioning (b) Dynamic Partitioning
   (c) Address translation (d) Logical and physical addresses.
9. What is the paging? Explain in brief.
10. Describe a mechanism by which one segment could belong to the address space of two different processes.

Answers of Self Assessment Questions

1. (a)	2. (a)	3. (b)	4. (a)
5. (b)	6. (a)	7. (c)	8. (b)

6.13 Further Studies

1. http://www.doc.ic.ac.uk/~ob3/Teaching/OperatingSystemsConcepts/Chapter08.pdf
2. Operating System Principles by Silberschatz, Galvin and Gagne

7
Virtual Memory

STRUCTURE

7.0 Objectives

7.1 Introduction

7.2 Concept of Virtual Memory

7.3 Demand Paging

7.4 Page Replacement

7.5 Thrashing

7.6 Demand Segmentation

7.7 Summary

7.8 Keywords

7.9 Review Questions

7.10 Further Studies

7.0 Objectives

After studying this chapter, you will able to:

- Understand the concept of virtual memory
- Discuss about demand paging
- Explain about page replacement
- Understand about thrashing
- Explain demand segmentation

7.1 Introduction

Processes in a system share the CPU and main memory with other processes. However, sharing the main memory poses some special challenges. As demand on the CPU increases, processes slow down in some reasonably smooth way. But if too many processes need too much memory, then some of them will simply not be able to run. When a program is out of space, it is out of luck. Memory is also vulnerable to corruption.

If some process inadvertently write to the memory used by another process, that process might fail in some bewildering fashion totally unrelated to the program logic. In order to manage memory more efficiently and with fewer errors, modern systems provide an abstraction of main memory known as virtual memory (VM). Virtual memory is an elegant interaction of hardware exceptions, hardware

address translation, main memory, disk files, and kernel software that provides each process with a large, uniform, and private address space.

With one clean mechanism, virtual memory provides three important capabilities.

(1). It uses main memory efficiently by treating it as a cache for an address space stored on disk, keeping only the active areas in main memory, and transferring data back and forth between disk and memory as needed.

(2). It simplifies memory management by providing each process with a uniform address space.

(3). It protects the address space of each process from corruption by other processes.

7.2 Concept of Virtual Memory

Virtual memory is one of the great ideas in computer systems. A major reason for its success is that it works silently and automatically, without any intervention from the application programmer. Since virtual memory works so well behind the scenes, why would a programmer need to understand it?

There are several reasons.

- *Virtual memory is central*. Virtual memory pervades all levels of computer systems, playing key roles in the design of hardware exceptions, assemblers, linkers, loaders, shared objects, files, and processes. Understanding virtual memory will help you better understand how systems work in general.
- *Virtual memory is powerful.* Virtual memory gives applications powerful capabilities to create and destroy chunks of memory, map chunks of memory to portions of disk files, and share memory with other processes. For example, did you know that you can read or modify the contents of a disk file by reading and writing memory locations? Or that you can load the contents of a file into memory without doing any explicit copying? Understanding virtual memory will help you harness its powerful capabilities in your applications.
- *Virtual memory is dangerous.* Applications interact with virtual memory every time they reference a variable, dereference a pointer, or make a call to a dynamic allocation package such as malloc. If virtual memory is used improperly, applications can suffer from perplexing and insidious memory related bugs. For example, a program with a bad pointer can crash immediately with a “Segmentation fault” or a “Protection fault,” run silently for hours before crashing, or scariest of all, run to completion with incorrect results. Understanding virtual memory and the allocation packages such as malloc that manage it, can help you avoid these errors.

7.3 Demand Paging

The Demand Paging is also same with the Simple Paging. But the Main Difference is that in the Demand Paging Swapping is used. Means all the Pages will be in and out from the Memory when they are required. When we specify a Process for the Execution then the Processes is stored firstly on the Secondary Memory which is also known as the Hard Disk.

When a user Request for any Operation then the Operating System performs the following instructions:-

1) First of all this will fetch all the instructions from the Physical Memory into the Logical Memory.
2) Decode all the instructions means this will find out which Operation has to be performed on the instructions.
3) Perform Requested Operation.

4) Stores the Result into the Logical Memory and if needed the Results will be Stored into the Physical Memory.

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: Draw the flow diagram of handling a page fault.

……..……

……..……

……...

Caution
To restart the instruction, we must reset the two registers to the values they had before we started the execution of the instruction.

Did You Know?
In 1961, the Burroughs Corporation independently released the first commercial computer with virtual memory, the B5000, with segmentation rather than paging.

7.4 Page Replacement

When a page fault occurs the operating system has to choose a page to remove from memory to make room for the page that has to be brought in. If the page is removed, has been modified while in memory, it must be rewritten to the disk to bring the disk copy up to date. If, however, the page has not been changed (e.g., a page contains program text), the disk copy is already up to date, so no rewrite is needed. The page to be read in just overwrites the page being evicted.
While it would be possible to pick a random page to replace at each page fault, system performance *is* much better if a page that *is* not heavily used is chosen. If a heavily used page is removed, it will probably have to be brought back in quickly, resulting in extra overhead. Much work has been done on the subject of page replacement algorithms, both theoretical and experimental. Below we will describe some of the most important algorithms.

7.4.1 The Optimal Page Replacement Algorithm

The best possible page replacement algorithm is easy to describe but impossible to implement. It goes like this. At the moment that a page fault occurs, some set of pages is in memory. One of these pages will be referenced on the very next instruction (the page containing that instruction). Other pages may not be referenced until 10, 100, or perhaps 1000 instructions later. Each page can be labelled with the number of instructions that will be executed before that page is first referenced.
The optimal page algorithm simply says that the page with the highest label should be removed. If one page will not be used for 8 million instructions and another page will not be used for 6 million instructions, removing the former pushes the page fault that will fetch it back as far into the future as possible. Computers, like people, try to put off unpleasant events for as long as they can. The only problem with this algorithm is that it is unrealizable. At the time of the page fault, the operating system has no way of knowing when each of the pages will be referenced next. (We saw a similar situation earlier with the shortest job first scheduling algorithm-how can the system tell which job is shortest? Still, by running a program on a simulator and keeping track of all page references, it is

possible to implement optimal page replacement on the *second* run by using the page reference information collected during the first run.

7.4.2 The Not Recently Used Page Replacement Algorithm

In order to allow the operating system to collect useful statistics about which pages are being used and which ones are not, most computers with virtual memory have two status bits associated with each page. *R* is set whenever the page is referenced (read or written). M is set when the page is written to (i. e., modified).

It is important to realize that these bits must be updated on every memory reference, so it is essential that they be set by the hardware. Once a bit has been set to 1, it stays 1 until the operating system resets it to 0 in software. If the hardware does not have these bits, they can be simulated as follows.

When a process is started up, all of its page table entries are marked as not in memory. As soon as any page is referenced, a page fault will occur. The operating system then sets the *R* bit (in its internal tables), changes the page table entry to point to the correct page, with mode READ ONLY, and restarts the instruction. If the page is subsequently written on, another page fault will occur, allowing the operating system to set the *M* bit and change the page's mode to READ/WRITE. The R and M bits can be used to build a simple paging algorithm as follows. When a process is start up, both page bits for all its pages are set to 0 by the operating system. Periodically (e.g., on each clock interrupt), the R bit is cleared, to distinguish pages that have not been referenced recently from those that have been. When a page fault occurs, the operating system inspects all the pages and divides them into four categories based on the current values of their *R* and *M* bits:

Class 0: not referenced, not modified.
Class 1: not referenced, modified.
Class 2: referenced, not modified.
Class 3: referenced, modified.

7.4.3 The First-in, First-out (FIFO) Page Replacement Algorithm

Another low-overhead paging algorithm is the FIFO (First-in, First-out) algorithm. To illustrate how this works, consider a supermarket that has enough shelves to display exactly k different products. One day, some company introduces a new convenience food—instant, freeze-dried, organic yogurt that can be reconstituted in a microwave oven. It is an immediate success, so our finite supermarket has to get rid of one old product in order to stock it. One possibility is to find the product that the supermarket has been stocking the longest (i.e., something it began selling 120 years ago) and get rid of it on the grounds that no one is interested any more. In effect, the supermarket maintains a linked list of all the products it currently sells in the order they were introduced.

The new one goes on the back of the list; the one at the front of the list is dropped. As a page replacement algorithm, the same idea is applicable. The operating system maintains a list of all pages currently in memory, with the page at the head of the list the oldest one and the page at the tail the most recent arrival. On a page fault, the page at the head is removed and the new page added to the tail of the list. When applied to stores, FIFO might remove mustache wax, but it might also remove flour, salt, or butter. When applied to computers the same problem arises. For this reason, FIFO in its pure form is rarely used.

7.4.4 The Second Chance Page Replacement Algorithm

A simple modification to FIFO that avoids the problem of throwing out a heavily used page is to inspect the R bit of the oldest page. If it is 0, the page is both old and unused, so it is replaced immediately. If the R bit is 1, the bit is cleared, the page is put onto the end of the list of pages, and its load time is updated as though it had just arrived in memory. Then the search continues.

The operation of this algorithm, called second chance, is shown in Figure 7.4. In Figure 7.4(a) we see pages A through H kept on a linked list and sorted by the time they arrived in memory.

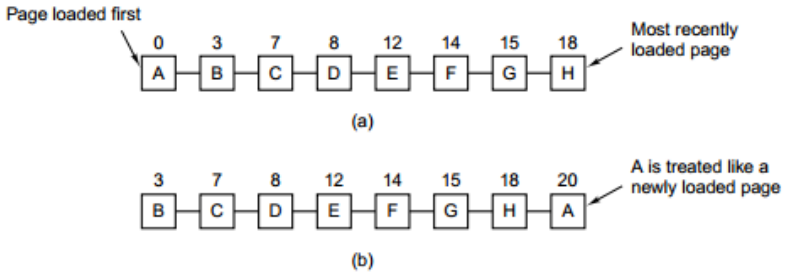


Figure 7.4: Operation of second chance, (a) Pages sorted in FIFO order, (b) Page list if a page fault occurs at time 20 and A has its R bit set. The numbers above the pages are their loading times.

Suppose that a page fault occurs at time 20. The oldest page is A, which arrived at time 0, when the process started. If A has the R bit cleared, it is evicted from memory, either by being written to the disk (if it is dirty), or just abandoned (if it is clean). On the other hand, if the R bit is set, A is put onto the end of the list and its ‘‘load time’’ is reset to the current time (20). The R bit is also cleared. The search for a suitable page continues with B. What second chance is doing is looking for an old page that has not been referenced in the previous clock interval. If all the pages have been referenced, second chance degenerates into pure FIFO. Specifically, imagine that all the pages in Figure 7.4(a) have their R bits set. One by one, the operating system moves the pages to the end of the list, clearing the R bit each time it appends a page to the end of the list. Eventually, it comes back to page A, which now has its R bit cleared. At this point A is evicted. Thus the algorithm always terminates.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Prepare the First-in, First-out (FIFO) page replacement algorithm.

……..……

……..……

……...

7.4.5 The Clock Page Replacement Algorithm

Although second chance is a reasonable algorithm, it is unnecessarily inefficient because it is constantly moving pages around on its list. A better approach is to keep all the page frames on a circular list in the form of a clock, as shown in Figure 7.5. A hand points to the oldest page.

When a page fault occurs, the page being pointed to by the hand is inspected. If its R bit is 0, the page is evicted, the new page is inserted into the clock in its place, and the hand is advanced one position. If R is 1, it is cleared and the hand is advanced to the next page. This process is repeated until a page is found with R = 0.

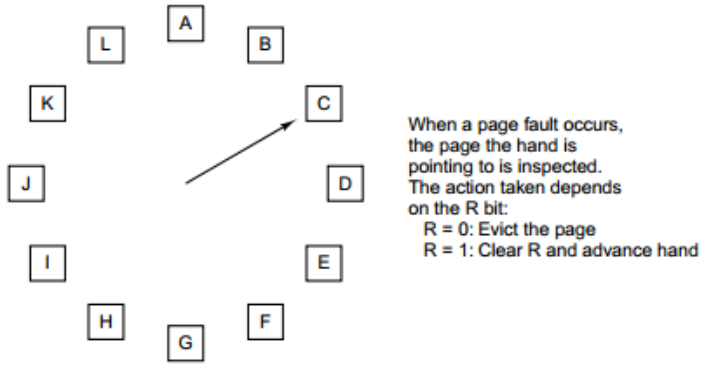


Figure 7.5: The clock page replacement algorithm.

Not surprisingly, this algorithm is called clock. It differs from second chance only in the implementation.

7.4.6 The Least Recently Used (LRU) Page Replacement Algorithm

A good approximation to the optimal algorithm is based on the observation that pages that have been heavily used in the last few instructions will probably be heavily used again in the next few. Conversely, pages that have not been used for ages will probably remain unused for a long time. This idea suggests a realizable algorithm: when a page fault occurs, throw out the page that has been unused for the longest time. This strategy is called LRU (Least Recently Used) paging.

Although LRU is theoretically realizable, it is not cheap. To fully implement LRU, it is necessary to maintain a linked list of all pages in memory, with the most recently used page at the front and the least recently used page at the rear. The difficulty is that the list must be updated on every memory reference. Finding a page in the list, deleting it, and then moving it to the front is a very time consuming operation, even in hardware (assuming that such hardware could be built). However, there are other ways to implement LRU with special hardware. Let us consider the simplest way first. This method requires equipping the hardware with a 64-bit counter, C, that is automatically incremented after each instruction. Furthermore, each page table entry must also have a field large enough to contain the counter. After each memory reference, the current value of C is stored in the page table entry for the page just referenced. When a page fault occurs, the operating system examines all the counters in the page table to find the lowest one. That page is the least recently used. Now let us look at a second hardware LRU algorithm. For a machine with n page frames, the LRU hardware can maintain a matrix of n × n bits, initially all zero. Whenever page frame k is referenced, the hardware first sets all the bits of row k to 1, and then sets all the bits of column k to 0. At any instant, the row whose binary value is lowest is the least recently used; the row whose value is next lowest is next least recently used, and so forth.

The workings of this algorithm are given in Figure 7.6 for four page frames and page references in the order

0 1 2 3 2 1 0 3 2 3

After page 0 is referenced, we have the situation of Figure 7.6(a). After page 1 is reference, we have the situation of Figure 7.6(b), and so forth.

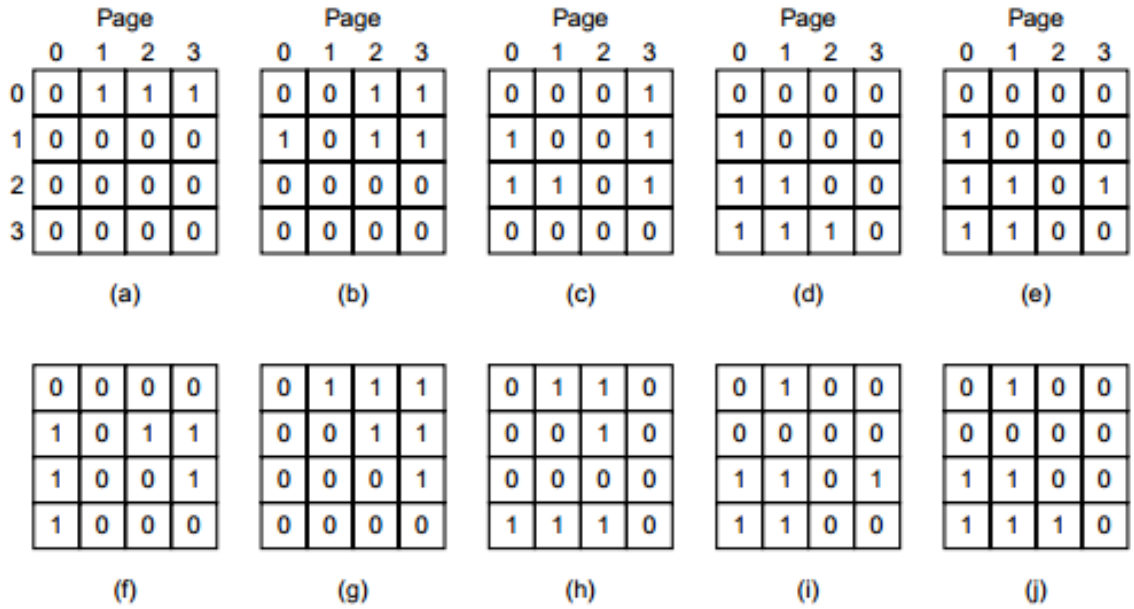


Figure 7.6: LRU using a matrix when pages are referenced in the order 0, 1, 2, 3, 2, 1, 0, 3, 2, 3.

7.4.7 Simulating LRU in Software

Although both of the previous LRU algorithms are realizable in principle, few, if any, machines have this hardware, so they are of little use to the operating system designer who is making a system for a machine that does not have this hardware. Instead, a solution that can be implemented in software is needed. One possibility is called the NFU (Not Frequently Used) algorithm. It requires a software counter associated with each page, initially zero. At each clock interrupt, the operating system scans all the pages in memory. For each page, the R bit, which is 0 or 1, is added to the counter. In effect, the counters are an attempt to keep track of how often each page has been referenced.

When a page fault occurs, the page with the lowest counter is chosen for replacement. The main problem with NFU is that it never forgets anything. For example, in a multi-pass compiler, pages that were heavily used during pass 1 may still have a high count well into later passes. In fact, if pass 1 happens to have the longest execution time of all the passes, the pages containing the code for subsequent passes may always have lower counts than the pass 1 pages. Consequently, the operating system will remove useful pages instead of pages no longer in use. Fortunately, a small modification to NFU makes it able to simulate LRU quite well. The modification has two parts. First, the counters are each shifted right 1 bit before the R bit is added in. Second, the R bit is added to the leftmost, rather than the rightmost bit. Figure 7.7 illustrates how the modified algorithm, known as aging, works.

Suppose that after the first clock tick the R bits for pages 0 to 5 have the values 1, 0, 1, 0, 1, and 1, respectively (page 0 is 1, page 1 is 0, page 2 is 1, etc.). In other words, between tick 0 and tick 1, pages 0, 2, 4, and 5 were referenced, setting their R bits to 1, while the other ones remain 0. After the six corresponding counters have been shifted and the R bit inserted at the left, they have the values shown in Figure 7.7(a). The four remaining columns show the six counters after the next four clock ticks.

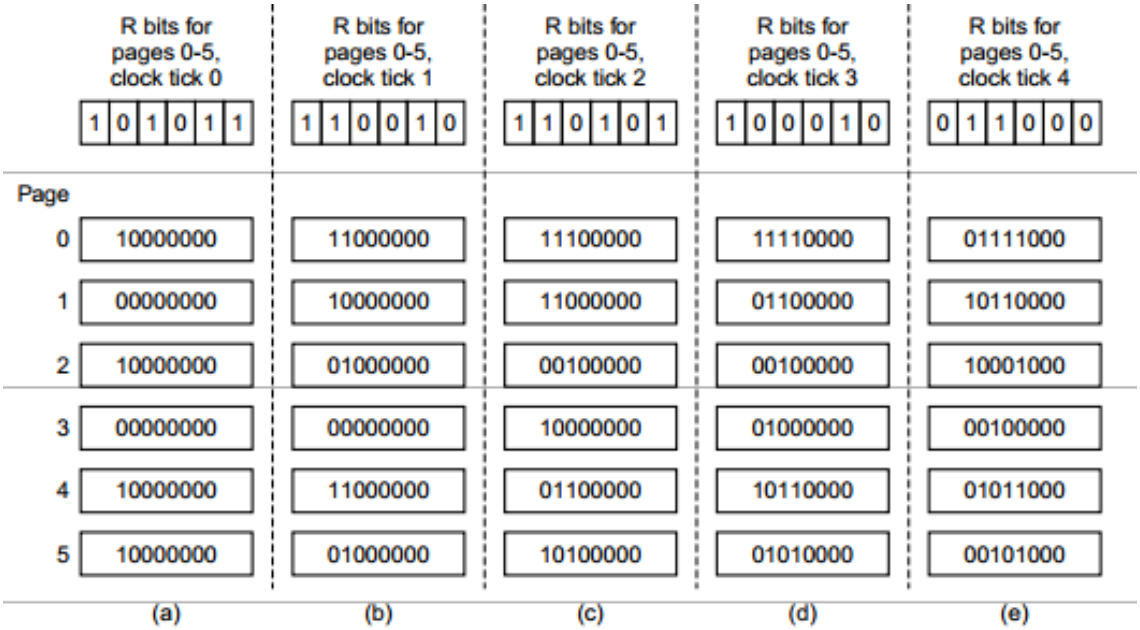


Figure 7.7: The aging algorithm simulates LRU in software. Shown are six pages for five clock ticks. The five clock ticks are represented by (a) to (e).

When a page fault occurs, the page whose counter is the lowest is removed. It is clear that a page that has not been referenced for, say, four clock ticks will have four leading zeros in its counter and thus will have a lower value than a counter that has not been referenced for three clock ticks.

7.4.8 The Working Set Page Replacement Algorithm

In the purest form of paging, processes are started up with none of their pages in memory. As soon as the CPU tries to fetch the first instruction, it gets a page fault, causing the operating system to bring in the page containing the first instruction. Other page faults for global variables and the stack usually follow quickly.

After a while, the process has most of the pages it needs and settles down to run with relatively few page faults. This strategy is called demand paging because pages are loaded only on demand, not in advance. Of course, it is easy enough to write a test program that systematically reads all the pages in a large address space, causing so many page faults that there is not enough memory to hold them all. Fortunately, most processes do not work this way. They exhibit a locality of reference, meaning that during any phase of execution, the process references only a relatively small fraction of its pages. Each pass of a multi-pass compiler, for example, references only a fraction of all the pages, and a different fraction at that.

It has been long known that most programs do not reference their address space uniformly, but that the references tend to cluster on a small number of pages. A memory reference may fetch an instruction, it may fetch data, or it may store data. At any instant of time, t, there exists a set consisting of all the pages used by the k most recent memory references. This set, w(k, t), is the working set. Because the k = 1 most recent references must have used all the pages used by the k > 1 most recent references, and possibly others, w(k, t) is a monotonically no decreasing function of k. The limit of w(k, t) as k becomes large is finite because a program cannot reference more pages than its address space contains, and few programs will use every single page. Figure 7.8 depicts the size of the working set as a function of k.

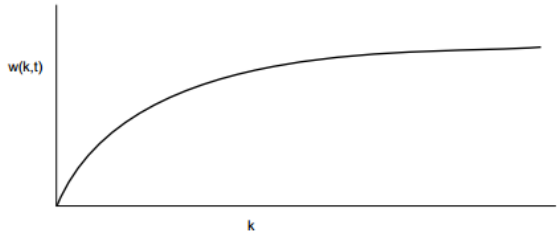


Figure 7.8: **The working set is the set of pages used by the k most recent memory references. The function w (k, t) is the size of the working set at time t.**

The algorithm works as follows. The hardware is assumed to set the R and M bits, as we have discussed before. Similarly, a periodic clock interrupt is assumed to cause software to run that clears the Referenced bit on every clock tick. On every page fault, the page table is scanned to look for a suitable page to evict. As each entry is processed, the R bit is examined. If it is 1, the current virtual time is written into the Time of last use field in the page table, indicating that the page was in use at the time the fault occurred. Since the page has been referenced during the current clock tick, it is clearly in the working set and is not a candidate for removal (τ is assumed to span multiple clock ticks).

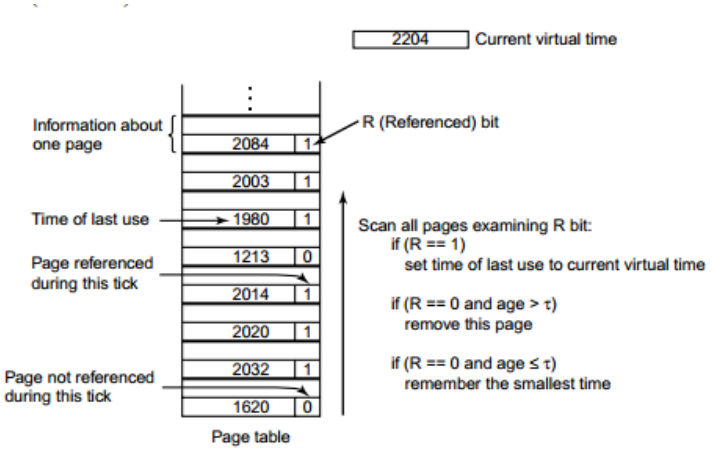


Figure 7.9: The working set algorithm.

If R is 0, the page has not been referenced during the current clock tick and may be a candidate for removal. To see whether or not it should be removed, its age, that is, the current virtual time minus its Time of last use is computed and compared to τ. If the age is greater than τ, the page is no longer in the working set. It is reclaimed and the new page loaded here. The scan continues updating the remaining entries. However, if R is 0 but the age is less than or equal to τ, the page is still in the working set. The page is temporarily spared, but the page with the greatest age (smallest value of Time of last use) is noted. If the entire table is scanned without finding a candidate to evict, that means that all pages are in the working set. In that case, if one or more pages with R = 0 were found, the one with the greatest age is evicted. In the worst case, all pages have been referenced during the current clock tick (and thus all have R = 1), so one is chosen at random for removal, preferably a clean page, if one exists.

7.4.9 The WSClock Page Replacement Algorithm

The basic working set algorithm is cumbersome since the entire page table has to be scanned at each page fault until a suitable candidate is located. An improved algorithm that is based on the clock

algorithm but also uses the working set information is called WSClock. Due to its simplicity of implementation and good performance, it is widely used in practice.

The data structure needed is a circular list of page frames, as in the clock algorithm, and as shown in Figure 7.10(a). Initially, this list is empty. When the first page is loaded, it is added to the list. As more pages are added, they go into the list to form a ring. Each entry contains the Time of last use field from the basic working set algorithm, as well as the R bit (shown) and the M bit (not shown).

As with the clock algorithm, at each page fault the page pointed to by the hand is examined first. If the R bit is set to 1, the page has been used during the current tick so it is not an ideal candidate to remove. The R bit is then set to 0, the hand advanced to the next page, and the algorithm repeated for that page. The state after this sequence of events is shown in Figure 7.10(b).

Now consider what happens if the page pointed to has R = 0, as shown in Figure 7.10(c). If the age is greater than τ and the page is clean, it is not in the working set and a valid copy exists on the disk. The page frame is simply claimed and the new page put there, as shown in Figure 7.10(d). On the other hand, if the page is dirty, it cannot be claimed immediately since no valid copy is present on disk. To avoid a process switch, the write to disk is scheduled, but the hand is advanced and the algorithm continues with the next page. After all, there might be an old, clean page further down the line that can be used immediately. In principle, all pages might be scheduled for disk I/O on one cycle around the clock. To reduce disk traffic, a limit might be set, allowing a maximum of n pages to be written back. Once this limit has been reached, no new writes are scheduled.

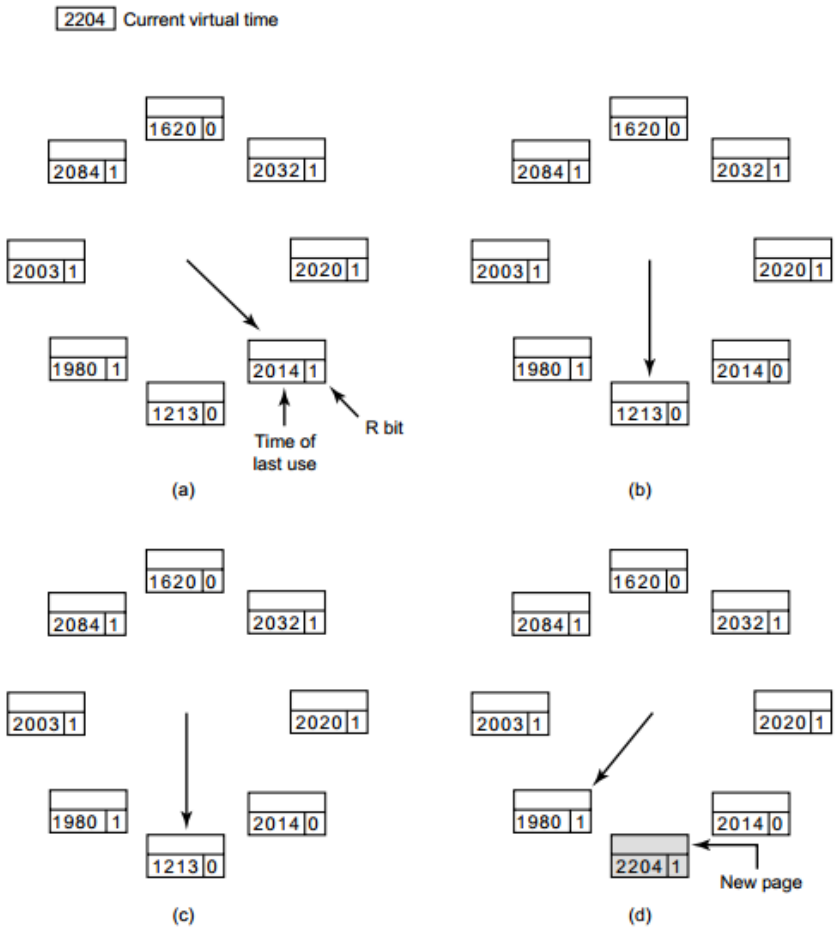


Figure 7.10: **Operation of the WSClock algorithm. (a) and (b) give an example of what happens when R = 1. (c) and (d) give an example of R = 0.**

What happens if the hand comes all the way around to its starting point?
There are two cases to distinguish:
1. At least one write has been scheduled.
2. No writes have been scheduled.
In the former case, the hand just keeps moving, looking for a clean page. Since one or more writes have been scheduled, eventually some write will complete and its page will be marked as clean. The first clean page encountered is evicted.
This page is not necessarily the first write scheduled because the disk driver may reorder writes in order to optimize disk performance.
In the latter case, all pages are in the working set; otherwise at least one write would have been scheduled. Lacking additional information, the simplest thing to do is claim any clean page and use it. The location of a clean page could be kept track of during the sweep. If no clean pages exist, then the current page is chosen and written back to disk.

Did You Know?

The first minicomputer to introduce virtual memory was the Norwegian NORD-1; during the 1970s, other minicomputers implemented virtual memory, notably VAX models running VMS.

Self Assessment Questions

1. ……is an elegant interaction of hardware address translation and kernel software that provides each process with a uniform and private address space.
(a). Main memory (b). Secondary memory
(c). Hard disk (d). Virtual memory

2. A .....manipulates entire processes, whereas a pager is concerned with the individual pages of a process.
(a). swapper (b). segmentation
(c). paging (d). None of these

3. ……can have a significant effect on the performance of a computer system.
(a). Main memory (b). Segmentation
(c). Demand paging (d). None of these

4. A good approximation to the optimal algorithm is based on the observation that pages.
(a). True (b). False

5. An improved algorithm that is based on the clock algorithm but also uses the working set information is called……
(a). counter clock (b). WSClock
(c). paging (d). None of these

7.5 Thrashing

If the number of frames allocated to a low-priority process falls below the minimum number required by the computer architecture, we must suspend that process execution. We should then page out its remaining pages, freeing all its allocated frames. This provision introduces a swap-in, swap-out level of intermediate CPU scheduling. In fact, look at any process that does not have “enough” frames.

Although it is technically possible to reduce the number of allocated frames to the minimum, there is some (larger) number of pages in active use. If the process does not have this number of frames, it will quickly page fault. At this point, it must replace some page. However, since all its pages are in active use, it must replace a page that will be needed again right away. Consequently, it quickly faults again, and again, and again. The process continues to fault, replacing pages for which it then faults and brings back in right away. This high paging activity is called thrashing

7.5.1 Cause of Thrashing

Thrashing results in severe performance problems. Consider the following scenario, which is based on the actual behaviour of early paging systems. The operating system monitors CPU utilization. If CPU utilization is too low, we increase the degree of multiprogramming by introducing a new process to the system. A global page-replacement algorithm is used; it replaces pages with no regard to the process to which they belong. Now suppose that a process enters a new phase in its execution and needs more frames. It starts faulting and taking frames away from other processes. These processes need those pages, however, and so they also fault, taking frames from other processes.. As they queue up for the paging device, the ready queue empties. As processes wait for the paging device, CPU utilization decreases. The CPU scheduler sees the decreasing CPU utilization, and increases the degree of multiprogramming as a result. The new process tries to get started by taking frames from running processes, causing more page faults, and a longer queue for the paging device. As a result, CPU utilization drops even further, and the CPU scheduler tries to increase the degree of multiprogramming even more. Thrashing has occurred and system throughput plunges. The page fault rate increases tremendously. As a result, the effective memory access time increases. No work is getting done, because the processes are spending all their time paging. This phenomenon is illustrated in the Figure 7.11, in which CPU utilization is plotted against the degree of multiprogramming. As the degree of multiprogramming increases, CPU utilization also increases, although more slowly, until a maximum is reached. If the degree of multiprogramming is increased even further, thrashing sets in and CPU utilization drops sharply. At this point, to increase CPU utilization and stop thrashing, we must decrease the degree of multiprogramming. We can limit the effects of thrashing by using a local replacement algorithm (or priority replacement algorithm). With local replacement, if one process starts thrashing, it cannot steal frames from another process and cause the latter to thrash also. Pages are replaced with regard to the process of which they are a part. However, if processes are thrashing, they will be in the queue for the paging device most of the time.

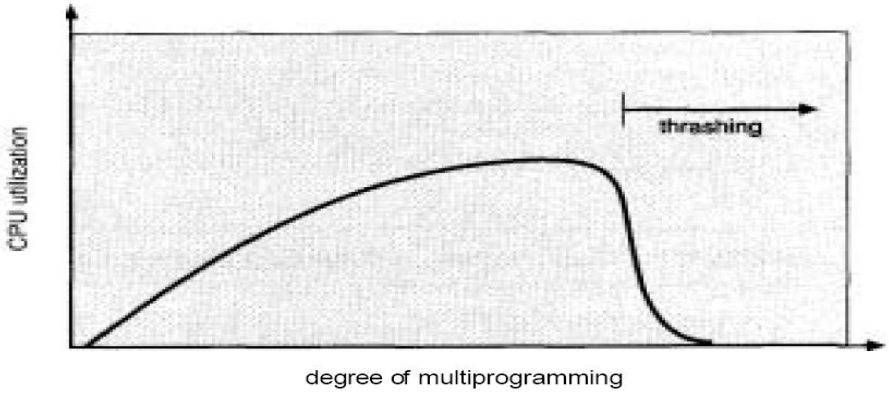


Figure 7.11: **Thrashing.**

The average service time for a page fault will increase, due to the longer average queue for the paging device. Thus, the effective access time will increase even for a process that is not thrashing.

To prevent thrashing, we must provide a process as many frames as it needs. But how do we know how many frames it "needs"? There are several techniques. The working-set strategy starts by looking at how many frames a process is actually using. This approach defines the locality model of process execution.

Caution
Faulting processes must use the paging device to swap pages in and out.

7.5.2 Locality Model

The locality model states that, as a process executes, it moves from locality to locality. A locality is a set of pages that are actively used together. A program is generally composed of several different localities, which may overlap. For example, when a subroutine is called, it defines a new locality. In this locality, memory references are made to the instructions of the subroutine, its local variables, and a subset of the global variables. When the subroutine is exited, the process leaves this locality, since the local variables and instructions of the subroutine are no longer in active use. We may return to this locality later. Thus, we see that localities are defined by the program structure and its data structures. The locality model states that all programs will exhibit this basic memory reference structure. Note that the locality model is the unstated principle behind the caching discussions so far in this book. If accesses to any types of data were random rather than patterned, caching would be useless. Suppose that we allocate enough frames to a process to accommodate its current locality. It will fault for the pages in its locality until all these pages are in memory; then, it will not fault again until it changes localities. If we allocate fewer frames than the size of the current locality, the process will thrash, since it cannot keep in memory all the pages that it is actively using.

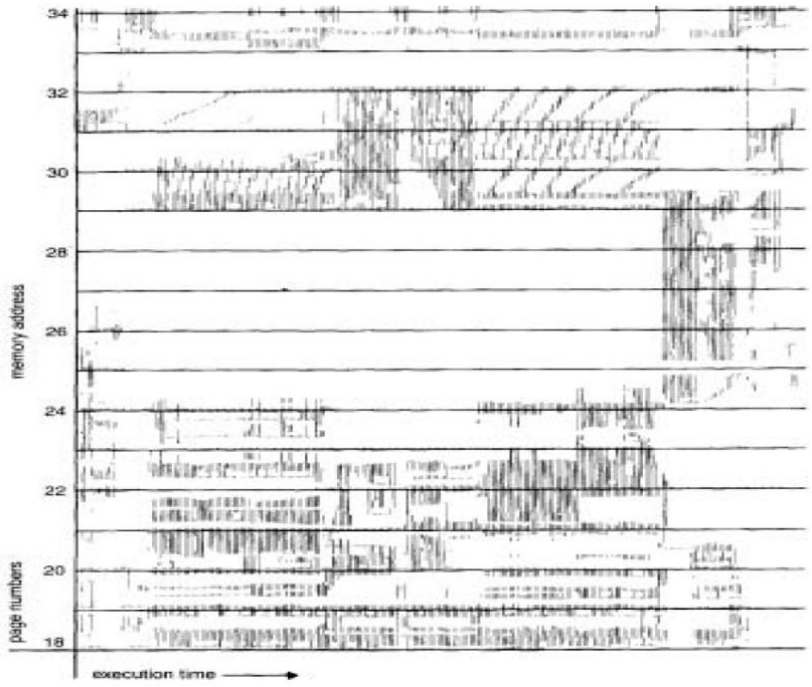


Figure 7.12: **Locality in a memory-reference pattern.**

7.5.3 Working-Set Model

The working-set model is based on the assumption of locality. This model uses a parameter, A, to define the working-set window. The idea is to examine the most recent A page references. The set of pages in the most recent A page references is the working set If a page is in active use, it will be in

the working set. If it is no longer being used, it will drop from the working set A time units after its last reference. Thus, the working set is an approximation of the program's locality. For example, given the sequence of memory references shown in if A = 10 memory references, then the working set at time t_1 is (1, 2, 5, 6, 7). By time t_2, the working set has changed to (3, 4). The accuracy of the working set depends on the selection of A. If A is too small, it will not encompass the entire locality; if A is too large, it may overlap several localities. In the extreme, if A is infinite, the working set is the set of pages touched during the process execution. The most important property of the working set is its size. If we compute the working-set size, WSSi, for each process in the system, we can then consider where D is the total demand for frames. Each process is actively using the pages in its working set. Thus, process i needs WSSi frames. If the total demand is greater than the total number of available frames (D > m), thrashing will occur, because some processes will not have enough frames. Use of the working-set model is then simple. The operating system monitors the working set of each process and allocates to that working set enough frames to provide it with its working-set size. If there are enough extra frames, another process can be initiated. If the sum of the working-set sizes increases, exceeding the total number of available frames, the operating system selects a process to suspend. The process pages are written out and its frames are reallocated to other processes. The suspended process can be restarted later. This working-set strategy prevents thrashing while keeping the degree of multiprogramming as high as possible. Thus, it optimizes CPU utilization. The difficulty with the working-set model is keeping track of the working set. The working-set window is a moving window. At each memory reference, a new reference appears at one end and the oldest reference drops off the other end. A page is in the working set if it is referenced anywhere in the working-set window. We can approximate the working-set model with a fixed interval timer interrupt and a reference bit. For example, assume A is 10,000 references and we can cause a timer interrupt every 5,000 references. When we get a timer interrupt, we copy and clear the reference-bit values for each page. Thus, if a page fault occurs, we can examine the current reference bit and 2 in-memory bits to determine whether a page was used within the last 10,000 to 15,000 references. If it was used, at least 1 of these bits will be on. If it has not been used, these bits will be off. Those pages with at least 1 bit on will be considered to be in the working set. Note that this arrangement is not entirely accurate, because we cannot tell where, within an interval of 5,000, a reference occurred. We can reduce the uncertainty by increasing the number of our history bits and the frequency of interrupts (for example, 10 bits and interrupts every 1,000 references). However, the cost to service these more frequent interrupts will be correspondingly higher.

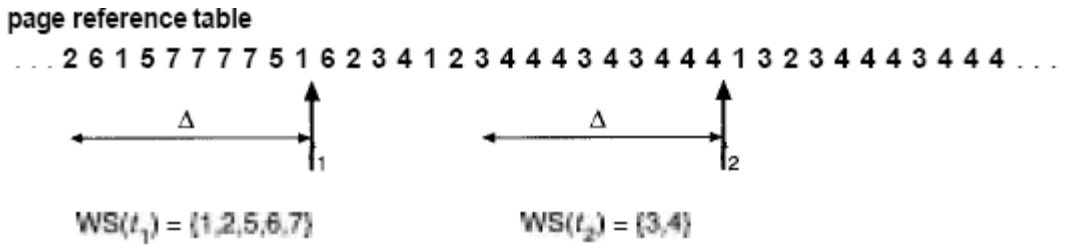


Figure 7.13: **Working-set model.**

7.5.4 Page-Fault Frequency

The working-set model is successful, and knowledge of the working set can be useful for pre-paging but it seems a clumsy way to control thrashing. A strategy that uses the page-fault frequency (PFF) takes a more direct approach.

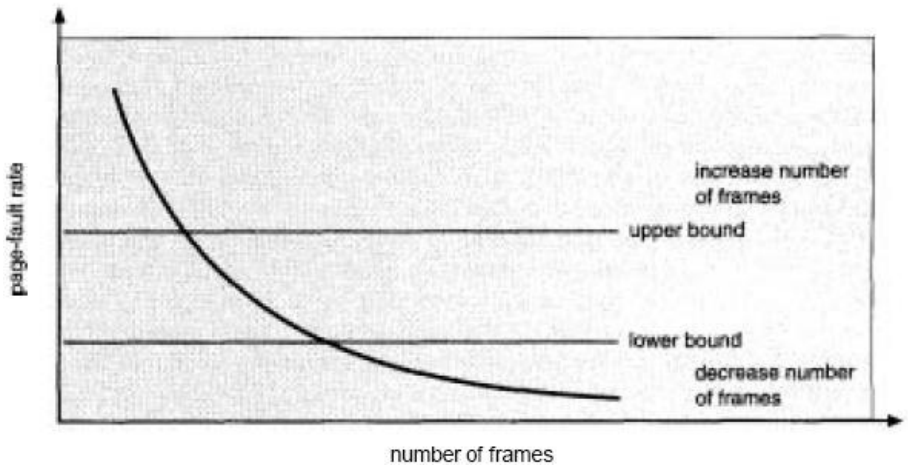


Figure 7.14: Page-fault frequency.

The specific problem is how to prevent thrashing. Thrashing has a high page-fault rate. Thus, we want to control the page-fault rate. When it is too high, we know that the process needs more frames. Similarly, if the page-fault rate is too low, then the process may have too many frames. We can establish upper and lower bounds on the desired page-fault rate. If the actual page-fault rate exceeds the upper limit, we allocate that process another frame; if the page-fault rate falls below the lower limit, we remove a frame from that process. Thus, we can directly measure and control the page-fault rate to prevent thrashing.

As with the working-set strategy, we may have to suspend a process. If the page-fault rate increases and no free frames are available, we must select some process and suspend it. The freed frames are then distributed to processes with high page-fault rates.

Table 7.1: Summary of page replacement algorithms

Algorithm	Comment
Optimal	Not implementable, but useful as a benchmark
NRU (Not Recently Used)	Very crude
FIFO (First-In, First-Out)	Might throw out important pages
Second chance	Big improvement over FIFO
Clock	Realistic
LRU (Least Recently Used)	Excellent, but difficult to implement exactly
NFU (Not Frequently Used)	Fairly crude approximation to LRU
Aging	Efficient algorithm that approximates LRU well
Working set	Somewhat expensive to implement
WSClock	Good efficient algorithm

7.6 Demand Segmentation

Although demand paging is considered the most efficient virtual memory system, a significant amount of hardware is required to implement it. When this hardware is lacking, less efficient means are sometimes devised to provide virtual memory. A case in point is demand segmentation.

Exercise: Check Your Progress 3

Case Study-Reasons for Low Virtual Memory

Correcting the Microsoft Windows “low virtual memory” warning is a balancing act dependent on several key factors. These variables include how much RAM is installed in your system, how much

hard drive space is available and how many programs you run concurrently. Windows will automatically attempt to remedy low virtual memory issues with varying degrees of success. If the automatic virtual memory adjustment does not permanently fix the problem, Microsoft provides several tools, integrated within Windows, to manually analyze and correct the problem.

Understanding Virtual Memory

- To explain the concept of virtual memory, think of the computer systems as having two types of memory: storage and working. Hard drives are one of the more common types of storage memory and are where the computer's software and data are stored. Computers also have a second type of memory known as RAM (random-access memory), which is strictly used as working memory. RAM has no ability to store information once the computer is shut down but is used as temporary working memory because it is significantly faster than hard drives. This fact explains why programs and data load into RAM memory as it is being worked on.

Virtual memory is a third, hybrid type of memory that uses your hard drive to read information from and write information to an extension of RAM when the RAM installed in your system is completely full.

Virtual Memory Settings

During a Windows installation, a default setting defining the amount of hard drive space used for virtual memory is created. For many users, this size is adequate, while others find it is set too low.

To view these settings in Windows 2000, click “Start,” then “My Computer,” and then “Properties,” followed by clicking on the “Advanced” tab. Click “Performance Options,” then clicks “Change” when you are presented with the Virtual Memory screen.

With XP, click “Start,” then “My Computer,” then “Properties,” and then select the “Advanced” tab. When you are presented with the Performance screen, click “Settings” and then the “Advanced” tab in the Performance Options dialog box. Click “Change” at the Virtual Memory screen.

For Windows Vista, click the “Start” button, then “Control Panel,” then select “System and Maintenance” and finally “System.” Look for Advanced system settings and enter your password when prompted. Click the “Advanced” tab, and under Performance, click “Settings.” Select the second “Advanced” tab and click the “Change” button in the Virtual Memory section. Remove the check from the “Automatically Manage Paging File Size for All Drives” and select the drive that you want to alter the virtual memory size on, assuming there are multiple drives installed.

As a general rule, and assuming that you have adequate hard drive space free, enter a value from 1 to 2 GB. Then close down everything to get back to your desktop. Depending on your version of Windows, a restart may be required.

Add More RAM

- By far, the best way to manage this problem is to maximize the RAM installed in your system to whatever the computer will support. Dollar for dollar, investing in RAM is the best performance-boosting upgrade anyone can make without making serious changes to their systems.

Questions

1. How do you change the Virtual Memory Settings?
2. What are the basic Reasons for Low Virtual Memory?

7.7 Summary

- Virtual memory is an elegant interaction of hardware exceptions, hardware address translation, main memory, disk files, and kernel software that provides each process with a large, uniform, and private address space.
- A swapper manipulates entire processes, whereas a pager is concerned with the individual pages of a process.
- A page fault could occur at any memory reference. If the page fault occurs on the instruction fetch, we can restart by fetching the instruction again.
- Demand paging can have a significant effect on the performance of a computer system. To see why, let us compute the effective access time for a demand paged memory.
- The working-set model is based on the assumption of locality. This model uses a parameter to define the working-set window.

7.8 Keywords

Demand Paging: A demand-paging system is similar to a paging system. Processes reside on secondary memory.

Locality Model: The locality model states that, as a process executes, it moves from locality to locality. A locality is a set of pages that are actively used together.

Secondary memory: This memory holds those pages that are not presentin main memory.

Thrashing: it quickly faults again, and again, and again. The process continues to fault, replacing pages for which it then faults and brings back in right away. This high paging activity is called thrashing.

Virtual Memory: Modern systems provide an abstraction of main memory known as virtual memory (VM) in order to manage memory more efficiently and with fewer errors.

7.9 Review Questions

1. Define the concept of virtual memory.
2. Explain the demand paging in detail.
3. Discuss the performance of demand paging.
4. Explain the page replacement algorithm.
5. Explain the process of working set page replacement algorithm.
6. What is thrashing? Explain its causes.
7. Why we use the page replacement algorithm? Explain.
8. Explain about the working set model.
9. Explain the page fault recovery.
10. What is demand segmentation?

Answers of Self Assessment Questions

1. (d) 2. (a) 3. (c) 4. (a) 5. (b)

7.10 Further Studies

1. http://my.safaribooksonline.com/book/operating-systems-and-server-administration/9788131715482/windows-xp/ch18lev1sec10
2. Operating System Principles by Silberschatz, Galvin and Gagne

8

Input Output Systems

STRUCTURE

8.0 Objectives

8.1 Introduction

8.2 Input-Output Devices

8.3 Hardware Support for I/O

8.4 I/O Communication Techniques

8.5 I/O Software Device Drivers

8.6 Performance Consideration

8.7 Summary

8.8 Keywords

8.9 Review Questions

8.10 Further Studies

8.0 Objectives

After studying this chapter, you will be able to:

- Describe the input-output devices
- Explain the hardware support for I/O
- Describe the I/O communication techniques
- Explain the I/O software device drivers
- Define a performance consideration

8.1 Introduction

One of the main functions of operating systems is to control all the computer's input/output (I/O) devices. It must issue commands to the devices, catch interrupts (I/O), and handle errors. It should also provide an interface between the devices and the rest of the system that is simple and easy to use. Different people look at I/O hardware in different ways. Electrical engineers look at it in term of chips, wires, power supplies, motors and all the other physical components that make up the hardware. Programmers look at the interface presented to the software the commands the hardware accepts, the functions it carries out, and the errors that can be reported back.

The control of devices connected to the computer is a major concern of operating-system designers. Because I/O devices vary so widely in their function and speed (consider a mouse, a hard disk, and a

CD-ROM jukebox), a variety of methods are needed to control them. These methods form the I/O sub-system of the kernel, which separates the rest of the kernel from the complexity of managing I/O devices.

The I/O-device technology exhibits two conflicting trends. On one hand, we see increasing standardization of software and hardware interfaces. This trend helps us to incorporate improved device generations into existing computers and operating systems. On the other hand, we see an increasingly broad variety of I/O devices. Some new devices are so unlike previous devices that it is a challenge to incorporate them into computers and operating systems.

This challenge is met by a combination of hardware and software techniques. The basic I/O hardware elements, such as ports, buses, and device controllers, accommodate a wide variety of I/O devices. To encapsulate the details and oddities of different devices, the kernel of an operating system is structured to use device driver modules. The device drivers present a uniform device access interface to the I/O subsystem, much as system calls provide a standard interface between the application and the operating system.

8.2 Input-Output Devices

The I/O (Input-Output devices) devices can be roughly divided into two categories: Block devices and character devices. A block device is one that stores information in fixes-size blocks, each one with its own address. Common block sizes range from 128 bytes to 1024 bytes. The essential property of a block device is that it is possible to read or write each block independently of all the other ones. In other words, at any instant, the program can read or write any of the blocks. Disks are block devices.

If you look closely, the boundary between device that are block addressable and those that are not is not well defined. Everyone agrees that a disk is a block addressable device because no matter where the arm currently is, it is always possible to seek another cylinder and then wait for the required block to rotate under the head. Now consider a magnetic tape containing blocks of 1K bytes. If the tape drive is given a command to read block N, it always rewind the tape and go forward until it comes to block N. This operation is analogous to a disk doing seek, except that it take much longer. Also, it may or may not be possible to rewrite one block in the middle of a tape. Even if were possible to use magnetic tapes as block devices that are stretching the point somewhat: they are normally not used that way.

The other type of I/O device is the character device. A character device delivers or accepts a stream of character, without regard to any block structure. It is not addressable and does not have any seek operation. Terminals, line printers, paper tapes, punched cards, network interface, mice (for pointing), and most other devices that are not disk like can be seen as character devices.

This classification scheme is not perfect. Some devices just do not fit in. However, the model of block and character devices is general enough that it can be used as a basis for making the I/O system device independent. A typical microcomputer system consists of a microprocessor plus memory and I/O interface. The various components that form the system are linked through buses that transfer instructions, data, addresses and control information among the components. The block diagram of a microcomputer system is shown in Figure 8.1.

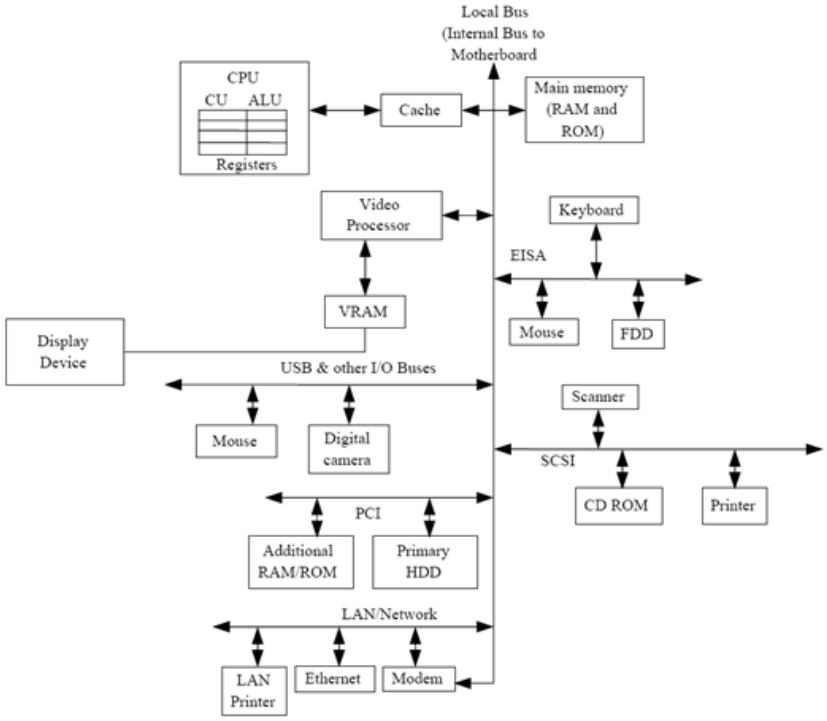


Figure 8.1: Block Diagram of a Microcomputer System.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Draw the block diagram of block devices and character devices.

……..……

……..……

……...

8.2.1 The Input /Output Interface

The Input/output interface provides a method for transferring information between internal storage and external I/O devices. Peripherals connected to a computer need special communication links for interfacing them with the CPU. The purpose of the communication link is to resolve the differences that exist between the central computer and each peripheral.

The major differences are:

- Peripherals are electromagnetic and electromechanical devices and their operations are different from the operation of the CPU and the memory, which are electronic devices.
- The data transfer rate of peripherals is usually slower than the transfer rate of the CPU, and consequently a synchronization mechanism may be needed.
- Data codes and formats in peripherals differ from the word format in the CPU and memory.
- The operating modes of peripherals are different from each other and each must be controlled so as not to disturb the operation of other peripherals connected to the CPU.

8.2.2 The Device Controllers and Its Structure

All the components of the computer communicate with the processor through the system bus. That means the I/O devices need to be attached to the system bus. However, I/O devices are not connected

directly to the computer's system bus. Instead they are connected to an intermediate electronic device interface called a device controller, which in turn is connected to the system bus. Hence a device controller is an interface between an I/O device and the system bus. On one side, it knows how to communicate with the I/O device connected to it, and on the other it knows how to communicate with the computer's CPU or processor and memory through the system bus.

Device Controller

A device controller need not necessarily control a single device. It can usually control multiple I/O devices. It comes in the form of an electronic circuit board that plugs directly into the system bus, and there is a cable from the controller to each device it controls. The cables coming out of the controller are usually terminated at the back panel of the main computer box in the form of connectors known as ports.

The Figure 8.2 illustrates how I/O devices are connected to a computer system through device controllers.

- Each I/O device is linked through a hardware interface called I/O Port.
- Single and Multi-port device controls single or multi-devices.
- The communication between I/O controller and Memory is through bus only in case of Direct Memory Access (DMA), whereas the path passes through the CPU for such communication in case of non-DMA.

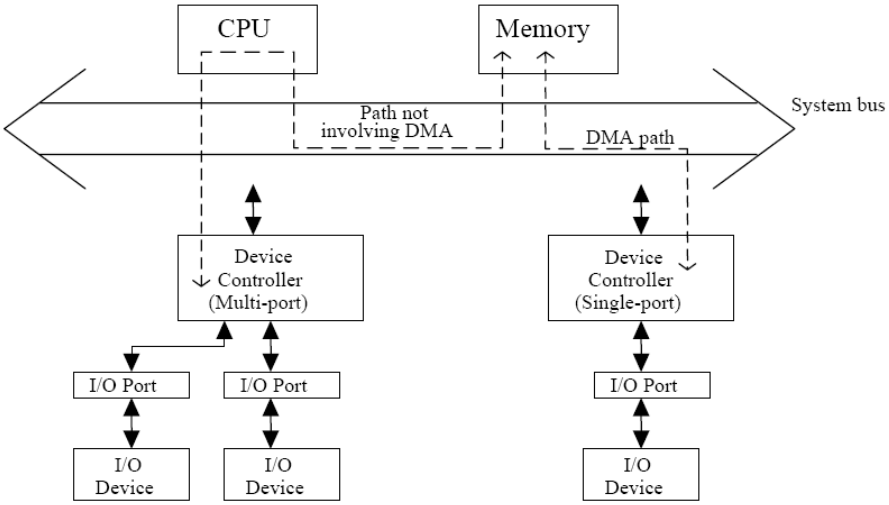


Figure 8.2: Connecting I/O Devices using Device Controller.

Using device controllers for connecting I/O devices to a computer system instead of connecting them directly to the system bus has the following advantages:

- A device controller can be shared among multiple I/O devices allowing many I/O devices to be connected to the system.
- I/O devices can be easily upgraded or changed without any change in the computer system.
- I/O devices of manufacturers other than the computer manufacturer can be easily plugged in to the computer system. This provides more flexibility to the users in buying I/O devices of their choice.

Structure of an I/O Interface

Due to the complexity and the number of external devices that the I/O interface control, there is no standard structure of I/O interface. Let us give a general structure to an I/O interfaces:

- There is a need of I/O logic, which should interpret and execute dialogue between the processor and I/O interface. Therefore, there need to be control lines between processors and I/O interface.
- The data line connecting I/O interface to the system bus must exist. These lines serve the purpose of data transfer.
- Data registers may act as buffer between processor and I/O interface.
- The I/O interface contains logic specific to the interface with each device that it controls.

Caution
Do not attempt to resize a partition on a device that is in use, it may create data lose.

Did You Know?
In 1978, Opto 22 created the first generation of digital I/O modules with plug-in racks for microprocessor-based control.

8.3 Hardware Support for I/O

Computers operate a great many kinds of devices. General types include storage devices (disks, tapes), transmission devices (network cards, modems), and human-interface devices (screen, keyboard, mouse). Other devices are more specialized. Consider the steering of a military fighter jet or of a space shuttle. In these aircraft, a human gives input to the flight computer via a joystick, and the computer sends output commands that cause motors to move rudders, flaps, and thrusters. Despite the incredible variety of I/O device that can be used with a computer, one needs only a few concepts to understand how the devices are attached, and how the software can control the hardware.

A device communicates with a computer system by sending signals over a cable or even through the air. The device communicates with the machine via a connection point termed a port (for example, a serial port). If one or more devices use a common set of wires, the connection is called a bus. In slightly more formal terms, a bus is a set of wires and a rigidly defined protocol that specifies a set of messages that can be sent on the wires. In terms of the electronics, the messages are conveyed by patterns of electrical voltages that are applied to the wires with defined timings. When device A has a cable that plugs into device B, and device B has a cable that plugs into device C, and device C plugs into a port on the computer, this arrangement is called a daisy chain. It usually operates as a bus. Buses are used widely in computer architecture. Figure 8.3 shows a typical Personal Computer (PC) bus structure. This Figure 8.3 shows a Peripheral Component Interconnect (PCI) bus (the common PC system bus) that connects the processor-memory subsystem to the fast devices, and an expansion bus that connects relatively slow devices such as the keyboard and serial and parallel ports. In the upper-right portion of the figure, four disks are connected together on a Small Computer System Interface (SCSI) bus that is plugged into a SCSI controller. A controller is a collection of electronics that can operate a port, a bus, or a device. A serial-port controller is an example of a simple device controller. It is a single chip in the computer that controls the signals on the wires of a serial port. By contrast, a SCSI bus controller is not simple. Because the SCSI protocol is complex, the SCSI bus controller is often implemented as a separate circuit board (a host adapter) that plugs into the computer. It typically contains a processor, microcode, and some private memory to enable it to process the SCSI protocol messages. Some devices have their own built-in controllers. If you look at a disk drive, you will see a circuit board attached to one side. This board is the disk controller. It implements the disk side of the protocol for some kind of connection (SCSI for instance).

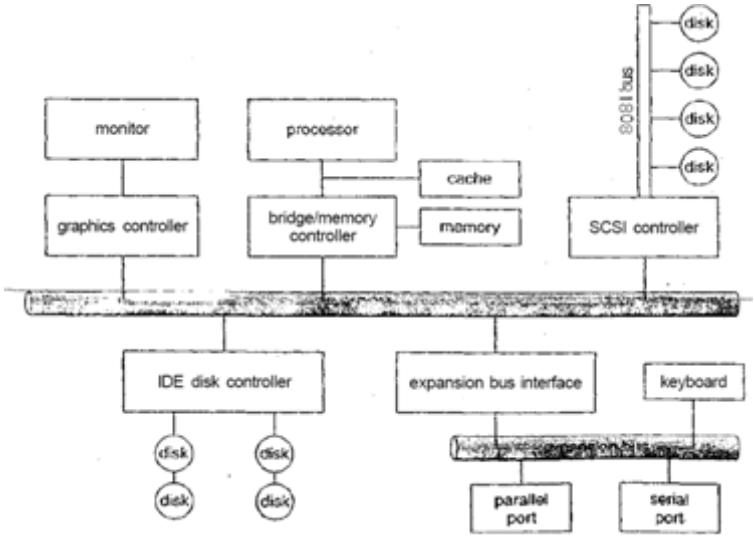


Figure 8.3: **A typical PC bus structure.**

It has microcode and a processor to do many tasks, such as bad-sector mapping, pre fetching, buffering, and caching. Because a common type of software fault is a write through an incorrect pointer to an unintended region of memory, a memory-mapped device register is vulnerable to accidental modification. Of course, protected memory helps to reduce this risk.

I/O address range (hexadecimal)	device
000-00F	DMA controller
020-021	interrupt controller
040-043	timer
200-20F	game controller
2F8-2FF	serial port (secondary)
320-32F	hard disk controller
378-37F	parallel port
3DO-3DF	graphics controller
3FO-3F7	diskette drive controller
3F8-3FF	serial port (primary)

Figure 8.4: Device I/O port locations on PC-compatible computers (partial).

8.3.1 Polling

The complete protocol for interaction between the host and a controller can be intricate, but the basic handshaking notion is simple. One explains handshaking by an example. Assume that 2 bits are used to coordinate the producer consumer relationship between the controller and the host. The controller indicates its state through the busy bit in the status register. (Recall that to set a bit means to write a 1 into the bit, and to dear a bit mean to write a 0 into it.) The controller sets the busy bit when it is busy working, and clears the busy bit when it is ready to accept the next command. The host signals its wishes via the command-ready bit in the command register. The host sets the command-ready bit when a command is available for the controller to execute. For this example, the host writes output through a port, coordinating with the Controller by handshaking as follows.

1. The host repeatedly reads the busy bit until that bit becomes clear.
2. The host sets the write bit in the command register and writes a byte into the data-out register.
3. The host sets the command-ready bit.

4. When the controller notices that the command-ready bit is set, it sets the busy bit.
5. The controller reads the command register and sees the write command. It reads the data-out register to get the byte, and does the I/O to the device.
6. The controller dears the command-ready bit, clears the error bit in the status register to indicate that the device I/O succeeded, and clears the busy bit to indicate that it is finished.

This loop is repeated for each byte. In step 1, the host is busy-waiting or polling: It is in a loop, reading the status register over and over until the busy bit becomes clear. If the controller and device are fast, this method is a reasonable one. But if the wait may be long, the host should probably switch to another task. But then how does the host know when the controller has become idle? For some devices, the host must service the device quickly, or data will be lost. For instance, when data are streaming in on a serial port or from a keyboard, the small buffer on the controller will overflow and data will be lost if the host waits too long before returning to read the bytes.

8.3.2 Interrupts

The basic interrupt mechanism works as follows. The CPU hardware has a wire called the interrupt request line that the CPU senses after executing every instruction. When the CPU detects that a controller has asserted a signal on the interrupt request line, the CPU saves a small amount of state, such as the current value of the instruction pointer, and jumps to the interrupt-handler routine at a fixed address in memory. The interrupt handler determines the cause of the interrupt performs the necessary processing and executes a return from interrupt instruction to return the CPU to the execution state prior to the interrupt. That the device controller raises an interrupt by asserting a signal on the interrupt request line, the CPU catches the interrupt and dispatches to the interrupt handler and the handler clears the interrupt by servicing the device. Figure 8.5 summarizes the interrupt-driven I/O cycle. This basic interrupt mechanism enables the CPU to respond to an asynchronous event, such as a device controller becoming ready for service. In a modern operating system, we need more sophisticated interrupt-handling features. First, we need the ability to defer interrupt handling during critical processing. Second, we need an efficient way to dispatch to the proper interrupt handler for a device, without first polling all the devices to see which one raise can distinguish between high- and low-priority interrupts, and can respond with the appropriate degree of urgency.

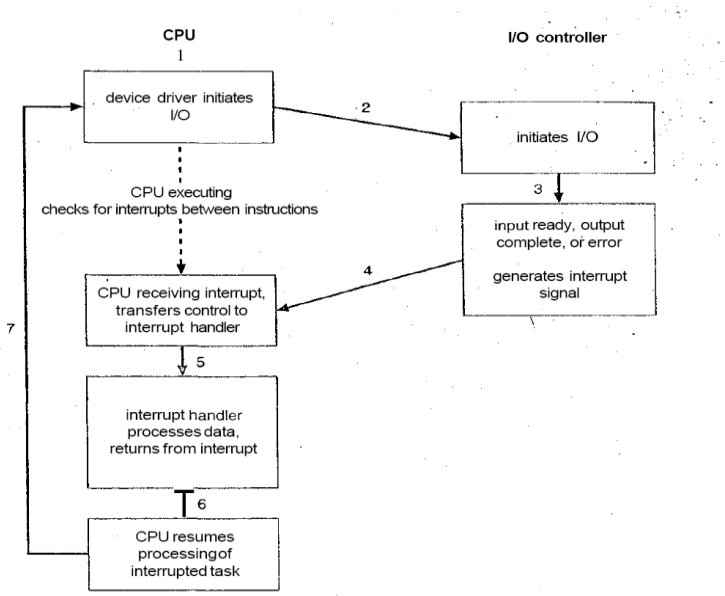


Figure 8.5: Interrupt-driven I/O cycle.

Table 8.1 illustrates the design of the interrupt vector for the Intel Pentium Processor. The first 32 events, which are non-maskable, are used to signal various error conditions. The events from 32 to 255, which are maskable, are used for purposes such as device-generated interrupts.

The interrupt mechanism also implements a system of interrupt priority levels. This mechanism enables the CPU to defer the handling of low-priority interrupts without masking off all interrupts, and makes it possible for a high priority interrupt to pre-empt the execution of a low-priority interrupt.

A modern operating system interacts with the interrupt mechanism in several ways. At boot time, the operating system probes the hardware buses to determine what devices are present, and installs the corresponding interrupt handlers into the interrupt vector

Table 8.1: Intel Pentium processor event vector table

vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detectedoverflow
5	bound range exception
6	invalidopcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

During I/O, interrupts are raised by the various device controllers when they are ready for service. These interrupts unify that output has completed, or that input data are available, or that a failure has been detected. The interrupt mechanism is also used to handle a wide variety of exceptions, such as dividing by zero, accessing a protected or nonexistent memory address, or attempting to execute a privileged instruction from user mode. The events that trigger interrupts have a common property:

They are occurrences that induce the CPU to execute an urgent, self-contained routine. An operating system has other good uses for an efficient hardware mechanism that saves a small amount of processor state, and then calls a privileged routine in the kernel. For example, many operating systems use the interrupt mechanism for virtual-memory paging. A page fault is an exception that raises an interrupt. The interrupt suspends the current process and jumps to the page fault handler in the kernel. This handler saves the state of the process, moves the process to the wait queue, performs page-cache management, schedules an I/O operation to fetch the page, schedules another process to resume execution, and then returns from the interrupt.

Another example is found in the implementation of system calls. A system call is a function that is called by an application to invoke a kernel service.

Did You Know?

SCSI was derived from “SASI”, (Shugart Associates System Interface) developed in 1978 and publicly disclosed in 1981.

8.4 I/O Communication Techniques

Binary information received from an external device is usually stored in memory for later processing. Information transferred from the central computer into an external device originates in the memory unit. Data transfer between the central computer and I/O devices may be handled in a variety of modes. Three techniques are possible for I/O operation.

These are:

- Programmed input/output
- Interrupt driven input/output
- Direct memory access

Figure 8.6 gives an overview of these three techniques

	Interrupt Required	**I/O interface to/from memory transfer (refer Figure 2)**
Programmed I/O	No	Through CPU
Interrupt-driven I/O	Yes	Through CPU
DMA	Yes	Direct to Memory

Figure 8.6: Overview of the three Input/ Output.

In programmed I/O, the I/O operations are completely controlled by the processor. The processor executes a program that initiates, directs and terminate an I/O operation. It requires a little special I/O hardware, but is quite time consuming for the processor since the processor has to wait for slower I/O operations to complete.

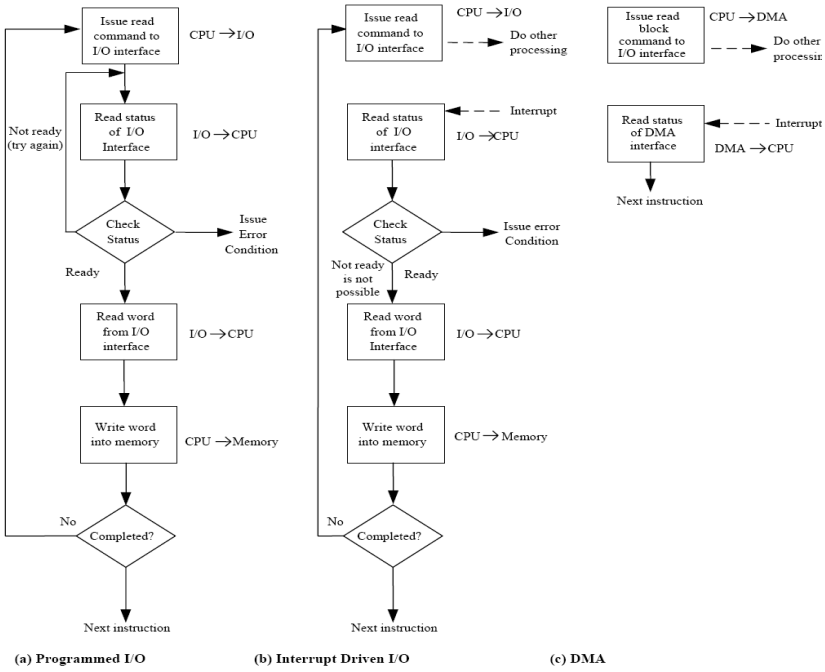


Figure 8.7: Three techniques of I/O.

8.4.1 Programmed Input /Output

Programmed input/output is a useful I/O method for computers where hardware costs need to be minimized.

The input or output operation in such cases may involve:

a) Transfer of data from I/O device to the processor registers.

b) Transfer of data from processor registers to memory.

With the programmed I/O method, the responsibility of the processor is to constantly check the status of the I/O device to check whether it is free or it has finished inputting the data. Thus, this method is very time consuming where the processor wastes a lot of time in checking and verifying the status of an I/O device. An example of the use of programmed I/O to read in a block of data from a peripheral device into memory.

I/O Commands

There are four types of I/O commands that an I/O interface may receive when it is addressed by a processor:

- Control: These commands are device specific and are used to provide specific instructions to the device, e.g. a magnetic tape requiring rewinding and moving forward by a block.
- Test: This command checks the status such as if a device is ready or not or is in error condition.
- Read: This command is useful for input of data from input device.
- Write: this command is used for output of data to output device.

I/O Instructions

An I/O instruction is stored in the memory of the computer and is fetched and executed by the processor producing an I/O-related command for the I/O interface. With programmed I/O, there is a close correspondence between the I/O-related instructions and the I/O commands that the processor issues to an I/O interface to execute the instructions. In systems with programmed I/O, the I/O interface, the main memory and the processors normally share the system bus. Thus, each I/O interface should interpret the address lines to determine if the command is for itself. There are two methods for doing so. These are called memory-mapped I/O and isolated I/O.

With memory-mapped I/O, there is a single address space for memory locations and I/O devices. The processor treats the status and data registers of I/O interface as memory locations and uses the same machine instructions to access both memory and I/O devices. For a memory-mapped I/O only a single read and a single write line are needed for memory or I/O interface read or write operations. These lines are activated by the processor for either memory access or I/O device access. With isolated I/O, there are separate controls lines for both memory and I/O device read or write operations. Thus a memory reference instruction does not affect an I/O device. In isolated I/O, the I/O devices and memory are addressed separately; hence separate input/output instructions are needed which cause data transfer between addressed I/O interface and processor

8.4.2 Interrupt-Driven Input /Output

The problem with programmed I/O is that the processor has to wait a long time for the I/O interface to see whether a device is free or wait till the completion of I/O. The result is that the performance of the processor goes down tremendously. What is the solution? What about the processor going back to do other useful work without waiting for the I/O device to complete or get freed up? But how will the processor be intimated about the completion of I/O or a device is ready for I/O? A well-designed mechanism was conceived for this, which is referred to as interrupt-driven I/O. In this mechanism,

provision of interruption of processor work, once the device has finished the I/O or when it is ready for the I/O, has been provided.

The interrupt-driven I/O mechanism for transferring a block of data is shown in Figure 8.7. Please note that after issuing a read command (for input) the CPU goes off to do other useful work while I/O interface proceeds to read data from the associated device. On the completion of an instruction cycle, the CPU checks for interrupts (which will occur when data is in data register of I/O interface and it now needs CPU's attention). Now CPU saves the important register and processor status of the executing program in a stack and requests the I/O device to provide its data, which is placed on the data bus by the I/O device. After taking the required action with the data, the CPU can go back to the program it was executing before the interrupt.

Interrupt-Processing

The occurrence of an interrupt fires a numbers of events, both in the processor hardware and software.

When an I/O device completes an I/O operation, the following sequence of hardware events occurs:

1. The device issues an interrupt signal to the processor.
2. The processor finishes execution of the current instruction before responding to the interrupt.
3. The processor tests for the interrupts and sends an acknowledgement signal to the device that issued the interrupt.
4. The minimum information required to be stored for the task being currently executed, before the CPU starts executing the interrupt routine (using its registers) are:
    (a) The status of the processor, which is contained in the register called program status word (PSW).
    (b) The location of the next instruction to be executed, of the currently executing program, which is contained in the program counter (PC).
5. The processor now loads the PC with the entry location of the interrupt-handling program that will respond to this interrupting condition. Once the PC has been loaded, the processor proceeds to execute the next instruction that is the next instruction cycle, which begins with an instruction fetch. Because the instruction fetch is determined by the contents of the PC, the result is that control is transferred to the interrupt-handler program.
6. The PC and PSW relating to the interrupted program have already been saved on the system stack. The contents of the processor registers are also needed to be saved on the stack that is used by the called interrupt servicing routine because these registers may be modified by the interrupt-handler. Here a user program is interrupted after the instruction at location N. The contents of all of the registers plus the address of the next instruction (N+1) are pushed on to the stack.
7. The interrupt handler next processes the interrupt. This includes determining of the event that caused the interrupt and also the status information relating to the I/O operation.
8. When interrupt processing is complete, the saved register values are retrieved from the stack and restored to the registers.
9. The final step is to restore the values of PSW and PC from the stack. As a result, the instruction to be executed will be from the previously interrupted program.

Thus, interrupt handling involves interruption of the currently executing program, execution of interrupt servicing program and restart of interrupted program from the point of interruption.

Design issues

Two design issues arise in implementing interrupt-driven I/O:

1) How does the processor determine which device issued the interrupt?

2) If multiple interrupts have occurred, how does the processor decide which one to be processed first?

To solve these problems, four general categories of techniques are in common use:

Multiple Interrupt Lines

The simplest solution to the problems above is to provide multiple interrupt lines, which will result in immediate recognition of the interrupting device. Priorities can be assigned to various interrupts and the interrupt with the highest priority should be selected for service in case a multiple interrupt occurs. But providing multiple interrupt lines is an impractical approach because only a few lines of the system bus can be devoted for the interrupt.

Software Poll

In this scheme, on the occurrence of an interrupt, the processor jumps to an interrupt service program or routine whose job it is to poll (roll call) each I/O interface to determine which I/O interface has caused the interrupt. This may be achieved by reading the status register of the I/O interface. Once the correct interface is identified, the processor branches to a device-service routine specific to that device. The disadvantage of the software poll is that it is time consuming.

Daisy Chain

This scheme provides a hardware poll. With this technique, an interrupt acknowledge line is chained through various interrupt devices. All I/O interfaces share a common interrupt request line. When the processor senses an interrupt, it sends out an interrupt acknowledgement. This signal passes through all the I/O devices until it gets to the requesting device. The first device which has made the interrupt request thus senses the signal and responds by putting in a word which is normally an address of interrupt servicing program or a unique identifier on the data lines. This word is also referred to as interrupt vector. This address or identifier in turn is used for selecting an appropriate interrupt-servicing program. The daisy chaining has an in-built priority scheme, which is determined by the sequence of devices on interrupt acknowledge line.

Bus Arbitration

In this scheme, the I/O interface first needs to control the bus and only after that it can request for an interrupt. In this scheme, since only one of the interfaces can control the bus, therefore only one request can be made at a time. The interrupt request is acknowledged by the CPU on response of which I/O interface places the interrupt vector on the data lines. An interrupt vector normally contains the address of the interrupt serving program. An example of an interrupt vector can be a personal computer, where there are several IRQs (Interrupt request) for a specific type of interrupt.

8.4.3 DMA (Direct Memory Access)

In both interrupt-driven and programmed I/O, the processor is busy with executing input/output instructions and the I/O transfer rate is limited by the speed with which the processor can test and service a device. What about a technique that requires minimal intervention of the CPU for input/output? These two types of drawbacks can be overcome with a more efficient technique known as DMA, which acts as if it has taken over control from the processor. Hence, the question is: why do we use DMA interface? It is used primarily when a large amount of data is to be transferred from the I/O device to the Memory.

DMA Function

Although the CPU intervention in DMA is minimized, yet it must use the path between interfaces that is the system bus. Thus, DMA involves an additional interface on the system bus. A technique called cycle stealing allows the DMA interface to transfer one data word at a time, after which it must return control of the bus to the processor. The processor merely delays its operation for one memory cycle to allow the directly memory I/O transfer to “steal” one memory cycle. When an I/O is requested, the

processor issues a command to the DMA interface by sending to the DMA interface the following information Which operations (read or write) to be performed, using the read or write control lines.

- The address of I/O devices, which is to be used, communicated on the data lines.
- The starting location on the memory where the information will be read or written to be communicated on the data lines and is stored by the DMA interface in its address register.
- The number of words to be read or written is communicated on the data lines and is stored in the data count register.

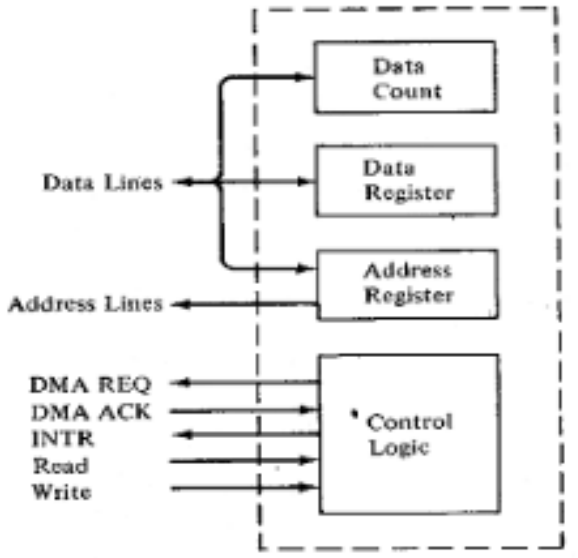


Figure 8.8: DMA block diagram.

The DMA interface transfers the entire block of data, one word at a time, directly to or from memory, without going through the processor. When the transfer is complete, the DMA interface sends an interrupt signal to the processor. Thus, in DMA the processor involvement can be restricted at the beginning and end of the transfer.

The DMA mechanism can be configured into a variety of ways. In which all interfaces share the same system bus. The DMA acts as the supportive processor and can use programmed I/O for exchanging data between memory and I/O interface through DMA interface. But once again this spoils the basic advantage of DMA not using extra cycles for transferring information from memory to/from DMA and DMA from/to I/O interface. The configuration suggests advantages over the one. In these systems a path is provided between I/O interface and DMA interface, which does not include the system bus. The DMA logic may become part of an I/O interface and can control one or more I/O interfaces. In an extended concept an I/O bus can be connected to this DMA interface. Such a configuration is quite flexible and can be extended very easily. In both these configurations, the added advantage is that the data between I/O interface and DMA interface is transferred off the system bus, thus eliminating the disadvantage we have witnessed for the first configuration.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Make a list of I/O of operation in a system.

……..……

……..……

……...

8.5 I/O Software Device Drivers

A device driver is a program routine that links a peripheral device to an operating system of a computer. It is essentially a software program that allows a user to employ a device, such as a printer, monitor, or mouse. It is written by programmers who comprehend the detailed knowledge of the device's command language and characteristics and contains the specific machine language necessary to perform the functions requested by the application. When a new hardware device is added to the computer, such as a CD-ROM drive, a printer, or a sound card, its driver must be installed in order to run it. The operating system "calls" the driver, and the driver "drives" the device. In Windows, for example, everything that is seen on the screen is the result of the display driver (video driver). The display driver effectuates the visual appearance of the screen according to the precise commands that Windows issues to it.

The driver is the link between the operating system and the peripheral device. If the peripheral device is changed, or if a bug is found in the driver, the driver must also be changed. A new version of the driver is then written and released by the manufacturer of the device.

The basic input/output (I/O) hardware features, such as ports, buses, and device controllers, accommodate a wide variety of I/O devices. To encapsulate the details and unique features of different devices, the kernel of an operating system is set up to use device driver modules. The device drivers present a uniform device-access interface to the I/O subsystem.

Each of the different types of I/O devices is accessed through a standardized set of functions-an interface. The tangible differences are encapsulated in kernel modules (i.e., device drivers) that internally are customized for each device, but that export and utilize one of the standard interfaces. A device driver sets the direct memory access (DMA) control registers to use appropriate source and destination addresses, and transfer length. The DMA controller is then instructed to begin the I/O operation. Refer to Figure 8.9 to see how a device driver (a mouse driver in this example) relates to the structure of the operating system.

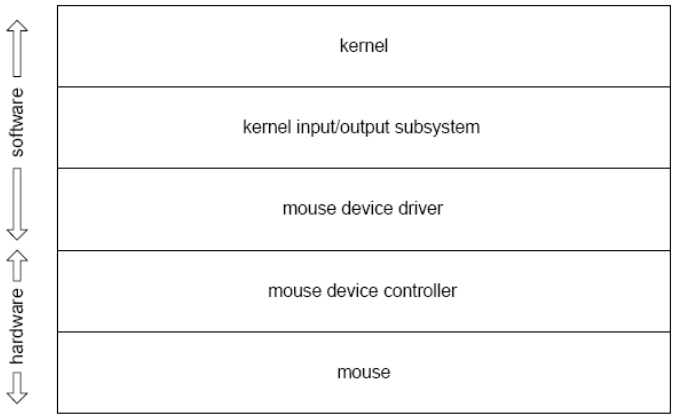


Figure 8.9: Mouse and mouse device driver in a kernel input/output structure of an operating system.

Device drivers are saved as files, and are called upon when a particular peripheral or hardware device is needed. On the Macintosh, for instance, they are stored in the extensions folder like extensions; their features are preset and cannot be modified. Once they are installed, the devices they control become available for use.

To provide an efficient and convenient access to the hard disk, the operating system requires the file system to allow the data to be stored, located, and retrieved easily. The file system is composed of several different levels. The lowest level (see Figure 8.10) is the input/output (I/O) control, and

consists of device drivers and interrupt handlers to transfer information between the memory and the hard disk. A device driver is the basically a translator. It is input consists of high-level commands, and it is output consists of low-level, hardware-specific instructions, which are utilized by the hardware controller, which interfaces the I/O device to the rest of the operating system. The device driver usually writes specific bit patterns to designated locations in the I/O controller's memory to let the controller know on which device location to act and what subsequent actions to provide.

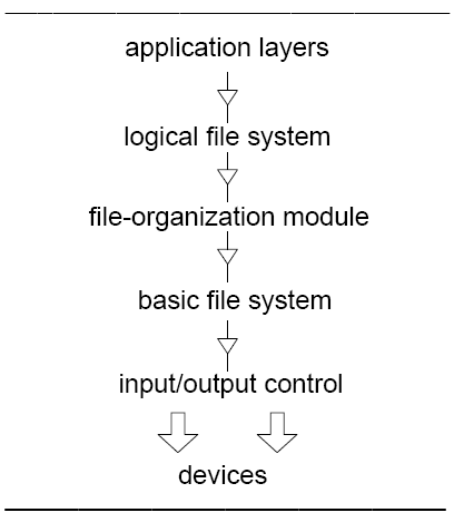


Figure 8.10: Layered file system of a computer.

Caution

There are no user-level file permissions. All file locations specified by the UTL_FILE_DIR parameters are valid, for both reading and writing, for all users of the file I/O procedures. This can override operating system file permissions.

Self Assessment Questions

1. The microcomputer has a……microprocessor, a number of RAM and ROM chips and an interface units communicates with various external devices through the I/O Bus.

(a) multiple (b) single

(c) Both (a) and (b) (d) None of these.

2. The I/O interface provides a method for transferring information between……and external I/O devices.

(a) internal storage (b) external storage

(c) virtual storage (d) cash memory

3. Peripherals are…….and electromechanical devices.

(a) electromagnetic (b) paramagnetic

(c) diamagnetic (d) All of these

4. An I/O interface is bridge between the processor and I/O devices. It controls the data exchange between the external devices and ……..

(a) main memory (b) external device

(c) processor registers (d) All of these

5. Programmed input/output is not a useful I/O method for computers where hardware costs need to be minimized.

(a) True (b) False

6. I/O commands are four types. Like……

(a) Control (b) Read

(c) Write (d) All of these

8.6 Performance Consideration

The Performance Consideration presents a number of considerations regarding implementation of Linux-based solutions on System platform. The information presented in this appendix is not intended to be the definitive word on performance and will not present specific performance characteristics of a given workload. As with anything involving performance, it is recommended that plan on prototyping solutions and be prepared to make changes as you go forward.

The information presented in this appendix can be broken down into three broad categories:

- Virtual I/O support: One of the key strengths to implementing Linux-based solutions on the System platform is the ability to have the I/O (disk) hosted by an OS partition and, thereby, extend the benefits of single-level store to the Linux operating system. We will considerations regarding performance as well as changes that have occurred in recent versions of OS that you should be aware of.
- Virtual network support: Another advantage of implementing Linux-based solutions on the System platform is the ability to build virtual Ethernet networks (LANs) inside the managed system (that is, with no physical hardware). The importance of the frame sizes of the network adapters as well as when it may be advantageous to implement multiple virtual LANs.
- Processor considerations: The System platform supports the ability to share processors between multiple logical partitions (LPARs) and to have the firmware (hypervisor) balance workload across the available processors through the uncapped partition support.

Did You Know?

Intel Corporation started the work on Peripheral Component Interconnect or PCI in 1990. On June 22 1992, PCI 1.0 was introduced in the computer world.

Exercise: Check Your Progress 3

Case Study: I/O-bound Multiuser System

A multiuser system in a company serves approximately 30 employees running a variety of packages including a simple database application (non-relational), an accounting package, and word processing software. At peak usage, there are complaints from users that response is slow and that characters are echoed with a noticeable time delay.

System configuration

The system configuration is:

- Uniprocessor 80486DX2 running at 66MHz.
- EISA bus.
- 32MB of RAM.
- 64MB of swap space.
- NBUF set to 3000.

- Two 1GB SCSI-2 hard disks.
- One 16-port and one 8-port non-intelligent serial card using 16450 UARTs.
- 22 ASCII terminals and PCs running terminal emulation software.
- One V.42 fax modem.

Defining a Performance Goal

The system administrator is tasked with improving the interactive performance of the system. Funds are available for upgrading the machine's subsystems if sufficient need is demonstrated. Any change to the system must be undertaken with minimal disruption to the users.

Collecting Data

The administrator ensures that system accounting is enabled using sar enable(ADM), and produces reports of system activity at five-minute intervals during the working week by placing the following line in root's crontab(C) file:

0 8-18 * * 1-5 /usr/lib/sa/sa2 -s 8:00 -e 18:00 -i 300 -A

=The administrator notes the times at which users report that the system response is slow and examines the corresponding operating system activity in the report (effectively using sar -u):

08:00:00	%usr	%sys	%wio	%idle
11:10:00	42	46	4	8
11:15:00	40	49	6	5
11:20:00	38	50	7	5
11:25:00	41	47	5	7

The system is spending a large amount of time in system mode and little time idle or waiting for I/O.

The length of the run queue shows that an unacceptably large number of user processes are lined up for running (sar -q statistics):

08:00:00	runq-sz	%runocc	swpq-sz	%swpocc
11:10:00	4.3	85		
11:15:00	7.8	98		
11:20:00	5.0	88		
11:25:00	3.5	72		

An acceptable number of processes on the run queue would be two or fewer.

At times when the system response seems acceptable, the system activity has the following pattern:

08:00:00	%usr	%sys	%wio	%idle
16:40:00	55	20	0	25
16:45:00	52	25	2	21
16:50:00	59	20	1	20
16:55:00	54	21	2	23

This shows that the system spends little time waiting for I/O and a large proportion of time in user mode. The %idle figure shows more than 20% spare CPU capacity on the system. The run queue statistics also show that user processes are getting fair access to run on the CPU:

08:00:00	runq-sz	%runocc	swpq-sz	%swpocc
16:40:00	1.0	22		
16:45:00	2.1	18		
16:50:00	1.6	9		
16:55:00	1.1	12		

Formulating a Hypothesis

From the CPU utilization statistics, it looks as though the system is occasionally spending too much time in system mode. This could be caused by memory shortages or too much overhead placed on the CPU by peripheral devices. The low waiting on I/O figures imply that memory shortage is not a

problem. If the system were swapping or paging, this would usually generate much more disk activity.
The administrator next examines the performance of the memory, disk and serial I/O subsystems to check on their performance.

Getting More Specifics

The memory usage figures for the period when the proportion of time spent in system mode (%sys) was high show the following pattern (sar -r statistics):

08:00:00	freemem	freeswp
11:10:00	1570	131072
11:15:00	1612	131072
11:20:00	1598	131072
11:25:00	1598	131072

The value of GPGSHI for this system is 300 and none of the swap space is allocated to processes -- there is no apparent evidence of swapping or paging to disk. This is confirmed by examining the reports for sar -w:

08:00:00	swpin/s	bswin/s	swpot/s	bswot/s	pswch/s
11:10:00	0.04	0.2	0.00	0.0	51
11:15:00	0.02	0.1	0.00	0.0	63
11:20:00	0.00	0.0	0.00	0.0	56
11:25:00	0.01	0.1	0.00	0.0	66

The zero values for swpot/s and bswot/s indicate that there was no swapping out activity.Examining the sar -q, sar -r and sar -w reports at other times shows occasional short periods of paging activity but these are correlated with batch payroll runs. It should be possible to reduce the impact of these on the system by rescheduling the jobs to run overnight.
The administrator next examines the buffer cache usage statistics for the same period (sar -b statistics):

08:00:00	bread/s	lread/s	%rcache	bwrit/s	lwrit/s	%wcache	pread/s	pwrit/s
11:10:00	27	361	93	5	16	68	0	0
11:15:00	35	320	89	7	22	66	0	0
11:20:00	22	275	92	5	15	65	0	0
11:25:00	22	282	96	9	27	67	0	0

These figures show hit rates on the buffer cache of about 90% for reads and 65% for writes. Approximately 30KB of data (bread/s + bwrit/s) is being read from or written to disk per second.
Disk performance is examined next using the statistics provided by sar -d:

08:00:00	device	%busy	avque	r+w/s	blks/s	avwait	avserv
11:10:00	Sdsk-0	0.91	3.70	2.37	13.15	12.42	4.60
	Sdsk-1	25.01	1.62	11.39	55.21	3.26	5.30

11:15:00 Sdsk-0 0.57 2.58 1.37 6.98 13.05 8.26 Sdsk-1 24.10 1.43 10.93 50.42 3.11 7.23
11:20:00 Sdsk-0 0.81 2.42 1.98 11.01 9.55 6.72 Sdsk-1 21.77 1.85 6.05 39.11 4.54 5.37
11:25:00 Sdsk-0 0.76 3.90 2.00 9.52 14.18 4.89 Sdsk-1 20.24 2.07 5.83 34.87 10.60 9.91

These results show that the busiest disk (Sdsk-1) has acceptable performance with a reasonably short request queue, acceptable busy values, and low wait and service times. The pattern of activity on the root disk (Sdsk-0) is such that the request queue is longer since requests are tending to arrive in

bursts. There is no evidence that the system is disk I/O bound though it may be possible to improve the interactive performance of some applications by increasing the buffer cache hit rates.

Making Adjustments to the System

Based on the evidence given above, the system would benefit from increasing the number of buffers in the buffer cache. Although the system does not show much sign of being disk I/O bound (sar -u shows %wio less than 15% at peak load), applications are placing a reasonably heavy demand on the second SCSI disk (Sdsk-1). This will affect the interactive response of programs which have to sleep if the data being requested cannot be found in the buffer cache. As the system does not appear to be short of memory at peak load, the system administrator may wish to experiment with doubling the size of the buffer cache by setting NBUF to 6000. Based on the evidence from sar -r that approximately 6MB (1500 4KB pages) of memory are free at peak load, doubling the size of the buffer cache will reduce this value to about 3MB. If the size of the buffer cache is increased, the system should be monitored to see:

- If the proportion of time spent waiting for I/O decreases (%wio reported by sar -u)
- If the buffer cache hit rates improve (%rcache and %wcache reported by sar -b)
- If disk activity on the second SCSI disk decreases appreciably as a result of increasing the number of buffers (%busy reported by sar -d)
- If the system is becoming short of memory (sar -r reports that freemem is dropping near to or below the value of GPGSHI)

If increasing the size of the buffer cache starts to make the system swap or page out intensively, the administrator should either reduce its size again or make more memory available in other ways.

If the interactive performance of applications is still less than desired, another possibility is to use intelligent serial I/O cards to relieve the processing overhead on the CPU. The serial multiport cards use 16450 UARTs and were previously used in two less powerful systems. It is possible that the CPU is spending too much time moving characters out to the serial lines on behalf of the serial cards. The CPU will do this whenever the applications need to refresh terminal screens to update database forms, word processor displays and so on.

Questions

1. What is the Multiuser System?
2. What is the idle figure in I/O-bound multiuser system?

8.7 Summary

- The Input /Output interface provides a method for transferring information between internal storage and external I/O devices.
- The Input /Output subsystem of a computer, referred to as I/O, provides an efficient mode of communication between the central system and the output environment.
- A device driver is a program routine that links a peripheral device to an operating system of a computer. It is essentially a software program that allows a user to employ a device, such as a printer, monitor, or mouse.
- An I/O instruction is stored in the memory of the computer and is fetched and executed by the processor producing an I/O-related command for the I/O interface.
- Data buffering is quite useful for the purpose of smoothing out the gaps in speed of processor and the I/O devices. The data buffers are registers, which hold the I/O information temporarily.

- The Performance Consideration presents a number of considerations regarding implementation of Linux-based solutions on System platform.

8.8 Keywords

***Block* Device:** It is one that stores information in fixes-size blocks, each one with its own address.
Character Device: It is delivers or accepts a stream of character, without regard to any block structure.
Controller: It is a collection of electronics that can operate a port, a bus, or a device.
Data Buffers: It is refer to registers, which hold the I/O information temporarily.
I/O Interface: It is bridge between the processor and I/O devices.
Page Fault: It is an exception that raises an interrupt.

8.9 Review Questions

1. Discuss about I/O device and device controllers.
2. What are the functions of an I/O interface?
3. What is a device driver?
4. Differentiate between device controller and device drivers.
5. Explain a DMA.
6. Differentiate between virtual I/O and virtual network support.
7. Why are using data buffering in I/O system?
8. Which techniques are used in I/O communication?
9. Define a hardware support for I/O.
10. Describe the one I/O device with example.

Answers for Self Assessment Questions

1. (b) 2 (a) 3 (a) 4 (d) 5 (b) 6 (d)

8.10 Further Studies

1. ***http://web.cs.wpi.edu/~cs3013/a07/week6-io.pdf***
2. Operating System Principles by Silberschatz, Galvin and Gagne

9

Disk Structure

STRUCTURE

9.0 Objectives

9.1 Introduction

9.2 Disk Drives

9.3 Disk Scheduling

9.4 Disk Management

9.5 Disk Reliability

9.6 Swap Space Management

9.7 Stable Storage Implementation

9.8 Summary

9.9 Keywords

9.10 Review Questions

9.11 Further Studies

9.0 Objectives

After studying this chapter, you will be able to:

- Discuss the characteristics of disk drives
- Explain the disk scheduling
- Describe the disk management
- Explain the disk reliability
- Explain the swap space management
- Discuss the stable storage implementation

9.1 Introduction

Modern disk drives are addressed as large one-dimensional arrays of logical blocks, where the logical block is the smallest unit of transfer. The size of a logical block is usually 512 bytes, although some disks can be low-level formatted to have a different logical block size, such as 1,024 bytes. The one-dimensional array of logical blocks is mapped onto the sectors of the disk sequentially. Sector 0 is the first sector of the first track on the outermost cylinder. The mapping proceeds in order through that

track, then through the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost.

By using this mapping, we can—at least in theory—convert a logical block number into an old-style disk address that consists of a cylinder number, a track number within that cylinder, and a sector number within that track. In practice, it is difficult to perform this translation, for two reasons. First, most disks have some defective sectors, but the mapping hides this by substituting spare sectors from elsewhere on the disk. Second, the number of sectors per track is not a constant on some drives.

9.2 Disk Drives

To model disk drives, we must understand how they behave. Thus, we begin with an overview of the current state of the art in no removable magnetic disk drives with embedded SCSI (Small Computer Systems Interconnect) controllers, since these are widely available.

Disk drives contain a mechanism and a controller. The mechanism is made up of the recording components (the rotating disks and the heads that access them) and the positioning components (an arm assembly that moves the heads into the correct position together with a track-following system that keeps it in place). The disk controller contains a microprocessor, some buffer memory, and an interface to the SCSI bus. The controller manages the storage and retrieval of data to and from the mechanism and performs mappings between incoming logical addresses and the physical disk sectors that store the information.

9.2.1 Recording Components

Modern disks range in size from 1.3 to 8 inches in diameter; 2.5, 3.5, and 5.25 inches are the most common sizes today. Smaller disks have less surface area and thus store less data than their larger counterparts; however, they consume less power, can spin faster, and have smaller sought distances. Historically, as storage densities have increased to where 2–3 GB can fit on a single disk, the next-smaller diameter in the series has become the most cost-effective and hence the preferred storage device.

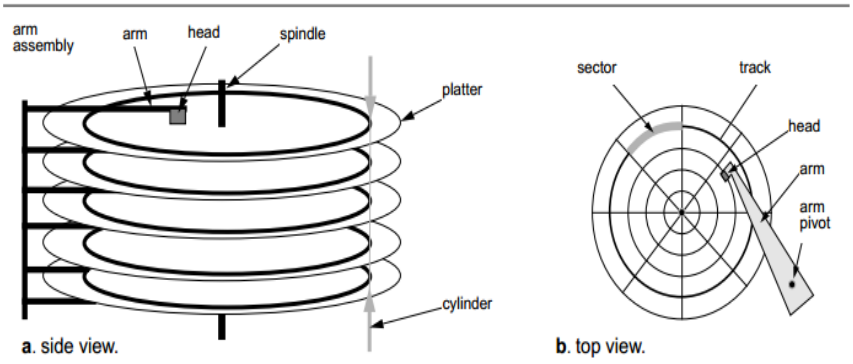


Figure 9.1: The mechanical components of a disk drive.

9.2.2 Positioning Components

Each data surface is set up to store data in a series of concentric circles, or tracks. A single stack of tracks at a common distance from the spindle is called a cylinder. Today's typical 3.5 inch disk has about 2,000 cylinders. As track densities increase, the notion of vertical alignment that is associated with cylinders becomes less and less relevant because track alignment tolerances are simply too fine. Essentially, then, we must consider the tracks on each platter independently.

To access the data stored in a track, the disk head must be moved over it. This is done by attaching each head to a disk arm a lever that is pivoted near one end on a rotation bearing. All the disk arms are attached to the same rotation pivot, so that moving one head causes the others to move as well. The rotation pivot is more immune to linear shocks than the older scheme of mounting the head on a linear slider. The positioning system's task is to ensure that the appropriate head gets to the desired track as quickly as possible and remains there even in the face of external vibration, shocks, and disk flaws (for example, no concentric and noncircular tracks).

Seeking

The speed of head movement, or seeking, is limited by the power available for the pivot motor (halving the seek time requires quadrupling the power) and by the arm's stiffness. Accelerations of 30–40 g are required to achieve good seek times, and too flexible an arm can twist and bring the head into contact with the platter surface. Smaller diameter disks have correspondingly reduced distances for the head to move. These disks have smaller, lighter arms that are easier to stiffen against flexing—all contributing to shorter seek times.

A seeks is composed of:

- A speedup, where the arm is accelerated until it reaches half of the seek distance or a fixed maximum velocity,
- A coast for long seeks, where the arm moves at its maximum velocity,
- A slowdown, where the arm is brought to rest close to the desired track, and
- A settle, where the disk controller adjusts the head to access the desired location.

Very short seeks (less than, say, two to four cylinders) are dominated by the settle time (1–3 ms). In fact, a seek may not even occur; the head may just resettle into position on a new track. Short seeks (less than 200–400 cylinders) spend almost all of their time in the constant acceleration phase, and their time is proportional to the square root of the seek distance plus the settle time. Long seeks spend most of their time moving at a constant speed, taking time that is proportional to distance plus a constant overhead. As disks become smaller and track densities increase, the fraction of the total seek time attributed to the settle phase increases.

"Average" seek times are commonly used as a Figure 9.1 of merit for disk drives, but they can be misleading. Such averages are calculated in various ways, a situation further complicated by the fact that independent seeks are rare in practice. Shorter seeks are much more common, although their overall frequency is very much a function of the workload and the operating system driving the disk. If disk requests are completely independent of one another, the average seek distance will be one third of the full stroke. Thus, some sources quote the one-third-stroke seeks time as the "average". Others simply quote the full-stroke time divided by three. Another way is to sum the times needed to perform one seek of each size and divide this sum by the number of different seek sizes. Perhaps the best of the commonly used techniques is to weight the seek time by the number of possible seeks of each size: Thus, there are N–1 different single-track seeks that can be done on a disk with N cylinders, but only one full-stroke seek. This emphasizes the shorter seeks, providing a somewhat better approximation to measured seek-distance profiles. What matters to people building models, however, is the seek-time-versus-distance profile. We encourage manufacturers to include these in their disk specifications, since the only alternative is to determine them experimentally.

Track Following System

Fine-tuning the head position at the end of seeks and keeping the head on the desired track is the function of the track following system. This system uses positioning information recorded on the disk at manufacturing time to determine whether the disk head is correctly aligned. This information can

be embedded in the target surface or recorded on a separate dedicated surface. The former maximizes capacity, so it is most frequently used in disks with a small number of platters. As track density increases, some form of embedded positioning data becomes essential for fine-grained control—perhaps combined with a dedicated surface for coarse positioning data. However, the embedded-data method alone is not good at coping with shock and vibration because feedback information is only available intermittently between data sectors.

The track following system is also used to perform a head switch. When the controller switches its data channel from one surface to the next in the same cylinder, the new head may need repositioning to accommodate small differences in the alignment of the tracks on the different surfaces. The time taken for such a switch (0.5–1.5 ms) is typically one third to one half of the time taken to do a settle at the end of a seek. Similarly, a track switch (or cylinder switch) occurs when the arm has to be moved from the last track of a cylinder to the first track of the next. This takes about the same time as the end-of-seek settling process. Since settling time increases as track density increases, and the tracks on different platters are becoming less well aligned, head-switching times are approaching those for track switching.

Nowadays, many disk drives use an aggressive, optimistic approach to head settling before a read operation. This means they will attempt a read as soon as the head is near the right track; after all, if the data are unreadable because the settle has not quite completed, nothing has been lost. (There is enough error correction and identification data in a misread sector to ensure that the data are not wrongly interpreted.) On the other hand, if the data are available, it might just save an entire revolution's delay. For obvious reasons, this approach is not taken for a settle that immediately precedes a write. The difference in the settle times for reads and writes can be as much as 0.75 ms.

Data Layout

A SCSI disk appears to its client computer as a linear vector of addressable blocks, each typically 256–1,024 bytes in size. These blocks must be mapped to physical sectors on the disk, which are the fixed-size data-layout units on the platters. Separating the logical and physical views of the disk in this way means that the disk can hide bad sectors and do some low-level performance optimizations, but it complicates the task of higher level software that is trying to second-guess the controller (for example, the 4.2 BSD UNIX fast file system).

Zoning

Tracks are longer at the outside of a platter than at the inside. To maximize storage capacity, linear density should remain near the maximum that the drive can support; thus, the amount of data stored on each track should scale with its length. This is accomplished on many disks by a technique called zoning, where adjacent disk cylinders are grouped into zones. Zones near the outer edge have more sectors per track than zones on the inside. There are typically 3–20 zones, and the number is likely to double by the end of the decade. Since the data transfer rate is proportional to the rate at which the media passes under the head, the outer zones have higher data transfer rates. For example, on a Hewlett-Packard C2240 3.5 inch disk drive, the burst transfer rate (with no inter track head switches) varies from 3.1 MB per second at the inner zone to 5.3 mbps at the outermost zone.

Track Skewing

Faster sequential access across track and cylinder boundaries is obtained by skewing logical sector zero on each track by just the amount of time required to cope with the most likely worst-case head- or track-switch times. This means that data can be read or written at nearly full media speed. Each zone has its own track and cylinder skew factors.

Sparing

It is prohibitively expensive to manufacture perfect surfaces, so disks invariably have some flawed sectors that cannot be used. Flaws are found through extensive testing during manufacturing, and a list is built and recorded on the disk for the controller's use.

So that flawed sectors are not used, references to them are remapped to other portions of the disk. This process, known as sparing, is done at the granularity of single sectors or whole tracks. The simplest technique is to remap a bad sector or track to an alternate location. Alternatively, slip sparing can be used, in which the logical block that would map to the bad sector and the ones after it are "slipped" by one sector or by a whole track. Many combinations of techniques are possible, so disk drive designers must make a complex trade-off involving performance, expected bad-sector rate, and space utilization. A concrete example is the HP C2240 disk drive, which uses both forms of track-level sparing: slip-track sparing at disk format time and single-track remapping for defects discovered during operation.

Did You Know?

The SCSI was derived from "SASI", the "Shugart Associates System Interface", developed c. 1978 and publicly disclosed in 1981.

9.2.3 Disk Controller

The disk controller mediates access to the mechanism, runs the track-following system, transfers data between the disk drive and its client, and, in many cases, and manages an embedded cache. Controllers are built around specially designed microprocessors, which often have digital signal processing capability and special interfaces that let them control hardware directly. The trend is toward more powerful controllers for handling increasingly sophisticated interfaces and for reducing costs by replacing previously dedicated electronic components with firmware.

Interpreting the SCSI requests and performing the appropriate computations takes time. Controller microprocessor speed is increasing just about fast enough to stay ahead of the additional functions the controller is being asked to perform, so controller over head is slowly declining. It is typically in the range 0.3–1.0 ms.

Bus Interface

The most important aspects of a disk drive's host channel are its topology, its transfer rate, and it is overhead. The SCSI is currently defined as a bus, although alternative versions are being discussed, as are encapsulations of the higher levels of the SCSI protocol across other transmission media, such as Fibre Channel. Most disk drives use the SCSI bus operation's synchronous mode, which can run at the maximum bus speed. This was 5 mbps with early SCSI buses; differential drivers and the "fast SCSI" specification increased this to 10 mbps a couple of years ago. Disks are now appearing that can drive the bus at 20 mbps ("fast, wide"), and the standard is defined up to 40 mbps. The maximum bus transfer rate is negotiated between the host computer SCSI interface and the disk drive. It appears likely that some serial channel such as Fibre Channel will become a more popular transmission medium at the higher speeds, partly because it would have fewer wires and requires a smaller connector. Because SCSI is a bus, more than one device can be attached to it. SCSI initially supported up to eight addresses, a figure recently doubled with the use of wide SCSI. As the number of devices on the bus increases, contention for the bus can occur, leading to delays in executing data transfers. This matters more if the disk drives are doing large transfers or if their controller overheads are high. In addition to the time attributed to the transfer rate, the SCSI bus interfaces at the host and disk also require time to establish connections and decipher commands. On SCSI, the cost of the low-level protocol for acquiring control of the bus is on the order of a few microseconds if the bus is idle.

The SCSI protocol also allows a disk drive to disconnect from the bus and reconnect later once it has data to transfer. This cycle may take 200 μs but allows other devices to access the bus while the disconnected device processes data, resulting in a higher overall throughput.

In older channel architectures, there was no buffering in the disk drive itself. As a result, if the disk was ready to transfer data to a host whose interface was not ready, then the disk had to wait an entire revolution for the same data to come under the head again before it could retry the transfer. In SCSI, the disk drive is expected to have a speed-matching buffer to avoid this delay, masking the asynchrony between the bus and the mechanism. Since most SCSI drives take data off the media more slowly than they can send it over the bus, the drive partially fills its buffer before attempting to commence the bus data transfer. The amount of data read into the buffer before the transfer is initiated is called the fence; its size is a property of the disk controller, although it can be specified on modern SCSI disk drives by a control command. Write requests can cause the data transfer to the disk's buffer to overlap the head repositioning, up to the limit permitted by the buffer's size. These interactions are illustrated in Figure 9.2.

Caching of Requests

The functions of the speed-matching buffer in the disk drive can be readily extended to include some form of caching for both reads and writes. Caches in disk drives tend to be relatively small (currently 64 kilobytes to 1 megabyte) because of space limitations and the relatively high cost of the dual ported static RAM needed to keep up with both the disk mechanism and the bus interface.

Read-ahead

A read that hits in the cache can be satisfied "immediately," that is, in just the time needed for the controller to detect the hit and send the data back across the bus. This is usually much quicker than seeking to the data and reading it off the disk, so most modern SCSI disks provide some form of read caching. The most common form is read-ahead—actively retrieving and caching data that the disk expects the host to request momentarily.

As we will show, read caching turns out to be very important when it comes to modelling a disk drive, but it is one of the least well specified areas of disk system behaviour. For example, a read that partially hits in the cache may be partially serviced by the cache (with only the no cached portion being read from disk), or it may simply bypass the cache altogether. Very large read requests may always bypass the cache. Once a block has been read from the cache, some controllers discard it; others keep it in case a subsequent read is directed to the same block.

Some early disk drives with caches did on-arrival read-ahead to minimize rotation latency for whole track transfers; as soon as the head arrived at the relevant track, the drive started reading into its cache. At the end of one revolution, the full track's worth of data had been read, and this could then be sent to the host without waiting for the data after the logical start point to be reread. (This is sometimes—rather unfortunately—called a "zero-latency read" and is also why disk cache memory is often called a track buffer.) As tracks get longer but request sizes do not, on-arrival caching brings less benefit; for example, with 8-Kbyte accesses to a disk with 32 KB tracks, the maximum benefit is only 25% of a rotation time.

On-arrival caching has been largely supplanted by simple read-ahead in 0 which the disk continues to read where the last host request left off. This proves to be optimal for sequential reads and allows them to proceed at the full disk bandwidth. (Without read ahead, two back-to-back reads would be delayed by almost a full revolution because the disk and host processing time for initiating the second read request would be larger than the inter-sector gap.) Even here there is a policy choice: Should the read-ahead be aggressive, crossing track and cylinder boundaries, or should it stop when the end of the track is reached? Aggressive read-ahead is optimal for sequential access, but it degrades random

accesses because head and track switches typically cannot be aborted once initiated, so an unrelated request that arrives while the switch is in progress can be delayed.

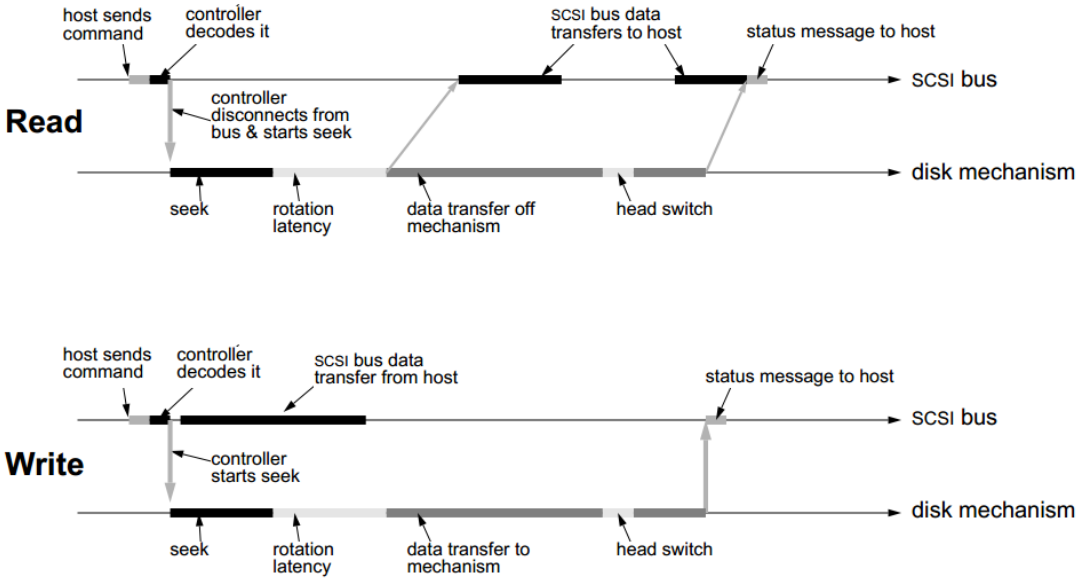


Figure 9.2: Overlap of bus phases and mechanism activity. The low-level details of bus arbitration and selection have been elided for simplicity.

A single read-ahead cache can provide effective support for only a single sequential read stream. If two or more sequential read streams are interleaved, the result is no benefit at all. This can be remedied by segmenting the cache so that several unrelated data items can be cached. For example, a 256 KB cache might be split into eight separate 32KB cache segments by appropriate configuration commands to the disk controller.

Write Caching

In most disk drives, the cache is volatile, losing its contents if power to the drive is lost. To perform write caching and prevent data loss, this kind of cache must be managed carefully. One technique is immediate reporting, which the HP-UX file system uses to allow back-to-back writes for user data. It allows selected writes to the disk to be reported as complete as soon as they are written into the disk's cache. Individual writes can be flagged "must not be immediate-reported"; otherwise, a write is immediately reported if it is the first write since a read or a sequential extension of the last write. This technique optimizes a particularly common case—large writes that the file system has split into consecutive blocks. To protect itself from power failures, the file system disables immediate reporting on writes to metadata describing the disk layout. Combining immediate reporting with read ahead means that sequential data can be written and read from adjacent disk blocks at the disk's full throughput.

Volatile write-cache problems go away if the disk's cache memory can be made non-volatile. One technique is battery-backed RAM, since a lithium cell can provide 10 year retention. Thus equipped, the disk drive is free to accept all the write requests that will fit in its buffer and acknowledge them all immediately. In addition to the reduced latency for write requests, two throughput benefits also result:

(1) Data in a write buffer are often overwritten in place, reducing the amount of data that must be written to the mechanism, and

(2) The large number of stored writes makes it possible for the controller to schedule them in near-optimal fashion, so that each takes less time to perform. These issues are discussed in more detail elsewhere.

As with read caching, there are several possible policies for handling write requests that hit data previously written into the disk's cache. Without non-volatile memory, the safest solution is to delay such writes until the first copy has been written to disk. Data in the write cache must also be scanned for read hits; in this case, the buffered copy must be treated as primary, since the disk may not yet have been written to.

Command Queuing
With SCSI, support for multiple outstanding requests at a time is provided through a mechanism called command queuing. This allows the host to give the disk controller several requests and let the controller determine the best execution order—subject to additional constraints provided by the host, such as "do this one before any of the others you already have." Letting the disk drive perform the sequencing gives it the potential to do a better job by using its detailed knowledge of the disk's rotation position.

9.3 Disk Scheduling

The processor is much faster than the disk, so it is highly likely that there will be multiple outstanding disk requests before the disk is ready to handle them. Because disks have non-uniform access times, re-ordering disk requests can greatly reduce the latency of disk requests. For instance, two requests to the same track are much faster to process in sequence than requests to different tracks. Similarly requests to nearby tracks are faster than requests to distant tracks. A crafty OS will then try to reduce latency and increase disk throughput by reordering disk requests to best utilize the disk. Following are descriptions of several disk scheduling policies. Performance of these algorithms is usually measured against so-called random scheduling where requests are selected randomly from the queue of outstanding requests. To explain these algorithms we are going to use the example of a disk with 200 tracks, and the read/write head starts at track 100. The request queue, in order, contains requests for tracks: 55, 58, 18, 90, 160, 38.

9.3.1 First-in First-out (FIFO)

With a FIFO scheduler, jobs are processed in queue order. If a process is kind and accesses tracks nearby, FIFO may not perform too badly. Even if a process is being nice to the disk scheduler, if there are many running processes, their requests are going to be interleaved, which will mess up the nice process-local ordering.

Track Number	Tracks Traversed
$100 \rightarrow 55$	45
$55 \rightarrow 58$	3
$58 \rightarrow 18$	40
$18 \rightarrow 90$	72
$90 \rightarrow 160$	70
$160 \rightarrow 38$	122
Average	59.67

9.3.2 Shortest Seek Time First (SSTF)

The SSTF stands for shortest seek time first. This algorithm is based on the observation that seek times are lower for nearby tracks. So SSTF picks the request from the queue closest to the current read/write head location. The only tricky part is if there are two jobs with the same distance (one would be towards the spindle, the other towards the edge). In this case, some kind of tie-breaking needs to be employed to pick one. For instance, you could just use a random number to effectively flip a coin and pick one.

Track Number	Tracks Traversed
$100 \rightarrow 90$	10
$90 \rightarrow 58$	32
$58 \rightarrow 55$	3
$55 \rightarrow 38$	17
$38 \rightarrow 18$	20
$18 \rightarrow 160$	142
Average	37.33

As you can see, this is a marked improvement over simple FIFO. However, if a process requests many nearby tracks it can dominate disk activity and greatly increase the latency for other processes (whose tracks are more distantly located). This condition is known as starvation, because one process is preventing the other processes from accessing the disk (starving them from disk access). This may be optimal however; it is not fair to the other processes. Fairness dictates that the latency be as evenly spread among running processes as possible. So other algorithms have been developed to prevent processes with high track locality from starving the other processes.

9.3.3 SCAN/LOOK

Starvation is a bad thing, so OS developers devised a scheduling algorithm based on the elevator algorithm. The SCAN services tracks in only one direction (either increasing or decreasing track number). When SCAN reaches the edge of the disk (or track 0), it reverses direction. The LOOK is the obvious optimization of having the read/write head reversed when the last track in that direction is serviced.

Increasing		Decreasing	
Track Number	Tracks Traversed	Track Number	Tracks Traversed
$100 \rightarrow 160$	60	$100 \rightarrow 90$	10
switch			
$160 \rightarrow 90$	70	$90 \rightarrow 58$	32
$90 \rightarrow 58$	32	$58 \rightarrow 55$	3
$58 \rightarrow 55$	3	$55 \rightarrow 38$	17
$55 \rightarrow 38$	17	$38 \rightarrow 18$	20
		switch	
$30 \rightarrow 18$	20	$18 \rightarrow 160$	142
Average	35.0		37.33

The LOOK behaves almost identically to SSTF, but avoids the starvation problem of SSTF. This is because LOOK is biased against the area recently traversed, and heavily favours tracks clustered at the outermost and innermost edges of the platter. The LOOK is also biased towards more recently arriving jobs (on average).

9.3.4 C-LOOK

The C-LOOK (circular LOOK) is an effort to remove the bias in LOOK for track clusters at the edges of the platter. The C-LOOK basically only scans in one direction. Either you sweep from the inside

out, or the outside in. When you reach the end, you just swing the head all the way back to the beginning. This actually takes advantage of the fact that many drives can move the read/write head at high speeds if you are moving across a large number of tracks (e.g. the seek time from the last track to track 0 is smaller than you would expect and usually considerably less than the time it would take to seek there one track at a time).

Increasing		Decreasing	
Track Number	Tracks Traversed	Track Number	Tracks Traversed
100 → 160	60	100 → 90	10
160 → 18	142	90 → 58	32
18 → 38	20	58 → 55	3
38 → 55	17	55 → 38	17
55 → 58	3	38 → 18	20
58 → 90	32	18 → 160	142
Average	45.67		37.33

9.3.5 N-LOOK and F-LOOK

N and F LOOK were designed to offset LOOK's bias towards recent jobs. Both algorithms partition the request queue into smaller sub queues and process the sub queues in order (oldest first). The N-LOOK is so called because the request queue is divided into N sub queues. F-LOOK is a simplification where there are only 2 queues, but they are used in a double-buffered fashion. While F-LOOK is processing one queue, all new requests go into the other one. For this example, we assume that the request queue is split into two, with the oldest one containing the requests for tracks: 55, 58, 18, 90. In this instance, N-LOOK and F-LOOK behave the same. Also notice, that in this configuration, it does not matter which direction the head was moving in, all requested tracks are less than 100 so it will only move in the direction of decreasing tracks.

Track Number	Tracks Traversed
100 → 90	10
90 → 58	32
58 → 55	3
55 → 18	37
Queue Switch	
Direction Switch	
18 → 38	20
38 → 160	122
Average	37.33

Even through the average number of tracks traversed is the same as LOOK in the worst case, N and F LOOK are in some sense, more fair than plain old LOOK. The sub queues system caps the maximum latency a process can expect between a request and it being serviced (unlike SSTF that can starve processes for arbitrary lengths of time).

Did You Know?

In the early 1990s, two of the biggest names in the PC world, IBM and Microsoft, joined forces to create OS/2, with the goal of making it the "next big thing" in graphical operating systems.

Caution
Formatting the partition may permanently destroy data that currently exists on the partition.

9.4 Disk Management

The operating system is responsible for several other aspects of disk management, too. Here we discuss disk initialization, booting from disk, and bad-block recovery.

9.4.1 Disk Formatting

A new magnetic disk is a blank slate. It is just platters of a magnetic recording material. Before a disk can store data, it must be divided into sectors that the disk controller can read and write. This process is called low-level formatting, or physical formatting. Low-level formatting fills the disk with a special data structure for each sector. The data structure for a sector typically consists of a header, a data area (usually 512 bytes in size), and a trailer. The header and trailer contain information used by the disk controller, such as a sector number and an error-correcting code (ECC). When the controller writes a sector of data during normal I/O, the ECC is updated with a value calculated from all the bytes in the data area. When the sector is read, the ECC is recalculated and is compared with the stored value. If the stored and calculated numbers are different, this mismatch indicates that the data area of the sector has become corrupted and that the disk sector may be bad. The ECC is an error-correcting code because it contains enough information that if only 1 or 2 bits of data have been corrupted, the controller can identify which bits have changed, and can calculate what their correct values should be. The ECC processing is done automatically by the controller whenever a sector is read or written.

Most hard disks are low-level formatted at the factory as a part, of the manufacturing process. This formatting enables the manufacturer to test the disk, and to initialize the mapping from logical block numbers to defect-free sectors on the disk. For many hard disks, when the disk controller is instructed to low-level format the disk, it can also be told how many bytes of data space to leave between the header and trailer of all sectors. It is usually possible to choose among a few sizes, such as 256, 512, and 1024 bytes. Formatting a disk with a larger sector size means that fewer sectors can fit on each track, but that also means fewer headers and trailers are written on each track, and thus increases the space available for user data. Some operating systems can handle only a sector size of 512 bytes.

To use a disk to hold files, the operating system still needs to record its own data structures on the disk. It does so in two steps. The first step is to partition the disk into one or more groups of cylinders. The operating system can treat each partition as though the latter were a separate disk.

9.4.2 Boot Block

For a computer to start running—for instance, when it is powered up or rebooted—it needs to have an initial program to run. This initial bootstrap program tends to be simple. It initializes all aspects of the system, from CPU registers to device controllers and the contents of main memory, and then starts the operating system. To do its job, it finds the operating system kernel on disk, loads that kernel into memory, and jumps to an initial address to begin the operating-system execution.

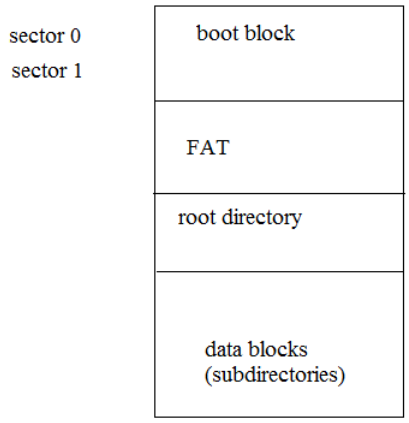


Figure 9.3: MS-DOS disk layout.

For most computers, the bootstrap is stored in read-only memory (ROM). This location is convenient, because ROM needs no initialization, and is at a fixed location that the processor can start executing when powered up or reset. And, since ROM is read only, it cannot be infected by a computer virus. The problem is that changing this bootstrap code requires changing the ROM hardware chips. For this reason, most systems store a tiny bootstrap loader program in the boot ROM, whose only job is to bring in a full bootstrap program from disk. The full bootstrap program can be changed easily: A new version is simply written onto the disk. The full bootstrap program is stored in a partition called the boot blocks, at a fixed location on the disk. A disk that has a boot partition is called a boot disk or system disk.

The code in the boot ROM instructs the disk controller to read the boot blocks into memory (no device drivers are loaded at this point), and then starts executing that code. The full bootstrap program is more sophisticated than the bootstrap loader in the boot ROM, and is able to load the entire operating system from a non-fixed location on disk, and to start the operating system running. Even so, the full bootstrap code may be small. For example, MS-DOS uses one 512 byte block for its boot program (see Figure 9.3).

9.4.3 Bad Blocks

Because disks have moving parts and small tolerances (recall that the disk head flies just above the disk surface), they are prone to failure. Sometimes, the failure is complete, and the disk needs to be replaced, and its contents restored from backup media to the new disk. More frequently, one or more sectors become defective. Most disks even come from the factory with bad blocks. Depending on the disk and controller in use, these blocks are handled in a variety of ways.

More sophisticated disks, such as the SCSI disks used in high-end PCs and most workstations, are smarter about bad-block recovery. The controller maintains a list of bad blocks on the disk. The list is initialized during the low-level format at the factory, and is updated over the life of the disk. Low-level formatting also sets aside spare sectors not visible to the operating system. The controller can be told to replace each bad sector logically with one of the spare sectors. This scheme is known as sector sparing or forwarding.

A typical bad-sector transaction might be as follows:

The operating system tries to read logical block 87.

- The controller calculates the ECC and finds that the sector is bad. It reports this finding to the operating system.

- The next time that the system is rebooted, a special command is run to tell the SCSI controller to replace the bad sector with a spare.
- After that, whenever the system requests logical block 87, the request is translated into the replacement sector's address by the controller.

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: Create the partitions with disk administrator and disk management.

……..……

……..……

……...

9.5 Disk Reliability

Disks used to be the least reliable component of a system. They still have relatively high failure rates, and their failure causes a loss of data and significant downtime while the disk is replaced and the data restored. Recovering from a disk crash may take hours, as backup copies of the lost data are found on tape and restored to the replacement disk. Under normal circumstances, the restored disk is not a precise image of the disk that crashed, because any changes to the data since the last backup are lost. Therefore, improving the reliability of disk systems is an important research topic.

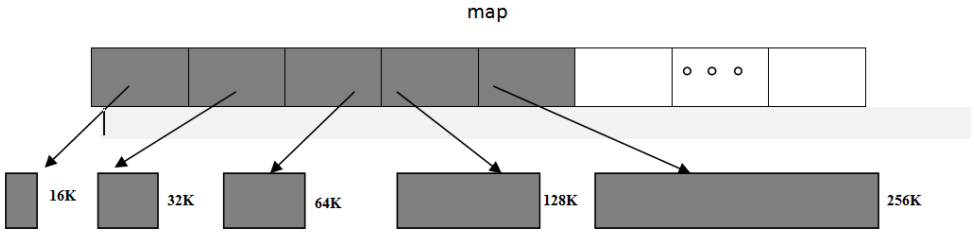


Figure 9.4: 4.3 BSD data-segment swap map.

Several improvements in disk-use techniques have been proposed. These methods involve the use of multiple disks working cooperatively. To improve speed, disk striping (or interleaving) uses a group of disks as one storage unit. Each data block is broken into several sub blocks, with one sub block stored on each disk. The time required to transfer a block into memory improves dramatically, because all the disks transfer their sub blocks in parallel. If the disks have their rotations synchronized, the performance improves further, because all the disks become ready to transfer their sub blocks at the same time/rather than waiting for the slowest rotational latency. The larger the number of disks that are striped together, the larger the total transfer rate of the system.

Did You Know?
The term RAID was first defined by David A. Patterson, Garth A. Gibson and Randy Katz at the University of California, Berkeley, in 1987.

Caution
A partition cannot be made larger than the space available on the device.

Self Assessment Questions

1. The application of digital signal processing may soon increase channel speeds their current.....per second.

(a) 90 MB (b) 100 MB
(c) 110 MB (d) 120 MB

2. The speed of head movement or..... is limited by the power available for the pivot motor and by the arm's stiffness.

(a) sparing (b) track skewing
(c) zoning (d) seeking

3..... is prohibitively expensive to manufacture perfect surfaces, so disks invariably have some flawed sectors that cannot be used.

(a) Sparing (b) Track skewing
(c) Zoning (d) Seeking

4..... algorithm is based on the observation that seek times are lower for nearby tracks.

(a) FIFO (b) SSTF
(c) SCAN (d) LOOK

9.6 Swap Space Management

Swap space management is another low-level task of the operating system. Virtual memory uses disk space as an extension of main memory. Since disk access is much slower than memory access, using swap space has a large effect on system performance. The main goal for the design and implementation of swap space is to provide the best throughput for the virtual memory system. How swap space is used, where swap space is located on disk, and how swap space is managed.

9.6.1 Swap Space Use

Swap space is used in various ways by different operating systems, depending on the implemented memory management algorithms. For instance, systems that implement swapping may use swap space to hold the entire process image, including the code and data segments. Paging systems may simply store pages that have been pushed out of main memory. The amount of swap space needed on a system can therefore vary depending on the amount of physical memory, the amount of virtual memory it is backing, and the way in which the virtual memory is used. It can range from a few megabytes of disk space to hundreds of megabytes or more. Some operating systems, such as UNIX, allow the use of multiple swap spaces. These swap spaces are usually put on separate disks, so the load placed on the I/O system by paging and swapping can be spread over the system's I/O devices.

Note that it is safer to overestimate than to underestimate swap space, because if a system runs out of swap space it may be forced to abort processes or may crash entirely. Overestimation results in wasted disk space that could otherwise be used for files/but does no other harm.

9.6.2 Swap Space Location

There are two places that a swap space can reside: Swap space can be carved out of the normal file system, or it can be in a separate disk partition. If the swap space is simply a large file within the file system, normal file-system routines can be used to create it, to name it, and to allocate its space. This approach is therefore easy to implement. Unfortunately, it is also inefficient. Navigating the directory

structure and the disk allocation data structures takes time, and (potentially) extra disk accesses. External fragmentation can greatly increase swapping times by forcing multiple seeks during reading or writing of a process image. We can improve performance by caching the block location information in physical memory, and by using special tools to allocate physically contiguous blocks for the swap file, but the cost of traversing the file system data structures still remains.

9.6.3 Swap Space Management

To illustrate the methods used to manage swap space, we now follow the evolution of swapping and paging in UNIX. The UNIX started with an implementation of swapping that copied entire processes between contiguous disk regions and memory. The UNIX evolved to a combination of swapping and paging as paging hardware became available.

In 4.3BSD, swap space is allocated to a process when the process is started. Enough space is set aside to hold the program, known as the text pages or the text segment, and the data segment of the process. Pre-allocating all the needed space in this way generally prevents a process from running out of swap space while it executes. When a process starts, its text is paged in from the file system. These pages are written out to swap when necessary, and are read back in from there, so the file system is consulted only once for each text page. Pages from the data segment are read in from the file system or, are created (if they are uninitialized), and are written to swap space and paged back in as needed. One optimization (for instance, when two users run the same editor) is that processes with identical text pages share these pages, both in physical memory and in swap space.

Two per-process swap maps are used by the kernel to track swap space use. The text segment is a fixed size, so its swap space is allocated in 512 K chunks, except for the final chunk, which holds the remainder of the pages; in 1K increments (see Figure 9.5). The data-segment swap map is more complicated, because the data segment can grow over time. The map is of fixed size, but contains swap addresses for blocks of varying size.

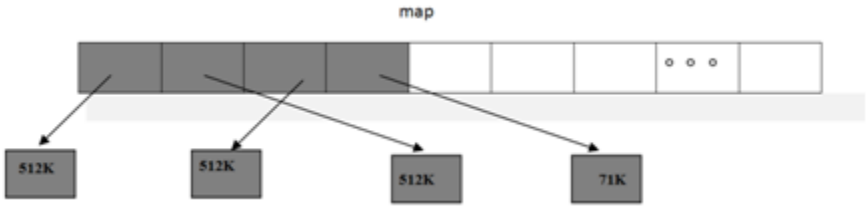


Figure 9.5: 4.3 BSD text segment swap map.

Given index i, a block pointed to by swap map entry i is of size 2^i x 16 K, to a maximum of 2 MB. This data structure is shown in Figure 9.4 (The block size minimum and maximum are variable, and can be changed at system reboot.) When a process tries to grow its data segment beyond the final allocated block in its swap area, the operating system allocates another block, twice as large as the previous one. This scheme results in small processes using only small blocks. It also minimizes fragmentation. The blocks of large processes can be found quickly, and the swap map remains small.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: List the advantages/features of hardware level RAID's.

……..……

……..……

……...

9.7 Stable Storage Implementation

The concept of a write-ahead log, which required the availability of stable storage. By definition, information residing in stable. Storage is never lost. To implement such storage, we need to replicate the needed information on multiple storage devices with independent failure modes. We need to coordinate the writing of updates in a way that guarantees that a failure during an update does not leave all the copies in a damaged state, and that, when we are recovering from a failure, we can force all copies to a consistent and correct value, even if there is another failure during the recovery.

A disk writes results in one of three outcomes:

- Successful completion: The data were written correctly on disk.
- Partial failure: A failure occurred in the midst of transfer, so only some of the sectors were written with the new data, and the sector being written during the failure may have been corrupted.
- Total failure: The failure occurred before the disk write started, so the previous data values on the disk remain intact.

An output operation is executed as follows:

1. Write the information onto the first physical block.
2. When the first write completes successfully, write the same information onto the second physical block.
3. Declare the operation complete only after the second write completes successfully.

Exercise: Check Your Progress 3

Case Study-Largest Regional Carrier Soars to New Heights with Flexible, Reliable Disk- and Tape-Based Storage Solutions

At SkyWest Airlines, a commitment to quality service with a competitive cost structure fuels continuing business growth in a highly competitive and demanding industry. A wholly owned subsidiary of SkyWest Inc., the airline focuses on proven best practices and the latest IT solutions to streamline and automate operations. With its recent acquisition of Atlantic Southeast Airlines (ASA), the St. George, Utah-based company has become the largest regional airline carrier network in the United States, serving more than 250 cities in North America, Mexico and the Caribbean.

With more than 14,000 employees and almost 400 aircraft, the airline provides some 2,400 daily departures for 25 million passengers annually. According to Josh Ursenbach, manager of systems engineering, SkyWest's ability to access, manage and protect critical data is key to keeping back-office, maintenance and flight operations running smoothly and efficiently. "We continually invest in and utilize leading-edge technologies to increase automation and boost overall productivity," he says.

To that end, the four-person systems engineering team oversees 100 Microsoft Windows 2000/2003 servers and 1,400 PCs while ensuring users with around-the-clock access to a host of mission-critical applications and databases. The team also safeguards nearly 2 TBs of data stored on an EMC Clariion CX400 Storage Area Network (SAN).

The Solution

In seeking an affordable system that could grow to meet SkyWest's increasing data backup and recovery requirements, Ursenbach consulted Scott Cooper, National Sales Manager at I/O Dynamics, who suggested the NEO tape library family from Overland Storage. The library's modular design and expansion-on demand capabilities seemed ideal for SkyWest's needs while providing flexibility to increase capacity as needed.

Each NEO 2000 module in the expandable chassis supports up to 30 LTO cartridge slots and up to two tape drives, providing up to 24 TBs of storage. "We liked how easily NEO interfaced with Fibre Channel, SCSI and iSCSI environments for complete configuration flexibility," said Ursenbach.

In early 2005, SkyWest deployed its first NEO 2000 and took advantage of the straightforward operation and additional capacity to migrate toward weekly full backups. The switch to LTO tape technology proved fortuitous as well, allowing SkyWest to reduce its weekly tape load from 12 to three tapes. Automating the process also enabled moving the library offsite to strengthen the airline's disaster recovery planning.

Next, the team decided to leverage disk-to-disk backup and virtual tape library (VTL) capabilities, prompting a review of offerings from Overland, StorageTek and Exabyte. Ursenbach was impressed with the diversity of Overland's REO 9000, a disk-based backup and recovery appliance that operated as a virtual tape library and standalone virtual tape drives. The simplicity of creating disk volumes (LUNs) for accommodating interim storage requirements-only available on the REO-also was a definite plus.

As a result, Ursenbach says Overland's REO was a "hands-down winner," so in late 2005, SkyWest purchased a REO 9000 with up to 44 TB of raw capacity.

Questions

1. What is advantage of REO's pre-built RAID arrays?
2. What are the challenges of SkyWest's increasing data?

9.8 Summary

- The track following system is used to perform a head switch.
- The maximum bus transfer rate is negotiated between the host computer SCSI interface and the disk drive.
- The LOOK is the obvious optimization of having the read/write head reversed when the last track in that direction is serviced.
- The C-LOOK (circular LOOK) is an effort to remove the bias in LOOK for track clusters at the edges of the platter.
- The operating system is responsible for several other aspects of disk management.
- The ECC processing is done automatically by the controller whenever a sector is read or written.
- A small fraction of the disk space is used to hold parity blocks.
- Swap space is used in various ways by different operating systems, depending on the implemented memory management algorithms.

9.9 Keywords

Disk Drives: A disk drive is a device implementing such a storage mechanism with fixed or removable media; with removable media the device is usually distinguished from the media as in compact disc drive and the compact disc.

Error-correcting Code (ECC): ECC is used to verify data transmissions by locating and correcting transmission errors. It is commonly used by RAM chips that include forward error correction, which ensures all the data being sent to and from the RAM is transmitted correctly.

Redundant Array of Independent Disks (RAID): Redundant array of independent disks is a storage technology that combines multiple disk drive components into a logical unit.

Shortest Seek Time First (SSTF): Shortest seek first (or shortest seek time first) is a secondary storage scheduling algorithm to determine the motion of the disk's arm and head in servicing read and write requests.

Small Computer System Interface (SCSI): Small computer system interface is a set of standards for physically connecting and transferring data between computers and peripheral devices.

9.10 Review Questions

1. What is disk drive?
2. What are characteristics of modern disk drives?
3. Discuss the mechanical components of a disk drive.
4. What is disk scheduling?
5. Describe the disk management.
6. What is the disk reliability?
7. What is use of swap space?
8. What is the swap space management?
9. How to stable storage implementation in disk?
10. Discuss the RAID hardware and software.

Answers for Self Assessment Questions

1 (b) 2 (d) 3 (a) 4 (b)

9.11 Further Studies

1. ***http://codex.cs.yale.edu/avi/os-book/OS8/os8c/slide-dir/PDF-dir/ch12.pdf***
2. Operating System Principles by Silberschatz, Galvin and Gagne

10

File Management

STRUCTURE

10.0 Objectives

10.1 Introduction

10.2 File Concepts

10.3 Directory Structure

10.4 File Sharing

10.5 Protection of File System

10.6 File System in Linux

10.7 Summary

10.8 Keywords

10.9 Review Questions

10.10 Further Studies

10.0 Objectives

After studying this chapter, you will be able to:

- Explain the concepts of file
- Discuss the directory structure
- Define the file sharing
- Explain the protection of file system
- Discuss the file system in Linux

10.1 Introduction

The file management, formerly known as data management, is the part of the operating system that controls the storing and accessing of data by an application program. The data may be on internal storage (for example, database), on external media (diskette, tape, printer), or on another system. File management, then, provides the functions that an application uses in creating and accessing data on the system and ensures the integrity of the data according to the definitions of the application. File management provides functions that allow you to manage files (create, change, override, or delete) using CL commands, and create and access data through a set of operations (for example, read, write, open, or close). File management also provides with the capability to access external devices and control the use of their attributes for creating and accessing data. If you want to make more efficient

use of printers and diskette devices, file management provides the capability of spooling data for input or output. For example, data being written to a printer can be held on an output queue until the printer is available for printing. On the IBM AS/400 system, each file (also called a file object) has a description that describes the file characteristics and how the data associated with the file is organized into records, and, in many cases, the fields in the records. Whenever a file is processed, the operating system (the Operating System/400 or OS/400 program) uses this description. You can create and access data on the system by using these file objects. File management defines and controls several different types of files. Each file type has associated CL commands to create and change the file, and you can also create and access data through the operations provided by file management.

10.2 File Concepts

From a beginner's point of view, a file could be defined as contiguous set of words (bits, bytes or instructions). More appropriately, a file is collection of data. We can store different kinds of information in a file; for example source codes, images, text documents, and so forth. Files are of different types depending on the information stored in a file and therefore each file has its own structure which is type dependent. Everything on a computer is stored as a file. It is the job of the operating system to map this sequence of words (i.e. files) into physical devices. This gives us another alternative definition of a file system - The part of operating system that maps various files to the physical devices is called as file management system. It is clear that the main objective of the file systems is to free the users of the details of storing the information in the physical devices.

10.2.1 Files Attributes

The particular information kept for each file varies from operating system to operating system. No matter what operating system one might be using, files always have certain attributes or characteristics. Different file attributes are discussed in this section of the report.

File name

The symbolic file name is the only information kept in human-read form. As it is obvious, a file name helps users to differentiate between various files. A file name generally consists of a string of characters. The string of characters prior to the "." is called as filename and the part after it is called the file extension (or file type) that differentiates between different types of files. We can have files with same names but different extensions and therefore we generally refer to a file with its name along with its extension and that forms a complete file name.

File type

A file type is required for the systems that support different types of files. As discussed earlier, file type is a part of the complete file name. We might have two different files; say "cs384report.doc" and "cs384report.txt". Therefore the file type is an important attribute which helps differentiating between files based on their types. File types indicate which application should be used to open a particular file.

Location

This is a pointer to the device and location on that device of the file. As it is clear from the attribute name, it specifies where the file is stored.

Size

Size attribute keeps track of the current size of a file in bytes, words or blocks.

Protection

Protection attribute of a file keeps track of the access-control information that controls who can do reading, writing, executing, and so on.

Usage count

This value indicates the number of processes that are currently using (have opened) a particular file.

Time, Date, and Process Identification

This information may be kept for creation, last modification, and last use. Data provided by this attribute is often helpful for protection and usage monitoring. Each process has its own identification number which contains information about file hierarchy.

10.2.2 File Operations

The operating system provides systems calls to create, write, read, reposition, truncate and delete files. The following discusses the specific duties a file system must do for each of the six basic file operations.

Creating a file

When creating a file, a space in the file system must be found for the file and then an entry for the new file must be made in the directory. The directory entry records the name of the file and the location in the file system.

Writing a file

To write a file, a system call is made specifying both the name and the file and the information to be written to the file. Given the name of the file, the system searches the directory to find the location of the file. The directory entry will need to store a pointer to the current block of the file (usually the beginning of the file). Using this pointer, the address of the next block can be computed where the information will be written. It is also important to make sure that the file is not overwritten in case of an append operation, i.e. when we are adding a block of data at the end of an already existing file.

Reading a file

To read a file, a system call is made that specifies the name of the file and where (in memory) the next block of the file should be put. Again, the directory is searched for the associated directory entry, and the directory will need a pointer to the next block to be read. Once the block is read, the pointer is updated.

Repositioning a file

When repositioning a file, the directory is searched for the appropriate entry, and the current file position is set to a given value. This file operation is also called as file seeks.

Truncating a file

The user may erase some contents of a file but keep its attributes. Rather than forcing the user to delete the file and then recreate it, this operation allows the all the attributes to remain unchanged, except the file size.

Deleting a file

To delete a file, the directory is searched for the named file. Having found the associated directory entry, the space allocated to the file is released (so it can be reused by other files) and invalidates the directory entry.

The six operations described comprise only the minimal set of required file operations. More commonly, we might also want to edit the file and modify its contents. A special case of editing a file is appending new information at the end of the file. Copies of the file can also be created, and since files are named objects, renaming an existing file may also be needed. If the file is a binary object format, we may also want to execute it. Also of use are facilities to lock sections of an open file for

multiprogramming access, to share sections, and even to map sections into memory or virtual-memory systems. This last function allows a part of the virtual address to be logically associated with section of a file. Reads and writes to that memory region are then treated as reads and writes to the file.

10.2.3 File Types

The file types (or extensions) are. File type is not only used to differentiate between two files having same file names but also to inform the operating system as to which application program should be used to perform operations on a particular file. There are many different file types supported by modern day operating systems. Some of the common file types (extensions) and the programs used to perform various operations in see Figure 10.1.

File Type	Usual Extension	Function
Executable	exe, com, bin	Ready to run the machine-language program
Object	obj, o	Compiled, machine language, not linked
Source Code	c, cc, java, pas asm, a	Source code in various languages
Batch	bat, sh	Commands to the command interpreter
Text	txt, doc	Textual data, documents
Word Processor	wp, tex, rtf, doc	Various word-processor formats
Library	lib, a, so, dll	Libraries of routines for programmers
Print or View	ps, pdf, jpg	ASCII or binary files in a format for printing or viewing
Archive	arc, zip, tar	Related files grouped into one file, sometimes compressed
Multumedia	mpeg, mov, rm	Binary file containing audio or A/V information

Figure 10.1: Common file types.

Caution

The write pointer must be updated ensuring successive writes that used to write a sequence of blocks to the file.

10.3 Directory Structure

The structure of the directories and the relationship among them are the main areas where file systems tend to differ, and it is also the area that has the most significant effect on the user interface provided by the file system.

The most common directory structures used by multi-user systems are:

- Single-level directory
- Two-level directory
- Tree-structured directory
- Acyclic graph directory

Single-level directory

In a single-level directory system, all the files are placed in one directory. This is very common on single-user operating systems. A single-level directory has significant limitations when the number of files increases or when there is more than one user. Since all files are in the same directory, they must have unique names. If there are two users who call their data file then the unique-name rule is violated. Even with a single user, as the number of files increases, it becomes difficult to remember the names of all the files in order to create only files with unique names. Figure 10.2 below shows the structure of a single-level directory system.

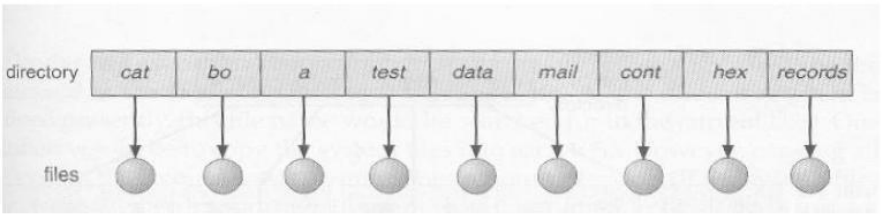


Figure 10.2: Single level directory.

Two-level directory

In the two-level directory system, the system maintains a master block that has one entry for each user. This master block contains the addresses of the directory of the users. There are still problems with two-level directory structure. This structure effectively isolates one user from another. This is an advantage when the users are completely independent, but a disadvantage when the users want to cooperate on some task and access files of other users. Some systems simply do not allow local files to be accessed by other users. (see Figure 10.3).

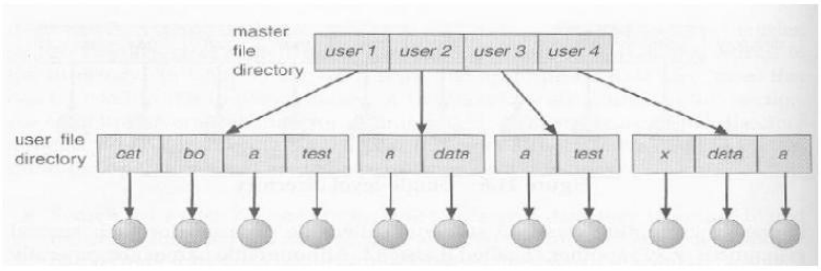


Figure 10.3: Two-level directory.

Tree-structured Directories

In the tree-structured directory, the directory themselves are considered as files. This leads to the possibility of having sub-directories that can contain files and subdirectories. An interesting policy decision in a tree-structured directory structure is how to handle the deletion of a directory. If a

directory is empty, its entry in its containing directory can simply be deleted. However, suppose the directory to be deleted is not empty, but contains several files or sub-directories then it becomes a bit problematic. Some systems will not delete a directory unless it is empty. Thus, to delete a directory, someone must first delete all the files in that directory. If these are any subdirectories, this procedure must be applied recursively to them so that they can be deleted too. This approach may result in a substantial amount of work. An alternative approach is just to assume that when a request is made to delete a directory, all of that directory's files and sub directories are also to be deleted. A typical tree-structured directory system is shown in Figure 10.4.

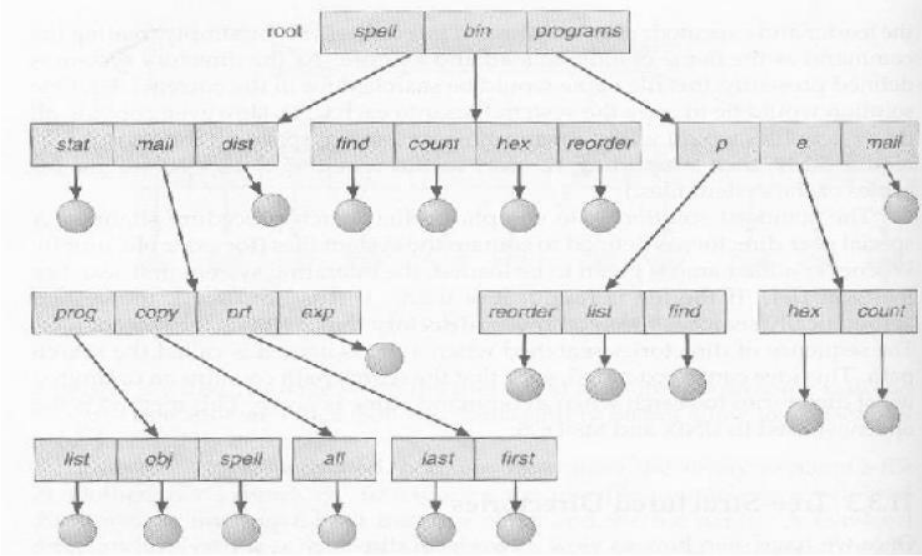


Figure 10.4: Tree-structured Directories.

Acyclic-graph Directories

The acyclic directory structure is an extension of the tree-structured directory structure. In the tree-structured directory, files and directories starting from some fixed directory are owned by one particular user. In the acyclic structure, this prohibition is taken out and thus a directory or file under directory can be owned by several users. Shows in Figure 10.5 an acyclic-graph directory structure.

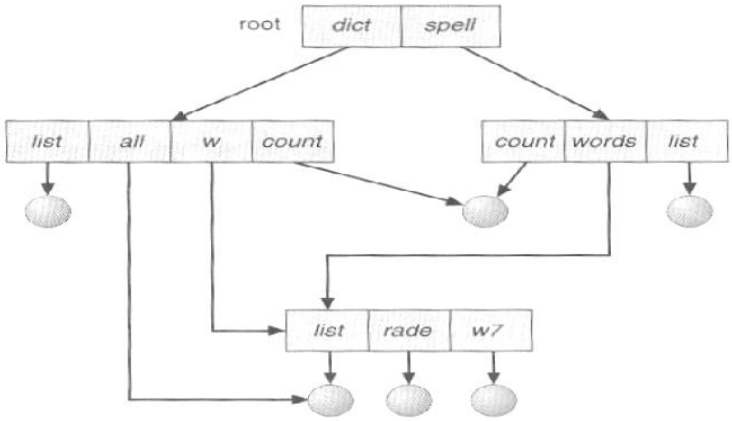


Figure 10.5: Acyclic graph directory.

Operations Performed on Directory

Just as with files, there are several basic operations on a directory:

- *Search for a File:* we need to be able to search the directory for a specific file.
- *Create a File:* We need to be able to create files and place the entry in the directory.
- *Delete a File:* We need to be able to delete a file and remove its entry from the directory.
- *List a Directory:* We need to be able to get a list of files in the directory.
- *Rename a File:* We need to be able to rename files in the directory.
- *Traverse the File System:* We need some way (or ability) to traverse the directory. This is useful in backups, compression (zipping the entire directory), etc.

10.3.1 Uses of Directories and Names

File system is used to organize files and other file system objects by using directories. A directory or folder is a file system object that contains other file system objects. Actually, a directory is an object that contains the name of file system objects. In a system with one-to-one mapping between names and objects, the distinction between name and object is not very important. In some file systems, objects can have multiple names or no name, In this situation, the distinction become important.

The entries in directories determine full pathname or names associated with a file system object. Starting with root directory, the full pathname is constructed by concatenating the sequence of names. Consider the file system in the following Figure 10.6. The directory D1 contains the names: D2, D3 and F1. The directory D2 contains no names. The directory D3 contains the names F1 and F2.

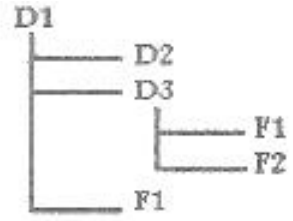


Figure 10.6: File system content.

Pre-process Root Directory

On some systems, it is possible to designate a directory that, will serve as the root directory for a process. Once this designation occurs, only the sub tree headed by that directory is visible to the process. On a system where file-sharing is desirable, each user's home directory could be made the root directory for all processes the user executes. The root of the file system would be the user's home directory to those processes. This capability may also be used by applications to provide additional security.

File systems allocate space in a granular manner, usually multiple physical units on the device. The file system is responsible for organizing files and directories, and keeping track of which areas of the media belong to which file and which are not being used. For example, in Apple DOS of the early 1980s, 256-byte sectors on 140 kilobyte floppy disk used a track/sector map.

Did You Know?

The first generation of Peer to peer software is Napster, a central server-based model that was eventually shut down. The second generation of P2P software is Gnutella and Kazaa, which are user-based models. Bit Torrent became the third generation of P2P networks.

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: Identify Structure of the Directories with Diagram.

……..……

……..……

……...

10.4 File Sharing

File sharing is the practice of distributing or providing access to digitally stored information, such as computer programs, multimedia (audio, images and video), documents, or electronic books. It may be implemented through a variety of ways. Common methods of storage, transmission and dispersion include manual sharing utilizing removable media, centralized servers on computer networks, World Wide Web-based hyperlinked documents, and the use of distributed peer-to-peer networking. With the "File Share" feature you can upload and share any of files of minutes for free! Files can be shared with anyone, including not registered 4shared members, by providing them with a simple web-based link to shared files. Use a handy 4shared email service to send a sharing link. Simply specify email addresses of the people you are going to share files with and choose the files. The link will be enclosed to your email and then used as an easy way to access your shared files.

Key features:

- Download accelerators can be used
- Multiple files uploading
- File upload progress bar
- Anonymous file uploading
- File sharing with different access levels
- Resume interrupted transfers
- Detailed file and account statistics
- Changing file description
- Automated thumbnails
- Photo album
- Web Grab for uploading files from the web
- Direct downloading link or embedding a file on your web page
- Zip Preview allows viewing content of a Zip file
- Zip files online
- Multiple files downloading
- Play music files
- Edit text files
- Search function

10.4.1 Types of File Sharing

Peer-to-peer File Sharing

Peer-to-peer file sharing (known in internet shorthand as "P2P") allows users to download media files such as music, movies, and games using a P2P software client that searches for other connected computers. The "peers" are computer systems connected to each other through the Internet. Thus, the only requirements for a computer to join peer-to-peer networks are an Internet connection and P2P

software. The difference between bit torrent and previous generations is that it creates a new network for every set of files instead of trying to create one big network of files using super nodes, web caches or servers. There are a number of factors that contribute to the widespread adoption and facilitation of peer-to-peer file sharing. These include increasing Internet bandwidth, the widespread digitization of physical media files, and the capabilities of home PC's increasing to better handle playing and storing digitized audio and video files. Users were able to transfer either one or more files from one computer to another across the Internet through various file transfers and file-sharing networks.

Legality of File Sharing

The legal debate surrounding file sharing has caused many lawsuits. In that particular lawsuit, the Supreme Court has ruled that the creators of P2P networks can be held responsible if the intent of their program is clearly to infringe on copyright laws.

On the other hand, file sharing is not necessarily illegal, even if the works being shared are covered by copyright. For example, some artists may choose to support freeware, shareware, open source, or anti-copyright, and advocate the use of file sharing as a free promotional tool. Nearly all freeware and open source software may be shared, under the rules specified in the license for that specific piece of software. Content in the public domain can also be freely shared.

Did You Know?

In 2004 there were an estimated 70 million people participating in online file sharing. According to a CBS News poll, in 2009 fifty-eight percent of Americans who follow the file sharing issue, considered it acceptable "if a person owns the music CD and shares it with a limited number of friends and acquaintances"; with 18 to 29 year olds this percentage reached as much as 70%.

Self Assessment Questions

1. Files are of different types depending on the……..in a file and therefore each file has its own structure which is type dependent.

(a). information stored (b). memory management
(c). file management (d). None of these.

2. The part of operating system that maps various files to the ……is called as file management system

(a). read only memory (b). physical devices
(c). random access memory (d). None of these.

3. The symbolic file name is the only information kept in……

(a). human-read form (b). memory management
(c). file management (d). None of these

4. A…….is required for the systems that support different types of files.

(a). file type (b). data type (c). file management (d). None of these

10.5 Protection of File System

When designing file systems, one of the main considerations is how to protect sharable files. Sometimes this is avoided by not permitting sharing of files and hence protection is quite easy to

implement. Another easy way of implementing file protection is to make all files public and provide complete sharing. However, neither of this is acceptable to users. There are files that users want to share and there are some which only a group of users should be allowed to access. The solution to this is to allow limited sharing. Protection is achieved by limiting the type of file access which can be made.

Access is permitted or denied depending upon several factors, one of which is the type of access requested. Several operations on files can be controlled.

Some of these are:

- Read - Read a file
- Write - Write a file
- Execute - Load and execute a file
- Append - Append information at the end of a file
- Delete - Free the space allocated to a file

The most common implementation of the file systems allow the owners of the file to do all operations, whereas other users can only invoke those operations that do not modify the file, e.g., file read. However, in some systems such as UNIX, the user can change the access control of a file such that he can let anybody access (modification allowed) the file or he can completely deny any user access to a file.

There are three most popular implementations of file protection:

- File naming - It depends upon the inability of a user to access a file he cannot name. This can be implemented by allowing only users to see the files they have created. But since most file systems allow only a limited number of characters for filenames, there is no guarantee that two users will not use the same filenames.
- Password Protection - This scheme associates a password to each file. If a user does not know the password associated to a file then he cannot access it. This is a very effective way of protecting files but for a user who owns many files, and constantly changes the password to make sure that nobody accesses these files will require that users have some systematic way of keeping track of their passwords.
- Access control - An access list is associated to each file or directory. The access list contains information on the type of users and accesses that they can do on a directory or file. An example is the following access list associated to a UNIX file or directory:

10.5.1 File Allocation Methods

One main problem in file management is how to allocate space for files so that disk space is utilized effectively and files can be accessed quickly. Three major methods of allocating disk space are contiguous, linked, and indexed. Each method has its advantages and disadvantages. Accordingly, some systems support all three mechanisms of file allocation. More commonly, a system will use one particular method for all files.

Contiguous Allocation

The contiguous allocation method requires each file to occupy a set of contiguous address on the disk. Disk addresses define a linear ordering on the disk. With this ordering, accessing block b+1 after block b normally requires no head movement. When head movement is needed (from the last sector of one cylinder to the first sector of the next cylinder), it is only one track. Thus, the number of disk seeks required for accessing contiguous allocated files in minimal. Contiguous allocation of a file is defined by the disk address and the length of the first block. If the file is n blocks long, and

starts at location b, then it occupies blocks b, b+1, b+2,…, b+n-1. The directory entry for each file indicates the address of the starting block and the length of the area allocated for this file.

Indexed Allocation

The indexed allocation method is the solution to the problem of both contiguous and linked allocation. This is done by bringing all the pointers together into one location called the index block. The index block will occupy some space and thus could be considered as an overhead of the method. In indexed allocation, each file has its own index block, which is an array of disk sector of addresses. Indexed allocation supports direct access, without suffering from external fragmentation. Any free block anywhere on the disk may satisfy a request for more space.

10.5.2 File System Reliability

It is very essential that a file system is reliable and long-lasting. The loss of invaluable data due to a file system failure could lead to catastrophic consequences. In this section, we will look at some of the issues involved in safeguarding the data. Having backup and checking for file consistency is couple of techniques that safeguard our data in case of a file system or a hardware crash. Backup technique is as simple as it sounds. It involves keeping another copy of the data on some other machine so that the copy could be used in case of a system failure. There are two types of backup techniques, namely full dump and incremental dump. Full dump simply refers to making a backup copy of the whole disk on another disk or machine. It is pretty obvious that the process of full dumping is time consuming as well as memory consuming. Incremental dump has some advantages over full dump. The simplest form of incremental dumping is to make a full dump periodically (say monthly or weekly) and to make a daily dump of only those files that have been modified since the last full dump. A better scheme could be to change only those files that have been changed since the last full dump. Such a scheme of data backup is time efficient as well as memory efficient. To implement this method, a list of dump times for each file must be kept on disk. The dump program should then check for each file on the hard disk and if it finds that the file has been modified since it was last dumped, it dumps the file yet again and changes the file's time of last dump to the current time. Another way of safeguarding data is to check for file consistency. The directory information in the main memory is more updated than that residing on the disk. In case of a system malfunction, if the table of open files is lost, all the changes made in the directories of open files are lost too. Hence, there arises an inconsistency between the state of some files and the directory structure. Many systems employ a "consistency checker" program that checks and corrects disk inconsistencies. Therefore, a file management system should have some consistency checking mechanism associated with it.

10.6 File System in Linux

A simple description of the UNIX system, also applicable to Linux, is this: "On a UNIX system, everything is a file; if something is not a file, it is a process." This statement is true because there are special files that are more than just files (named pipes and sockets, for instance), but to keep things simple, saying that everything is a file is an acceptable generalization. A Linux system, just like UNIX, makes no difference between a file and a directory, since a directory is just a file containing names of other files. Programs, services, texts, images, and so forth, are all files. Input and output devices, and generally all devices, are considered to be files, according to the system. In order to manage all those files in an orderly fashion, man likes to think of them in an ordered tree-like structure on the hard disk, as we know from MS-DOS (Disk Operating System) for instance. The large branches contain more branches, and the branches at the end contain the tree's leaves or normal

files. Now we will use this image of the tree, but we will find out later why this is not a fully accurate image.

Sorts of Files

Most files are just files, called regular files; they contain normal data, for example text files, executable files or programs, input for or output from a program and so on.

While it is reasonably safe to suppose that everything you encounter on a Linux system is a file, there are some exceptions.

- *Directories:* Files that are lists of other files.
- *Special Files:* The mechanism used for input and output.
- *Links:* A system to make a file or directory visible in multiple parts of the system's file tree.
- *(Domain) Sockets:* A special file type, similar to TCP/IP sockets, providing inter process networking protected by the file system's access control.
- *Named Pipes:* Act more or less like sockets and form a way for processes to communicate with each other, without using network socket semantics.

Table 10.1: File types in a long list

Symbol	Meaning
-	Regular file
d	Directory
l	Link
c	Special file
s	Socket
p	Named pipe
b	Block device

Partition

Most people have a vague knowledge of what partitions are, since every operating system has the ability to create or remove them. It may seem strange that Linux uses more than one partition on the same disk, even when using the standard installation procedure, so some explanation is called for. One of the goals of having different partitions is to achieve higher data security in case of disaster. By dividing the hard disk in partitions, data can be grouped and separated. When an accident occurs, only the data in the partition that got the hit will be damaged, while the data on the other partitions will most likely survive. This principle dates from the days when Linux did not have journal file systems and power failures might have lead to disaster. The use of partitions remains for security and robustness reasons, so a breach on one part of the system does not automatically mean that the whole computer is in danger. This is currently the most important reason for partitioning.

A simple example: a user creates a script, a program or a web application that starts filling up the disk. If the disk contains only one big partition, the entire system will stop functioning if the disk is full. If the user stores the data on a separate partition, then only that (data) partition will be affected, while the system partitions and possible other data partitions keep functioning. Mind that having a journal file system only provides data security in case of power failure and sudden disconnection of storage devices. This does not protect your data against bad blocks and logical errors in the file system. In those cases, you should use a RAID (Redundant Array of Inexpensive Disks) solution.

Partition Layout and Types

There are two kinds of major partitions on a Linux system:

- *Data Partition:* Normal Linux system data, including the root partition containing all the data to start-up and run the system.
- *Swap Partition*: Expansion of the computer's physical memory, extra memory on hard disk.

10.6.1 More File System Layout

Visual

For convenience, the Linux file system is usually thought of in a tree structure. On a standard Linux system you will find the layout generally follows the scheme presented below in Figure 10.7.

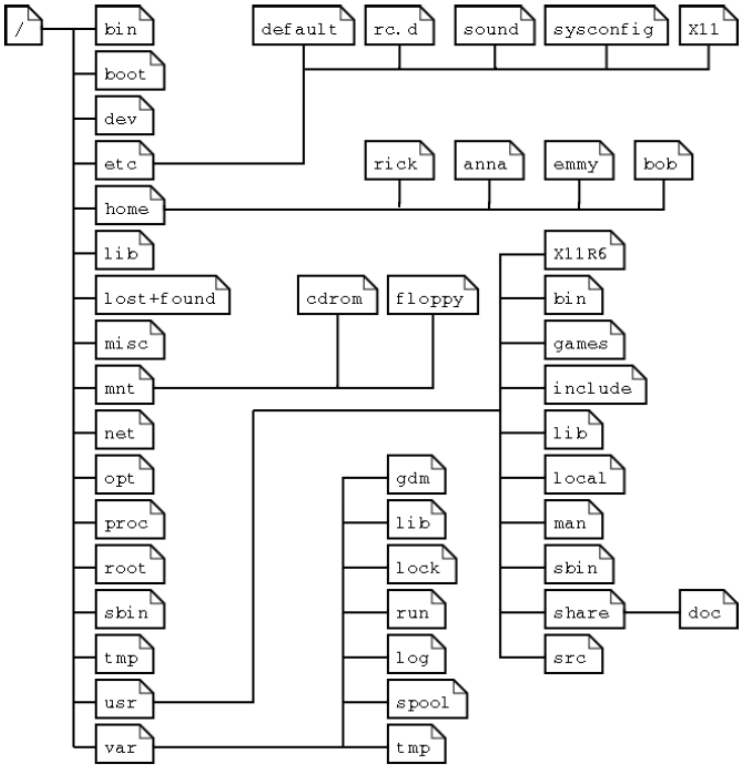


Figure 10.7: Linux file system layout.

Did You Know?

Linux's first file system was minix, which was borrowed from the Minix OS. Linus Torvalds adopted this file system because it was an efficient and relatively bug-free piece of existing software that postponed the need to design a new file system from scratch.

Self Assessment Questions

5. ……this is a pointer to the device and location on that device of the file. As it is clear from the attribute name, it specifies where the file is stored.

(a). Space (b). Location (b). Memory (d). None of these

6. ……attribute keeps track of the current size of a file in bytes, words or blocks.

(a). Size (b). Location (c). Protection (d). None of these

7. The operating system provides systems calls to create, write, read, reposition, truncate and delete files.

(a). True (d). False

8. To delete a file, the directory is searched for the……

(a). operating system (b). named file

(c). file management (d). None of these

9. The most common directory structures used by multi-user systems are…….

(a). single-level directory (b). two-level directory

(c). Both (a) and (b) (d). None of these

10. A…….has significant limitations when the number of files increases or when there is more than one user.

(a). single-level directory (b). two-level directory

(c). tree-structured directory (d). None of these

10.7 Summary

- The file management, formerly known as data management, is the part of the operating system that controls the storing and accessing of data by an application program.
- In a single-level directory system, all the files are placed in one directory.
- The using memory on a hard disk is naturally slower than using the real memory chips of a computer.
- The kernel is on a separate partition as well in many distributions, because it is the most important file of system.
- One of the goals of having different partitions is to achieve higher data security in case of disaster.

10.8 Keywords

Data Partition: The normal Linux system data, including the root partition containing all the data to start up and run the system.

File Management: It provides functions that allow you to manage files (create, change, override, or delete) using CL commands, and create and access data through a set of operations (for example, read, write, open, or close).

Operating System: It provides systems calls to create, write, read, reposition, truncate and delete files.

Swap Partition: It is expansion of the computer's physical memory, extra memory on hard disk.

Sub Directory: The directory under root (/) directory is subdirectory which can be created, renamed by the user.

10.9 Review Questions

1. What are the concepts of file system and operations?
2. Explain the types of file.
3. What is the directory structure?

4. What are the uses of directories and names?
5. Define the types of file sharing.
6. What are the protections of file system?
7. Explain the methods of file allocation.
8. Discuss the file system in Linux.
9. What are differentiating between file sharing and management?
10. Define the file system layout.

Answers of Self Assessment Questions

1. (a)	2. (b)	3. (a)	4. (a)	5. (b)
6. (a)	7. (a)	8. (b)	9. (c)	10. (a)

10.10 Further Studies

1. *File Management by Patricia Brown*
2. Operating System Principles by Silberschatz, Galvin and Gagne

Computer Networks & Internet Concept

(MCA/PGDCA - 106)



Jaipur National University
Directorate of Distance Education

Established by Government of Rajasthan
Approved by UGC under Sec 2(f) of UGC ACT 1956
(**Recognised by Joint Committee of UGC-AICTE-DEC, Govt. of India**)

1

Introduction to Networking

STRUCTURE

1.0 Objectives

1.1 Introduction

1.2 Network

1.3 Need of Networks

1.4 Uses of Computer Networks

1.5 Application of Computer Network

1.6 Network Criteria

1.7 Network Hardware and Software

1.8 Types of Networks

1.9 Summary

1.10 Keywords

1.11 Review Questions

1.12 Further Studies

1.0 Objectives

After studying this chapter, you will be able to:

- Discuss the introduction to network
- Define the computer networks
- Explain the need and uses of computer network
- Discuss the applications of network and criteria

1.1 Introduction

Today computer is available in many offices and homes and therefore there is a need to share data and programs among various computers. With the advancement of data communication facilities the communication between computers has increased and thus it has extended the power of computer beyond the computer room. Now a user sitting at one place can communicate with computers of any remote site through communication channel. The aim of this chapter is to introduce you the various aspects of computer network.
In the world of computers, networking is the practice of linking two or more computing devices together for the purpose of sharing data. Networks are built with a mix of computer hardware and computer software.

1.2 Network

A network comprises two or more computers that have been connected in order to enable them to communicate with each other, and share resources and files.

1.2.1 Computer Networks

A computer network is interconnection of various computer systems located at different places. In computer network two or more computers are linked together with a medium and data communication devices for the purpose of communication data and sharing resources. The computer that provides resources to other computers on a network is known as server. In the network the individual computers, which access shared network resources, are known as nodes.

Parts of a network

There are five basic components of a network: clients, servers, channels, interface devices and operating systems.

Servers: Sometimes called host computers, servers are powerful computers that store data or applications and connect to resources that are shared by the users of a network.

Clients: These computers are used by the users of the network to access the servers and shared resources (such as hard disks and printers). These days, it is typical for a client to be a personal computer that the users also use for their own non-network applications.

Channels: Called the network circuit, the channel is the pathway over which information travels between the different computers (clients and servers) that comprises the network.

Interface devices: These are hardware devices that connect clients and servers (and sometimes other networks) to the channel. Examples include modems and network interface cards.

Operating systems: The network operating system is the software of the network. It serves a similar purpose that the operating system serves in a stand-alone computer.

1.2.2 Understanding Network Channels

Network channels come in a variety of types, speeds and capabilities. For our purposes, there are four important dimensions of channels:

Transmission medium: This is the actual physical medium of the channel. Computer network channels use either wire line or wireless media.

Wire line media: Also called guided media and line-based media. In networks that use wire line media, the transmission of information takes place on a wire or cable. The three types of wire line media are twisted-pair wire, coaxial cable and fibre-optic cable. (Try and find examples of each of these media, and their relative speeds). While twisted-pair and coaxial cable are more commonly used today, fibre optic cables are becoming increasingly popular.

Wireless media also called radiated media. As the name indicates, in networks that use wireless media, there is no physical wire along which information travels; instead, information is transmitted through the air, from one transmission station to the next. Networking examples include radio, cellular, microwave and satellite. Broadcast TV and FM radio use wireless transmission as well (though the underlying engineering is a little different).

***Transmission rate or bandwidth*:** This property of a network channel describes how fast information can be transmitted over the channel. It is measured in bits per second people very commonly use the term bandwidth to mean transmission rate.

***Transmission directional capability*:** The direction in which information can be transmitted over a channel depends on whether the channel is simple, half-duplex or full-duplex.

Simplex: Information can be transmitted only in one direction.
Half-duplex: Information can be transmitted in both directions, but only in one direction at a time.
Full-duplex: Information can be transmitted in both directions simultaneously.
***Signal type*:** There are two signal types analog and digital. It is a little hard to understand the exact difference without discussing a lot of electrical engineering and physics, so we would not go there. What you need to take away is that:

- Analog signals are 'continuous' (they take on a wide range of values) and digital signals are 'discrete', and binary.
- Digital signals are more 'natural' for computer networks, since, as we know, computers represent all information in binary.
- The reason why we have to worry about analog signals is because the communications. Channels that predated computer networks (like telephone lines, cable TV lines and radio transmitters) were all designed to carry analog signals.

Did You Know?
Early networks of communicating computers included the military radar system Semi-Automatic Ground Environment (SAGE), started in the late 1950s.

1.3 Need of Networks

As businesses grows, so does the requirements. Companies that start out small grow into large organizations where they have thousands of offices spread across different parts of the globe. Companies needed a cost effective solution for connecting all their businesses in different places and thus computer networks were invented.
In olden times, a single computer used to process all the requirements of a large organization or university. Gone are those days where centralized computing was the norm. With the advent of technology and the reduction in prices of computers, businesses found it more cost effective to have a lot of computers do the job a single computer did. This proved effective in many ways:

1. Higher reliability
2. Better efficiency
3. Ability to troubleshoot better
4. Reduced latency

1.4 Uses of Computer Networks

A computer network is an interconnection of computers, printers, scanners and other hardware devices and software applications. Networks connect users within a defined physical space (such as within an office building). The Internet is a network that connects users from all parts of the world. Educational institutions, government agencies, health care facilities, banking and other financial institutions, and residential applications use computer networking to send and receive data and share resources.

Communication and Access to Information: The primary purpose of computer networking is to facilitate communication. A network allows a user to instantly connect with another user, or network, and send and receive data. It allows remote users to connect with one other via videoconferencing, virtual meetings and digital emails. Computer networks provide access to online libraries, journals, electronic newspapers, chat rooms, social networking Websites, email clients and the World Wide Web. Users can benefit from making

online bookings for theatres, restaurants, hotels, trains and airplanes. They can shop and carry out banking transactions from the comfort of their homes.

Computer networks allow users to access interactive entertainment channels, such as video on demand, interactive films, interactive and live television, multi-person real-time games and virtual-reality models.

Resource Sharing: Computer networks allow users to share files and resources. They are popularly used in organizations to cut costs and streamline resource sharing. A single printer attached to a small local area network (LAN) can effectively service the printing requests of all computer users on the same network. Users can similarly share other network hardware devices, such as modems, fax machines, hard drives and removable storage drives.

Networks allow users to share software applications, programs and files. They can share documents (such as invoices, spreadsheets and memos), word processing software, videos, photographs, audio files, project tracking software and other similar programs. Users can also access, retrieve and save data on the hard drive of the main network server.

Centralized Support and Administration: Computer networking centralizes support, administration and network support tasks. Technical personnel manage all the nodes of the network, provide assistance, and troubleshoot network hardware and software errors. Network administrators ensure data integrity and devise systems to maintain the reliability of information through the network. They are responsible for providing high-end antivirus, anti-spyware and firewall software to the network users. Unlike a stand-alone system, a networked computer is fully managed and administered by a centralized server, which accepts all user requests and services them as required.

1.5 Application of Computer Network

In the short time they have been around, data communication networks have become an indispensable part of business, industry, and entertainment. Some of the network applications in different fields are the following:

Marketing and Sales: Computer networks are used extensively in both marketing and sales organizations. Marketing professionals use them to collect, exchange, and analyze data relating to customer needs and product development cycles. Sales applications include teleshopping, which uses order-entry computers or telephones connected to an order-processing network, and on-line reservation services for hotels, airlines, and so on.

Financial Services: Today's financial services are totally dependent on computer networks. Applications include credit history searches, foreign exchange and investment services, and electronic funds transfer (EFT), which allows a user to transfer money without going into a bank (an automated teller machine is a kind of electronic funds transfer; automatic pay check deposit is another).

Manufacturing: Computer networks are used today in many aspects of manufacturing, inch the manufacturing process itself. Two applications that use networks to provide essential services are computer-assisted design (CAD) and computer-assisted manufacturing (CAM), both of which allow multiple users to work on a project simultaneously.

Electronic Messaging**:** Probably the most widely used network application is electronic mail (e-mail).

Directory Services: Directory services allow lists of files to be stored in a central location to speed worldwide search operations.

Information Services: Network information services include bulletin boards and data banks. A World Wide Web site offering the technical specifications for a new product is an information service.

Electronic data interchange (EDI). The EDI allows business information (including documents such as purchase orders and invoices) to be transferred without using paper.

Teleconferencing: Teleconferencing allows conferences to occur without the participants being in the same place. Applications include simple text conferencing (where participants communicate through their keyboards

and computer monitors), voice conferencing (where participants at a number of locations communicate simultaneously over the phone), and video conferencing (where participants can see as well as talk to one another).

Cellular Telephone: In the past, two parties wishing to use the services of the telephone company had to be linked by a fixed physical connection. Today's cellular networks make it possible to maintain wireless phone connections even while travelling over large distances.

Cable Television: Future services provided by cable television networks may include video on request, as well as the same information, financial, and communications services currently provided by the telephone companies and computer networks.

1.6 Network Criteria

To be considered effective and efficient, a network must meet a number of criteria. The most important of these are performance, reliability, and security Figure 1.1

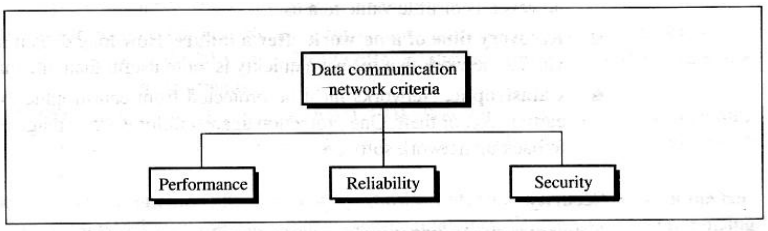


Figure1.1: Network Criteria.

Performance

Performance can be measured in many ways, including transit time and response time. Transit time is the amount of time required for a message to travel from one device to another. Response time is the elapsed time between an inquiry and a response. The performance of a network depends on a number of factors, including the number of users, the type of transmission medium, the capabilities of the connected hardware, and the efficiency of the software.

- ***Number of users:*** Having a large number of concurrent users can slow response time in a network not designed to coordinate heavy traffic loads. The design of a given network is based on an assessment of the average number of users that will be communicating at any one time. In peak load periods, however, the actual number of users can exceed the average and thereby decrease performance. How a network responds to loading is a measure of its performance.
- ***Type of transmission medium:*** The medium defines the speed at which data can travel through a connection (the data rate). Today's networks are moving to faster and faster transmission media, such as fiber-optic cabling. A medium that can carry data at 100 megabits per second is 10 times more powerful than a medium that can carry data at only 10 megabits per second. However, the speed of light imposes an upper bound on the data rate.
- ***Hardware:*** The types of hardware included in a network affect both the speed and capacity of transmission. A higher-speed computer with greater storage capacity provides better performance.
- ***Software:*** The software used to process data at the sender, receiver, and intermediate nodes also affects network performance. Moving a message from node to node through a network requires processing to transform the raw data into transmittable signals, to route these signals to the proper destination, to ensure error-free delivery, and to recast the signals into a form the receiver can use. The software that provides these services affects both the speed and the reliability of a network link. Well-designed software can speed the process and make transmission more effective and efficient.

Reliability: In addition to accuracy of delivery, network reliability is measured by frequency of failure, the time it takes a link to recover from a failure, and the network's robustness in a catastrophe.
Frequency of failure: All networks fail occasionally. A network that fails often, however, is of little value to a user.
Recovery time of a network after a failure: A network that recovers quickly is more useful than one that does not.
Security: Network security issues include protecting data from unauthorized access and viruses.

- *Unauthorized access*: For a network to be useful, Protection can be accomplished at a number of levels. At the lowest level are user identification codes and passwords. At a higher level are encryption techniques. In these mechanisms, data are systematically altered in such a way that if they are intercepted by an unauthorized user, they will be unintelligible.
- *Viruses:* Because a network is accessible from many points, it can be susceptible to computer viruses. A virus is an illicitly introduced code that damages the system. A good network is protected from viruses by hardware and software designed specifically for that purpose.

Caution
Networks must be protected from catastrophic events such as fire, earthquake, or theft. One protection against unforeseen damage is a reliable system to back up network software.

1.7 Network Hardware and Software

1.7.1 Networking Hardware

Network adapter cards: expansion cards that provide the physical connection between each computer and the network. The card installs into a slot on your computer, just like a sound card or modem card. Some newer computers have a network adapter already built into the system. Laptop computers often use a card that slides into a PC card slot.



Figure 1.2: Network adapter card.

Network hub: the central connection point for network cables that connect to computers or other devices on a network. The hub has several network cable jacks or ports that you use to connect network cables to computers. The hub contains circuitry that enables each computer to communicate with any other computer connected to the hub (see Figure 1.3).



Figure 1.3: Network Hub.

Network cables: special, unshielded twisted-pair (UTP) cables used to connect each computer to the hub. The cable you need is Category 5 UTP cable with a square plastic RJ-45 connector on each end.

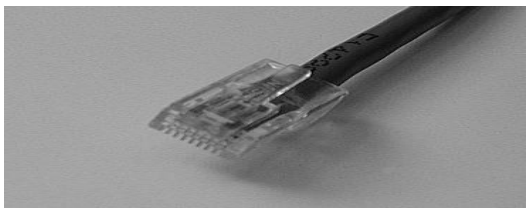


Figure 1.4: Network cable with RJ-45 connector.

All the networking hardware described here is known as Ethernet. Ethernet is the industry-wide standard for computer networks. Standard Ethernet networks transmit data at 10 million bits per second (Mbps). A newer Ethernet standard, called Fast Ethernet, transmits data at 100 Mbps. Computer networks often contain a mixture of 10 Mbps and 100 Mbps devices.

Wireless Networking Hardware

You may want to network computers where it is expensive or difficult to run network cables, for example, between two rooms or two buildings. However, recent advances in wireless networking technology make wireless networking practical and affordable. New wireless standards have facilitated the development of wireless products with good performance and the ability to integrate easily into a wired Ethernet network. The Ethernet standard for wireless networking is the IEEE 802.11b wireless standard. The 802.11b standard supports wireless connections at speeds up to 11 Mps, comparable to 10 Mbps wired Ethernet. Wireless industry leaders formed the Wireless Ethernet Compatibility Alliance (WECA) to certify cross-vendor compatibility with the 802.11b standard. These products display the WECA "Wireless Fidelity" (Wi-Fi) logo.

Figure 1.5: Wireless Fidelity Logo.

Suppose you want to network a few computers together in a small area where it would be expensive to have network cabling installed in an existing building. Or perhaps you just have a desktop computer and a notebook computer at home and you would like to be able to roam the house with the notebook computer and perhaps even browse the Web from the hammock in the back yard. Wireless Ethernet makes all this possible. You can install wireless adapters in each computer and form a wireless network Figure 1.6.





Figure 1.6: PC Card and USB Wireless Adapters.

1.7.2 Networking Software

Microsoft Windows includes the networking software you need to set up a small network of computers that can share documents and printers. All you need to do is install the Windows components for networking, and configure your computer to use these networking features. Most of the software you need will be installed when you install a network adapter in your computer. No matter which networking hardware you choose to install, the information will help you set up and use your network. After you have installed the networking drivers for your network adapter, the networking software built into Windows will work with your hardware.

Caution

While protecting the network, sensitive data must be protected from unauthorized access.

1.8 Types of Networks

There are many different types of networks. However, from an end user's point of view there are two basic types:

1.8.1 Local Area Network (LAN)

The LAN is a computer network that spans a relatively small area. Most LANs are confined to a single building or group of buildings. However, one LAN can be connected to other LAN's over any distance via telephone lines and radio waves. A system of LANs connected in this way is called a wide-area network (WAN).

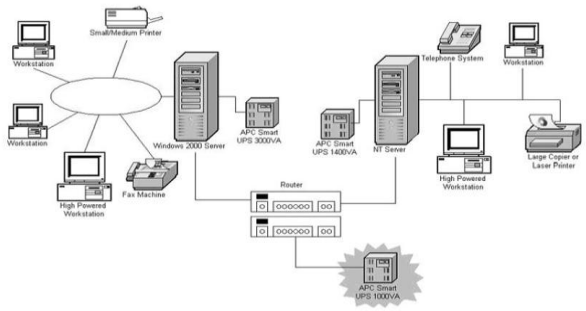


Figure.1.7: Local Area Network.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Draw the basic flow diagram of communication in the network.

……..……

……..……

……...

1.8.2 Wide-Area Networks (WANs)

The computers are farther apart and are connected by telephone lines or radio waves.

In addition to these types, the following characteristics are also used to categorize different types of networks.

Topology: The geometric arrangement of a computer system. Common topologies include bus, star, and ring.

Protocol

The protocol defines a common set of rules and signals that computers on the network use to communicate. One of the most popular protocols for LANs is called Ethernet. Another popular LAN protocol for PCs is the IBM token-ring network.

Architecture

Networks can be broadly classified as using either peer-to-peer or client/server architecture. Computers on a network are sometimes called nodes. Computers and devices that allocate resources for a network are called servers.

The types of networks can be further classified into two more divisions:

1.8.3 Peer-to-Peer Networking

This is a simple network configuration that requires some basic know-how to set up. Each of the interconnected machines shares dual capability and responsibility on the network. That is to say, that each machine serves a dual purpose or role, i.e. they are both clients and servers to some extent.

The server capability of the machines is very basic. The services provided by each, is no more than the ability to share resources like files, folders, disk drives and printers. They even have the ability to share Internet access.

However, the server functionality of these machines stops there. They cannot grant any of the benefits mentioned previously, since these are functions provided only by a dedicated server operating system.

Because all machines on the network have equal status, hence the term peers; there is no centralized control over shared resources. Sharing is endorsed or repealed by each machine's user. Passwords can be assigned to each individual shared resource whether it is a file, folder, drive or peripheral, again done by the user.

Although this solution is workable on small networks, it introduces the possibility that users may have to know and remember the passwords assigned to every resource and then re-learn them if the user of a particular machine decides to change them! Due to this flexibility and individual discretion, institutionalised chaos is the norm for peer-to-peer networks.

Security can also be a major concern, because users may give passwords to other unauthorised users, allowing them to access areas of the network that the company does not permit. Furthermore, due to lack of centralisation, it is impossible for users to know and remember what data lives on what machine, and there are no restrictions to prevent them from over-writing the wrong files with older versions of the file. This of course cripples attempts to organise proper backups.

It may appear that peer-to-peer networks are hardly worthwhile. However, they offer some powerful incentives, particularly for smaller organisations. Networks of this type are the cheapest and easiest to install, requiring only Windows XP, a network card for each machine and some cabling. Once connected, users can start to share information immediately and get access to devices.

1.8.4 Client/Server Networks

Server based networks, or client/server networks as they are properly called, has a machine at the heart of its operations called the server. A server is a machine that provides services over a network by responding to client requests. Servers rarely have individuals operating it, and even then, it is usually to install, configure or manage its capabilities. The server’s essential role on the network is to be continuously available to handle the many requests generated by its clients.

Server-based networks provide centralised control of the entire network environment. The computer systems used for this role are generally more powerful than end-user machines, incorporating faster CPUs, more memory, larger disk drives and other drive types installed, like a tape drive for backup purposes. These are required, because servers are dedicated to handling multiple simultaneous requests from their client communities.

Server based networks provide centralised verification of user accounts and passwords. Only valid account name and password combinations are allowed access to the network. Client/Server networks typically require a single login to the network itself, meaning that users need to remember long password lists to access various resources. Concentrations of resources on a single server mean that they are easier to find, as opposed to the peer-to-peer model, were resources were distributed throughout the network since they were attached to multiple machines. The server being a central data repository, means that not only is data more accessible to users, but it also makes life much easier in terms of performing backups, since the data is in a location know to the administrator.

Server-based networks are easier to scale. Peer-to-peer networks bog down seriously as they grow beyond ten users, and serious slow up with 20 users. On the other hand, client/server networks can handle a few users; up to a thousand users as such networks grow to keep pace with an organisations growth and expansion.

Unlike peer-to-peer networks, client/server networks do not come cheap. The server machine itself may cost several thousands of pounds, along with the software to make it run; another thousand pounds. Because of the complex nature of this kind of networking environment, a dedicated administrator is required to be on site at all times to be involved in the day to day running of the network. Hiring an individual of this nature adds considerably to the cost of client/server networks.

Lastly, because the networks operability is so dependent upon the server, this introduces a single point of failure, if the server goes down the network goes down. There are measures available, that can legislate for such failures, and however these techniques add even more cost to this solution.

Did You Know?

In 1965 Thomas Merrill and Lawrence G. Roberts created the first wide area network (WAN).

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Draw the block diagram of peer to peer network.

……..……

……..……

……...

1.8.5 Various Server Types

The multiple types of servers or types of network servers are as follows:

Application Servers

Sometimes referred to as a type of middleware, application servers occupy a large chunk of computing territory between database servers and the end user, and they often connect the two.

Middleware is a software that connects two otherwise separate applications For example, there are a number of middleware products that link a database system to a Web server This allows users to request data from the database using forms displayed on a Web browser and it enables the Web server to return dynamic Web pages based on the user's requests and profile.

The term middleware is used to describe separate products that serve as the glue between two applications. It is, therefore, distinct from import and export features that may be built into one of the applications. Middleware is sometimes called plumbing because it connects two sides of an application and passes data between them. Common middleware categories include:

- TP monitors
- DCE environments

- RPC systems
- Object Request Brokers (ORBs)
- Database access systems
- Message Passing

Audio/Video Servers

Audio/video servers bring multimedia capabilities to Web sites by enabling them to broadcast streaming multimedia content. Streaming is a technique for transferring data such that it can be processed as a steady and continuous stream. Streaming technologies are becoming increasingly important with the growth of the Internet because most users do not have fast enough access to download large multimedia files quickly. With streaming, the client browser or plug-in can starts displaying the data before the entire file has been transmitted.

For streaming to work, the client side receiving the data must be able to collect the data and send it as a steady stream to the application that is processing the data and converting it to sound or pictures. This means that if the streaming client receives the data more quickly than required, it needs to save the excess data in a buffer if the data does not come quickly enough, however, the presentation of the data will not be smooth.
There are a number of competing streaming technologies emerging. For audio data on the Internet, the de facto standard is Progressive Network's Real Audio.

Chat Servers

Chat servers enable a large number of users to exchange information in an environment similar to Internet newsgroups that offer real-time discussion capabilities. Real time means occurring immediately. The term is used to describe a number of different computer features. For example, real-time operating systems are systems that respond to input immediately. They are used for such tasks as navigation, in which the computer must react to a steady flow of new information without interruption. Most general-purpose operating systems are not real-time because they can take a few seconds, or even minutes, to react.
Real time can also refer to events simulated by a computer at the same speed that they would occur in real life. In graphics animation, for example, a real-time program would display objects moving across the screen at the same speed that they would actually move.

Fax Servers

A fax server is an ideal solution for organizations looking to reduce incoming and outgoing telephone resources but that need to fax actual documents.

FTP Servers

One of the oldest of the Internet services, File Transfer Protocol makes it possible to move one or more files securely between computers while providing file security and organization as well as transfer control.

Groupware Servers

A GroupWare server is software designed to enable users to collaborate, regardless of location, via the Internet or a corporate Intranet and to work together in a virtual atmosphere.

IRC Servers

An option for those seeking real-time capabilities, Internet Relay Chat consists of various separate networks (or "nets") of servers that allow users to connect to each other via an IRC network.

List Servers

List servers offer a way to better manage mailing lists, whether they are interactive discussions open to the public or one-way lists that deliver announcements, newsletters, or advertising.

Mail Servers

Almost as ubiquitous and crucial as Web servers, mail server's move and store mail over corporate networks via LANs and WANs and across the Internet.

News Servers

News servers act as a distribution and delivery source for the thousands of public news groups currently accessible over the USENET news network. USENET is a worldwide bulletin board system that can be accessed through the Internet or through many online services The USENET contains more than 14,000 forums called newsgroups that cover every imaginable interest group. It is used daily by millions of people around the world.

Proxy Servers

Proxy servers sit between a client program typically a Web browser and an external server (typically another server on the Web) to filter requests, improve performance, and share connections.

Telnet Servers

A Telnet server enables users to log on to a host computer and perform tasks as if they are working on the remote computer itself.

Web Servers

At its core, a Web server serves static content to a Web browser by loading a file from a disk and serving it across the network to a user's Web browser. The browser and server talking to each other using HTTP mediate this entire exchange.

Did You Know?

The first widely used telephone switch that used true computer control was introduced by Western Electric in 1965.

Self Assessment Questions

1.……have been widely used in many areas of business since the mid-1960s.

(a). Database management systems (b). Networking

(c). Management (d). None of these

2.These systems must address a number of problems, such as……

(a). Security (b). Accuracy

(c). Both a and b (d). None of these

3……..the central connection point for network cables that connect to computers or other devices on a network.

(a). Network (b). Network hub

(c). Network adapter cards (d). None of these

4……expansion cards that provide the physical connection between each computer and the network.

(a). Network cards (b). Pen cards

(c). Network adapter cards (d). None of these

5. ……are more 'natural' for computer networks, since, as we know, computers represent all information in binary.

(a). Analog signals (b). Network signals

(c). Digital signals (d). None of these

Exercise: Check Your Progress 3

Case Study: Computer Network Technologies

The Client

CNT was growing rapidly. In 2003, the company had doubled in size, and soon found its legacy system unable to keep up with its growth. Integration was also an issue. With key information residing on multiple databases, CNT could not leverage the Internet to drive efficiency from its HR processes. In addition, CNT's payroll process with an outsourced payroll provider involved collecting payroll information on hand-written worksheets.

The Challenge

The company studied the problem, and found that in order to process HR data more efficiently and streamline its business processes, it needed a single, integrated and web-enabled HR application to collect and process employee time, benefit elections and other employee changes. The company also needed to integrate its stand-alone recruiting solution with its HR applications to allow managers to conduct online reviews.

The Solution

CNT decided to act quickly, and set a goal to select and implement a new solution by 2005. After reviewing its options, CNT found that Oracle's HRMS application best met its needs for a fully functional, webcapable and integrated HR and Payroll system.

The Benefits

CNT's HR-related business processes are now managed from a single, integrated database solution that eliminates redundant data entry. Management reporting information and visibility is improved, because there is no need to extract the necessary data from a variety of systems. Payroll processes, including check writing and tax filing, are now handled out of the same system as other HR-related tasks.

Q.1: Identify Client.

Q.2: Give an Solution of CNT.

1.9 Summary

- The modern form of communication like e-mail and Internet is possible only because of computer networking.
- Data Routing is the process of finding the most efficient route between source and destination before sending the data.
- In simplex mode the communication take place in one direction. The receiver receives the signal from the transmitting device.
- In half-duplex mode the communication channel is used in both directions, but only in one direction at a time. Thus a half-duplex line can alternately send and receive data.
- The computer that provides resources to other computers on a network is known as server.
- In the network the individual computers, which access shared network resources, are known as nodes.

1.10 Keywords

Communication Satellite: The problem of line-sight and repeaters are overcome by using satellites which are the most widely used data transmission media in modern days.

Data sequencing: A long message to be transmitted is broken into smaller packets of fixed size for error free data transmission.

Internet: The newest type of network to be used within an organisation is an internet or Internet Web. Such networks enable computers (or network) of any type to communicate easily.

Transmission: Communication of data achieved by the processing of signals.

Teleconferencing: It refers to electronic meetings that involve people who are at physically different sites. Telecommunication technology allows participants to interact with one another without travelling to the same location.

1.11 Review Questions

1. What is the model and communication task?
2. What are the needs of computer networks?
3. Differentiate between half-duplex and full-duplex.
4. What is the use of computer networks?
5. What is the application of computer network?
6. Differentiate between LAN and WAN.
7. How many types of server and network?
8. What are the software and hardware networking?
9. What are the network criteria?
10. What are the hub and network cables?

Answers for Self Assessment Questions

1. (a) 2. (c) 3. (b) 4. (c) 5. (c)

1.12 Further Studies

1. Computer Networks by V.S.Bagad, I.A.Dhotre
2. Computer Networks: A Systems Approach by Larry L. Peterson, Bruce S. Davie

2

Data Transmission

STRUCTURE

2.0 Objectives

2.1 Introduction

2.2 Transmission Technologies

2.3 Analog and Digital Data

2.4 Data Communication

2.5 Base Band, Broadband

2.6 Asynchronous and Synchronous Transmission

2.7 Comparison of Base band and Broadband Signalling

2.8 Summary

2.9 Keywords

2.10 Review Questions

2.11 Further Studies

2.0 Objectives

After studying this chapter, you will be able to:

- Explain the Base Band, Broadband
- Discuss the Analog and Digital Data
- Define the transmission impairment
- Discuss the Shannon capacity

2.1 Introduction

Data transmission is the transfer of data from point-to-point often represented as an electro-magnetic signal over a physical point-to-point or point-to-multipoint communication channel. Examples of such channels are copper wires, optical fibers, wireless communication channels, and storage media. The term usually refers to digital communications (i.e. digital bit stream), but may include analog data transmission as well.

Data transmission is a subset of the field of data communications, which also includes computer networking or computer communication applications and networking protocols, for example routing and switching.

Data transmission, Sending and receiving data via cables (e.g., telephone lines or fiber optics) or wireless relay systems. Because ordinary telephone circuits pass signals that fall within the frequency range of voice communication (about 300–3,500 hertz), the high frequencies associated with data transmission suffer a loss of

amplitude and transmission speed. Digital computers use a modem to transform outgoing digital electronic data; a similar system at the receiving end translates the incoming signal back to the original electronic data. Specialized data-transmission links carry signals at frequencies higher than those used by the public telephone network.

The purpose of a network is to transmit information from one computer to another. To do this, you first have to decide how to encode the data to be sent, in other words its computer representation. This will differ according to the type of data, which could be:

- Audio data
- Text data
- Graphical data
- Video data

Data representation can be divided into two categories:

- Digital representation: which means that the information is encoded as a set of binary values, in other words a sequence of 0s and 1s
- Analogue representation: which means that the data will be represented by the variation in a continuous physical quantity

2.2 Transmission Technologies

There is no generally accepted taxonomy into which all computer networks fit, but two dimensions stand out as important: Transmission Technology and Scale.

Classification based on Transmission Technology:

Computer networks can be broadly categorized into two types based on transmission technologies:

1. Broadcast networks
2. Point-to-point networks

Broadcast Networks

A broadcast network has a single communication channel that is shared by all the machines on the network.

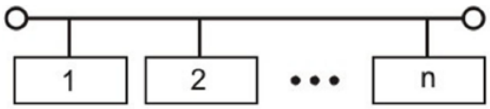


Figure 1: Broadcast Network

All the machines on the network send and receive messages, called packets. Each packet conation an address field specifies the intended recipient. Upon receiving a packet, machine checks the address field. If packet is intended for itself, it processes the packet; if packet is not intended for itself it is simply ignored. When a packet is transmitted by one machine and received by all the machines on the network. This mode of operation is known as Broadcast Mode. Some Broadcast systems also support transmission to a sub-set of machines, something known as Multicasting.

Point-to-point networks

This type of network consists of many connections between individual pairs of machines. To go from the source to the destination, a packet of information on this type of network may have to first visit one or more intermediate machines. Often multiple routes, of different length are possible, so routing algorithms play an

important role in point-to-point networks. A network based on point-to-point communication is shown in Fig 2.

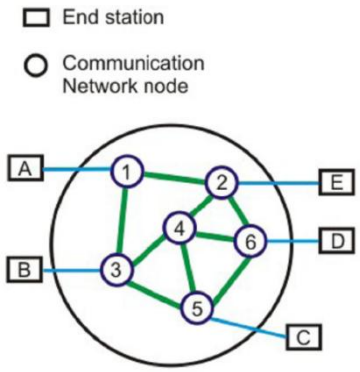


Figure: Communication network based on point-to-point communication

2.3 Analog and Digital Data

Data can be analog or digital. The term analog data refers to information that is continuous: digital data refers to the information that has discrete states. For example, consider an analog clock that has hour, minute, and second hands. These give information in a continuous form. The movements of the hands are continuous. Analog data such as the sound made by human voice, take on continuous values. When someone speaks, an analog wave is created in the air. This can be captured by a microphone and converted to an analog signal or sampled and converted to a digital signal. Eg., Video.

Digital data take on discrete values. For example, data are stored in computer memory in the form of 0s and 1s. They can be converted to digital signal or modulated into an analog signal for transmission across a medium. Eg., Text and Character Strings. A number of codes have been devised by which characters are represented by a sequence of bits. The most commonly used text code now a day is the International Reference Alphabet (IRA). The U.S National version of IRA is referred to as the American National Standard Code for Information Interchange (ASCII).

Analog and Digital Signals

Like the data they represent, signals can be either analog or digital. An analog signal has infinitely many levels of intensity over a period of time. As the wave moves from value A to value B, it passes through and includes an infinite number of values along its path. A digital signal, on the other hand, can have only a limited number of defined values. Although each value can be any number, it is often as simple as 1 and 0.

In a communication system, data are propagated from one point to another by means of electromagnetic signals. As we saw, analog signal is a continuously varying electromagnetic wave that may be propagated over a variety of media. A Digital signal is a sequence of voltage pulses that may be transmitted over a wire medium. Usually, a constant positive voltage represents a binary 0 and a constant negative voltage represents a binary 1.

Advantage of digital signaling is that it is cheaper than analog signaling. The disadvantage is that it suffers from more attenuation than analog signaling.

Analog and Digital Transmission

Both analog and digital signals may be transmitted on suitable transmission media. The way these signals are treated is a function of the transmission system.

Methods for Data Transmission

There are two methods involved in data transmission. One is Analog transmission and the other is Digital transmission. Analog transmission is a means of transmitting analog signals without regard to their content. In this case, the signal may represent analog or digital data. In both scenario's, the analog signal becomes weak (attenuate) after a certain distance.

Digital transmission is concerned with the content of the signal. Digital signals cannot travel long distances. Hence repeaters are used to regenerate the digital bit patterns. A repeater receives the digital signal, recovers the bit patterns of 0's and 1's and retransmits a new signal. By this, attenuation is overcome. The same technique may be used with an analog signal if it carries a digital data. Most of the telecommunication companies and service providers are switching over to digital transmission. The reasons are,

- Digital Technology
- Data Integrity
- Capacity Utilization
- Security and Privacy
- Integration

2.4 Data Communication

We all are acquainted with some sorts of communication in our day to day life. For communication of information and messages we use telephone and postal communication systems. Similarly data and information from one computer system can be transmitted to other systems across geographical areas. Thus data transmission is the movement of information using some standard methods. These methods include electrical signals carried along a conductor, optical signals along an optical fibers and electromagnetic areas.

Suppose a manager has to write several letters to various clients. First he has to use his PC and Word Processing package to prepare the letter, if the PC is connected to all the client's PC through networking, he can send the letters to all the clients within minutes. Thus irrespective of geographical areas, if PCs are connected through communication channel, the data and information, computer files and any other programs can be transmitted to other computer systems within seconds. The modern form of communication like e-mail and Internet is possible only because of computer networking.

Basic Elements of a Communication System

The following are the basic requirements for working of a communication system.

1. The sender (source) who creates the message to be transmitted
2. A medium that carries the message
3. The receiver (sink) who receives the message

In data communication four basic terms are frequently used. They are:

- Data: A collection of facts in raw forms that become information after processing.
- Signals: Electric or electromagnetic encoding of data.
- Signalling: Propagation of signals across a communication medium.
- Transmission: Communication of data achieved by the processing of signals.

Data Transmission Modes

There are three ways for transmitting data from one point to another as shown in Fig. 5.1

1. Simplex: In simplex mode the communication can take place in one direction. The receiver receives the signal from the transmitting device. In this mode the flow of information is Uni-directional. Hence it is rarely used for data communication.

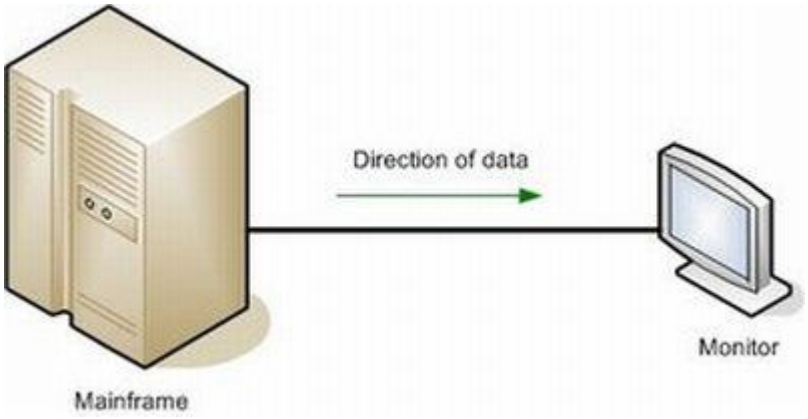


Figure. 3 Simplex

2. Half-duplex: In half-duplex mode the communication channel is used in both directions, but only in one direction at a time. Thus a half-duplex line can alternately send and receive data.

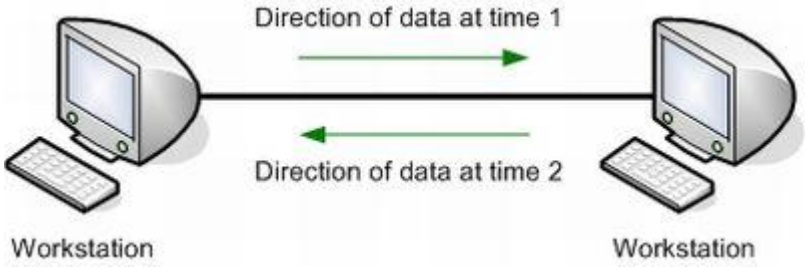


Figure.4: Half Duplex

3. Full-duplex: In full duplex the communication channel is used in both directions at the same time. Use of full-duplex line improves the efficiency as the line turnaround time required in half-duplex arrangement is eliminated. Example of this mode of transmission is the telephone line.

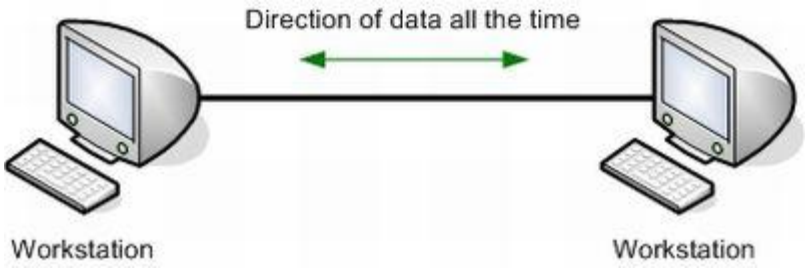


Figure.5: Full Duplex

2.5 Base Band, Broadband

2.5.1 Baseband Transmission

Baseband systems use digital signalling over a single channel. Signals flow in the form of discrete pulses of electricity or light. Figure 6 shows a baseband transmission with a bidirectional digital wave. With baseband transmission, the entire communication channel capacity is used to transmit a single data signal. The digital signal uses the complete bandwidth of the cable, which constitutes a single channel. The term bandwidth refers to the data transfer capacity, or speed of transmission, of a digital communications system as measured in bits per second (bps).



Figure 6: Baseband transmissions showing digital wave.

As the signal travels along the network cable, it gradually decreases in strength and can become distorted. If the cable length is too long, the received signal can be unrecognizable or misinterpreted.
As a safeguard, baseband systems sometimes use repeaters to receive incoming signals and retransmit them at their original strength and definition.
This increases the practical length of a cable.

2.5.2 Broadband Transmission

Broadband systems, as shown in Figure 7, use analog signalling and a range of frequencies. With analog transmission, the signals are continuous and no discrete. Signals flow across the physical medium in the form of electromagnetic or optical waves. With broadband transmission, signal flow is unidirectional.



Figure 7: Broadband Transmission.

If sufficient total bandwidth is available, multiple analog transmission systems, such as cable television and network transmissions, can be supported simultaneously on the same cable.
Each transmission system is allocated a part of the total bandwidth. All devices associated with a given transmission system, such as all computers using a LAN cable, must then be tuned so that they use only the frequencies that are within the allocated range.
While baseband systems use repeaters, broadband systems use amplifiers to regenerate analog signals at their original strength.
In broadband transmission, signals flow in one direction only, so there must be two paths for data flow in order for a signal to reach all devices. There are two common ways to do this:
1 Through mid-split broadband configuration, the bandwidth is divided into two channels, each using a different frequency or range of frequencies. One channel transmits signals; the other receives signals.
2 In dual-cable broadband configuration, each device is attached to two cables. One cable is used to send, and the other is used to receive.

2.6 Asynchronous and Synchronous Transmission

Data transmission through a medium can be either asynchronous or synchronous. In asynchronous transmission data is transmitted character by character as you go on typing on a keyboard. Hence there are irregular gaps between characters. However, it is cheaper to implement, as you do not have to save the data before sending. On the other hand, in the synchronous mode, the saved data is transmitted block by block. Each block can contain many characters. Synchronous transmission is well suited for remote communication between a computer and related devices like card reader and printers.

Self Assessment Questions

1. Transmission media are usually categorized as……..
(a) Fixed or unfixed (b) Guided or unguided

(c) Determinate or indeterminate (d). Metallic or nonmetallic

2. Radio communication frequencies range from……
(a). 3 KHz to 300 KHz (b). 300 KHz to 3 GHz
(c). 3 KHz to 300 GHz (d). 3 KHz to 3000 GHz

3. In fiber optics, the signal source is……waves.
(a) Light (b) Radio
(c) Infrared (d) Very low frequency

4. As frequency increase, the period ……
(a) Doubles (b)Increases
(c) Remains the same (d) Decreases

5. In cellular telephony, a service area is divided into small regions called……
(a) Cell (b) Cell offices
(c) MTSOs (d) Relay sites

2.7 Comparison of Base band and Broadband Signalling

S.N.	Parameter	Baseband	Broadband
1	Signalling type used	Digital signalling.	Analog signalling
2	Frequency spectrum	Entire frequency spectrum of the cable is consumed by a digital signal.	Frequency spectrum of cable is divided into sections of bandwidths.
3	Number of channels	Single channels	Multiple channels for data audio signals and video signals
4	Frequency division multiplexing	FDM is not possible	FDM is possible
5	Direction of transmission	A signal inserted at any point on the medium propagates in both direction i.e. bidirectional.	Signal is unidirectional
6	Length(Distance)	Only up to few kilometres	Up to tens of kilometres.

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: A line has a signal-to-noise ratio of 1000 and a bandwidth of 4000 KHz. What the maximum data rate supported by this line?

……..……

……..……

……...

Exercise: Check Your Progress 2

Case Study-History of Network Transmission

AT&T built its original long distance network from copper wire strung on telephone poles. Telephone calls travelled down these wires as analog signals, electrical waves of analogous form to the original voices. Each call required two wires to form a complete electrical circuit. Telephone signals weaken from electrical resistance as they travel down the wires. But thicker wires have lower resistance, so wires as thick as 1/6 inch were used on circuits such as the original New York-Chicago line of 1892.

After 1904, properly spaced loading coils, which effectively reduce the resistance in the line, allowed for longer lines and thinner wires. Vacuum-tube repeaters, introduced in 1914, made it possible to amplify or strengthen the signals, allowing for still longer lines and still thinner wires. An electrical trick called a "phantom circuit" allowed two pairs of wires to carry three calls.

Demand grows

In the 1910s, AT&T developed several new technologies to meet the growing demand on major routes. Underground cables carried more wires in less space and provided protection from the weather. AT&T installed its first underground cable between Philadelphia and Washington, D.C., in 1912. Carrier-current systems sent several calls down a single pair of wires by superimposing the calls on higher frequency currents, rather than transmitting the signals on their natural voice frequencies. AT&T installed its first carrier system between Baltimore and Pittsburgh in 1918. That system carried four calls on a single pair of wires.

Broadband begins

Still higher calling volume, and the beginning of work on television, fueled AT&T's invention of the first broadband transmission medium, broadband copper coaxial cable. AT&T installed its first experimental coaxial cable between New York and Philadelphia in 1936. The first "regular" installation connected Minneapolis, Minn., and Stevens Point, Wis., in 1941. This L1 coaxial-cable system could carry 480 telephone conversations or one television program. Subsequent coaxial-cable systems had much higher capacity. The L5 systems of the 1970s could carry 132,000 calls or more than 200 television programs.

Microwave radio relay

Coaxial-cable systems developed in tandem with microwave radio relay, a broadband system by which conversations and television travelled via radio along a series of towers. The first such system, with seven towers on seven hilltops, connected New York and Boston in 1947. This system carried 2,400 conversations; later systems carried as many as 19,200.

Microwave relay had lower construction and maintenance costs than coaxial cable, particularly across difficult terrain. By the 1970s, radio-relay systems carried 70% of AT&T's voice and 95 % of its television traffic.

Fiber-optic systems

In the 1980s, both coaxial cable and microwave relay gave way to an entirely new system - fiber-optics. Fiber-optic systems use rapid pulses of light traveling on fibers of ultra-pure glass. It was a digital rather than an analog medium, and particularly well suited for transmitting data as well as voice.

Glass fibers, as thin as a human hair, make up the highways for modern fiber optic communication systems.

AT&T installed its first fiber-optic route between Washington, D.C., and New York in 1983. In 1989, AT&T announced that it would retire all its analog transmission facilities. Within a few years, the analog coaxial-cable and radio-relay systems were relegated to back-up duty. Meanwhile, continuing advances in fiber-optic technology greatly increased the capacity of the new systems, a process that continues today.

Questions

1. Discuss the brief history of Network Transmission.
2. Write the history of broadband**.**

2.8 Summary

- Signals travel from transmitter to receiver via a path. This path, called the medium, guided or unguided.
- A guided medium is contained within physical boundaries, while an unguided, medium is boundless.
- Radio waves used to transmit data. These waves use unguided are usually propagated through the air.
- Fiber-optic cables are composed of a glass or plastic inner core surrounded by cladding, all encased in an outside jacket.
- Satellite communication uses a satellite in geosynchronous orbit to relay signals. A system of three correctly spaced satellites covers most of the earth.
- The Shannon capacity is a formula to determine the theoretical maximum data rate for a channel.

2.9 Keywords

Cellular telephony: Cellular telephony is moving fast toward integrating the existing system with satellite communication.

Guided media: It provides a conduit from one device to another; include twisted-pair cable, coaxial cable, and fiber-optic cable. A signal travelling along any of these media is directed and contained by the physical limits of the medium.

Optical fiber: Optical fiber is a glass or plastic cable that accepts and transports signals in the form of light.

Reflection: When the angle of incidence becomes greater than the critical angle, a new phenomenon occur called reflection.

Satellite transmission: Satellite transmission is much like line-of-sight microwave transmission in which one of the stations is a satellite orbiting the earth.

2.10 Review Questions

1. Which parts of the electromagnetic spectrum are used for communication?
2. Name the two major categories of transmission media.
3. How do guided media differ from unguided media?
4. What are the three major classes of guided media?
5. What is the major advantage of shielded twisted pair over unshielded twisted pair?
6. Why is coaxial cable superior to twisted-pair cable?
7. What happens to a beam of light as it travels to a less dense medium? What happens if it travels to a denser medium?
8. What are the three criteria used to evaluate transmission media?
9. What is reflection?
10. Discuss the modes for propagating light along optical channels.

Answers for Self Assessment Questions

1. (b) 2. (c) 3. (b) 4. (d) 5. (a)

2.11 Further Studies

1. Digital Transmission Systems by David R. Smith
2. Computer Networks: A Systems Approach by Larry L. Peterson, Bruce S. Davie

3

W OSI Model

STRUCTURE

3.0 Objectives

3.1 Introduction

3.2 Open System Interconnection Model

3.3 Functions of the OSI Layers

3.4 Reference models

3.5 Summary

3.6 Keywords

3.7 Review Questions

3.8 Further Studies

3.0 Objectives

After studying this chapter, you will be able to:

- Explain the open system interconnection model
- Discuss the functions of the ISO/OSI layers

3.1 Introduction

An ISO standard that covers all aspects of network communications is the Open Systems Interconnection (OSI) model. An open system is a model that allows any two different systems to communicate regardless of their underlying architecture. Vendor-specific protocols close off communication between unrelated systems. The purpose of the OSI model is to open communication between different systems without requiring changes to the logic of the underlying hardware and software. The OSI model is not a protocol; it is a model for understanding and designing a network architecture that is flexible, robust, and interoperable.

3.2 Open System Interconnection Model

The Open Systems Interconnection model is a layered framework for the design of network systems that allows for communication across all types of computer systems. It consists of seven separate but related layers, each of which defines a segment of the process of moving information across a network (see Figure 3.1). Understanding the fundamentals of the OSI model provides a solid basis for exploration of data communication.

3.2.1 Layered Architecture

The OSI model is built of seven ordered layers: physical (layer 1), data link (layer 2), network (layer 3), transport (layer 4), and session (layer 5), and presentation (layer 6), and application (layer 7). Figure 3.2 shows the layers involved when a message is sent from device A to device B. As the message travels from A to B, it may pass through many intermediate nodes. These intermediate nodes usually involve only the first three layers of the OSI model. In developing the model, the designers distilled the process of transmitting data down to its most fundamental elements. They identified which networking function had related used and collected those factions into discrete groups that became the layer.

Each layer defines a family of function distinct from those of the other the layer. By defining and localizing function the designers created an architecture that is both comprehensive and flexible. Most important, the OSI model allows complete transparency between otherwise incompatible systems.

A mnemonic for remembering the layers of the OSI model is: *"Please Do Not Touch Steve's Pet Alligator*" (Physical, Data Link, Network, Transport, Session, Presentation, and Application).

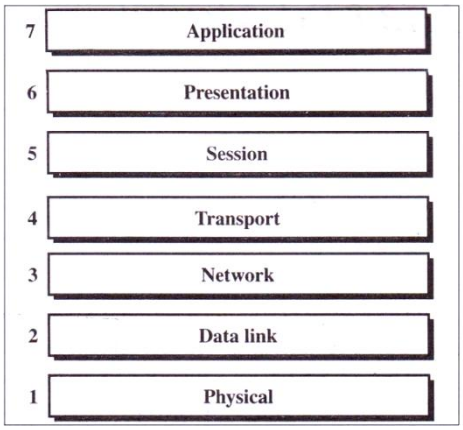


Figure 3.1: The OSI model.

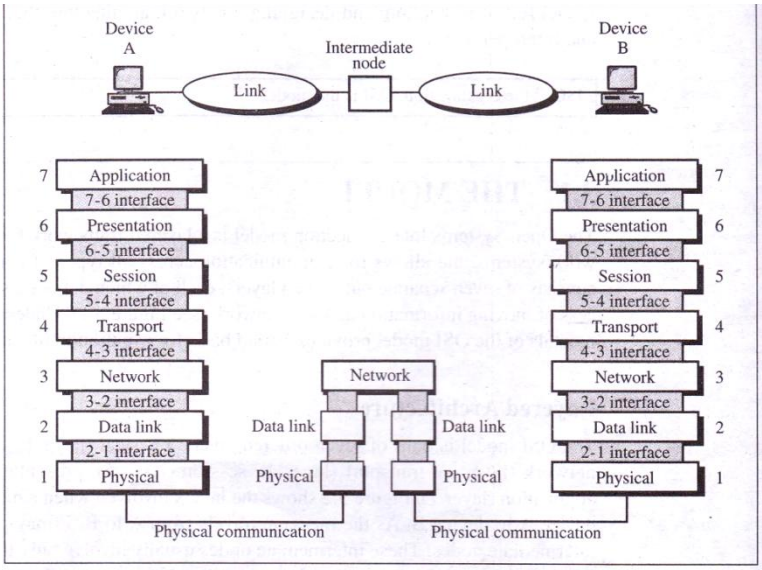


Figure 3.2: OSI layers.

Did You Know?

The International Standards Organization (ISO)) is a multinational body dedicated to worldwide agreement on international standards established in 1947.

3.2.2 Peer-to-Peer Processes

Within a single machine, each layer calls upon the services of the layer just below it. Layer 3, for example, uses the services provided by layer 2 and provides services for layer 4. Between machines, layer x on one machine communicates with layer x on another machine. This communication is governed by an agreed-upon series of rules and conventions called protocols. The processes on each machine that communicate at a given layer are called peer-to-peer processes. Communication between machines is therefore a peer-to-peer process using the protocols appropriate to a given layer.

At the physical layer, communication is direct: Machine A sends a stream of bits to machine B. At the higher layers, however, communication must move down through the layers on machine A, over to machine B, and then back up through the layers. Each layer in the sending machine adds its own information to the message it receives from the layer just above it and passes the whole package to the layer just below it. This information is added in the form of headers or trailers (control data added to the beginning or end of a data parcel). Headers are added to the message at layers 6, 5, 4, 3, and 2. A trailer is added at layer 2.

Headers are added to the data at layers 6, 5, 4, 3, and 2. Trailers are usually added only at layer 2.

At layer 1 the entire package is converted to a form that can be transferred to the receiving machine. At the receiving machine, the message is unwrapped layer by layer, with each process receiving and removing the data meant for it. For example, layer 2 removes the data meant for it, and then passes the rest to layer 3. Layer 3 removes the data meant for it and passes the rest to layer 4, and so on.

3.2.3 Interfaces between Layers

The passing of the data and network information down through the layers of the sending machine and back up through the layers of the receiving machine is made possible by an interface between each pair of adjacent layers. Each interface defines what information and services a layer must provide for the layer above it. Well-defined interfaces and layer functions provide modularity to a network. As long as a layer still provides the expected services to the layer above it, the specific implementation of its functions can be modified or replaced without requiring changes to the surrounding layers.

3.2.4 Organization of the Layers

The seven layers can be thought of as belonging to three subgroups. Layers 1, 2, and 3-physical, data link, and network-are the network support layers; they deal with the physical aspects of moving data from one device to another (such as electrical specifications, physical connections, physical addressing, and transport timing reliability). Layers 5, 6, and 7-session, presentation, and application-can be thought of as the user support layers; they allow interoperability among unrelated software systems. Layer 4, the transport layer, ensures end-to-end reliable data tr. mission while layer 2 ensures reliable transmission on a single link. The upper OSI layers are almost always implemented in software; lower layers are a combination of hardware and software, except for the physical layer, which is mostly hardware.

In Figure 3.3, which gives an overall view of the OSI layers, L7 data means data unit at layer 7, L6 data means the data unit at layer 6, and so on. The process starts out at layer 7 (the application layer), then moves from layer to layer in descending sequential order. At each layer (except layers 7 and 1), a header is added to the data unit. At layer 2, a trailer is added as well. When the formatted data unit passes through the physical layer (layer 1), it is changed into an electromagnetic signal and transported along a physical link.

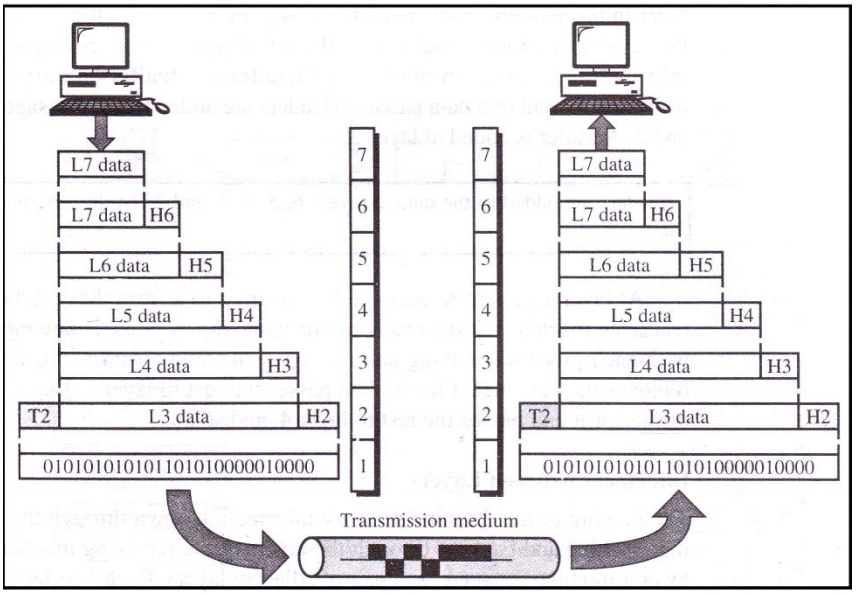


Figure 3.3: An exchange using the OSI model.

Upon reaching its destination, the signal passes into layer 1 and is transformed back into bits. The data units then move back up through the OSI layers. As each block of data reaches the next higher layer, the headers and trailers attached to it at the corresponding sending layer are removed, and actions appropriate to that layer are taken. By the time it reaches layer 7, the message is again in a form appropriate to the application and is made available to the recipient.

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: Draw the block diagram of OSI Model based on working.

……..……

……..……

……...

3.3 Functions of the OSI Layers

In this section we briefly describe the functions of each layer in the OSI model.

3.3.1 Physical Layer

The physical layer coordinates the functions required to transmit a bit stream over a physical medium. It deals with the mechanical and electrical specifications of the interface and transmission medium. It also defines the procedures and functions that physical devices and interfaces have to perform for transmission to occur. Figure 3.4 shows the position of the physical layer with respect to the transmission medium and the data link layer.

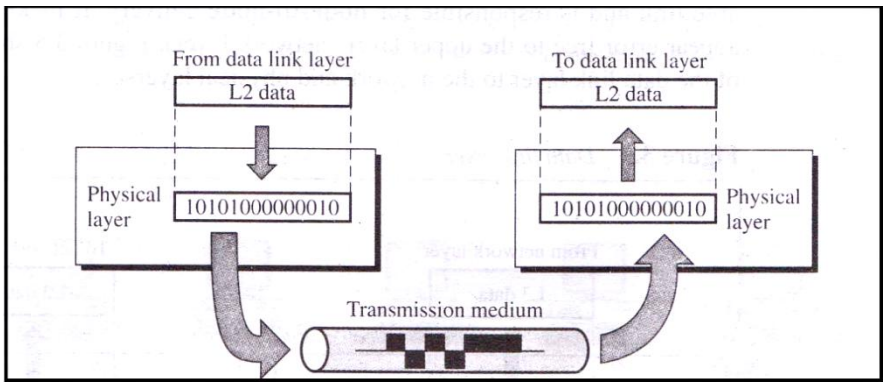


Figure 3.4: Physical layer.

The physical layer is concerned with the following:

- ***Physical characteristics of interfaces and media:*** The physical layer defines the characteristics of the interface between the devices and the transmission medium. It also defines the type of transmission medium.
- ***Representation of bits***: The physical layer data consist of a stream of bits (sequence of 0's and 1's) without any interpretation. The physical layer defines the type of encoding (how 0's and 1's are changed to signals).
- ***Data rate***: The transmission rate the number of bits sent each second-is also defined by the physical layer. In other words, the physical layer defines the duration of a bit, which is how long it lasts.
- ***Synchronization of bits***: The sender and receiver must be synchronized at the bit level. In other words, the sender and the receiver clocks must be synchronized.
- ***Line configuration***: The physical layer is concerned with the connection of devices to the medium. In a point-to-point configuration, two devices are connected together through a dedicated link. In a multipoint configuration, a link is shared between several devices.
- ***Physical topology:*** The physical topology defines how devices are connected to make a network. Devices can be connected using a mesh topology (every device connected to every other device), a star topology (devices are connected through a central device), a ring topology (every device is connected to the next, forming a ring), or a bus topology (every device on a common link).
- ***Transmission mode:*** The physical layer also defines the direction of transmission between two devices: simplex, half-duplex, or full-duplex. In the simplex mode, only one device can send; the other can only receive. The simplex mode is at one way communication. In the half-duplex mode, two devices can send and receive, but not at the same time. In a full-duplex (or simply duplex) mode, two devices can send and receive at the same time.

Caution

While synchronization of bits, bits must be encoded into signals-electrical or optical.

3.3.2 Data Link Layer

The data link layer transforms the physical layer, a raw transmission facility, to a reliable link and is responsible for node-to-node delivery. It makes the physical layer appear error free to the upper layer (network layer). Figure 3.5 shows the relationship of the data link layer to the network and physical layers.

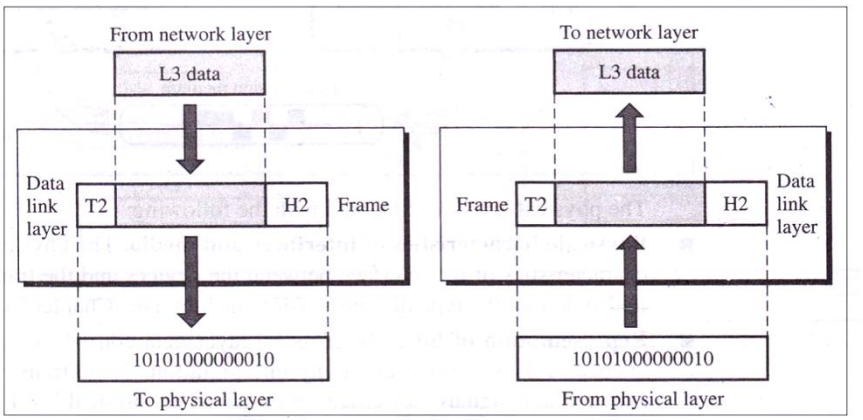


Figure 3.5: Data Link Layer.

Specific responsibilities of the data link layer include the following:

Framing: The data link layer divides the stream of bits received from the network layer into manageable data units called frames.

Physical addressing: If frames are to be distributed to different systems on the network, the data link layer adds a header to the frame to define the physical address of the sender (source address) and/or receiver (destination address) of the frame. If the frame is intended for a system outside the sender's network, the receiver address is the address of the device that connects one network to the next.

Flow control. If the rate at which the data are absorbed by the receiver is less than the rate produced in the sender, the data link layer imposes a flow control mechanism to prevent overwhelming the receiver.

Error control: The data link layer adds reliability to the physical layer by adding mechanisms to detect and retransmit damaged or lost frames. It also uses a mechanism to prevent duplication of frames. Error control is normally achieved through a trailer added to the end of the frame.

Access control: When two or more devices are connected to the same link, data link layer protocols are necessary to determine which device has control over the link at any given time.

Example:

In Figure 3.6 a node with physical address 10 sends a frame to a node with physical address 87. The two nodes are connected by a link. At the data link level this frame contains physical (link) addresses in the header. These are the only addresses needed. The rest of the header contains other information needed at this level. The trailer usually contains extra bits needed for error detection.

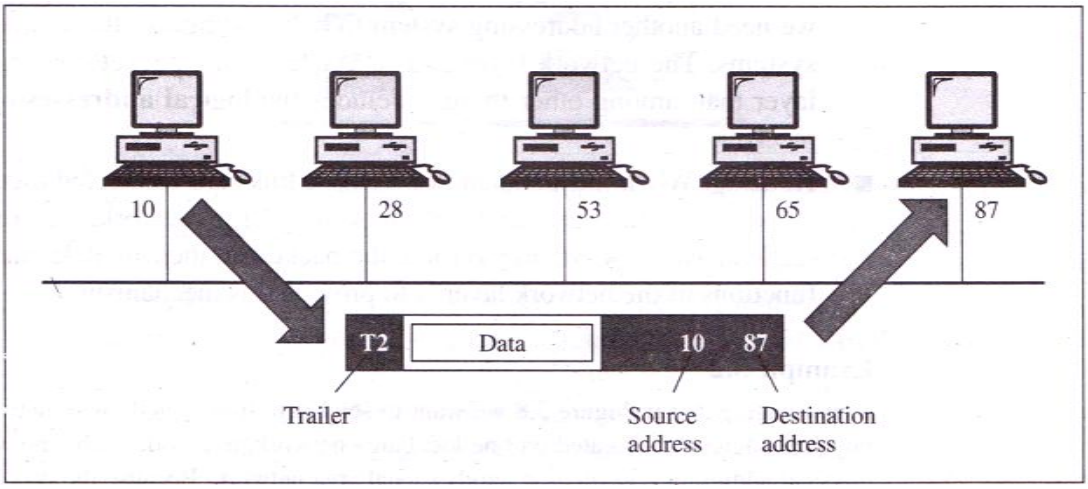


Figure 3.6: Node with physical address.

3.3.3 Network Layer

The network layer is responsible for the source-to-destination delivery of a packet possibly across multiple networks (links). Whereas the data link layer oversees the delivery of the packet between two systems on the same network (links), the network layer ensures that each packet gets from its point of origin to its final destination. If two systems are connected to the same link, there is usually no need for a network layer. However, if the two systems are attached to different networks (links) with connecting devices between the networks, (links), there is often a need for the network layer to accomplish source-to-destination delivery. Figure 3.7 shows the relationship of the network layer to the data link and transport layers.

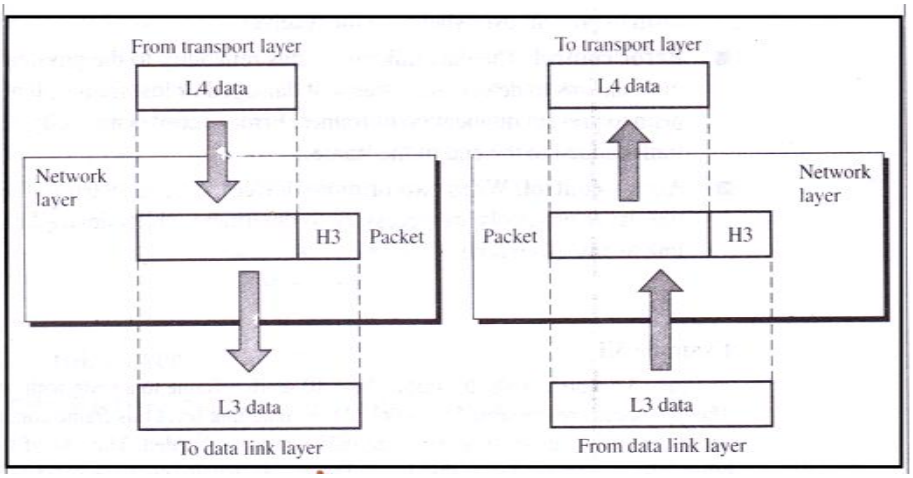


Figure 3.7: Network layer.

Specific responsibilities of the network layer include the following:

Logical addressing: The physical addressing implemented by the data link layer handles the addressing problem locally. If a packet passes the network boundary, we need another addressing system to help distinguish the source and destination systems. The network layer adds a header to the packet coming from the upper layer that, among other things, includes the logical addresses of the sender and receiver.

Routing: When independent networks or links are connected together to create an internetwork (a network of networks) or a large network, the connecting devices (called routers or gateways) route the packets to their final destination. One of the functions of the network layer is to provide this mechanism.

Example:

Now imagine that in Figure 3.8 we want to send data from anode with network address A and physical address 10, located on one local area network, to a node with a network address P and physical address 95, and located on another local area network. Because the two devices are located on different networks, we cannot use physical addresses only; the physical addresses have only local jurisdiction. What we need here are universal addresses that can pass through the bound arise of local area networks. The network (logical) addresses have this characteristic. The packet at the network layer contains the logical addresses, which remain the same from the original source to the final destination. They will not change when we go from network to network. However, the physical addresses will change when the packet moves from one network to another. The box with the R is a router (internetwork device).

3.3.4 Transport Layer

The transport layer is responsible for source-to-destination (end-to-end) delivery of the entire message. Whereas the network layer oversees end-to-end delivery of individual packets, it does not recognize any

relationship between those packets. It treats each one independently, as though each piece belonged to a separate message, whether or not it does. The transport layer, on the other hand, ensures that the whole message arrives intact and in order, overseeing both error control and flow control at the source to-destination level. Figure 3.8 shows the relationship of the transport layer to the network and session layers.

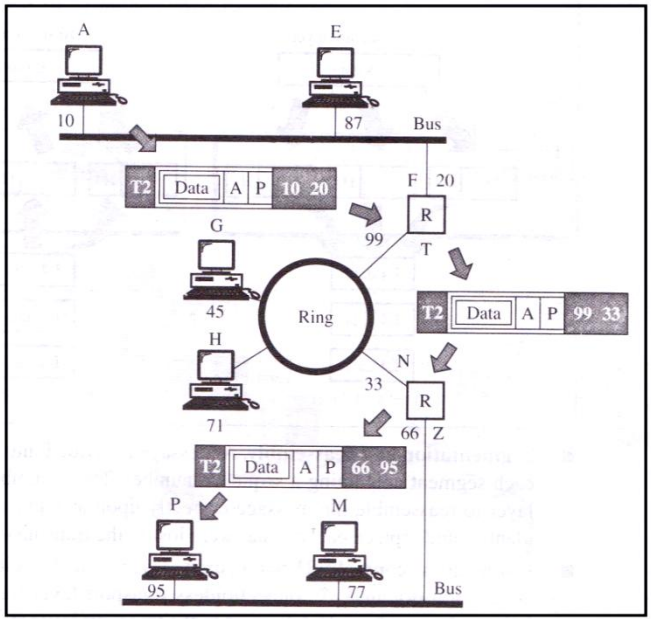


Figure 3.8: Network layer.

For added security, the transport layer may create a connection between the two end ports. A connection is a single logical path between the source and destination that is associated with all packets in a message. Creating a connection involves three steps: connection establishment, data transfer, and connection release. By confining transmission of all packets to a single pathway, the transport layer has more control over sequencing, flow, and error detection and correction.

Specific responsibilities of the transport layer include the following:

Service-point addressing: Computers often run several programs at the same time. For this reason, source-to-destination delivery means delivery not only from one computer to the next but also from a specific process (running program) on one computer to a specific process (running program) on the other. The transport layer header therefore must include a type of address called a service-point address (or port address). The network layer gets each packet to the correct computer; the transport layer gets the entire message to the correct process on that computer.

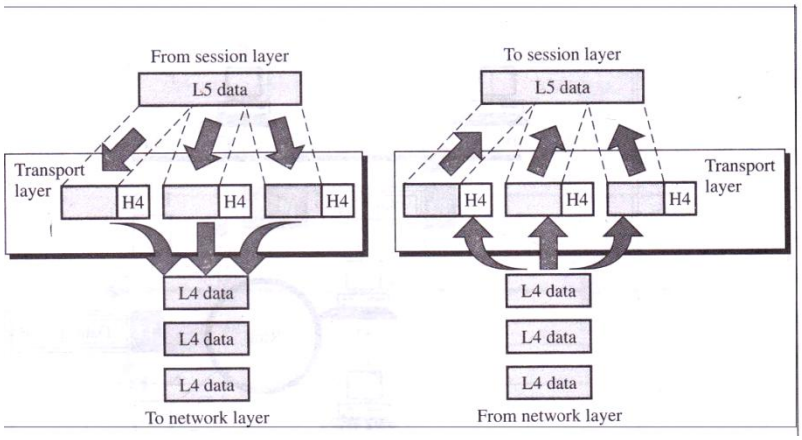


Figure 3.9: Transport layer.

Segmentation and reassembly: A message is divided into transmittable segments, each segment containing a sequence number. These numbers enable the transport layer to reassemble the message correctly upon arriving at the destination and to identify and replace packets that were lost in the transmission.

Connection control: The transport layer can be either connectionless or connection-oriented. A connectionless transport layer treats each segment as an independent packet and delivers it to the transport layer at the destination machine. A connection-oriented transport layer makes a connection with the transport layer at the destination machine first before delivering the packets. After all the data are transferred, the connection is terminated.

Flow control. Like the data link layer, the transport layer is responsible for flow control. However, flow control at this layer is performed end to end rather than across a single link.

Error control: Like the data link layer, the transport layer is responsible for error control. However, error control at this layer is performed end to end rather than across a single link. The sending transport layer makes sure that the entire message arrives at the receiving transport layer without error (damage, loss, or duplication). Error correction is usually achieved through retransmission.

Example:

Figure 3.10 shows an example of a transport layer. Data coming from the upper layers have service-point (port) addresses j and k (j is the address of the sending application and k is the address of the receiving application). Since the data size is larger than the network layer can handle, the data are split into two packets, each packet retaining the service-point addresses (j and k). Then in the network layer, network addresses (A and P) are added to each packet. The packets may travel on different paths and arrive at the destination either in order or out of order. The two packets are delivered to the destination network layer, which is responsible for removing the network layer headers. The two packets are now passed to the transport layer, where they are combined for delivery to the upper layers.

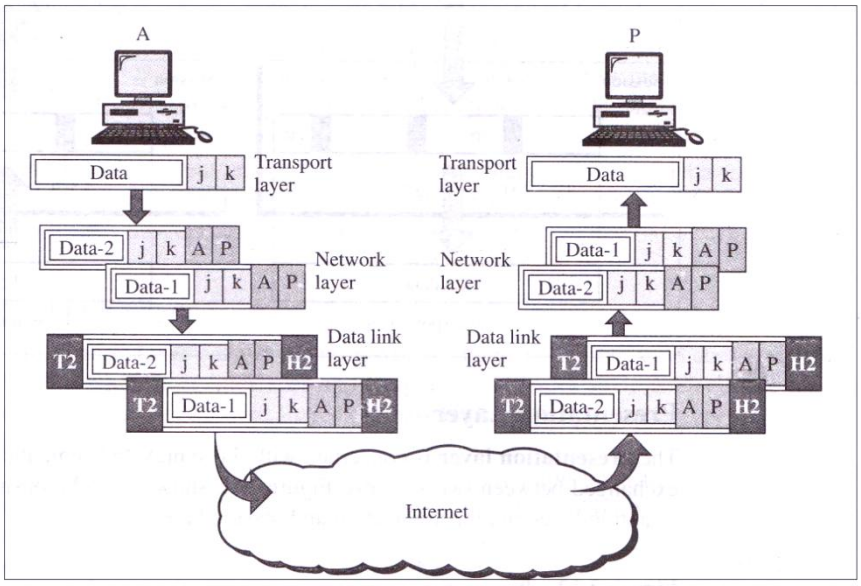


Figure 3.10: Transport layer.

3.3.5 Session Layer

The services provided by the first three layers (physical, data link, and network) are not sufficient for some processes. The session layer is the network dialog controller. It establishes, maintains, and synchronizes the interaction between communicating systems.

Specific responsibilities of the session layer include the following:

Dialog control: The session layer allows two systems to enter into a dialog. It allows the communication between two processes to take place either in half-duplex (one way at a time) or full-duplex (two ways at a time). For example, the dialog between a terminal connected to a mainframe can be half-duplex.

Synchronization: The session layer allows a process to add checkpoints (synchronization points) into a stream of data. For example, if a, system is sending a file of 2000 pages, it is advisable to insert checkpoints after every 100 pages to ensure that each 100-page unit is received and acknowledged independently. In this case, if a crash happens during the transmission of page 523, retransmission begins at page 501: pages 1 to 500 need not be retransmitted. Figure 3.11 illustrates the relationship of the session layer to the transport and presentation layers.

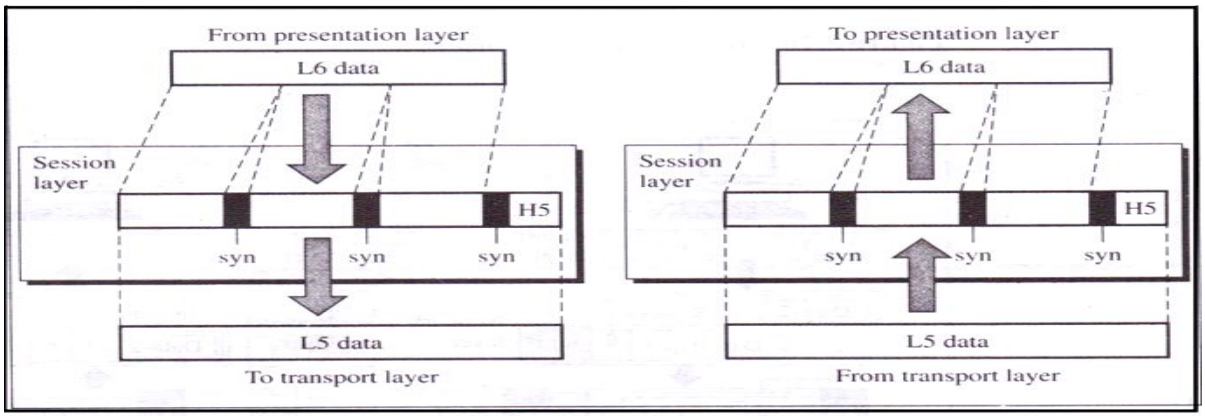


Figure 3.11: Session layer.

3.3.6 Presentation Layer

The presentation layer is concerned with the syntax and semantics of the information exchanged between two systems. Figure 3.12 shows the relationship between the presentation layer and the application and session layers.

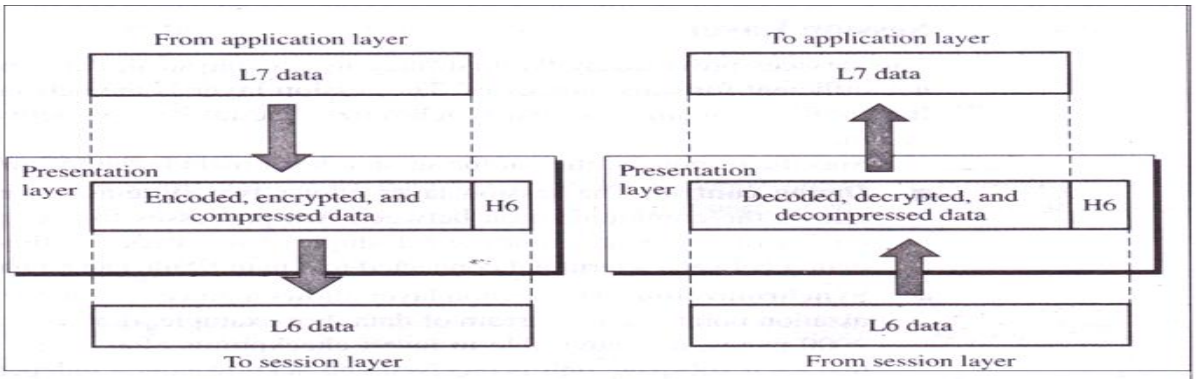


Figure 3.12 Presentation layer.

Specific responsibilities of the presentation layer include the following:

Translation: The processes (running programs) in two systems are usually exchanging information in the form of character strings, numbers, and so on. Because different computers use different encoding systems, the presentation layer is responsible for interoperability between these different encoding methods. The presentation layer at the sender changes the, information from its sender-dependent format into a common format. The presentation layer at the receiving machine changes the common format into its receiver-dependent format.

Encryption: To carry sensitive information, a system must be able to assure privacy. Encryption means that the sender transforms the original information to another form and sends the resulting message out over the network. Decryption reverses the original process to transform the message back to its original form.
Compression: Data compression reduces the number of bits to be transmitted. Data compression becomes particularly important in the transmission of multimedia such as text, audio, and video.

3.3.7 Application Layer

The application layer enables the user, whether human or software, to access the network. It provides user interfaces and support for services such as electronic mail, remote file access and transfer, shared database management, and other types of distributed information services.
Figure 3.13 shows the relationship of the application layer to the user and the presentation layer. Of the many application services available, the Figure 3.13 shows only three: X.400 (message-handling services); X.500 (directory services); and file transfer, access, and management (FTAM). The user in this example uses X.400 to send an e-mail message. Note that no headers or trailers are added at this layer:

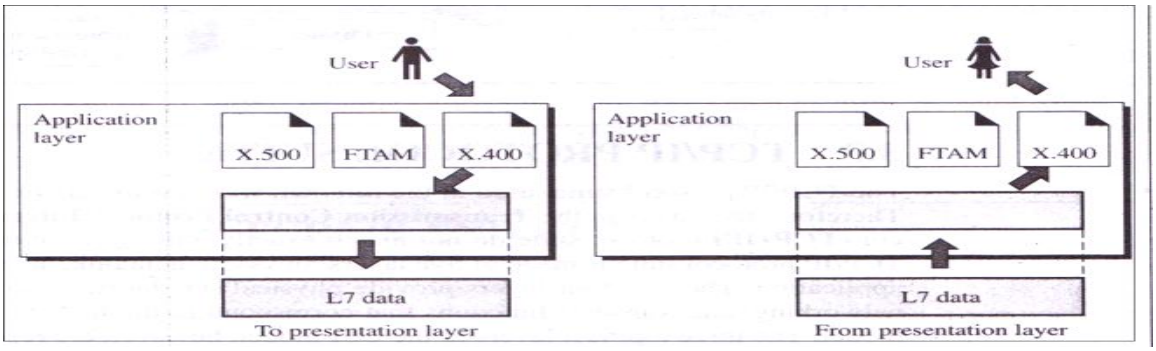


Figure 3.13: Application layer.

Specific services provided by the application layer include the following:
Network virtual terminal: A network virtual terminal is a software version of a physical terminal and allows a user to log on to a remote host. To do so, the application creates a software emulation of a terminal at the remote host. The user's computer talks to the software terminal, which, in turn, talks to the host, and vice; versa. The remote host believes it is communicating with one of its own terminals and allows you to log on.
File transfer, access, and management (FTAM): This application allows a user to access files in a remote computer (to make changes or read data), to retrieve files from a remote computer; and to manage or control files in a remote computer.
Mail services: This application provides the basis for e-mail forwarding and storage.
Directory services: This application provides distributed database sources and access for global information about various objects and services.

Exercise: Check Your Progress 2
Note: i) Use the space below for your answer.
Ex1: Prepare a diagram which shows the data communication concept of OSI layer.
……..……

……..……

……...

Did You Know?

The OSI model is essentially the same as the DSA (distributed systems architecture) model developed in 1977.

Summary of OSI Layer Functions

The functions of the seven layers are summarized in Figure 3.14.

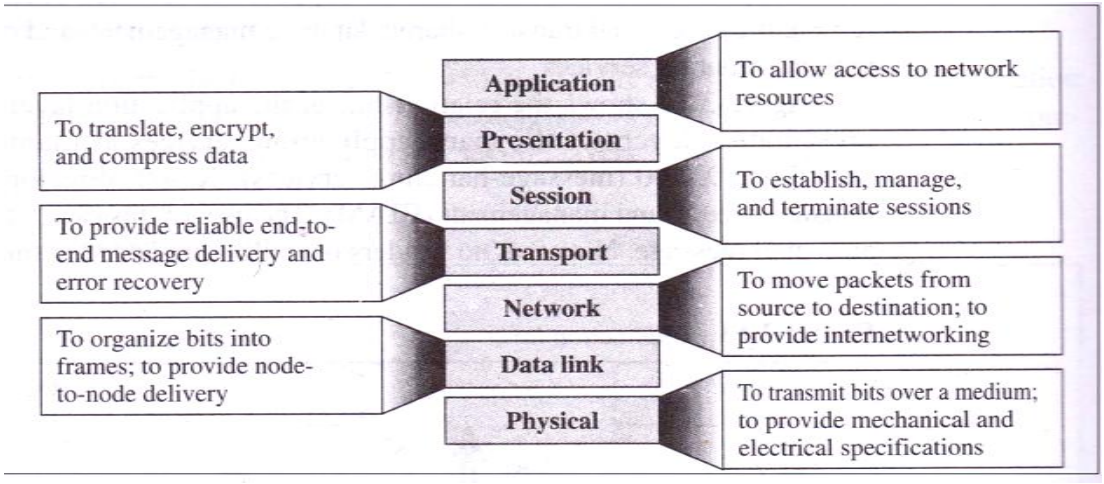


Figure 3.14: Summary of layer OSI functions.

Self Assessment Questions

1. The OSI model consists of…….layers.

(a). three (b). five

(c). seven (d). eight

2. The ……..layer decides the location of synchronization points.

(a). transport (b). session

(c). presentation (d). application

3. The end-to-end delivery of the entire message is the responsibility of the layer.

(a) network (b). transport

(c). session (d). presentation

4. The…….layer is the layer closest to the transmission medium.

(a). physical (b) data link

(c). network (d). transport

5. In the……..layer, the data unit is called a frame.

(a). physical (b) data link

(c). network (d). transport

6. Decryption and encryption of data are the responsibility of the……..layer.

(a) physical (b) data link

(c) presentation (d). Session

3.4 Reference models

Two major approaches:

- The seven-layer OSI/ISO model – Open Systems Interconnection, currently maintained by the International Organization for Standards.
- The five-layer TCP/IP model – Transmission Control Protocol/Internet Protocol.

OSI/ISO

- Physical layer – controls electrical and mechanical aspects of data transmission, e.g., voltage levels, cable lengths, and so on.
- Data-link layer – addresses the transmission of data frames (or packets) over a physical link between network entities, includes error correction.
- Network layer – establishes paths for data between computers and determines switching among routes between computers, determines how to disaggregate messages into individual packets.
- Transport layer – deals with data transfer between end systems and determines flow control.
- Session layer – creates and manages sessions when one application process requests access to another applications process, e.g., MS Word importing a spread sheet from Excel.
- Presentation layer – determines syntactic representation of data, e.g., agreement on character code like ASCII/Unicode.
- Application layer – establishes interface between a user and a host computer, e.g., searching in a database application.

TCP/IP

- Physical layer – not really part of this model, since TCP and IP deal with software; usually thought to refer to all hardware beneath the network layer.
- Network or data link layer – defined by whatever the Internet Protocol will run over, e.g., a token-ring network.
- Internet or network layer – provides network addressing and routing, providing a common address space and connecting heterogeneous networks. IP runs here.
- Transport layer – manages data-consistency by providing a reliable byte stream between nodes on a network. TCP and User Datagram Protocol (UDP) run here.
- Process and applications layer – provides application services to users and programs.

Exercise: Check Your Progress 3

Case study: OSI reference model

OSI (Open Systems Interconnect) refers to computer network architecture structured into layers.

The physical layer is concerned the transfer of electrical signals. The physical layer of a school network can be shown in a diagram of the network hardware.

The data link layer is concerned with the rules of data exchange. A student logging on to the school network does so via a username and a password.

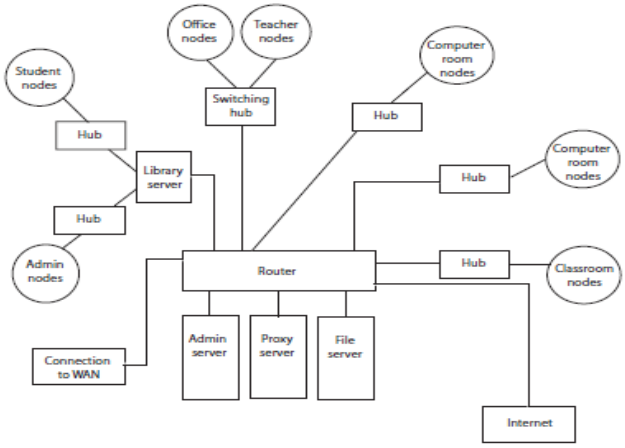
The network logon script controls the data link at logon. It establishes if a link exists to the tree, context and server. The user name and password need to be verified to determine if the student has access rights to the network.

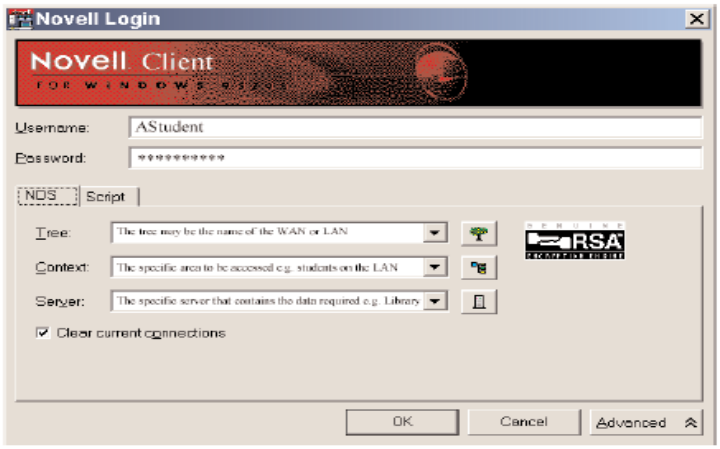
The network layer controls the communication procedures routing packets of data by the most efficient route available. This will depend on network traffic.

The transport layer checks to ensure that data is correctly received. This may involve the use of error checking methods such as parity and checksum.

The session layer works with the transport layer to allow the student access to the network files (according to the student's level of network access and the rights allocated to the student). This layer controls the access

while the student remains logged on to the school network. When the student logs off the network, the session is terminated and the access rights are cancelled until the next person logs on.
The presentation layer establishes protocols for the handling of the data display.
The application layer protocols control the transferring of files and manage the transfer. If the student requests word-processing files from their user directory, then the protocols for the word-processing program are activated and the files transferred according to the rules of that application.





Questions

1 Sketch a basic OSI model to show the various layers in the correct order.
2 Label the diagram to show the function of each layer.

3.5 Summary

- The International Standards Organization (ISO) created a model called the Open Systems Interconnection (OSI), which allows diverse systems to communicate.
- The seven-layer OSI model provides guidelines for the development of universally compatible architecture, hardware, and software.
- The transport layer links the network support layers and the user support layers.
- The data link layer is responsible for delivering data units from one station to the next without errors.

- The session layer establishes, maintains, and- synchronizes the interactions between communicating devices.
- The TCP/IP, a five-layer hierarchical protocol suite developed before the OSI model, is the protocol suite used in the Internet.

3.6 Keywords

Application layer: The application layer enables the users to access the network.
Network layer: The network layer is responsible for the source-to-destination delivery of a packet across multiple network links.
Physical layer: The physical layer coordinates the functions required to transmit a bit stream over a physical medium.
Presentation layer: The presentation layer ensures interoperability between communicating devices through transformation of data into a mutually agreed-upon format.
Transport layer: The transport layer is responsible for the source-to-destination delivery of the entire message.

3.7 Review Questions

1. Which OSI layers are the network support layers?
2. Which OSI layers are the user support layers?
3. Differentiate between network and transport layer delivery.
4. How are OSI and ISO related to each other?
5. List the layers of the OSI model.
6. How does information get passed from one OSI layer to the next?
7. What are headers and trailers and how do they get added and removed?
8. What are the concerns of the physical layer?
9. What are the responsibilities of the data link layer?
10. The transport layer creates a connection between the source and destination. What are the three events involved in a connection?

Answers for Self Assessment Questions

1. (c) 2. (b) 3. (b) 4. (a) 5. (b) 6. (c)

3.8 Review Questions

1. http://www.crammasteronline.com/data/cmdata/NETPLUSN10004/Books/ec2_netplus004c04.pdf
2. Computer Networks: A Systems Approach by Larry L. Peterson, Bruce S. Davie

4

Real World Networks

STRUCTURE

4.0 Objectives

4.1 Introduction

4.2 Ethernet

4.3 FDDI (Fiber Distributed Data Interface)

4.4 Network Operation

4.5 ATM (Asynchronous Transfer Mode)

4.6 ATM Service Categories

4.7 ARCNET

4.8 AppleTalk

4.9 Summary

4.10 Keywords

4.11 Review Questions

4.12 Further Studies

4.0 Objectives

After studying this chapter, you will be able to:

- Discuss the Ethernet
- FDDI (fiber distributed data interface)
- Discuss the network operation
- Define the ATM (asynchronous transfer mode
- Explain the ATM service categories

4.1 Introduction

The main interest of real world network has been in understanding the structural properties and patterns in the evolution of large graphs and networks. What does a “normal” social network look like? How will it evolve over time? How can we spot “abnormal” interactions (e.g., spam) in a time-evolving e-mail graph? The large networks can be divided into two parts:

- The study of statistical properties and models that govern the generation and evolution of large real-world networks. We view the network as a big complex system, observe its static and temporal properties and patterns to design models that capture and help us understand the temporal and static patterns of real-world networks.
- The study of the network by starting from individual nodes and small communities. We are especially interested in modeling the spread of influence and information over the network and the substructures of the network, called cascades, which this process creates. We aim to find common and abnormal sub-network patterns and understand the propagation of influence, information, diseases and computer viruses over the network. Once we know the propagation patterns and structure, we devise algorithms for efficiently finding influential nodes.

4.2 Ethernet

IEEE 802.3 supports a LAN standard originally developed by Xerox and la - extended by a joint venture between Digital Equipment Corporation, Intel Corporation, and Xerox. This was called Ethernet.

IEEE 802.3 defines two categories: baseband and broadband, as shown in Figure 4.1. The word base specifies a digital signal (in this case, Manchester encoding). The word broad specifies an analog signal (in this case, PSK encoding). IEEE divides the baseband category into five different standards: 10Base5, 10Base2. 10Base-T, l0Base5, and 100Base-T. The first number (10, 1, or 100) indicates the data rate in Mbps. The last number or letter (5, 2, 1, or T) indicates the maximum cable length or the type of cable. IEEE defines only one specification for the broadband category: lOBroad36. Again, the first number (10) indicates the data rate. The last number defines the maximum cable length. However, the maximum cable length restriction can be changed using networking devices such as repeaters or bridges.

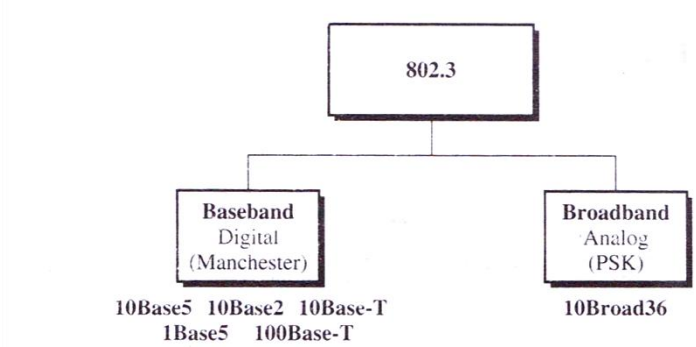


Figure 4.1: IEEE 802.3.

4.2.1 Access Method: CSMA/CD

Whenever multiple users have unregulated access to a single line, there is a danger of signals overlapping and destroying each other. Such overlaps, which turn the signals into unusable noise, are called collisions. As traffic increases on a multiple-access link, so do collisions. A LAN therefore needs a mechanism to coordinate traffic, minimize the number of collisions that occur, and maximize the number of frames that are delivered successfully. The access mechanism used in an Ethernet is called carrier sense multiple access with collision detection (CSMA/CD, standardized in IEEE 802.3).

The CSMA/CD is the result of an evolution from multiple accesses (MA) to carrier sense multiple accesses (CSMA), and, finally, to carrier senses multiple accesses with collision detection (CSMA/CD). The original design was a multiple access method in which every workstation had equal access to a link. In MA, there was no provision for traffic coordination. Access to the line was open to any node at any time, with the assumption that the odds of two devices competing for access at the same time were small enough to be unimportant. Any

station wishing to transmit did so, and then relied on acknowledgments to verify that the transmitted frame had not been destroyed by other traffic on the line.

In a CSMA system, any workstation wishing to transmit must first listen for existing traffic on the line. A device listens by checking for a voltage. If no voltage is detected, the line is considered idle and the transmission is initiated. CSMA cuts down on the number of collisions but does not eliminate them. Collisions can still occur. If another station has transmitted too recently for its signal to have reached the listening station, the listener assumes the line is idle and introduces its own signal onto the line.

The final step is the addition of collision detection (CD). In CSMA/CD the station wishing to transmit first listens to make certain the link is free, and then transmits its data, then listens again. During the data transmission, the station checks the line for the extremely high voltages that indicate a collision. If a collision is detected, the station quits the current transmission and waits a predetermined amount of time for the line to clear, then sends its data again.

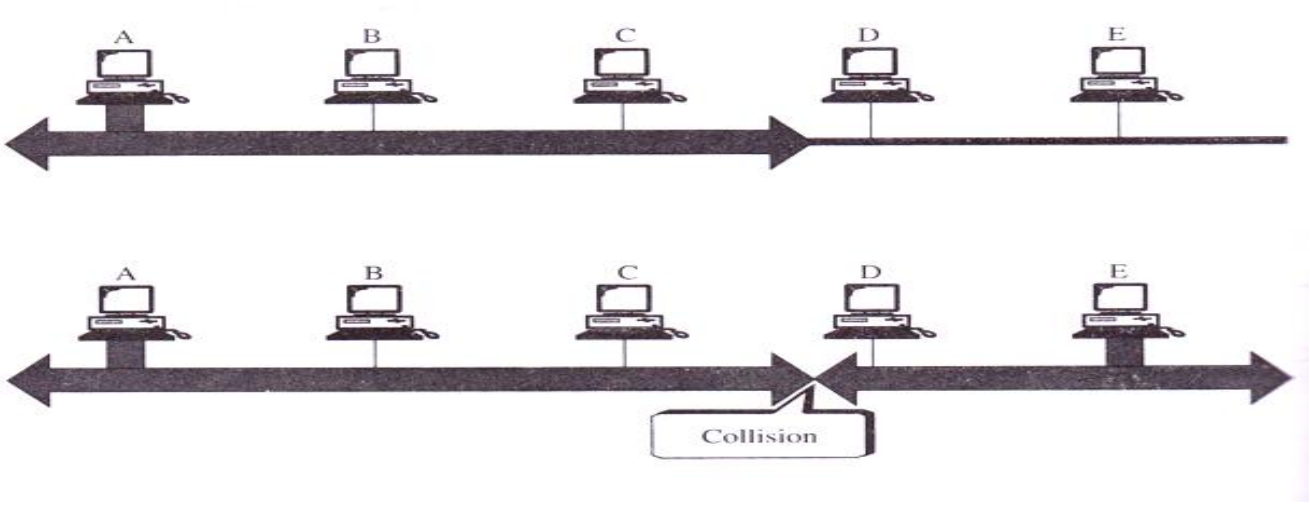


Figure 4.2: collision in CSMA/CD.

Addressing

Each station on an Ethernet network (such as a PC, workstation, or printer) has its network interface card (NIC). The NIC usually fits inside the station and provides station with a six-byte physical address. The number on the NIC is unique.

4.2.2 Electrical Specification

Signalling

The baseband systems use Manchester digital encoding .There is broadband system, 10Broad36. It uses digital/analog conversion (differential PSK).

Data Rate

Ethernet LANs can support data rates between 1 and 100 Mbps.

4.2.3 Frame Format

IEEE 802.3 specifies one type of frame containing seven fields: preamble, SFD, DA, SA, length/type of PDU, 802.2 frames, and the CRC. Ethernet does not provide any mechanism for acknowledging received frames, making it what is known as an unreliable medium. Acknowledgments must be implemented at the higher layers.

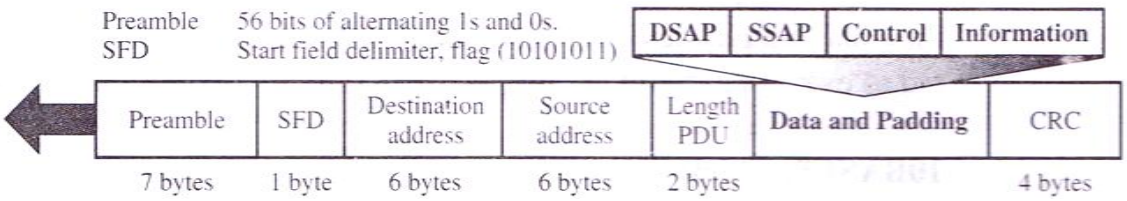


Figure 4.3: 802.3 MAC frame.

Preamble: The first field of the 802.3 frame, the preamble, contains seven bytes (56 bits) of alternating 0's and l's that alert the receiving system to the coming frame and enable it to synchronize its input timing. The pattern 1010101 provides only an alert and a timing pulse; it can be too easily aliased to be useful in indicating the beginning of the data stream. HDLC combined the alert, timing, and start synchronization into a single field: the flag. IEEE 802.3 divides these three functions between the preamble and the second field, the start frame delimiter (SFD).

Start frame delimiter (SFD): The second field (one byte: 10101011) of the 802.3 frame signals the beginning of the frame. The SFD tells the receiver that everything that follows is data, starting with the addresses.

Destination addresses (DA): The destination address (DA) field is allotted six bytes and contains the physical address of the packet's next destination. A system's physical address is a bit pattern encoded on its network interface card (NIC). Each NIC has a unique address that distinguishes it from any other NIC. If the packet must cross from one LAN to another to reach its destination, the DA field contains the physical address of the router connecting the current LAN to the next one. When the packet reaches the target network, the DA field contains the physical address of the destination device.

Source address (SA): The source address (SA) field is also allotted six bytes and contains the physical address of the last device to forward the packet. That device can be the sending station or the most recent router to receive and forward the packet.Length/type of PDU: These next two bytes indicate the number of bytes in the coming PDU. If the length of the PDU is fixed, this field can be used to indicate type, or as a base for other protocols. For example, Novell and the Internet use it to identify the network layer protocol that is using the PDU.

802.2 frames (PDU): This field of the 802.3 frame contains the entire 802.2 frame as a modular, removable unit. The PDU can be anywhere from 46 to 1500 bytes long, depending on the type of frame and the length of the information field. The PDU is generated by the upper (LLC) sub layer, and then linked to the 802.3 frame.

CRC: The last field in the 802.3 frame contains the error detection information, in this case a CRC-32.

Implementation

Although the bulk of the IEEE Project 802 standard focuses on the data link layer of the OSI model, the 802 model also defines some of the physical specifications for each of the protocols defined in the MAC layer. In the 802.3 standard, the IEEE defines the types of cable, connections, and signals that are to be used in each of five different Ethernet implementations. All Ethernet LANs are configured as logical buses, although.

4.2.4 10BASE5: Thick Ethernet

The first of the physical standards defined in the IEEE 802.3 model is called 10Base5, thick Ethernet, or Thicknet. The nickname derives from the size of the cable, which is roughly the size of garden hose and too stiff to bend with your hands. 10Base5 is a bus topology LAN that uses base and signalling and has a maximum segment length of 500 meters.

Networking devices (such as repeaters and bridges), can be used to overcome the size limitation of local area networks. In thick Ethernet, a local area network can be divided into segments by connecting devices. In this case, the length of each segment is limited to 500 meters. However, to reduce collisions, the total; length of the

bus should not exceed 2500 meters (five segments). Also, the standard demands that each station be separated from each neighbour by 2.5 meters (200 stations per segment and 1000 stations total).

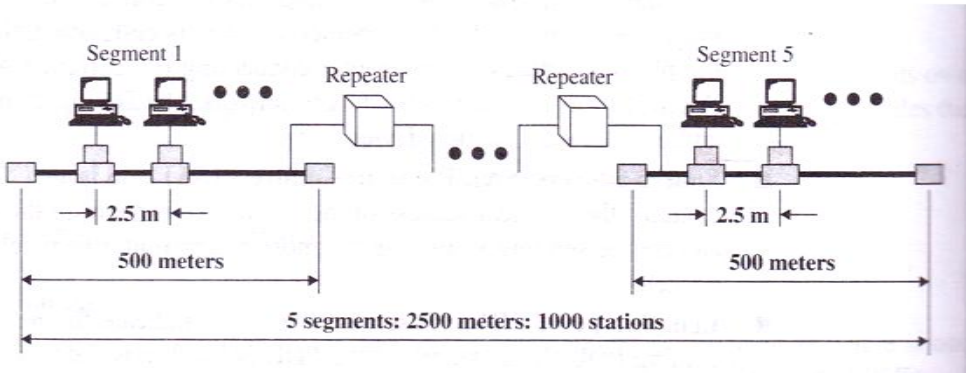


Figure 4.4: Ethernet segments.

The physical connectors and cables utilized by 10Base5 include coaxial cable, network interface cards, transceivers, and attachment unit interface (AUI) cables.

RG-8 Cable RG-8 cable (RG stands for radio government) is a thick coaxial cable that provides the backbone of the IEEE 802.3 standard.

Transceiver Each station is attached by an AUI cable to an intermediary device called a medium attachment-unit (MAU) or, more commonly, a transceiver (short for transmitter-receiver). The transceiver performs the CSMA/CD function of checking for voltages and collisions on the line and may contain a small buffer. It also serves as the connector that attaches a station to the thick coaxial cable itself via a tap.

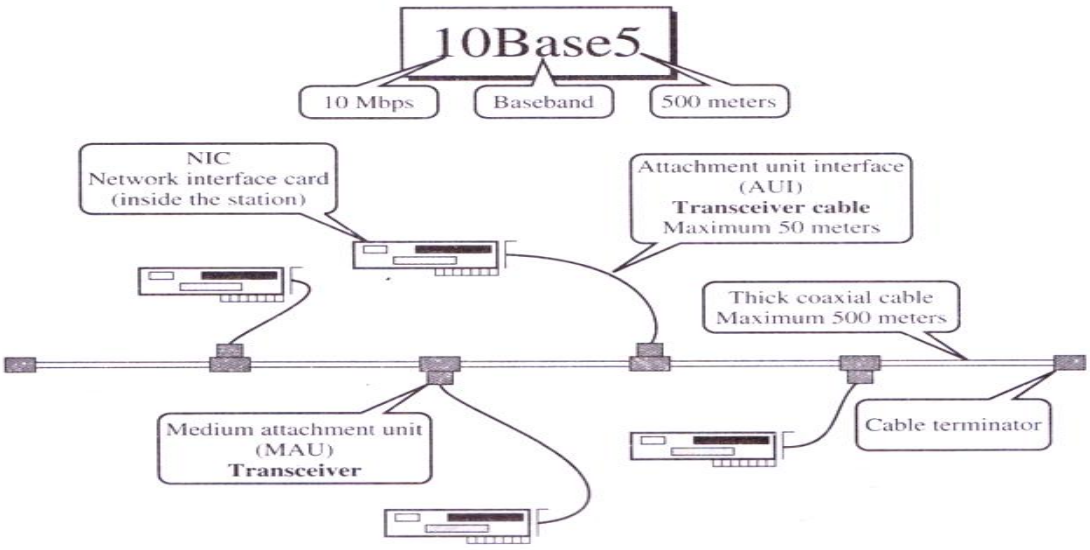


Figure 4.5: Topology of 10BASE5.

AUI Cables Each station is linked to its corresponding transceiver by an attachment unit interface (AUI), also called a transceiver cable. An AUI is a 15-wire cable with plugs that performs the physical layer interface functions between the station and the transceiver. Each end of an AUI terminates in a DB-15 (15-pin) connector. One connector plugs into a port on the NIC, the other into a port on the transceiver. AUIs are

restricted to a maximum length of 50 meters, allowing for some flexibility in placement of stations relative to the 10BASE5 backbone cable.

Transceiver Tap Each transceiver contains a connecting mechanism, called a tap because it allows the transceiver to tap into the line at any point. The tap is a thick cable-sized well with a metal spike in the centre. The spike is attached to wires inside the transceiver. When the cable is pressed into the well, the spike pierces the jacket and sheathing layers and makes an electrical connection between the transceiver and the cable. This kind of connector is often called a vampire tap because it bites the cable.

4.2.5 10BASE2: Thin Ethernet

The second Ethernet implementation defined by the IEEE 802 series is called 10Base2 or thin Ethernet. Thin Ethernet (also called Thinnet, cheapnet, cheapernet, and thin-wire Ethernet) provides an inexpensive alternative to 10Base5 Ethernet, with the same data rate. Like 10Base5, 10Base2 is a bus topology LAN. The advantages of thin Ethernet are reduced cost and ease of installation (the cable is lighter weight and more flexible than that used in thick Ethernet). The disadvantages are shorter range (185 meters as opposed to the 500 meters available with thick Ethernet) and smaller capacity (the thinner cable accommodates fewer stations). In many situations-such as a small number of users on a UNIX-based minicomputer or a network of personal computers workstations-these disadvantages are irrelevant, and the cost savings make 10Base2 the better choice.

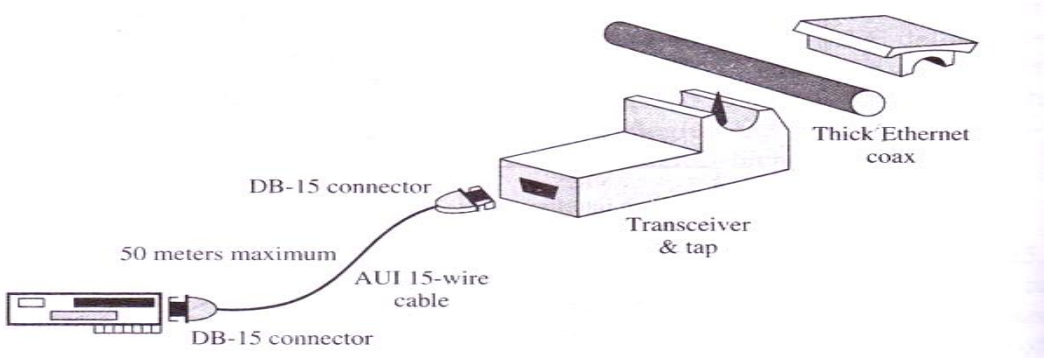


Figure 4.6: Transceiver connections in 10BASE2.

The physical layout of 10Base2 is illustrated in Figure 4.6. The connectors cables utilized are: NICs, thin coaxial cable, and BNC-T connectors. In this technology the transceiver circuitry has moved into the NIC, and the transceiver tap has been replaced by a connector that splices the station directly into the cable, eliminating the need for AUI cables.

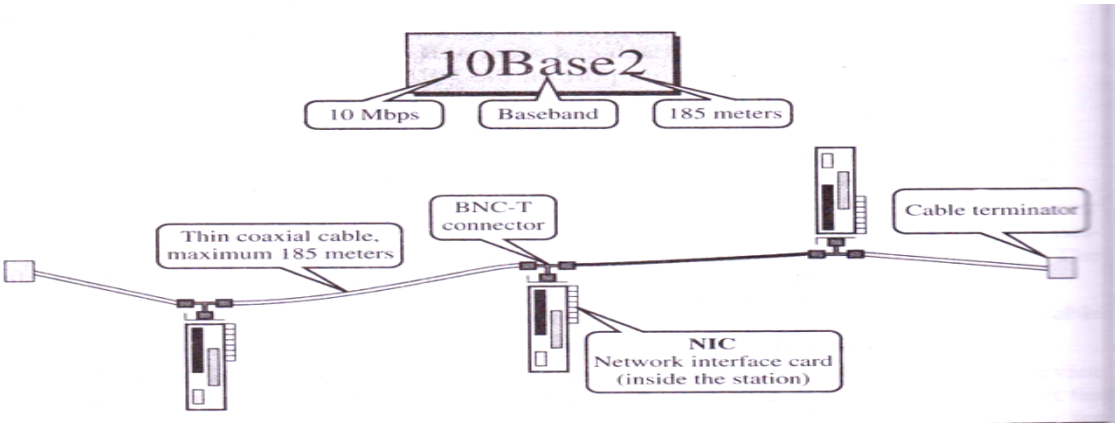


Figure 4.7: Topology of 10BASE2.

NIC The NICs in a thin Ethernet system provide all of the same functionality as those in a thick Ethernet system, plus the functions of the transceivers. That means that a 10Base2 NIC not only provides the station with an address but also checks for voltages on the link.

Thin Coaxial Cable The cable required to implement the 10Base2 standard is RG-58. These cables are relatively easy to install and move around (especially inside existing buildings where cabling must be pulled through the walls and ceilings).

BNC-T the BNC-T connector is a T -shaped device with three ports: one for the NIC and one each for the input and output ends of the cable.

4.2.6 10Base-T: Twisted-Pair Ethernet

The most popular standard defined in the IEEE 802.3 series is 10Base-T (also called twisted-pair Ethernet), a star-topology LAN using unshielded twisted pair (UTP) cable instead of coaxial cable. It supports a data rate of 10 Mbps and has a maximum length (hub to station) of 100 meters.

Instead of individual transceivers, 10Base-T Ethernet places all of its networking operations in an intelligent hub with a port for each station. Stations are linked into the hub by four-pair RJ-45 cable (eight-wire unshielded twisted-pair cable) terminating at each end in a male-type connector much like a telephone jack (see Figure 4.8). The hub fans out any transmitted frame to all of its connected stations. Logic in the NIC assures that the only station to open and read a given frame is the station to which that frame is addressed.

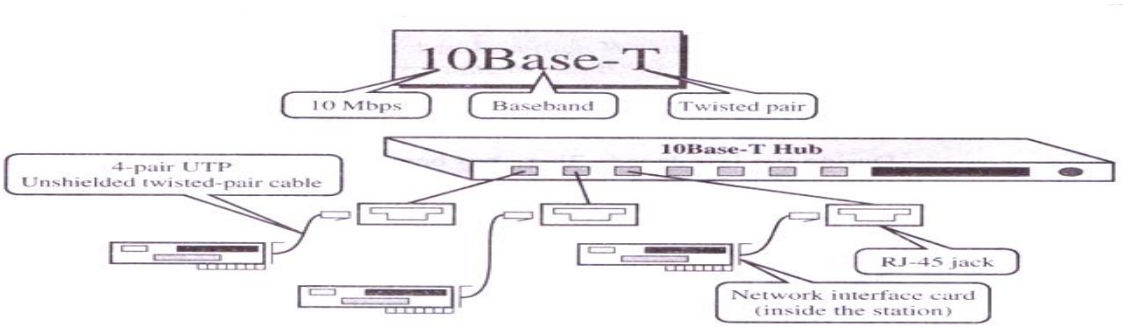


Figure 4.8: 10Base-T topology.

As Figure 4.8 shows, each station contains an NIC. A length of four-pair UTP of not more than 100 meters connects the NIC in the station to the appropriate port in the 10Base-T hub.

The weight and flexibility of the cable and the convenience of the RJ-45 jack and plug make 10Base-T the easiest of the 802.3 LANs to install and reinstall. When a station needs to be replaced, a new station can simply be plugged in.

1Base5: StarLAN

StarLAN is an AT&T product used infrequently today because of its slow speed. At only 1 Mbps, it is 10 times slower than the three standards discussed above.

What is interesting about StarLAN is its range, which can be increased by a mechanism called daisy chaining. Like 10Base-T, StarLAN uses twisted-pair cable to connect stations to a central intelligent hub. Unlike 10Base-T, which requires that each station have its own dedicated cable into the hub, StarLAN allows as many as 10 stations to be linked, each to the next, in a chain in which only the lead device connects to the hub (see Figure 4.9).

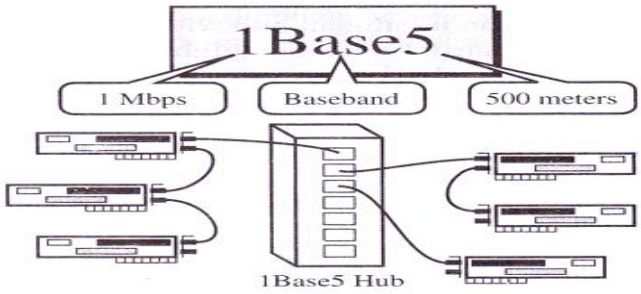


Figure 4.9: 1Base5.

4.2.7 Other Ethernet Networks

During the last decade, there has been an evolution in Ethernet networks. Several new schemes have been devised to improve the performance and the speed of Ethernet LANs. We will discuss three of these efforts here: Switched Ethernet, Fast Ethernet and Gigabit Ethernet.

Switched Ethernet

Switched Ethernet is an attempt to improve the performance of 10Base-T The 10Base-T Ethernet is a shared media network, which- means that the entire involved in each transmission. This is because the topology, though physically a star is logically a bus. When a station sends a frame to a hub, the frame is sent out from all ports (interfaces) and every station will receive it. In this situation, only one station can send a frame at any time. If two stations try to send frames simultaneously, there is a collision.

Figure 4.10 shows this situation. Station A is sending a frame to station E. The frame is received by the hub and is sent to every station. All of the cabling in the system is involved in this transmission. Another way to think about this is that one transmission uses the entire capacity of 10 Mbps; if one station uses it, no other station can.

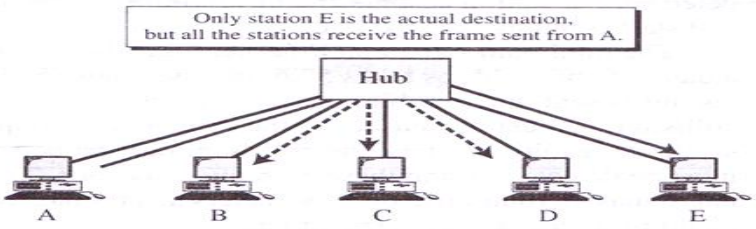


Figure 4.10: An Ethernet network using a hub.

However, if we replace the hub with a switch, a device that can recognize the destination address and can route the frame to the port to which the destination station is connected, the rest of the media are not involved in the transmission process. This means that the switch can receive another frame from another station at the same time and can route this frame to its own final destination. In this way, theoretically, there is no collision.

Using a switch, instead of a hub, we can theoretically increase the capacity of a network with N devices to N x 10 Mbps because 10Base-T uses two pairs of UTP for full-duplex communication.

Figure 4.11 shows a Switched Ethernet. When station A is sending a frame to station E , station B can also send a frame to station D without any collision.

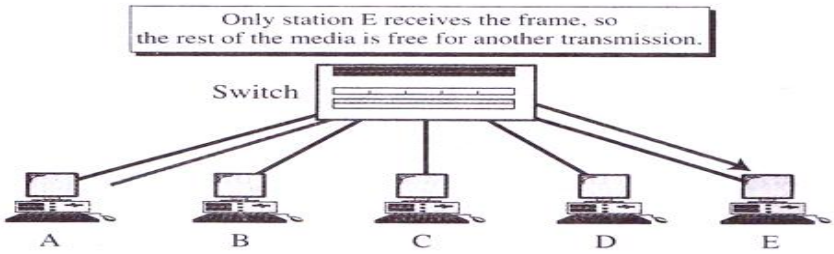


Figure 4.11: An Ethernet using a switch.

Fast Ethernet

With new applications such as computer-aided design (CAD), image processing, and real-time audio and video being implemented on LANs, there is a need for a LAN with a data rate higher than 10 Mbps. Fast Ethernet operates at 100 Mbps.

In the physical layer, the specification developed for Fast Ethernet is a star topology similar to 10Base-T; however, to match the physical layer to different re available, IEEE has designed two categories of Fast Ethernet: 100Base-X and 100Base-T4. The first uses two cables between the station and the hub; the - uses four. 100Base-X itself is divided into two types: 100Base-TX and 100Base-FX (see Figure 4.12).

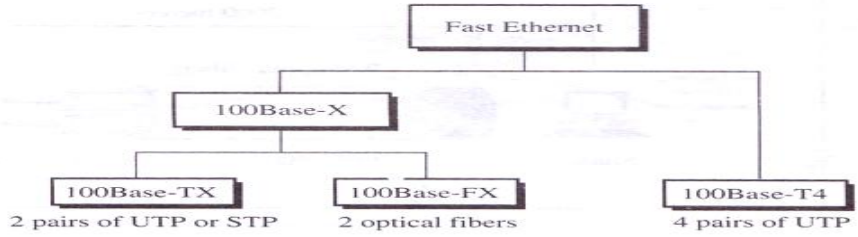


Figure 4.12: Categories of Fast Ethernet implementations.

100Base-TX

The 100Base-TX design uses two category 5 unshielded twisted-pair (UTP) or shielded twisted-pair (STP) cables to connect a station to the hub. One pair is carry frames from the station to the hub and the other to carry frames from the station. The encoding is 4B/5B to handle the 100 Mbps; the signaling is NRZ-I, The distance between the station and the hub (or switch) should be less than 100 meters (see Figure 4.13).

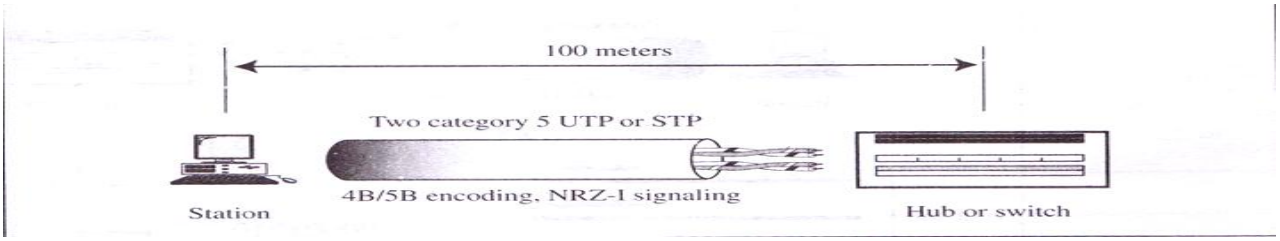


Figure 4.13: 100Base-TX implementation.

100Base-FX

The 100Base-FX design uses two optical fibers, one to carry frames from the station to the hub and the other from the hub to the station. The encoding is 4B/SB and signaling is NRZ-I. The distance between the station and the hub (or switch) should be less than 2000 meters (see Figure 4.14).

100Base-T4

The 100Base-T4 scheme was designed in an effort to avoid rewiring. It requires four pairs of category 3 (voice grade) UTP that are already available for telephone service inside most buildings. Two of the four pairs are bidirectional; the other two are unidirectional. This means that in each direction, three pairs are used at the same time to carry data. Because a 100-Mbps data rate cannot be handled by a voice-grade UTP, the specification splits the 100-Mbps flow of data into three 33.66-Mbps flows. To reduce the baud rate of the transmission, a method called 8B/6T (eight binary/six ternary) is used in which each block of eight bits is transformed into six bauds of three voltage levels (positive, negative, and zero). Figure 4.15 shows the scheme and an encoding example.

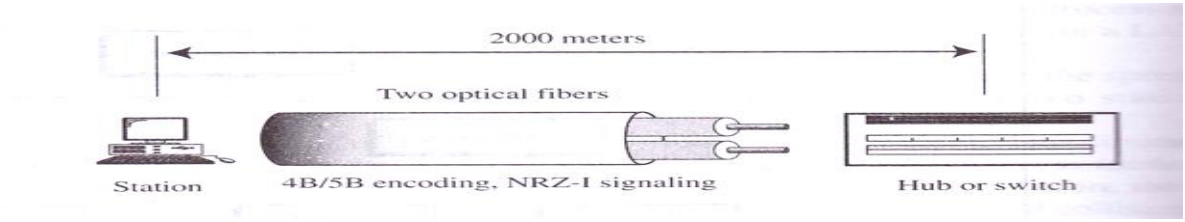


Figure 4.14:100Base-FX.

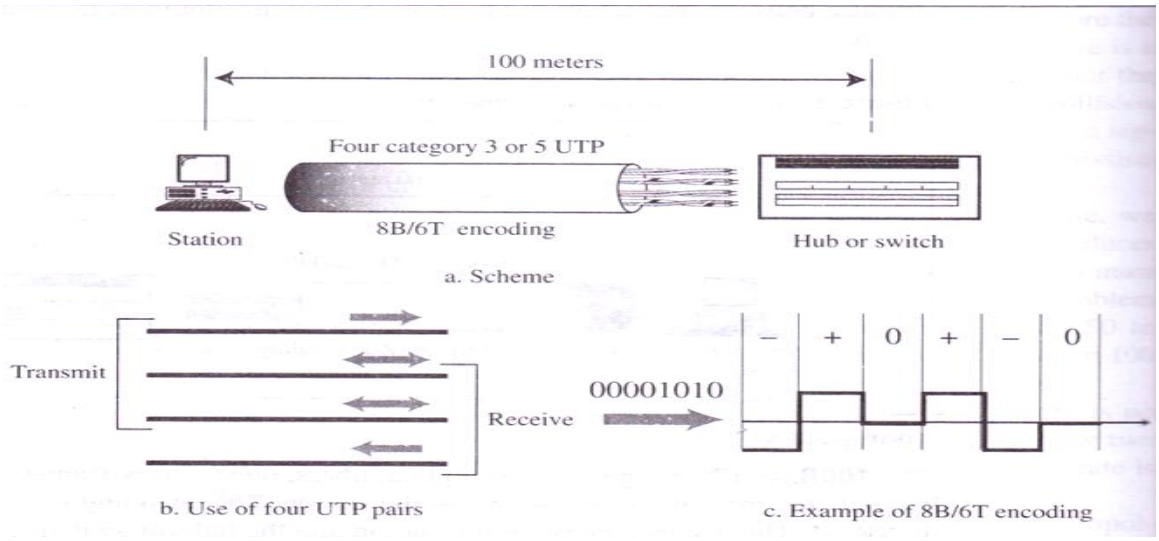


Figure 4.15: 100Base-T4.

Gigabit Ethernet

The migration from 10 Mbps to 100 Mbps encouraged the IEEE 802.3 design Gigabit Ethernet, which has a data rate of 1000 Mbps or 1 Gbps. The strategy is the same; the MAC layer and the access method remain the same, but the domain is reduced. The physical layer-the transmission media and the system-however, changes. Gigabit Ethernet is mainly designed to use optical fiber although the protocol does not eliminate the use of twisted pair cables. Gigabit Ethernet usually serves as a backbone to connect Fast Ethernet networks. An example is show in Figure 4.16.

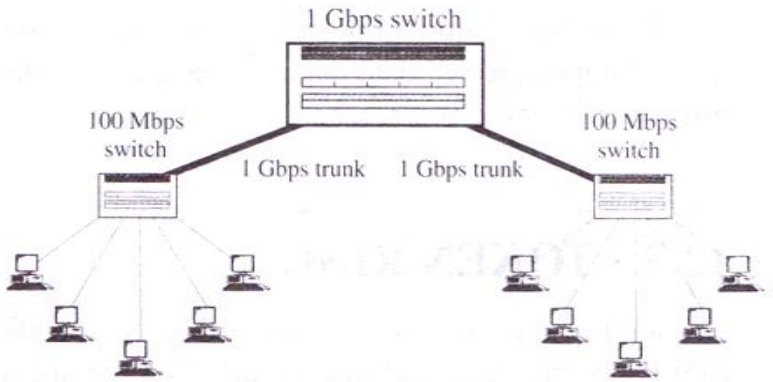


Figure 4.16: Use of Gigabit Ethernet.

Four implementations have been designed for Gigabit Ethernet: 1000Base-LX, 1000Base-SX, 1000Base-CX, and 1000Base-T. The encoding is 8B/IOB, which means a group of 8 binary bits are encoded into a group of 10 binary bits. Table 4.1 shows the features of the four implementations.

Table 4.1 Comparison between the Gigabit Ethernet implementations

Feature	1000Base-SX	1000Base-LX	1000Base-CX	1000Base-T
Medium	Optical fiber (multimode)	Optical fiber (multi- or single-mode)	STP	UTP
Signal	Short-wave laser	Long-wave laser	Electrical	Electrical
Max. distance	550 m	550 m (multimode) 5000 m (single mode)	25 m	25 m

Did You Know?

Ethernet was developed at Xerox PARC between 1973 and 1974. It was inspired by ALOHA net, which Robert Metcalfe had studied as part of his PhD dissertation.

4.3 FDDI (Fiber Distributed Data Interface)

Fiber distributed data interface (FDDI) is a local area network protocol standardized by ANSI and the ITU-T (ITU-T X.3). It supports data rates of 100 34bps and prop ides a high-speed alternative to Ethernet and Token Ring. 'When FDDI was designed, speeds of 100 Mbps required fiber-optic cable. Today, however, comparable speeds are available using copper cable. The copper version of FDDI is known as CDDI.

4.3.1 Access Method: Token Passing

In FDDI, access is limited by time. A station may send as many frames as it can within its allotted access period, with the proviso that real-time data be sent first.

To implement this access mechanism, FDDI differentiates between two types of data frames: synchronous and asynchronous. Synchronous here refers to information that is real-time, while as while asynchronous refers to information that is not. These frames are usually called S-frames and A-frames.

Each station that captures the token is required to send S-frames first. In fact, it must send its S-frames whether or not it is time allotment has run out. Any remaining time may then be used to send A-frames. To understand how this mechanism ensures fair and timely link access, it is necessary to understand isle FDDI time registers and timers.

Time Registers

FDDI defines three time registers to control circulation of the token and distribute link access opportunities among the nodes equitably. Values are set when the ring is initialized and do not vary in the course of operation. The registers are called synchronous allocation (SA), target token rotation time (TTRT), and absolute maximum time (AMT).

Synchronous Allocation (SA) The SA register indicates the length of time allowed each station for sending synchronous data. This value is different for each station and is negotiated during initialization of the ring.

Target Token Rotation Time (TTRT) The TTRT register indicates the average time required for a token to circulate around the ring exactly once (the elapsed time between a token's arrival at a given station and its next arrival at the same station). Because it is an average, the actual time of any rotation may be greater or less than this value.

Absolute Maximum Time (AMT) The AMT register holds a value equal to twice the TTRT. A token may not take longer than this time to make one rotation of the ring. If it does, some station or stations are monopolizing the network and the ring must be reinitialized.

Timers

Each station contains a set of timers that enable it to compare actual timings with the values contained in the registers. Timers can be set and reset, and their values decremented or incremented at a rate set by the system clock. The two timers used by FDDI are called the token rotation timer (TRT) and token holding timer (THT). Time taken by the token to complete a cycle. In our implementation, we use an incrementing TRT for simplicity, although some implementations may use a decrementing timer.

Token Holding Timer (THT) The THT begins running as soon as the token is received. Its function is to show how much time remains for sending asynchronous frames once the synchronous frames have been sent. In our implementation, we use decrementing THT for simplicity, although some implementations may use an incrementing one. In addition, we allow the value of THT to become negative (to make the concept easier to understand) although a real timer may stay at zero.

Station Procedure

When a token arrives, each station follows this procedure:

I. THT is set to the difference between TTRT and TRT (THT = TTRT - TRT).

2. TRT is reset to zero (TRT = 0).

3. The station sends its synchronous data.

4. The station sends asynchronous data as long as the value of THT is positive.

Example:

Figure 4.17 show how FDDI access works. We have simplified example by showing only four stations and making the following assumption,: the TTRT is 30 time units; the time required for the token to go from one station to am is 1 time unit, each station is allowed to send two synchronous data units per turn: each station has a lot of asynchronous data to send (waiting in buffers).

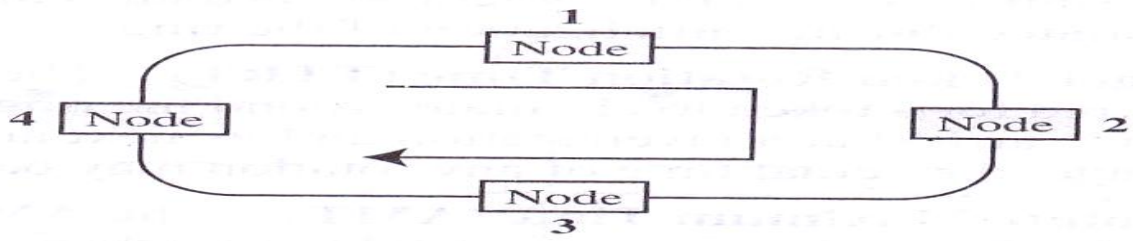


Figure 4.17: FDDI operation.

In round 0, the token travels from station to station; each station sets its TRT time to 0. No data transfer occurs in this round.
In round 1, station 1 receives the token at time 4; its TRT is now 4 (in round. was 0; it took 4 time units for the token to return). THT is set to 26 (THT = TTI!'" TRT = 30 - 4). TRT is reset to 0. Now station I sends 2 data unit equiv synchronous data. THT is decremented to 24 (26 - 2), so station I can send 24 equivalents of asynchronous data.

In the same round, station 2 follows the same procedure. The token arriving time is now 31 because the token arrived at station 1 at time 4. It was held 1-6 time units (2 for synchronous data and 24 for asynchronous data) and it took I time unit for the token to travel between stations (4 + 26 + 1= 31).
Note that the asynchronous allocation time is almost equally distributed between stations. In round 1, station 1 has the opportunity to send 24 time unit equivalents of asynchronous data, but the other stations did not have such an opportunity. However in rounds 2, 3, and 4, station 1 was deprived of this privilege, but other stations (one in each round) had the opportunity to send. In round 2, station 2 sent 16; and in round 4, station 4 sent 16.

Round	*Station 1*	*Station 2*	*Station 3*	*Station 4*
0	Arriving Time: 0 TRT = 0	Arriving Time: 1 TRT = 0	Arriving Time: 2 TRT = 0	Arriving Time: 3 TRT = 0
1	Arriving Time: 4 TRT is now 4 THT = 30 – 4 = 26 **TRT = 0** Syn Data: 2 THT is now 24 **Asyn Data: 24**	Arriving Time: 31 TRT is now 30 THT = 30 – 30 = 0 **TRT = 0** Syn Data: 2 THT is now –2 Asyn Data: 0	Arriving Time: 34 TRT is now 32 THT = 30 – 32 = –2 **TRT = 0** Syn Data: 2 THT is now –4 Asyn Data: 0	Arriving Time: 37 TRT is now 34 THT = 30 – 34 = –4 **TRT = 0** Syn Data: 2 THT is now –6 Asyn Data: 0
2	Arriving Time: 40 TRT is now 36 THT = 30 – 36 = –6 **TRT = 0** Syn Data: 2 THT is now –8 Asyn Data: 0	Arriving Time: 43 TRT is now 12 THT = 30 – 12 = 18 **TRT = 0** Syn Data: 2 THT is now 16 **Asyn Data: 16**	Arriving Time: 62 TRT is now 28 THT = 30 – 28 = 2 **TRT = 0** Syn Data: 2 THT is now 0 Asyn Data: 0	Arriving Time: 65 TRT is now 28 THT = 30 – 28 = 2 **TRT = 0** Syn Data: 2 THT is now 0 Asyn Data: 0
3	Arriving Time: 68 TRT is now 28 THT = 30 – 28 = 2 **TRT = 0** Syn Data: 2 THT is now 0 Asyn Data: 0	Arriving Time: 71 TRT is now 28 THT = 30 – 28 = 2 **TRT = 0** Syn Data: 2 THT is now 0 Asyn Data: 0	Arriving Time: 74 TRT is now 12 THT = 30 – 12 = 18 **TRT = 0** Syn Data: 2 THT is now 16 **Asyn Data: 16**	Arriving Time: 93 TRT is now 28 THT = 30 – 28 = 2 **TRT = 0** Syn Data: 2 THT is now 0 Asyn Data: 0
4	Arriving Time: 96 TRT is now 28 THT = 30 – 28 = 2 **TRT = 0** Syn Data: 2 THT is now 0 Asyn Data: 0	Arriving Time: 99 TRT is now 28 THT = 30 – 28 = 2 **TRT = 0** Syn Data: 2 THT is now 0 Asyn Data: 0	Arriving Time: 102 TRT is now 28 THT = 30 – 28 = 2 **TRT = 0** Syn Data: 2 THT is now 0 Asyn Data: 0	Arriving Time: 105 TRT is now 12 THT = 30 – 12 = 18 **TRT = 0** Syn Data: 2 THT is now 16 **Asyn Data: 16**

Addressing

FDDI uses a six-byte address which is imprinted on the NIC card similar to Ethernet addresses.

4.3.2 Electrical Specification

Signalling (Physical Layer)

FDDI uses a special encoding mechanism called four bits/five bits (4B/SB). In this system, each four-bit segment of data is replaced by a five-bit code before being: encoded in NRZ. The NRZ-I used here inverts on the 1(see Figure 4.18).

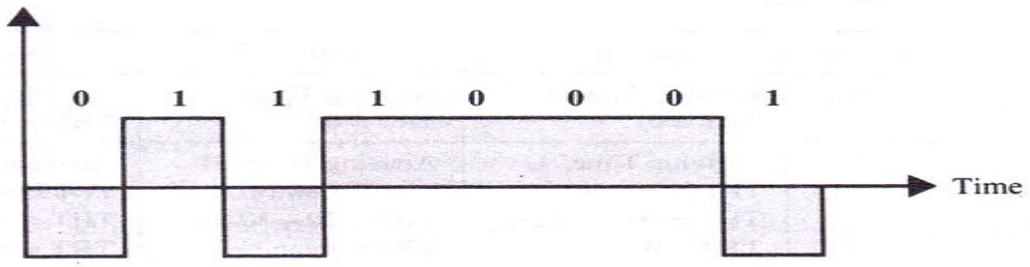


Figure 4.18: Encoding.

The reason for this extra encoding step is that, although NRZ-I provides adequate synchronization under average circumstances, sender and receiver may go out of synchronization anytime the data includes a long sequence of Os. 4B/SB encoding transforms each four-bit data segment into a five-bit unit that contains no more than two consecutive Os. Each of the 16 possible four-bit patterns is assigned a five-bit pattern M represent it. These five-bit patterns have been carefully selected so that even sequential data units cannot result in sequences of more than three Os (none of the five-bit patterns start with more than one 0 or end with more than two Os); see Table 4.2.

Table 4.2: 4B/5B encoding

Data Sequence	*Encoded Sequence*	*Data Sequence*	*Encoded Sequence*
0000	11110	1000	10010
0001	01001	1001	10011
0010	10100	1010	10110
0011	10101	1011	10111
0100	01010	1100	11010
0101	01011	1101	11011
0110	01110	1110	11100
0111	01111	1111	11101

Five-bit codes that have not been assigned to represent a four-bit counterpart are used for control (see Table 4.3). The SD field contains the J and K codes, and the ED field contains the symbols TT. To guarantee that these control codes do not endanger synchronization or transparency, the designers specify bit patterns that can never occur in the data field. In addition, their order is controlled to limit the number of sequential bit patterns possible. A K always follows a J, and an H is never followed by an R.

Table 4.3: 4B15B control symbol

Control Symbol	*Encoded Sequence*
Q (Quiet)	00000
I (Idle)	11111
H (Halt)	00100
J (Used in start delimiter)	11000
K (Used in start delimiter)	10001
T (Used in end delimiter)	01101
S (Set)	11001
R (Reset)	00111

The FDDI standard divides transmission functions into four protocols: physical medium dependent (PMD), physical (PHY), media access control (MAC), and logical link control (LLC). These protocols correspond to the physical and data link layers of the OSI model (see Figure 4.19). In addition, the standard specifies a fifth protocol (used for station management).

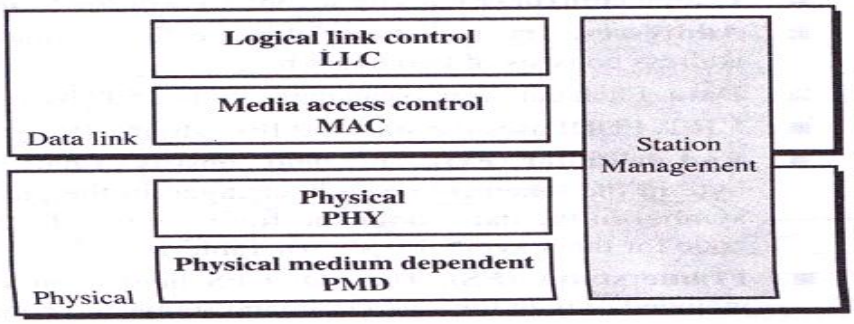


Figure 4.19: FDDI layers.

4.3.3 Media Access Control

The FDDI MAC layer is almost identical to that defined for Token Ring. Howe - although the functions are similar, the FDDI MAC frame itself is different enough warrant an independent discussion of each field (see Figure 4.20).

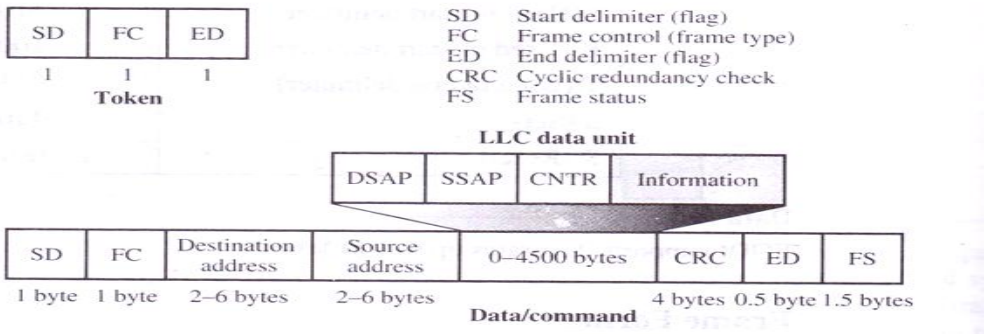


Figure 4.20: FDDI frame types.

Each frame is preceded by 16 idle symbols (1111), for a total of 64 bits, to initialize clock synchronization with the receiver.

Frame Fields There are eight fields in the FDDI frame:

• Start delimiter (SD). The first byte of the field is the frame's starting flag. As in Token Ring, these bits are replaced in the physical layer by the control codes (violations) J and K (the five-bit sequences used to represent J and K are shown in Table 4.3).
• Frame control (FC). The second byte of the frame identifies the frame type.
• Addresses. The next two fields are the destination and source addresses. Each address consists of two to six bytes.
• Data. Each data frame can carry up to 4500 bytes of data.
• CRC. FDDI uses the standard IEEE four-byte cyclic redundancy check.
• End delimiter (ED). This field consists of half a byte in the data frame or a full byte in the token frame. It is changed in the physical layer with one T violation symbol in the data/command frame or two T symbols in the token frame. (The code for the T-symbol is shown Table 4.3.)
• Frame status (FS). The FDDI FS field is similar to that of Token Ring. It is included only in the data/command frame and consists of 1.5 bytes.

Implementation: Physical Medium Dependent (PMD) Layer

The physical medium dependent (PMD) layer defines the required connections and electronic components. Specifications for this layer depend on whether the transmission medium used is fiber-optic or copper cable.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: The data rate of 10base5 in 10 Mbps. how long does it take to create the smallest frame? Show your calculation.

……..……

……..……

……...

Dual Ring

FDDI is implemented as a dual ring (see Figure 4.21). In most cases, data transmission is confined to the primary ring. The secondary ring is provided in case the primary fails.

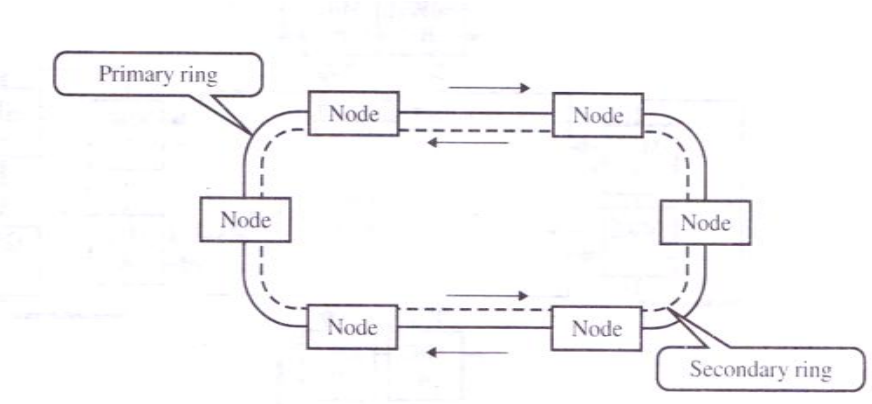


Figure 4.21: FDDI ring.

The secondary ring makes FDDI self-healing. Whenever a problem occurs on the primary ring, the secondary can be activated to complete data circuits and maintain service (see Figure 4.22).

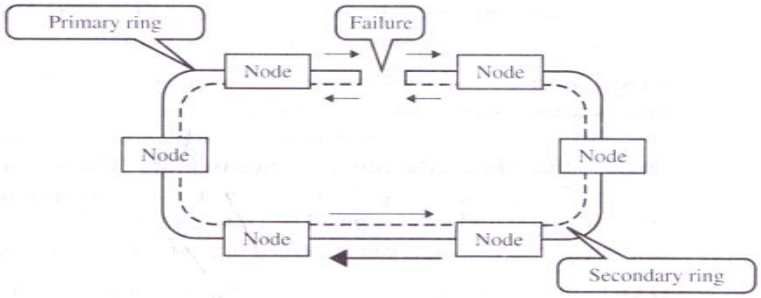


Figure 4.22: FDDI ring after a failure.

Nodes connect to one or both rings using a media interface connector (MIC) that can be either male or female depending on the requirements of the-station.

Nodes

FDDI defines three types of nodes: dual attachment station (DAS), single attachment station (SAS), and dual attachment concentrator (DAC); see Figure 4.23.

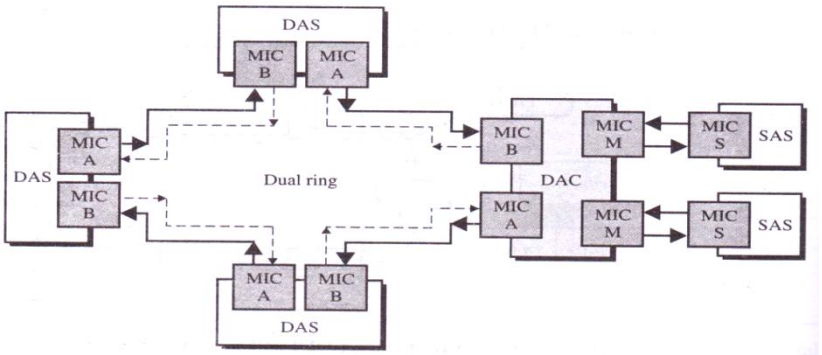


Figure 4.23: Node connections.

DAS*:* A dual attachment station (DAS) has two MICs (called MIC A and MIC A MIC B and connects to both rings. To do so requires an expensive NIC with two inputs and two outputs. The connection to both rings gives it improved reliability and throughput these improvements, however, are predicated on the stations remaining on. Faults are bypassed by a station's making a wrap connection from the primary ring to the secondary to switch signals from one input to another output. However, for DAS stations to make this switch, they must be active (turned on).

SAS: Most workstations, servers, and minicomputers attach to the ring in single attachment station (SAS) mode. An SAS has only one MIC (called MIC S) and therefore can connect only to one ring. Robustness is achieved by connecting SASs to intermediate ring nodes, called dual attachment concentrators (DACs), rather than to the FDDI ring directly. This configuration allows each workstation to operate through a simple NIC with only one input and one output. The concentrator (DAC) provides the connection to the, dual ring. Faulty stations can be turned off and bypassed to keep the ring, alive

DAC: As mentioned above, a dual attachment concentrator (DAC) connects an SAS to the dual ring. It provides wrapping (diverting traffic from one ring to the other to bypass a failure) as well as control functions. It uses MIC M to-connect to an SAS.

4.4 Network Operation

Network operation diverges into two categories: peer-to-peer networks and client-to-server operating systems. Each of these methods of tying multiple devices into a given system has its unique advantages from the

flexibility of adding new software and ease of expansion that is the hallmark of client-to-server operating systems, to the ease of setup and inexpensive operating cost that is main attractor to peer-to-peer connections. Disadvantages arise when networks require troubleshooting as client-to-server networks often have a dedicated IT staff to handle the potentially expensive problems whereas peer-to-peer networks have only each other to solve problems large and small.

4.4.1 Cost versus Maintenance

Of the two main networking operation systems, a peer-to-peer network is the lowest cost in terms of initial start up. Users do not need to purchase a centralized server to store data as all information is stored on and accessed from computers connected to the network. A client-to-server network operating system is of course the more expensive route as a centralized server is required when the network is first established and will require a staff or some form of information technology personnel to adequately service it. Cost (an initial disadvantage) is mitigated by the return investment in technical staff when a problem arises with the network and a trained professional is on-hand to troubleshoot the issue.

4.4.2 Dependency on the System

Client-to-Server network operations depend on the operating system to function. Should the centralized server go down or malfunction, operations will cease across the entire network. Peer-to-peer systems have the advantage of an existing operating system for each computer linked into the network to depend on so each computer can function easily as a separate unit. As information can only be stored on systems linked into the network in peer-to-peer systems, if a computer does go down the information stored there will not be accessible for the rest of the network until it is repaired.

4.4.3 Versatility

New technology is easily integrated into client-to-server network connections as the operation is controlled centrally. Of course when this new technology is integrated into the system, a given staff must then be trained to use the new technology which can be time consuming and have a few pitfalls as workers integrate the new system into existing protocols. Peer-to-peer systems depend largely on existing software platforms installed on computers linked to the network and while systems for the entire network cannot be changed, each user is able to customize a work station to optimize personal efficiency.

Caution

If your computer are part of a large network, your must verify with your network administrator that the computer names, domain name and other information used in setting up windows server 2003,windows XP, and ISA Server 2004 otherwise it will conflicts with network operations.

4.5 ATM (Asynchronous Transfer Mode

Asynchronous Transfer Mode (ATM) is the cell relay protocol designed by the ATM Forum and adopted by the ITU-T. The combination of ATM and B-ISDN will allow high-speed interconnection of all the world's networks. In fact, ATM can be thought of as the “highway” of the information superhighway.

4.5.1 Design Goals

Among the challenges faced by the designers of ATM, six stand out. First and foremost is the need for a transmission system to optimize the use of high-data-rate transmission media, in particular optical fiber. In addition to offering large bandwidths, newer transmission media and equipment are dramatically less

susceptible to noise degradation. A technology is needed to take advantage of both factors and thereby maximize data rates.

Second is the need for a system that can interface with existing systems, such as the various packet networks, and to provide wide area interconnectivity between them without lowering their effectiveness or requiring their replacement. ATM is potentially as effective a LAN and short-haul mechanism as it is a WAN mechanism. Its proponents hope that it will eventually replace the existing systems. Until that happens, however, the protocol provides mechanisms for mapping the packets and frames of other systems onto ATM cells.

Third is the need for a design that can be implemented inexpensively so that cost would not be a barrier to adoption. If ATM is to become the backbone of international communications, as intended, it must be available at low cost to every user who wants it.

Fourth, the new system must be able to work with and support the existing telecommunications hierarchies (local loops, local providers, long-distance carriers, and so on).

Fifth, the new system must be connection-oriented to ensure accurate and predictable delivery.

And last but not least, one objective is to move as many of the functions to hardware as possible (for speed) and eliminate as many software functions as possible (again for speed). Before discussing the solutions to these design requirements, it is useful to examine some of the problems associated with existing systems. In ATM, some software functions have moved to hardware; this can increase the data rate.

4.5.2 Packet Networks

Data communications today are based on packet switching and packet networks a packet is a combination of data and overhead bits that can be passed through the network as a self-contained unit. The overhead bits, in the form header and trailer, act as an envelope that provides identification and addressing information as well as the data required for routing, flow control, error control, and so on.

Different protocols use packets of varying size and intricacy. As networks become more complex, the information that must be carried in the header becomes more extensive. The result is larger and larger headers relative to the size of the data unit. In response, some protocols have enlarged the size of the data unit to make header use more efficient (sending more data with the same size header). Unfortunately, large data fields create waste. If there is not much information to transmit, much of the field goes unused. To improve utilization, some protocols provide variable packet sizes to users. We now have packets that can be as long as 65,545 bytes sharing long-haul links packets of fewer than 200 bytes.

4.5.3Mixed Network Traffic

As you can imagine, the variety of packet sizes makes traffic unpredictable: A great deal of header information must be read, and each bit counted and evaluated to ensure the integrity of every packet. Internetworking among the different packet networks is slow and expensive at best, and impossible at worst.

Another problem is that of providing consistent data-rate delivery when packet sizes are unpredictable and can vary so dramatically. To get the most out of broadband technology, traffic must be time-division multiplexed onto shared paths. Imaging the results of multiplexing packets from two networks with different requirements (and packet designs) onto one link (see Figure 4.24). What happens when line 1 uses large packets (usually data packets) while line 2 uses very small packets (the norm for and video information)?

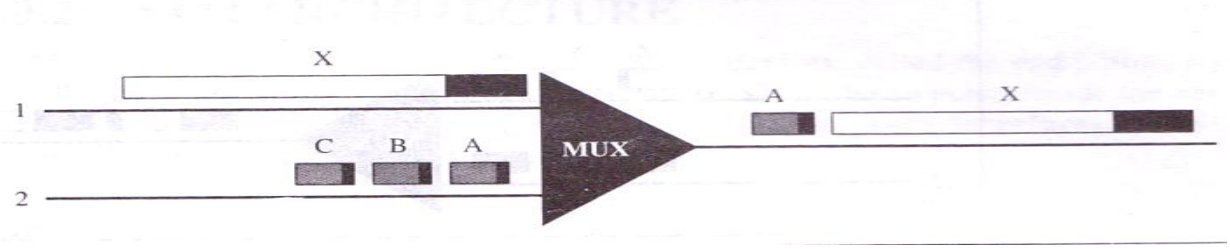


Figure 4.24: multiplexing using different packet size.

If line 1's gigantic packet X arrives at the multiplexer even a moment earlier than line 2's packets, the multiplexer puts packet X onto the new path first. After all, even if line 2's packets have priority, the multiplexer has no way of knowing to wait for them and processes the packet that has arrived. Packet A must therefore wait for the entire X bit stream to move into place before it can follow. The sheer size of X creates an unfair delay for packet A. The same imbalance can affect all of the packets from line 2. As an analogy, imagine yourself in a car arriving at a crossroad just after a parade has arrived from the opposite direction. The parade takes the same outbound road that you need to take but, because it arrived just before you did, it is already beginning its turn when you get there. You have to wait for the entire parade to turn onto the road before you can follow. Now imagine that you had been following another car that made the turn before the parade. You will now arrive at your shared destination separated by a huge gap of time.

Caution

Switches multiplexers and routers must incorporate elaborate software systems to manage the various sizes of packets in the mixed network traffic.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Define ATM.

……..……

……..……

……...

4.5.4 Cell Networks

Many of the problems associated with packet internetworking are solved by adopting a concept called cell networking. A cell is a small data unit of fixed size. In a cell network, which uses the cell as the basic unit of data exchange, all data are loaded into identical cells that can be transmitted with complete predictability and uniformity. As packets of different sizes and formats reach the cell network from a tributary network, they are split into multiple small data units of equal length and loaded into cells. The cells are then multiplexed with other cells and routed through the cell network. Because each cell is the same size and all are small, the problems associated with multiplexing different-sized packets are avoided.

A cell network uses the cell as the basic unit of data exchange. A cell is defined as a small. Fixed-sized block of information.

Advantages of Cells

Figure 4.25 shows the multiplexer with the two lines sending cells instead of packets. Packet X has been segmented into three cells: X, Y, and Z. only the first cell from line 1 gets put on the link before the first cell from line 2. The cells from the two lines are interleaved so that none suffers a long delay.

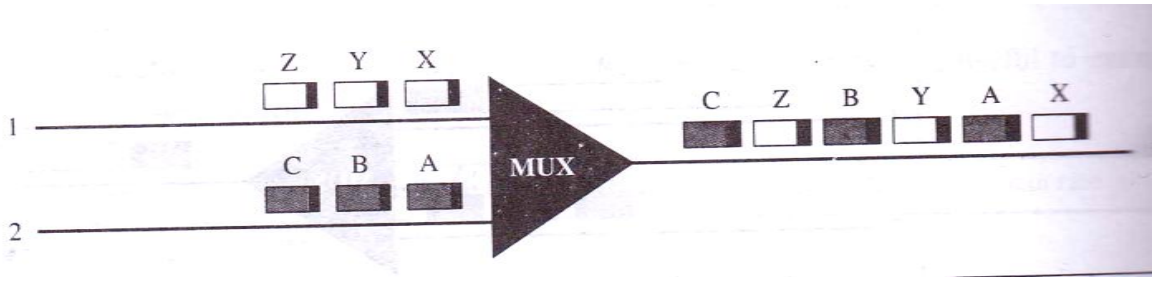


Figure 4.25: Multiplexing using cell.

A second advantage of this same scenario is that the high speed of the links coupled with the small size of the cells means that, despite interleaving, cells from each line arrive at their respective destinations in an approximation of a continuous stream (much as a movie appears to your brain to be continuous action when in fact it is really a series of separate still photos). In this way, a cell network can handle real-time missions, such as a phone call, without the parties being aware of the segmentation or multiplexing at all.

In addition, the predictability conferred by a fixed cell size allows switches and terminals to treat each cell as a unit rather than as a bit stream. In other words, to a cell network the smallest unit is a cell, not a bit. This distinction makes network operation not only more efficient but also cheaper. Switching and multiplexing can be implemented in hardware rather than software, resulting in devices that are less expensive both to produce and to maintain.

4.5.5 Asynchronous TDM

ATM uses asynchronous time-division multiplexing that is why it is called Asynchronous Transfer Mode-to multiplex cells coming from different channels. It uses fixed-size slots (size of a cell). ATM multiplexers fill a slot with a cell from any input channel that has a cell; the slot is empty if none of the channels has a cell to send.

Figure 4.26 shows how cells from three inputs are multiplexed. At the first tick of the clock, channel 2 has no cell (empty input slot), so the multiplexer fills the slot a cell from the third channel. When all the cells from all the channels are multiplexed the output slots are empty.

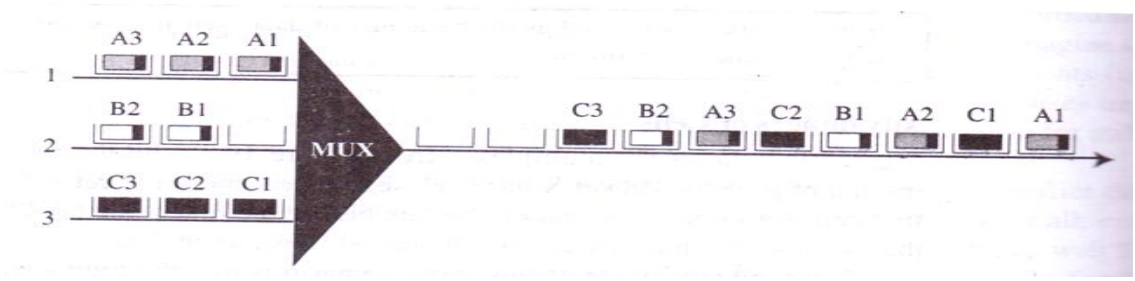


Figure 4.26: ATM multiplexing.

Did You Know?

ATM was developed in the late 1980s to meet the needs of the Broadband Integrated Services Digital Network.

4.6 ATM Service Categories

The ATM Forum defines four service classes: CBR, VBR, ABR, and UBR. VBR is divided into two subclasses VBR-RT and VBR-NRT (see Figure 4.27).

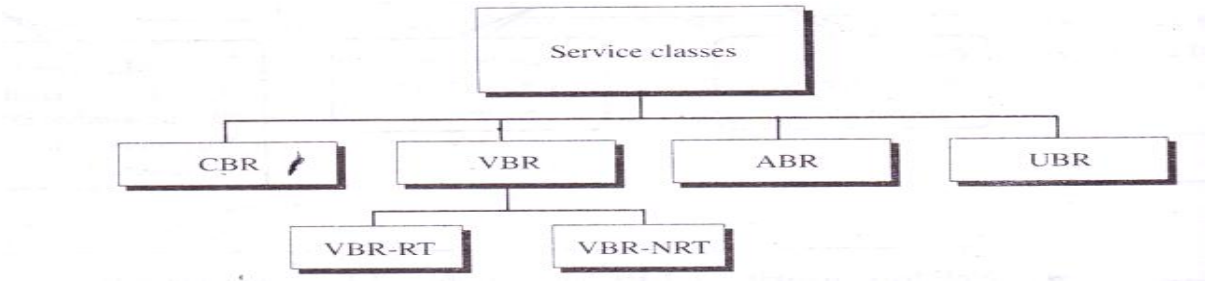


Figure 4.27: Service classes.

• CBR. The constant bit rate (CBR) class is designed for customers that need real-time audio or video services. The service is similar to that provided by a dedicated line such as a T -line.
• VBR. The variable bit rate (VBR) class is divided into two subclasses: real time (VBR-RT) and nonreal time (VBR-NRT). VBR-RT is designed for those users that need real-time services (such as voice and video transmission) and use compression techniques to create a variable bit rate. VBR-NRT is designed for users that do not need real-time services but use compression techniques to create a variable bit rate.
• ABR. The available bit rate (ABR) class delivers cells at a minimum rate. If more network capacity is available, this minimum rate can be exceeded. ABR is particularly suitable for applications that are bursty in nature.
• UBR. The unspecified bit rate (UBR) class is a best-effort delivery service that does not guarantee anything.

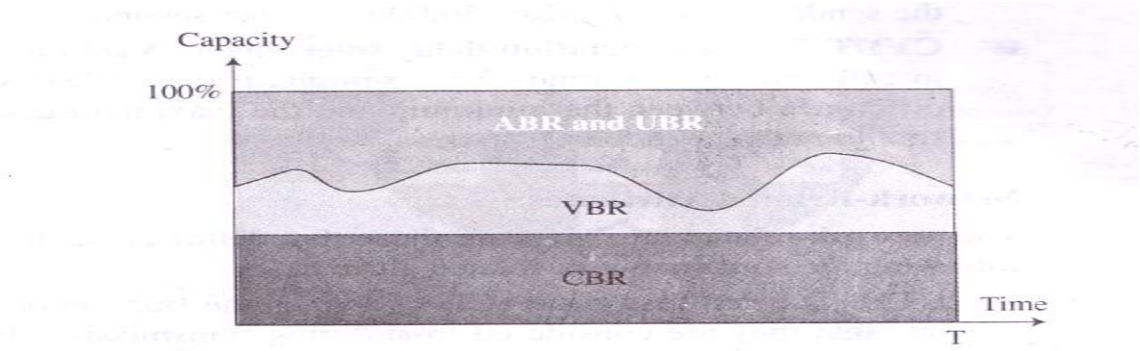


Figure 4.28: shows the relationship of different classes to the total capacity of the network.

4.6.1 Quality of Service (QoS)

The quality of service (QoS) defines a set of attributes related to the performance of the connection. For each connection, the user can request a particular attribute. Each service class is associated with a set of the attributes. We can categorize the attributes into those related to the user and those related to the network. Figure 4.29 shows the two categories and some important attributes in each category.

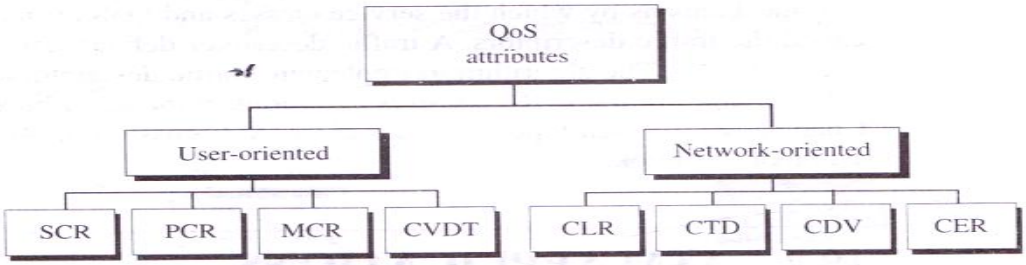


Figure 4.29: Quality of Service.

User-Related Attributes

User-related attributes are those attributes that define how fast the user wants to send data. These are negotiated at the time of contract between, a user and a network. The following are some user-related attributes:

• SCR. The sustained cell rate (SCR) is the average cell rate over a long time interval. The actual cell rate may be lower or higher than this value, but the average should be equal to or less than the SCR.
• PCR. The peak cell rate (PCR) defines the sender's maximum cell rate. The user's cell rate can sometimes reach this peak, as long as the SCR is maintained.
• MCR. The minimum cell rate (MCR) defines the minimum cell rate acceptable to the sender. For example, if the MCR is 50,000, the network must guarantee that the sender can send at least 50,000 cells per second.

- CVDT. The cell variation delay tolerance (CVDT) is a measure of the variation in cell transmission times. For example, if the CVDT is 5 ns, this means that difference between the minimum and the maximum delays in delivering the cell should not exceed 5 ns.

Network-Related Attributes

The network-related attributes are those that define characteristics of the network. The following are some network-related attributes:

CLR: The cell loss ratio (CLR) defines the fraction of cells lost (or delivered so late that they are considered lost) during transmission. For example, if the sender sends 100 cells and one of them is lost, the CLR is

$CLR = 1/100 = 10^{-2}$

CTD: The cell transfer delay (CTD) is the average time needed for a cell to from source to destination. The maximum CTD and the minimum CTD are also considered attributes.

CDV: The cell delay variation (CDV) is the difference between the CTD maximum and the CTD minimum.

CER: The cell error ratio (CER) defines the fraction of the cells delivered in error.

4.6.2 Traffic Descriptors

The mechanisms by which the service classes and QoS attributes are implemented are called the traffic descriptors. A traffic descriptor defines how the system enforces and polices traffic. The algorithm to implement traffic descriptors is called the cell rate algorithm (GCRA). It uses variations of the leaky bucket algorithm for each type of service class.

Self Assessment Questions

1. In Ethernet, the source address field in the MAC frame is.....address.

(a) the original sender's physical (b) the previous station's physical
(c) the next destination's physical (d) the original sender's service port

2. ATM multiplexes cell using…….

(a) asynchronous FDM (b) synchronous FDM
(c) asynchronous TDM (d) synchronous TDM

3. ATM can use ……as a transmission medium

(a) twisted-pair cable (b) coaxial cable
(c) fiber-optic cable (d) All of the above

4. . …… is the most widely used local area network protocol.

(a) Token Ring (b) Token Bus
(c) Ethernet (d) None of the above

4.7 ARCNET

ARCNET, once quite popular in office automation, has reinvented itself into an embedded networking technology that is frequently found in applications such as industrial control, building automation, transportation, robotics and gaming. Like Ethernet and Controller Area Network (CAN), ARCNET is a data-link layer technology with no defined application layer. Designers write their own application layer to meet their particular needs and frequently do not advertise the fact that ARCNET is being used in their product.

ARCNET incorporates a token-passing protocol where media access is determined by the station with the token. When a station receives the token, it can either initiate a transmission to another station or it must pass

the token to its logical neighbour. All stations are considered peers and no one station can consume all the bandwidth since only one packet can be sent each token pass. This scheme avoids collisions and gives ARCNET its greatest advantage in real-time applications—it is deterministic! By being deterministic, the designer can accurately predict the time it takes for a particular station to gain access to the network and send a message. This is of particular importance for control or robotic applications where timely responses or coordinated motion are needed.

4.8 AppleTalk

AppleTalk was designed with a transparent network interface. That is, the interaction between client computers and network servers requires little interaction from the user. In addition, the actual operations of the AppleTalk protocols are invisible to end users, who see only the result of these operations. Two versions of AppleTalk exist: AppleTalk Phase 1 and AppleTalk Phase 2.

AppleTalk Phase 1, which is the first AppleTalk specification, was developed in the early 1980s strictly for use in local workgroups. Phase 1 therefore has two key limitations: its network segments can contain no more than 127 hosts and 127 servers, and it can support only non extended networks.

AppleTalk Phase 2, which is the second enhanced AppleTalk implementation, was designed for use in larger internetworks. Phase 2 addresses the key limitations of AppleTalk Phase 1 and features a number of improvements over Phase 1. In particular, Phase 2 allows any combination of 253 hosts or servers on a single AppleTalk network segment and supports both nonextended and extended networks.

4.8.1 AppleTalk Network Components

AppleTalk networks are arranged hierarchically. Four basic components form the basis of an AppleTalk network: *sockets*, *nodes*, *networks*, and *zones*. Figure 4.30 illustrates the hierarchical organization of these components in an AppleTalk internetwork. Each of these concepts is summarized in the sections that follow.

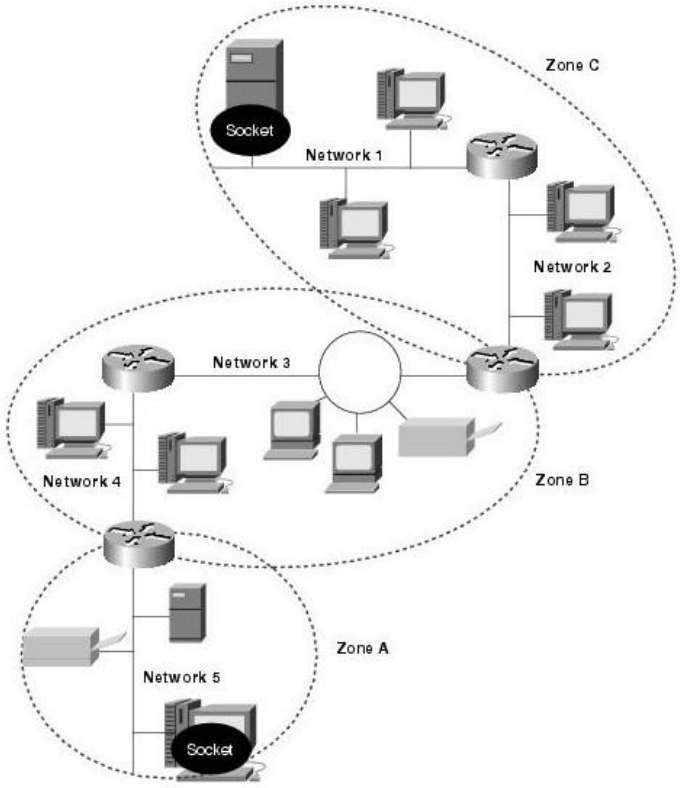


Figure 4.30: The AppleTalk internetwork consists of a hierarchy of components**.**

Sockets

An AppleTalk socket is a unique, addressable location in an AppleTalk node. It is the logical point at which upper-layer AppleTalk software processes and the network-layer Datagram-Delivery Protocol (DDP) interact. These upper-layer processes are known as *socket clients*. Socket clients own one or more sockets, which they use to send and receive datagrams. Sockets can be assigned statically or dynamically. Statically assigned sockets are reserved for use by certain protocols or other processes. Dynamically assigned sockets are assigned by DDP to socket clients upon request. An AppleTalk node can contain up to 254 different socket numbers. Figure 4.31 illustrates the relationship between the sockets in an AppleTalk node and DDP at the network layer.

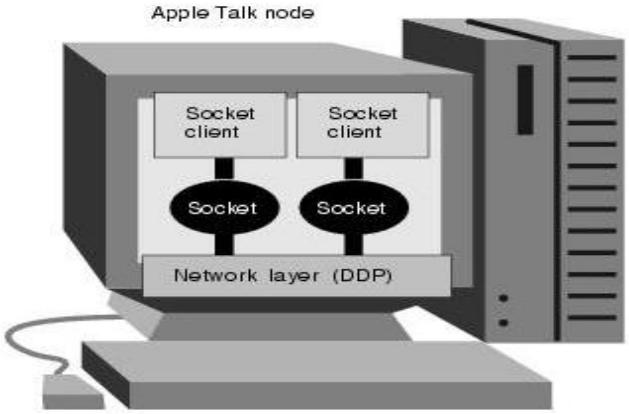


Figure 4.31: Socket clients use sockets to send and receive datagrams**.**

Nodes

An AppleTalk node is a device that is connected to an AppleTalk network. This device might be a Macintosh computer, a printer, an IBM PC, a router, or some other similar device. Within each AppleTalk node exist numerous software processes called sockets. As discussed earlier, the function of these sockets is to identify the software processes running in the device. Each node in an AppleTalk network belongs to a single network and a specific zone.

Networks

An AppleTalk network consists of a single logical cable and multiple attached nodes. The logical cable is composed of either a single physical cable or multiple physical cables interconnected by using bridges or routers. AppleTalk networks can be nonextended or extended. Each is discussed briefly in the following sections.

No extended Networks

A *no extended* AppleTalk network is a physical-network segment that is assigned only a single network number, which can range between 1 and 1,024. Network 100 and Network 562, for example, are both valid network numbers in a no extended network. Each node number in a nonextended network must be unique, and a single no extended network segment cannot have more than one AppleTalk Zone configured on it. (A zone is a logical group of nodes or networks.) AppleTalk Phase 1 supports only nonextended networks, but as a rule, nonextended network configurations are no longer used in new networks because they have been superseded by extended networks. Figure 4.32 illustrates a nonextended AppleTalk network.

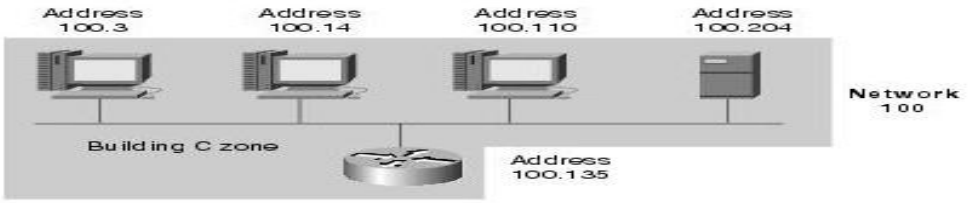


Figure 4.32: A nonextended network is assigned only one network number**.**

Extended Networks

An *extended* AppleTalk network is a physical-network segment that can be assigned multiple network numbers. This configuration is known as a *cable range*. AppleTalk cable ranges can indicate a single network number or multiple consecutive network numbers. The cable ranges Network 3-3 (unary) and Network 3-6, for example, are both valid in an extended network. Just as in other protocol suites, such as TCP/IP and IPX, each combination of network number and node number in an extended network must be unique, and its address must be unique for identification purposes. Extended networks can have multiple AppleTalk zones configured on a single network segment, and nodes on extended networks can belong to any single zone associated with the extended network. Extended network configurations have, as a rule, replaced nonextended network configurations. Figure 4.33 illustrates an extended network.

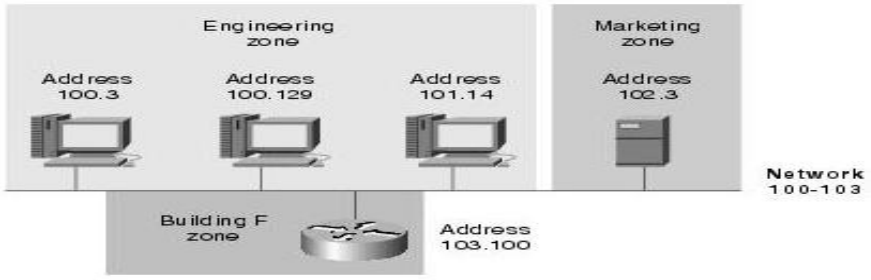


Figure 4.33: An extended network can be assigned multiple network numbers.

Zones

An AppleTalk *zone* is a logical group of nodes or networks that is defined when the network administrator configures the network. The nodes or networks need not be physically contiguous to belong to the same AppleTalk zone. Figure 4.34 illustrates an AppleTalk internetwork composed of three noncontiguous zones.

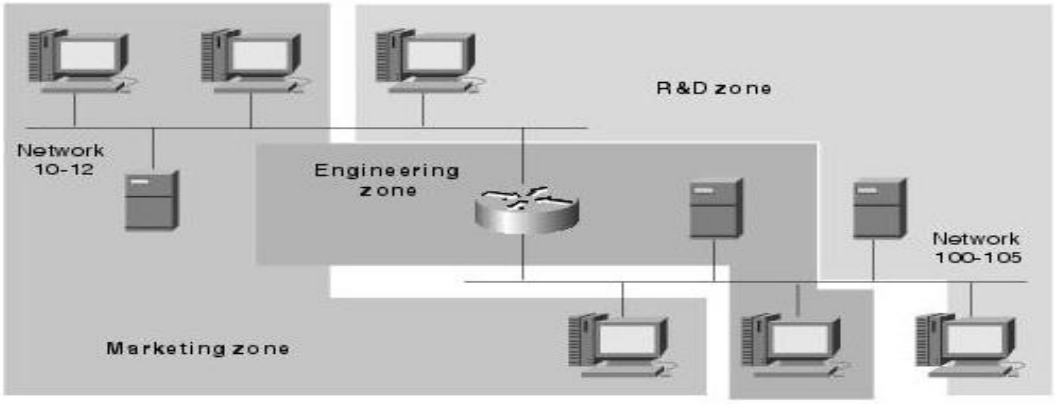


Figure 4.34: Nodes or networks in the same zone need not be physically contiguous.

Exercise: Check Your Progress 3

Note: i) Use the space below for your answer.

Ex1: Draw the hierarchy of AppleTalk internetwork consists of components.

……..……

……..……

……...

Did You Know?

AppleTalk was intended to be part of a project known as Macintosh Office, which would consist of a host machine providing routing, printer sharing and file sharing. However this project was cancelled in 1986.

4.9 Summary

- In fast Ethernet, the data rate is increased to 100Mbps, but the collision domain is reduced to 250 meters.
- Asynchronous Transfer Mode (ATM) is the cell relay protocol designed by the ATM Forum and adopted by the ITU-T. The combination of ATM and B-ISDN will allow high-speed interconnection of all the world's networks.
- User-related attributes are those attributes that define how fast the user wants to send data. These
- AppleTalk networks are the interaction between client computers and network servers requires little interaction from the user.
- AppleTalk networks are arranged hierarchically. Four basic components form the basis of an AppleTalk network: sockets, nodes, networks, and zones.
- ARCNET incorporates a token-passing protocol where media access is determined by the station with the token.

4.10 Keywords

Cell Relay: Asynchronous Transfer Mode (ATM) is the cell relay protocol designed by the ATM Forum and adopted by the ITU-T.

FDDI: Fiber distributed data interface (FDDI) is a local area network protocol using optical fiber as a medium, with a 100-Mbps data rate.

Gigabit Ethernet: It is mainly designed to use optical fiber although the protocol does not eliminate the use of twisted pair cables.

NIC: Each station on an Ethernet network has its network interface card (NIC). The NIC usually fits inside the station and provides station with a six-byte physical address.

WAN: ATM is potentially as effective a LAN and short-haul mechanism as it is a WAN mechanism.

4.11 Review Questions

1. What is a collision?
2. What is ATM technology?
3. How do we find out where ATM is available?
4. What are PPD and EPD?
5. How is the collision domain to the data rate in Ethernet network?
6. Compare the data transmission rates for traditional Ethernet Fast Ethernet Gigabit Ethernet
7. How are ATM cell multiplexed?

8. What is the purpose of service classes?
9. What is an AppleTalk?
10. What are the sockets, nodes, networks, and zones?

Answers for Self Assessment Questions

1. (b) 2. (c) 3. (c) 4. (c)

4.12 Further Studies

1. The Real-world Network Troubleshooting Manual: Tools, Techniques, and Scenarios by Alan Sugano
2. Computer Networks: A Systems Approach by Larry L. Peterson, Bruce S. Davie

5

IEEE 802 Standards

STRUCTURE

5.0 Objectives

5.1 Introduction

5.2 LAN Architecture

5.3 IEEE 802 Standards

5.4 Media Access Control

5.5 Summary

5.6 Keywords

5.7 Review Questions

5.8 Further Studies

5. 0 Objectives

After studying this chapter, you will be able to:

- Define LAN architecture
- Discuss about the IEEE 802 Standards
- Explain the Wireless LANS
- Describes the bridges

5.1 Introduction

A Local Area Network (LAN) is a group of computers and associated devices that share a common communications line or wireless link. Typically, connected devices share the resources of a single processor or server within a small geographic area (for example, within an office building). Usually, the server has applications and data storage that are shared in common by multiple computer users. A local area network may serve as few as two or three users (for example, in a home network) or as many as thousands of users (for example, in an FDDI network).

An engineer (IEEE) has produced a set of standards for LAN architectures. Although token ring and Ethernet were both created before the IEEE standards, the IEEE specifications for IEEE 802.3 (Ethernet) and IEEE 802.5 (token ring) now provide vendor-neutral standards for these important LAN technologies.

5.2 LAN Architecture

LAN (Local Area Network) refers to a group of computers interconnected into a network so that they are able to communicate, exchange information and share resources (e.g. printers, application programs, database etc).

In other words, the same computer resources can be used by multiple users in the network, regardless of the physical location of the resources.

Each computer in a LAN can effectively send and receive any information addressed to it. This information is in the form of data "packets". The standards followed to regularize the transmission of packets, are called LAN standards. There are many LAN standards as Ethernet, Token Ring, and FDDI etc. Usually LAN standards differ due to their media access technology and the physical transmission medium.

A LAN is a high-speed, fault-tolerant data network that covers a relatively small geographic area. It typically connects workstations, personal computers, printers, and other devices. LANs offer computer users many advantages, including shared access to devices and applications, file exchange between connected users, and communication between users via electronic mail and other applications.

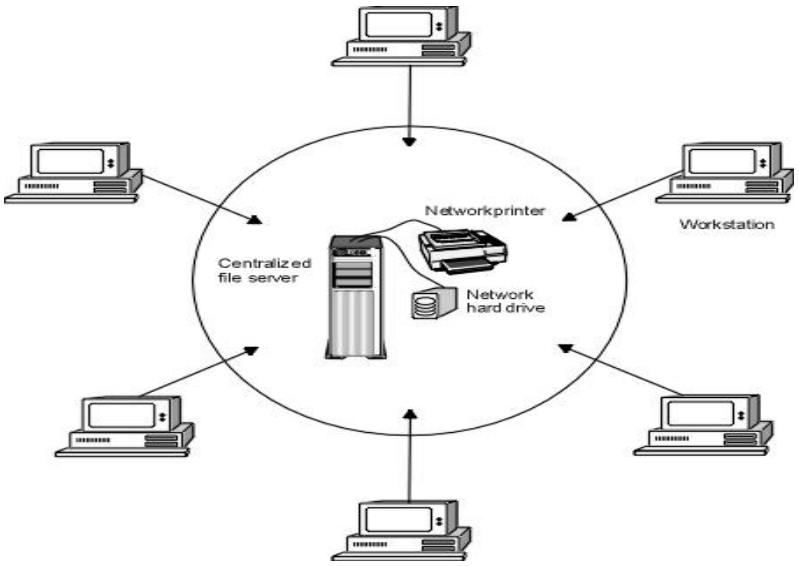


Figure 5.1: LAN architecture.

Did You Know?

In 1991 NCR Corporation/AT&T invented the precursor to IEEE 802.11 in Nieuwegein.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: How LAN Architecture works? Define with using graphs.

……..……

……..……

……...

5.3 IEEE 802 Standards

The IEEE (Institute of Electrical and Electronic Engineers) is a technical association of industry professionals with a common interest in advancing all communications technologies. This topic describes the standards developed by the LAN/MAN Standards Committee (LMSC), which develops LAN (local area network) and MAN (metropolitan area network) standards, mainly for the lowest two layers in the OSI reference model. LMSC is also called the IEEE Project 802, so the standards it develops are referenced as IEEE 802 standards. In general, IEEE 802 standards define physical network interfaces such as network interface cards, bridges,

routers, connectors, cables, and all the signaling and access methods associated with physical network connections.

802.1 Higher Layer LAN Protocols Working Group
802.2 Logical Link Control Working Group (Inactive)
802.3 Ethernet Working Group
802.4 Token Bus Working Group (Inactive)
802.5 Token Ring Working Group
802.6 Metropolitan Area Network Working Group (Inactive)
802.7 Broadband TAG (Inactive)
802.8 Fiber Optic TAG
802.9 Isochronous LAN Working Group
802.10 Security Working Group
802.11 Wireless LAN Working Group
802.12 Demand Priority Working Group
802.14 Cable Modem Working Group
802.15 Wireless Personal Area Network (WPAN) Working Group
802.16 Broadband Wireless Access Working Group
802.17 Resilient Packet Ring Working Group

5.3.1 IEEE 802.3 (CSMA/CD)

Carrier Sense Multiple Access / Collision Detection, a set of rules determining how network devices respond when two devices attempt to use a data channel simultaneously (called a collision). Standard Ethernet networks use CSMA/CD to physically monitor the traffic on the line at participating stations. If no transmission is taking place at the time, the particular station can transmit. If two stations attempt to transmit simultaneously, this causes a collision, which is detected by all participating stations. After a random time interval, the stations that collided attempt to transmit again. If another collision occurs, the time intervals from which the random waiting time is selected are increased step by step. This is known as exponential back off.

CSMA/CD is a type of contention protocol. Networks using the CSMA/CD procedure are simple to implement but do not have deterministic transmission characteristics. The CSMA/CD method is internationally standardized in IEEE 802.3 and ISO 8802.3.

- IEEE 802.3 is a collection of IEEE standards defining the physical layer and the media access control (MAC) sub-layer of the data link layer of wired Ethernet.
- This is generally a LAN technology with some WAN applications. Physical connections are made between nodes and/or infrastructure devices (hubs, switches, routers) by various types of copper or fiber cable.
- IEEE 802.3 is a technology that can support the IEEE 802.1network architecture.
- The maximum packet size is 1518 bytes, although to allow the Q-tag for Virtual LAN and priority data in 802.3ac it is extended to 1522 bytes. If the upper layer protocol submits a PDU (Protocol data unit) less than 64 bytes, 802.3 will pad the data field to achieve the minimum 64 bytes.

Physical Connections

IEEE 802.3 specifies several different physical layers, whereas Ethernet defines only one. Each IEEE 802.3 physical layer protocol has a name that summarizes its characteristics. The coded components of an IEEE 802.3 physical-layer name are shown in Figure 5.2.

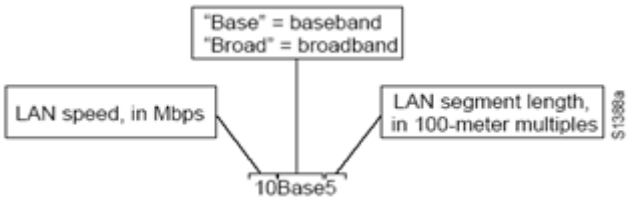


Figure 5.2: IEEE 802.3 Physical-Layer Name Components.

A summary of Ethernet Version 2 and IEEE 802.3 characteristics appears in Table 5.1.

Table 5.1 Ethernet Version 2 and IEEE 802.3 Physical Characteristics

		IEEE 802.3 Values				
Characteristic	**Ethernet Value**	**10Base5**	**10Base2**	**1Base5**	**10BaseT**	**10Broad36**
Data rate (Mbps)	10	10	10	1	10	10
Signaling method	Baseband	Baseband	Baseband	Baseband	Baseband	Broadband
Maximum segment length (m)	500	500	185	250	100 Unshielded twisted-pair wire	1800
Media	50-ohm coax (thick)	50-ohm coax (thick)	50-ohm coax (thin)	Unshielded twisted-pair wire	Unshielded twisted-pair wire	75-ohm coax
Topology	Bus	Bus	Bus	Star	Star	Bus

Ethernet is most similar to IEEE 802.3 10Base5. Both of these protocols specify a bus topology network with a connecting cable between the end stations and the actual network medium. In the case of Ethernet, that cable is called a transceiver cable. The transceiver cable connects to a transceiver device attached to the physical network medium. The IEEE 802.3 configuration is much the same, except that the connecting cable is referred to as an attachment unit interface (AUI), and the transceiver is called a medium attachment unit (MAU). In both cases, the connecting cable attaches to an interface board (or interface circuitry) within the end station.

Frame Formats

Ethernet and IEEE 802.3 frame formats are shown in Figure 5.3.

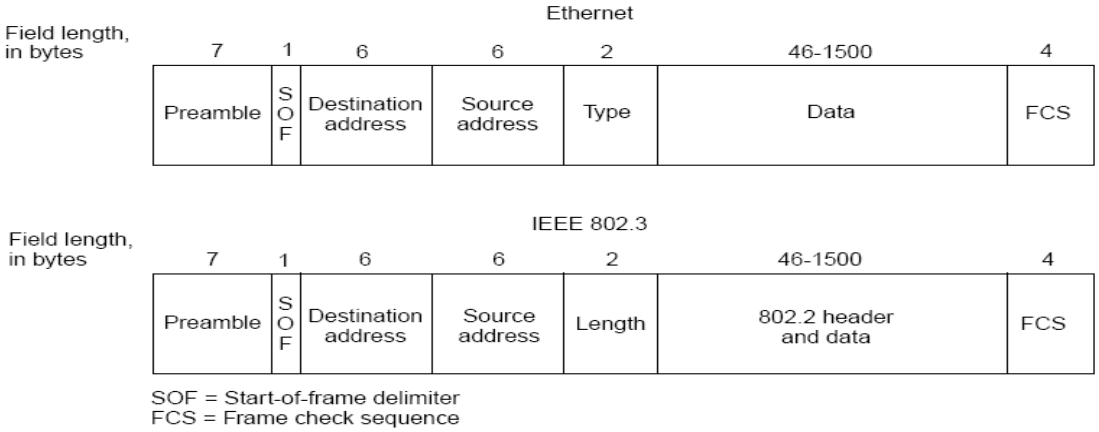


Figure 5.3: Ethernet and IEEE 802.3 Frame Formats.

Both Ethernet and IEEE 802.3 frames begin with an alternating pattern of ones and zeros called a preamble. The preamble tells receiving stations that a frame is coming.

The byte before the destination address in both an Ethernet and IEEE 802.3 frame is a start-of-frame (SOF) delimiter. This byte ends with two consecutive one bits, which serve to synchronize the frame reception portions of all stations on the LAN.

Immediately following the preamble in both Ethernet and IEEE 802.3 LANs are the destination and source address fields. Both Ethernet and IEEE 802.3 addresses are 6 bytes long. Addresses are contained in hardware on the Ethernet and IEEE 802.3 interface cards. The first 3 bytes of the addresses are specified by the IEEE on a vendor-dependent basis, while the last 3 bytes are specified by the Ethernet or IEEE 802.3 vendor. The source address is always a unicast (single node) address, while the destination address may be unicast, multicast (group), or broadcast (all nodes).

In Ethernet frames, the 2-byte field following the source address is a type field. This field specifies the upper-layer protocol to receive the data after Ethernet processing is complete.

In IEEE 802.3 frames, the 2-byte field following the source address is a length field, which indicates the number of bytes of data that follow this field and precede the frame check sequence (FCS) field.

Following the type/length field is the actual data contained in the frame. After physical-layer and link-layer processing is complete, this data will eventually be sent to an upper-layer protocol. In the case of Ethernet, the upper-layer protocol is identified in the type field. In the case of IEEE 802.3, the upper-layer protocol must be defined within the data portion of the frame, if at all. If data in the frame is insufficient to fill the frame to its minimum 64-byte size, padding bytes are inserted to ensure at least a 64-byte frame.

After the data field is a 4-byte FCS field containing a cyclic redundancy check (CRC) value. The CRC is created by the sending device and recalculated by the receiving device to check for damage that might have occurred to the frame in transit.

Did You Know?

Originally, IBM developed Token Ring network in the 1970s.

5.3.2 Token Bus (IEEE 802.4)

IEEE 802.4 defines the medium access control (MAC) layer for bus networks that use a token-passing mechanism (token bus networks). This is an application of the concepts used in token ring networks. The main difference is that the endpoints of the bus do not meet to form a physical ring. The IEEE 802.4 Working Group is disbanded.

Although Ethernet was widely used in the offices, but people interested in factory automation did not like it because of the probabilistic MAC layer protocol. They wanted a protocol which can support priorities and has predictable delay. These people liked the conceptual idea of Token Ring network but did not like its physical implementation as a break in the ring cable could bring the whole network down and ring is a poor fit to their linear assembly lines. Thus a new standard, known as Token bus, was developed, having the robustness of the Bus topology, but the known worst-case behavior of a ring. Here stations are logically connected as a ring but physically on a Bus and follows the collision-free token passing medium access control protocol. So the motivation behind token bus protocol can be summarized as:

- The probabilistic nature of CSMA/ CD leads to uncertainty about the delivery time; which created the need for a different protocol
- The token ring, on the hand, is very vulnerable to failure.
- Token bus provides deterministic delivery time, which is necessary for real time traffic.
- Token bus is also less vulnerable compared to token ring.

Functions of a Token Bus

It is the technique in which the station on bus or tree forms a logical ring that is the stations are assigned positions in an ordered sequence, with the last number of the sequence followed by the first one as shown in Figure 5.4. Each station knows the identity of the station following it and preceding it.

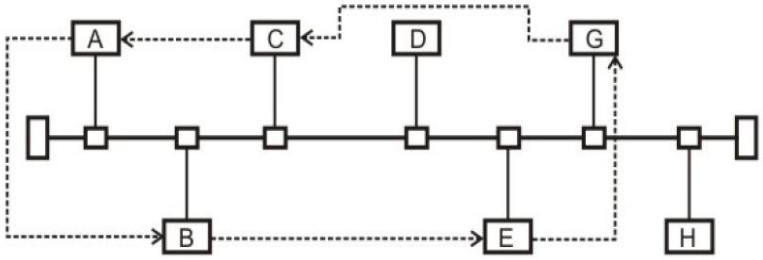


Figure 5.4: Token Bus topology.

A control packet known as a Token regulates the right to access. When a station receives the token, it is granted control to the media for a specified time, during which it may transmit one or more packets and may poll stations and receive responses when the station is done, or if its time has expired then it passes token to next station in logical sequence. Hence, steady phase consists of alternate phases of token passing and data transfer.

The MAC sub layer consists of four major functions: the interface machine (IFM), the access control machine (ACM), the receiver machine (RxM) and the transmit machine (TxM).

IFM interfaces with the LLC sub layer. The LLC sub layer frames are passed on to the ACM by the IFM and if the received frame is also an LLC type, it is passed from RxM component to the LLC sub layer. IFM also provides quality of service.

The ACM is the heart of the system. It determines when to place a frame on the bus, and responsible for the maintenance of the logical ring including the error detection and fault recovery. It also cooperates with other stations ACM's to control the access to the shared bus, controls the admission of new stations and attempts recovery from faults and failures.

The responsibility of a TxM is to transmit frame to physical layer. It accepts the frame from the ACM and builds a MAC protocol data unit (PDU) as per the format.

The RxM accepts data from the physical layer and identifies a full frame by detecting the SD and ED (start and end delimiter). It also checks the FCS field to validate an error-free transmission.

Frame Form

The frame format of the Token Bus is shown in Figure 5.5. Most of the fields are same as Token Ring. So, we shall just look at the Frame Control Field in Table 5.2.

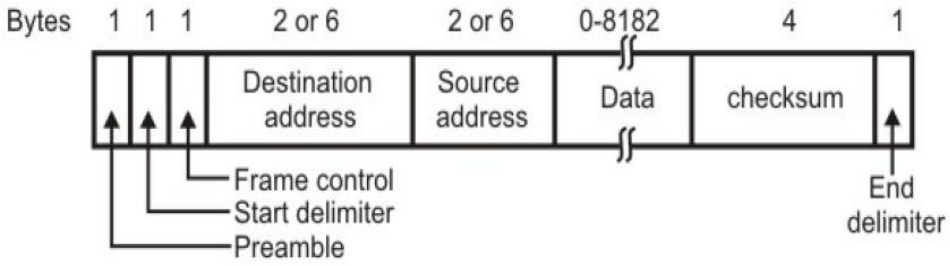


Figure 5.5: Token Bus frame format.

Table 5.2: The Frame Control Field

Frame Control	Name	Use
0000 0000	Claim-Token	Ring Initialization
0000 0001	Solicit-successor -1	Addition to the Ring
0000 0010	Solicit-successor -2	Addition to the Ring
0000 0011	Who-follows	Recovery from lost token
0000 0100	Resolve Contention	Multiple station to join the Ring
0000 1000	Token	Pass the Token
0000 1100	Set-Successor	Deletion from the ring

Logical ring maintenance

The MAC performs the following functions as part of its maintenance role of the ring. Addition to the Ring: Non-participating stations must periodically be granted the opportunity to insert themselves into the ring. Each node in the ring periodically grants an opportunity for new nodes to enter the ring while holding the token. The node issues a solicit–successor–1 packet, inviting nodes with an address between itself and the next node in logical sequence to request entrance. The transmitting node then waits for a period of time equal to one response window or slot time (twice the end-to-end propagation delay of the medium). If there is no request, the token holder sets its successor node to be the requesting node and transmits the token to it; the requester sets the linkages accordingly and proceeds. If more than one node requests, to enter the ring, the token holder will detect a garbled transmission. The conflict is resolved by addressed based contention scheme; the token holder transmits a resolved contention packet and waits for four response windows. Each requester can transmit in one of these windows, based on the first two bits of its address. If requester hears anything before its windows comes up, it refrains from requesting entrance. If a token holder receives a valid response, then it can proceed, otherwise it tries again and only those nodes that request the first time are allowed to request this time, based on the second pair of bits in their address. This process continues until a valid request is received or no request is received, or a maximum retry count is reached. In latter cases, the token holder passes the token to logical successor in the ring.

Deletion from Ring: A station can voluntarily remove itself from the ring by splicing together its predecessor and successor. The node which wants to be deleted from the ring waits until token comes to it, then it sends a set successor packet to its predecessor, instructing it to splice to its successor.

Fault Management: Errors like duplicate address or broken ring can occur. A suitable management scheme should be implemented for smooth functioning. It is done by the token-holder first, while holding the token, node may hear a packet, indicating that another node has the token. In this case, it immediately drops the token by reverting to listener mode, and the number of token holders drops immediately from one to zero. Upon completion of its turn, it immediately issues a data or token packet. The sequence of steps is as follows:

i. After sending the token, the token issuer will listen for one slot time to make sure that its predecessor is active.
ii. If the issuer does not hear a valid packet, it reissues the token to the same successor one more time.
iii. After two failures, the issuer assumes that its successor has failed and issues a "who-follows" packet, asking for the identity of the node that follows the failed node. The issuer should get back a set successor packet from the second node down the time. If so, the issuer adjusts its linkage and issues a token (back to step i).
iv. If the issuing node gets a response to its "who-follows" packet, it tries again.
v. If the "who-follows" tactic fails, the node issues a solicit-successor-2 packet with full address range (i.e. every node is invited to respond). If this packet works then the ring is established and procedure continues.
vi. If two attempts in step (v) fail, it assumes that a catastrophe has happened; perhaps the node receiver has failed. In any case, the node ceases the activity and listen the bus.

Ring Initialization: Ring is to be initialized by starting the token passing. This is necessary when the network is being setup or when ring is broken down. Some decentralized algorithms should take care of, who starts first, who starts second, etc. it occurs when one or more stations detects a lack of bus activity lasting longer than a specific time. The token may get lost. This can occur on a number of occasions. For example, when network has been just powered up, or a token holding station fails. Once its time out expires, a node will issue a claim token packet. Contending clients are removed in a similar fashion to the response window process.

5.3.3 Token Ring (IEEE 802.5)

Stations on a token ring LAN are logically organized in a ring topology with data being transmitted sequentially from one ring station to the next with a control token circulating around the ring controlling access. This token passing mechanism is shared by ARCNET, token bus, and FDDI, and has theoretical advantages over the stochastic CSMA/CD of Ethernet.

It is still IBM's primary local-area network (LAN) technology. The related IEEE 802.5 specification is almost identical to and completely compatible with IBM's Token Ring network. In fact, the IEEE 802.5 specification was modeled after IBM Token Ring, and on the same lines. The term Token Ring is generally used to refer to both IBM's Token Ring network and IEEE 802.5 networks.

The medium access mechanism used by Ethernet (CSMA/CD) may results in collision. Nodes attempt to a number of times before they can actually transmit, and even when they start transmitting there are chances to encounter collisions and entire transmission need to be repeated. And all this become worse one the traffic is heavy i.e. all nodes have some data to transmit. Apart from this there is no way to predict either the occurrence of collision or delays produced by multiple stations attempting to capture the link at the same time. So all these problems with the Ethernet gives way to an alternate LAN technology, Token Ring.

Token Ring and IEEE802.5 are based on token passing MAC protocol with ring topology. They resolve the uncertainty by giving each station a turn on by one. Each node takes turns sending the data; each station may transmit data during its turn. The technique that coordinates this turn mechanism is called Token passing; as a Token is passed in the network and the station that gets the token can only transmit. As one node transmits at a time, there is no chance of collision.

Stations are connected by point-to-point links using repeaters. Mainly these links are of shielded twisted-pair cables. The repeaters function in two basic modes: Listen mode, Transmit mode. A disadvantage of this topology is that it is vulnerable to link or station failure. But a few measures can be taken to take care of it.

Differences between Token Ring and IEEE 802.5

Both of these networks are basically compatible, although the specifications differ in some ways.

- IEEE 802.5 does not specify a topology, although virtually all IEEE 802.5 implementations are based on the star topology. While IBM's Token Ring network explicitly specifies a star, with all end stations attached to a device called a Multi-Station Access Unit (MSAU).
- IEEE 802.5 does not specify a media type, although IBM Token Ring networks use twisted-pair wire.
- There are few differences in routing information field size of the two.

Token Ring Operation

Token-passing networks move a small frame, called a token, around the network. Possession of the token grants the right to transmit. If a node receiving the token has no information to send, it passes the token to the next end station. Each station can hold the token for a maximum period of time.

If a station possessing the token does have information to transmit, it seizes the token, alters 1 bit of the token (which turns the token into a start-of-frame sequence), appends the information that it wants to transmit, and sends this information to the next station on the ring. While the information frame is circling the ring, no token is on the network (unless the ring supports early token release), which means that other stations wanting to

transmit must wait. Therefore, collisions cannot occur in Token Ring networks. If early token release is supported, a new token can be released immediately after a frame transmission is complete.

The information frame circulates around the ring until it reaches the intended destination station, which copies the information for further processing. The information frame makes a round trip and is finally removed when it reaches the sending station. The sending station can check the returning frame to see whether the frame was seen and subsequently copied by the destination station in error-free form. Then the sending station inserts a new free token on the ring, if it has finished transmission of its packets. Unlike CSMA/CD networks (such as Ethernet), token-passing networks are deterministic, which means that it is possible to calculate the maximum time that will pass before any end station will be capable of transmitting. Token Ring networks are ideal for applications in which delay must be predictable and robust network operation is important.

Priority System

Token Ring networks use a sophisticated priority system that permits certain user-designated, high-priority stations to use the network more frequently. Token Ring frames have two fields that control priority: the priority field and the reservation field. Only stations with a priority equal to or higher than the priority value contained in a token can seize that token. After the token is seized and changed to an information frame, only stations with a priority value higher than that of the transmitting station can reserve the token for the next pass around the network. When the next token is generated, it includes the higher priority of the reserving station. Stations that raise a token's priority level must reinstate the previous priority after their transmission is complete.

Ring Maintenance

There are two error conditions that could cause the token ring to break down. One is the lost token in which case there is no token the ring, the other is the busy token that circulates endlessly. To overcome these problems, the IEEE 802 standard specifies that one of the stations be designated as 'active monitor'. The monitor detects the lost condition using a timer by time-out mechanism and recovers by using a new free token. To detect a circulating busy token, the monitor sets a 'monitor bit' to one on any passing busy token. If it detects a busy token with the monitor bit already set, it implies that the sending station has failed to remove its packet and recovers by changing the busy token to a free token. Other stations on the ring have the role of passive monitor. The primary job of these stations is to detect failure of the active monitor and assume the role of active monitor. A contention-resolution is used to determine which station to take over.

Physical Layer

The Token Ring uses shielded twisted pair of wire to establish point-point links between the adjacent stations. The baseband signaling uses differential Manchester encoding. To overcome the problem of cable break or network failure, which brings the entire network down, one suggested technique, is to use wiring concentrator as shown in Figure 5.6.

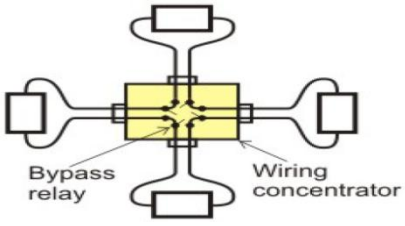


Figure 5.6 Star Connected Ring topology.

It imposes the reliability in an elegant manner. Although logically the network remains as a ring, physically each station is connected to the wire center with two twisted pairs for 2-way communication. Inside the wire center, bypass relays are used to isolate a broken wire or a faulty station. This Topology is known as Star-Connected Ring.

Frame Format

Token Ring and IEEE 802.5 support two basic frame types: tokens and data/command frames. Tokens are 3 bytes in length and consist of a start delimiter, an access control byte, and an end delimiter. Data/command frames vary in size, depending on the size of the Information field. Data frames carry information for upper-layer protocols, while command frames contain control information and have no data for upper-layer protocols.

Token Frame Fields

Start Delimiter	Access Control	Ending delimiter

Token Frame contains three fields, each of which is 1 byte in length:

- *Start delimiter (1 byte):* Alerts each station of the arrival of a token (or data/command frame). This field includes signals that distinguish the byte from the rest of the frame by violating the encoding scheme used elsewhere in the frame.
- *Access-control (1 byte):* Contains the Priority field (the most significant 3 bits) and the Reservation field (the least significant 3 bits), as well as a token bit (used to differentiate a token from a data/command frame) and a monitor bit (used by the active monitor to determine whether a frame is circling the ring endlessly).
- *End delimiter (1 byte):* Signals the end of the token or data/command frame. This field also contains bits to indicate a damaged frame and identify the frame that is the last in a logical sequence.

Data/Command Frame Fields

Start Delimiter	Access Control	Frame Control	Destination address	Source address	Data	Frame check sequence	End Delimiter	Frame Status

Data/command frames have the same three fields as Token Frames, plus several others. The Data/command frame fields are described below:

- *Frame-control byte (1 byte)*—Indicates whether the frame contains data or control information. In control frames, this byte specifies the type of control information.
- *Destination and source addresses (2-6 bytes)*—Consists of two 6-byte address fields that identify the destination and source station addresses.
- *Data (up to 4500 bytes)*—Indicates that the length of field is limited by the ring token holding time, which defines the maximum time a station can hold the token.
- *Frame-check sequence (FCS- 4 byte)*—Is filed by the source station with a calculated value dependent on the frame contents. The destination station recalculates the value to determine whether the frame was damaged in transit. If so, the frame is discarded.
- *Frame Status (1 byte)*—This is the terminating field of a command/data frame. The Frame Status field includes the address-recognized indicator and frame-copied indicator.

5.3.4 Comparison between IEEE 802.3, 802.4 and 802.5

A comparison of the three standards for different functions is shown in Table 5.3and results of the analysis of the performance of the three standards are summarized below:

- The CSMA/CD protocol shows strong dependence on the parameter ‘a’, which is the ratio of the propagation time to the transmission time. It offers shortest delay under light load and it is most sensitive under heavy load conditions.
- Token ring is least sensitive to different load conditions and different packet sizes.

- Token bus is highly efficient under light load conditions.

Table 5.3 Comparison of the three standards

Function	CSMA/CD	Token bus	Token ring
Access determination	Contention	Token	Token
Packet length restriction	64 bytes (Greater than 2.Tprop)	None	None
Priority	Not supported	Supported	Supported
Sensitivity to work load	Most sensitive	Sensitive	Least sensitive
Principle advantage	Simplicity, wide installed base	Regulated/fair access	Regulated/fair access
Principle disadvantage	Nondeterministic delay	Complexity	Complexity

5.3.5 Compare Token Passing with CSMA/CD

Ethernet and IEEE 802.3 specify similar technologies. Both are CSMA/CD LANs. Stations on a CSMA/CD LAN can access the network at any time. Before sending data, CSMA/CD stations "listen" to the network to see if it is already in use. If it is, the station wishing to transmit waits. If the network is not in use, the station transmits. A collision occurs when two stations listen for network traffic, "hear" none, and transmit simultaneously. In this case, both transmissions are damaged, and the stations must retransmit at some later time. *Back off* algorithms determine when the colliding stations retransmit. CSMA/CD stations can detect collisions, so they know when they must retransmit.

Both Ethernet and IEEE 802.3 LANs are broadcast networks. In other words, all stations see all frames, regardless of whether they represent an intended destination. Each station must examine received frames to determine if the station is a destination. If so, the frame is passed to a higher protocol layer for appropriate processing.

Differences between Ethernet and IEEE 802.3 LANs are subtle. Ethernet provides services corresponding to Layers 1 and 2 of the OSI reference model, while IEEE 802.3 specifies the physical layer (Layer 1) and the channel-access portion of the link layer (Layer 2), but does not define a logical link control protocol. Both Ethernet and IEEE 802.3 are implemented in hardware. Typically, the physical manifestation of these protocols is either an interface card in a host computer or circuitry on a primary circuit board within a host computer.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Identify IEEE 802 Standards.

……..……

……..……

……...

5.4 Media Access Control

The Media Access Control is often said to be a sub-layer of the OSI data Link layer. On every network interface adaptor card there is a set of computer chips that handle communication with the physical media (copper wire, fiber optic cable or the air) by controlling the communication signal (electricity, light or radio

frequencies) over the physical media. The computer chips that control the electricity transmitted and received on a copper wire are MAC-related hardware.
The MAC sub layer provides the means to access the physical medium used for communication. The MAC sub layer also communicates with the Logical Link Control (LLC) sub-layer above it allowing it to access and speak to the upper layer network protocols such as IP.

5.4.1 MAC Addresses

The IEEE 802.3 Ethernet and 802.5 Token Ring protocols specify that the MAC sub-layer must supply a 48-bit (6 byte) address. The MAC address is most frequently represented as 12 hexadecimal digits. The MAC address uniquely identifies a specific network device and MAC addresses must be unique on a given LAN (a network of computing devices in a single subnet of IP addresses). The first 12-bit portion of the MAC address identifies the vendor of the network device; the last 12-bit portion identifies the unique id of the device itself. When looking at a hexadecimal representation of the MAC address, the first six hexadecimal digits identify the vendor and the last six hexadecimal digits identify the specific network interface card.

5.4.2 Duplicate MAC Addresses

Manufacturers reuse MAC addresses and they ship cards with duplicate addresses to different parts of the United States or the World so that there is only a very small chance two computers with network cards with the same MAC address will end up on the same network.
MAC addresses are 'burned' into the Network Interface Card (NIC), and cannot be changed. ARP and RARP on how IP addresses are translated into MAC addresses and vice versa.
In order for a network device to be able to communicate, the MAC address it is using must be unique. No other device on that local network subnet can use that MAC address. If two devices have the same MAC address (which occurs more often than network administrators would like), neither computer can communicate properly. On an Ethernet LAN, this will cause a high number of collisions. Duplicate MAC addresses on the same LAN are a problem. Duplicate MAC addresses separated by one or more routers are not a problem since the two devices would not see each other and will use the router to communicate.

5.4.3 MAC Sub layer

In the OSI protocol stack, channel allocation is addressed in the Medium access control (MAC) sub layer. This is a sub layer of the Data Link Layer - considered to be below the Logical Link Control (LLC) sub-layer. Many LAN technologies, such as Ethernet are based on this type of architecture. The MAC layer provides an unreliable connectionless service; if required, the LLC layer can convert this into a reliable service by using an ARQ protocol.

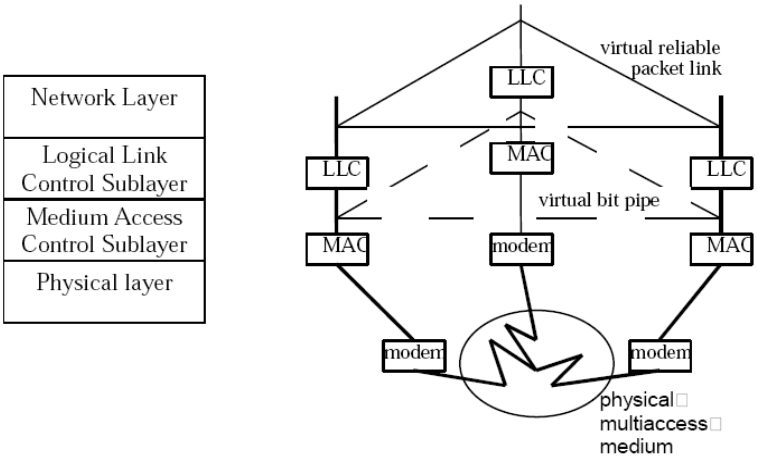


Figure 5.7: MAC sub layer

5.4.4 MAC Frame Format

Since there are various types of Network Interfaces (Ethernet, Token Ring, FDDI etc.) the MAC frame format differs by protocol according to it is design. However most will have at a minimum the following fields:

Table 5.4 MAC frame format

MAC CONTROL	DESTINATION MAC ADDR	SOURCE MAC ADDR	LLC PDU	CRC

MAC Control Field

The MAC control field contains all information used for flow control, connection establishment and teardown as well as error control. Not all protocols provide for establishment/teardown, flow control and error recovery. The content of this field is dependent upon the specified standards for that particular data link layer protocol (Ethernet, Token Ring, FDDI etc.)

Destination/Source Mac Fields

The source MAC address field contains the MAC address of the source machine--the transmitting device (since some computers with MAC addresses are not called computers-cell phones have MAC addresses), and the destination device is the receiver. The destination MAC is closer to the 'front' (left side in the diagram) of the frame for easier scanning, mostly because it is the destination device that is important as that is the device we are trying to reach.

When the receiver responds to the frame, it will use the source address to generate the destination portion of the frame it sends out. In other words, the source MAC in the frame received becomes the destination MAC in the frame transmitted as a response.

LLC PDU Field

When network communication protocols such as Ethernet or FDDI or Token Ring, they are described as being Physical and Data Link layer protocols-they perform functions that are said to be Physical and Data Link Layer functions as listed in the OSI Model of networking. For Ethernet and Token Ring the Data Link layer is described as being broken into two sub-layers, the MAC sub layer (for the MAC address and Media Access Control functions) and the Logical Link Control sub layer LLC.

The Logical Link Control Packet Data Unit field (LLC PDU) contains data from the from the LLC sub-layer of the data link layer protocol (eg. Ethernet, FDDI, Token Ring etc.). The LLC information is used to keep track of which piece of data is sent to which IP address and application. For example, the LLC information helps a Web browser keep track of which data being received is part of an image in a Web page, and which data is the text in the body of the web page itself.

CRC Checksum Field

This field contains what is called a 'checksum' that is the product of a Cyclic Redundancy Check (CRC check). A CRC check is a mathematical formula that uses the data as input and produces a numeric result that is almost as unique as the input data. Using the CRC checksum value it is possible to verify the integrity of the frame. Before transmitting the frame, the source computer calculates the checksum and places the checksum value in this field. The receiving computer looks at the same data in the frame and also calculates the checksum. If the CRC it calculates is different from the CRC checksum in the CRC checksum field, the CRC check has failed. Frames that fail this checksum test are discarded because there is a near certainty that the frame is damaged.

Ethernet protocols

Ethernet protocols refer to the family of local-area network (LAN) covered by the IEEE 802.3. In the Ethernet standard, there are two modes of operation: half-duplex and full-duplex modes. In the half duplex mode, data are transmitted using the popular Carrier-Sense Multiple Access/Collision Detection (CSMA/CD) protocol on a shared medium. The main disadvantages of the half-duplex are the efficiency and distance limitation, in which the link distance is limited by the minimum MAC frame size. This restriction reduces the efficiency

drastically for high-rate transmission. Therefore, the carrier extension technique is used to ensure the minimum frame size of 512 bytes in Gigabit Ethernet to achieve a reasonable link distance.
Four data rates are currently defined for operation over optical fiber and twisted-pair cables:

- 10 Mbps - 10Base-T Ethernet (IEEE 802.3)
- 100 Mbps - Fast Ethernet (IEEE 802.3u)
- 1000 Mbps - Gigabit Ethernet (IEEE 802.3z)
- 10-Gigabit - 10 Gbps Ethernet (IEEE 802.3ae).

Did You Know?
Victor "Vic" Hayes, a Senior Research Fellow at the Delft University of Technology, has been called the "father of Wi-Fi". He was involved in designing the initial standards within the IEEE.

Self Assessment Question

1. A LAN is a……, fault-tolerant data network that covers a relatively small geographic area.
(a) High-speed (b) Low-speed
(c) Medium-speed (d) None of these.

2. ……..define physical network interfaces such as network interface cards, bridges, routers, connectors, cables, and all the signaling and access methods associated with physical network connections.
(a) TCP/IP (b) IEEE 802 standards
(c) FTP (d) None of these

3. IEEE 802.3 is a technology that can support the ……network architecture.
(a) IEEE 802.2 (b) IEEE 802.5
(c) IEEE 802.1 (d) None of these

4. ……..is a collection of IEEE standards defining the physical layer and the media access control (MAC) sub-layer of the data link layer of wired Ethernet?
(a) IEEE 802.2 (b) IEEE 802.6
(c) IEEE 802.4 (d) IEEE 802.3

5. ……..defines the medium access control(MAC) layer for bus networks that use a token-passing mechanism (token bus networks).
(a) IEEE 802.4 (b) IEEE 802.3
(c) IEEE 802.5 (d) IEEE 802.7

6. ……..provides deterministic delivery time, which is necessary for real time traffic.
(a) Token ring (b) Token bus
(c) Both (a) and (b) (d) None of these

5.5 Summary

- IEEE has produced a set of standards for LAN architectures. Although token ring and ethernet were both created before the IEEE standards, the IEEE specifications for IEEE 802.3 (ethernet) and IEEE 802.5 (token ring) now provide vendor-neutral standards for these important LAN technologies.

- LAN (Local Area Network) refers to a group of computers interconnected into a network so that they are able to communicate, exchange information and share resources (e.g. printers, application programs, database etc).
- IEEE 802.3 is a collection of IEEE standards defining the physical layer and the media access control (MAC) sub-layer of the data link layer of wired Ethernet.
- The MAC sublayer consists of four major functions: the interface machine (IFM), the access control machine (ACM), the receiver machine (RxM) and the transmit machine (TxM).

5.6 Keywords

Bridge: It is a network communication device that is used to connect one segment of the network with another that uses the same protocol.

***CRC check*:** It is a mathematical formula that uses the data as input and produces a numeric result that is almost as unique as the input data.

IEEE (Institute of Electrical and Electronic Engineers): It is a technical association of industry professionals with a common interest in advancing all communications technologies.

***LAN*:** It is a group of computers and associated devices that share a common communications line or wireless link.

***MAC addresses*:** These are 'burned' into the Network Interface Card (NIC), and cannot be changed. ARP and RARP on how IP addresses are translated into MAC addresses and vice versa.

***Wireless LANs*:** These networks are set up to provide wireless connectivity within a finite coverage area.

5.7 Review Questions

1. Explain the LAN architecture.
2. What are the IEEE 802 standards? Explain briefly.
3. Define IEEE 802.3 (CSMA/CD).
4. What is the main purpose token bus (IEEE 802.4)?
5. Write the short notes token ring (IEEE 802.5).
6. Differentiate between token passings with CSMA/CD.
7. Explain the media access control.
8. What is the concept of Wireless LANs?
9. Explain the bridges.
10. Discuss latest trends in LAN technologies LAN devices.

Answers for Self Assessment Questions

1. (a) 2. (b) 3. (c) 4. (d) 5(a) 6. (b)

5.8 Further Studies

1. IEEE 802.11 Handbook: A Designer's Companion (IEEE Standards Wireless Networks) by Al Petrick,Bob O'Hara
2. Computer Networks: A Systems Approach by Larry L. Peterson, Bruce S. Davie

6

TCP/IP Reference Model

STRUCTURE

6.0 Objectives

6.1 Introduction

6.2 Overview of TCP/IP

6.3 TCP/IP Protocol

6.4 User Datagram Protocol

6.5 Internet Control Message Protocol (ICMP)

6.6 Address Resolution Protocol (ARP)

6.7 Reverse Address Resolution Protocol (RARP)

6.8 Simple Mail Transfer Protocol (SMTP)

6.9 File Transfer Protocol

6.10 Dynamic Host Configuration Protocol (DHCP)

6.11 Remote Login (rlogin)

6.12 Network File System (NFS)

6.13 Summary

6.14 Keywords

6.15 Review Questions

6.16 Further Studies

6.0 Objectives

After studying this chapter, you will be able to:

- Discuss the overview of TCP/IP
- Explain the TCP/IP protocol
- Define internet control message protocol
- Explain the reverse address resolution protocol

6.1 Introduction

Transmission Control Protocol/Internet Protocol (TCP/IP) is an industry standard protocols that is designed for large networks consisting of network segments that are connected by routers. TCP/IP is the protocol that is

used on the Internet, which is the collection of thousands of networks worldwide that connect research facilities, universities, libraries, government agencies, private companies, and individuals.

6.2 Overview of TCP/IP

TCP and IP were developed by a Department of Defence (DOD) research project to connect a number different networks designed by different vendors into a network of networks (the "Internet"). It was initially successful because it delivered a few basic services that everyone needs (file transfer, electronic mail, remote logon) across a very large number of client and server systems. Several computers in a small department can use TCP/IP (along with other protocols) on a single LAN. The IP component provides routing from the department to the enterprise network, then to regional networks, and finally to the global Internet. On the battlefield a communications network will sustain damage, so the DOD designed TCP/IP to be robust and automatically recover from any node or phone line failure. This design allows the construction of very large networks with less central management. However, because of the automatic recovery, network problems can go undiagnosed and uncorrected for long periods of time.

As with all other communications protocol, TCP/IP is composed of layers:

- IP- is responsible for moving packet of data from node to node. IP forwards each packet based on a four byte destination address (the IP number). The Internet authorities assign ranges of numbers to different organizations. The organizations assign groups of their numbers to departments. IP operates on gateway machines that move data from department to organization to region and then around the world.
- TCP- is responsible for verifying the correct delivery of data from client to server. Data can be lost in the intermediate network. TCP adds support to detect errors or lost data and to trigger retransmission until the data is correctly and completely received.
- Sockets -is a name given to the package of subroutines that provide access to TCP/IP on most systems?

Did You Know?

In 1974, the Transmission Control Protocol (TCP) was specified in detail. TCP replaced NCP and provided enhanced reliable communication services.

6.3 TCP/IP Protocol

The TCP/IP protocol maps to a four-layer conceptual model known as the DARPA model, which was named after the U.S. government agency that initially developed TCP/IP. The four layers of the DARPA model are: Application, Transport, Internet, and Network Interface. Each layer in the DARPA model corresponds to one or more layers of the seven-layer OSI model. Figure 6.1 shows the architecture of the TCP/IP protocol.

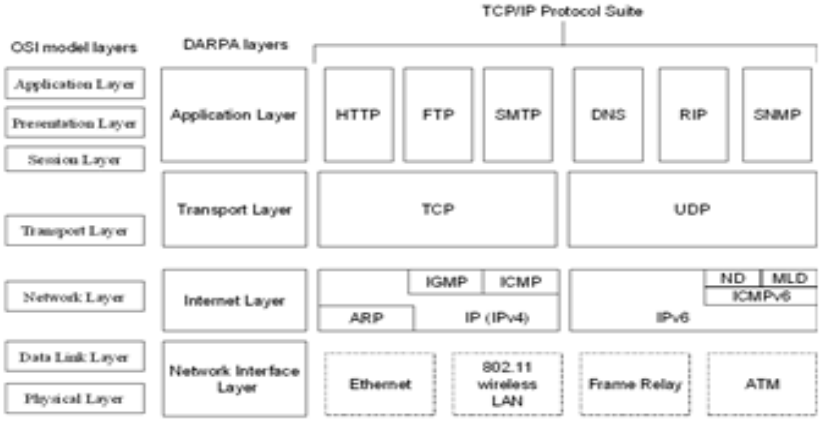


Figure 6.1: The architecture of the TCP/IP protocol.

The TCP/IP protocol suite has two sets of protocols at the Internet layer:

- IPv4, also known as IP, is the Internet layer in common use today on private intranets and the Internet.
- IPv6 is the new Internet layer that will eventually replace the existing IPv4 Internet layer.

TCP/IP is made up of two acronyms, TCP, for Transmission Control Protocol, and IP, for Internet Protocol. TCP handles packet flow between systems and IP handles the routing of packets. However, that is a simplistic answer that we will expound on further.

All modern networks are now designed using a layered approach. Each layer presents a predefined interface to the layer above it. By doing so, a modular design can be developed so as to minimize problems in the development of new applications or in adding new interfaces.

The ISO/OSI protocol with seven layers is the usual reference model. Since TCP/IP was designed before the ISO model was developed it has four layers; however the differences between the two are mostly minor.

OSI Protocol Stack

7. Application -- End user services such as email.
6. Presentation -- Data problems and data compression
5. Session -- Authenication and authorization
4. Transport -- Gaurentee end-to-end delivery of packets
3. Network -- Packet routing
2. Data Link -- Transmit and receive packets
1. Physical -- The cable or physical connection itself.

6.3.1 Network Interface Layer

The Network Interface layer (also called the Network Access layer) sends TCP/IP packets on the network medium and receives TCP/IP packets off the network medium. TCP/IP was designed to be independent of the network access method, frame format, and medium. Therefore, you can use TCP/IP to communicate across differing network types that use LAN technologies—such as Ethernet and 802.11 wireless LAN—and WAN technologies—such as Frame Relay and Asynchronous Transfer Mode (ATM). By being independent of any specific network technology, TCP/IP can be adapted to new technologies.

The Network Interface layer of the DARPA model encompasses the Data Link and Physical layers of the OSI model. The Internet layer of the DARPA model does not take advantage of sequencing and acknowledgment services that might be present in the Data Link layer of the OSI model. The Internet layer assumes an unreliable Network Interface layer and that reliable communications through session establishment and the sequencing and acknowledgment of packets is the responsibility of either the Transport layer or the Application layer.

6.3.2 Internet Layer

The Internet layer responsibilities include addressing, packaging, and routing functions. The Internet layer is analogous to the Network layer of the OSI model.

The core protocols for the IPv4 Internet layer consist of the following:

- The Address Resolution Protocol (ARP) resolves the Internet layer address to a Network Interface layer address such as a hardware address.
- The Internet Protocol (IP) is a routable protocol that addresses, routes, fragments, and reassembles packets.
- The Internet Control Message Protocol (ICMP) reports errors and other information to help you diagnose unsuccessful packet delivery.
- The Internet Group Management Protocol (IGMP) manages IP multicast groups.

The core protocols for the IPv6 Internet layer consist of the following:

- IPv6 is a routable protocol that addresses and routes packets.
- The Internet Control Message Protocol for IPv6 (ICMPv6) reports errors and other information to help you diagnose unsuccessful packet delivery.
- The Neighbor Discovery (ND) protocol manages the interactions between neighboring IPv6 nodes.
- The Multicast Listener Discovery (MLD) protocol manages IPv6 multicast groups.

6.3.3 Transport Layer

The Transport layer (also known as the Host-to-Host Transport layer) provides the Application layer with session and datagram communication services. The Transport layer encompasses the responsibilities of the OSI Transport layer. The core protocols of the Transport layer are TCP and UDP.

TCP provides a one-to-one, connection-oriented, reliable communications service. TCP establishes connections, sequences and acknowledges packets sent, and recovers packets lost during transmission.

In contrast to TCP, UDP provides a one-to-one or one-to-many, connectionless, unreliable communications service. UDP is used when the amount of data to be transferred is small (such as the data that would fit into a single packet), when an application developer does not want the overhead associated with TCP connections, or when the applications or upper-layer protocols provide reliable delivery. TCP and UDP operate over both IPv4 and IPv6 Internet layers.

6.3.4 Application Layer

The Application layer allows applications to access the services of the other layers, and it defines the protocols that applications use to exchange data. The Application layer contains many protocols, and more are always being developed.

The most widely known Application layer protocols help users exchange information:

- The Hypertext Transfer Protocol (HTTP) transfers files that make up pages on the World Wide Web.
- The File Transfer Protocol (FTP) transfers individual files, typically for an interactive user session.
- The Simple Mail Transfer Protocol (SMTP) transfers mail messages and attachments.

Did You Know?

In 1981, the Internet Protocol (IP) (also known as IP version 4 [IPv4]) was specified in detail. IP provides addressing and routing functions for end-to-end delivery.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: What is Protocol?

……..……

……..……

……...

6.4 User Datagram Protocol

UDP provides a connectionless datagram service that offers unreliable, best-effort delivery of data transmitted in messages. This means that neither the arrival of datagrams nor the correct sequencing of delivered packets is guaranteed. UDP does not retransmit lost data. UDP messages consist of a UDP header and a UDP payload, also known as a message. RFC 768 defines UDP. Applications use UDP if they do not require an acknowledgment of receipt of data, and they typically transmit small amounts of data at one time. NetBIOS

name service, NetBIOS datagram service, and SNMP are examples of services and applications that use UDP. Table 6.1 lists and describes the key fields in the UDP header.

Table 6.1: Key fields in the UDP header

Field	Description
Source Port	UDP port of sending application.
Destination Port	UDP port of destination application.
Checksum	A simple mathematical calculation that is used to check for bit-level errors in the UDP message.

UDP Ports

To use UDP, an application must supply the IP address and UDP port number of the source and destination applications. A port provides a location for sending messages. A unique number identifies each port. UDP ports are distinct and separate from TCP ports even though some of them use the same number. Just like TCP ports, UDP port numbers below 1024 are well-known ports that IANA assigns. Table 6.2 lists a few well-known UDP ports.

Table 6.2 Well-known UDP ports

UDP Port Number	Description
53	Domain Name System (DNS) name queries
69	Trivial File Transfer Protocol (TFTP)
137	NetBIOS name service
138	NetBIOS datagram service
161	SNMP

Caution

Be aware while accessing Internet Websites wrong data accessing can get the virus may be cause to harm the system.

6.5 Internet Control Message Protocol (ICMP)

ICMP, defined in RFC 792, reports and helps troubleshoot errors for packets that are undeliverable. For example, if IPv4 cannot deliver a packet to the destination host, ICMP on the router or the destination host sends a Destination Unreachable message to the sending host. Table 6.3 lists and describes the most common ICMP messages.

Table 6.3 Common ICMP messages

ICMP Message	Description
Echo	The Ping tool sends ICMP Echo messages to troubleshoot network problems by checking IPv4 connectivity to a particular node.
Echo Reply	Nodes send Echo Reply messages to respond to ICMP Echo messages.

Redirect	Routers send Redirect messages to inform sending hosts of better routes to destination IPv4 addresses.
Source Quench	Routers send Source Quench messages to inform sending hosts that their IPv4 packets are being dropped due to congestion at the router. The sending hosts then send packets less frequently.
Destination Unreachable	Routers and destination hosts send Destination Unreachable messages to inform sending hosts that their packets cannot be delivered.

ICMP contains a series of defined Destination Unreachable messages. Table 6.4 lists and describes the most common messages.

Table 6.4 Common ICMP destination unreachable messages

Destination Unreachable Message	**Description**
Host Unreachable	Routers send Host Unreachable messages when they cannot find routes to destination IPv4 addresses.
Protocol Unreachable	Destination IPv4 nodes send Protocol Unreachable messages when they cannot match the Protocol field in the IPv4 header with an IPv4 client protocol that is currently in use.
Port Unreachable	IPv4 nodes send Port Unreachable messages when they cannot match the Destination Port field in the UDP header with an application using that UDP port.
Fragmentation Needed and DF Set	IPv4 routers send Fragmentation Needed and DF Set messages when fragmentation must occur but the sending node has set the Do not Fragment (DF) flag in the IPv4 header.

ICMP does not make IPv4 a reliable protocol. ICMP attempts to report errors and provide feedback on specific conditions. ICMP messages are carried as unacknowledged IPv4 packets and are themselves unreliable.

6.6 Address Resolution Protocol (ARP)

When IP sends packets over a shared access, broadcast-based networking technology such as Ethernet or 802.11 wirelesses LAN, the protocol must resolve the media access control (MAC) addresses corresponding to the IPv4 addresses of the nodes to which the packets are being forwarded, also known as the next-hop IPv4 addresses. As RFC 826 defines, ARP uses MAC-level broadcasts to resolve next-hop IPv4 addresses to their corresponding MAC addresses.

Based on the destination IPv4 address and the route determination process, IPv4 determines the next-hop IPv4 address and interface for forwarding the packet. IPv4 then hands the IPv4 packet, the next-hop IPv4 address, and the next-hop interface to ARP.

If the IPv4 address of the packet's next hop is the same as the IPv4 address of the packet's destination, ARP performs a direct delivery to the destination. In a direct delivery, ARP must resolve the IPv4 address of the packet's destination to its MAC address.

ARP Cache

To minimize the number of broadcast ARP Request frames, many TCP/IP protocol implementations incorporate an ARP cache, which is a table of recently resolved IPv4 addresses and their corresponding MAC addresses. ARP checks this cache before sending an ARP Request frame. Each interface has its own ARP cache.

Depending on the vendor implementation, the ARP cache can have the following qualities:

ARP cache entries can be dynamic (based on ARP replies) or static. Static ARP cache entries are permanent, and you add them manually using a TCP/IP tool, such as the Arp tool provided with Windows. Static ARP cache entries prevent nodes from sending ARP requests for commonly used local IPv4 addresses, such as those for routers and servers. The problem with static ARP cache entries is that you must manually update them when network adapter equipment changes. Dynamic ARP cache entries have time-out values associated with them so that they are removed from the cache after a specified period of time. For example, dynamic ARP cache entries for Windows are removed after no more than 10 minutes.

To view the ARP cache on a Windows–based computer, type arp -a at a command prompt. You can also use the Arp tool to add or delete static ARP cache entries.

ARP Process

When sending the initial packet as the sending host or forwarding the packet as a router, IPv4 sends the IPv4 packet, the next-hop IPv4 address, and the next-hop interface to ARP. Whether performing a direct or indirect delivery, ARP performs the following process:

1. Based on the next-hop IPv4 address and interface, ARP checks the appropriate ARP cache for an entry that matches the next-hop IPv4 address. If ARP finds an entry, ARP skips to step 6.
2. If ARP does not find an entry, ARP builds an ARP Request frame. This frame contains the MAC and IPv4 addresses of the interface from which the ARP request is being sent and the IPv4 packet's next-hop IPv4 address. ARP then broadcasts the ARP Request frame from the appropriate interface.
3. All nodes on the subnet receive the broadcasted frame and process the ARP request. If the next-hop address in the ARP request corresponds to the IPv4 address assigned to an interface on the subnet, the receiving node updates its ARP cache with the IPv4 and MAC addresses of the ARP requestor. All other nodes silently discard the ARP request.
4. The receiving node that is assigned the IPv4 packet's next-hop address formulates an ARP reply that contains the requested MAC address and sends the reply directly to the ARP requestor.
5. When the ARP requestor receives the ARP reply, the requestor updates its ARP cache with the address mapping. With the exchange of the ARP request and the ARP reply, both the ARP requestor and ARP responder have each other's address mappings in their ARP caches.
6. The ARP requestor sends the IPv4 packet to the next-hop node by addressing it to the resolved MAC address. Figure 6.2 shows this process.

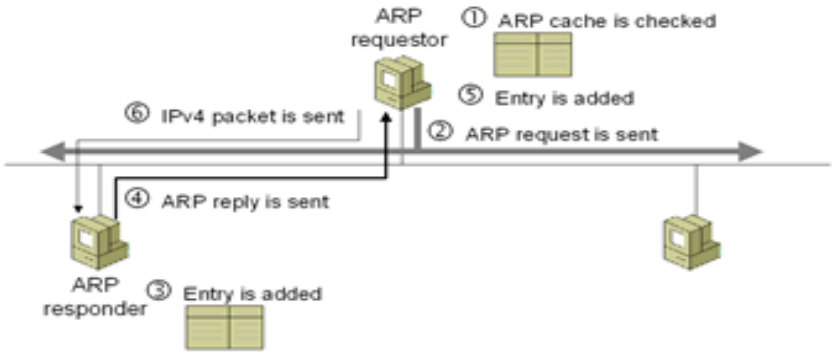


Figure 6.2: The ARP address resolution process.

6.7 Reverse Address Resolution Protocol (RARP)

RARP (Reverse Address Resolution Protocol) allows a physical machine in a local area network to request its IP address from a gateway server's Address Resolution Protocol (ARP) table or cache. A network administrator creates a table in a local area network's gateway router that maps the physical machine (or Media Access Control - MAC address) addresses to corresponding Internet Protocol addresses (IP address). When a new machine is set up, its RARP client program requests from the RARP server on the router to be sent its IP address. Assuming that an entry has been set up in the router table, the RARP server will return the IP address to the machine which can store it for future use.

RARP is available for Ethernet, Fiber Distributed-Data Interface, and Token Ring LANs. ARP (Address Resolution Protocol) performs the opposite function as the RARP: mapping of an IP address to a physical machine address.

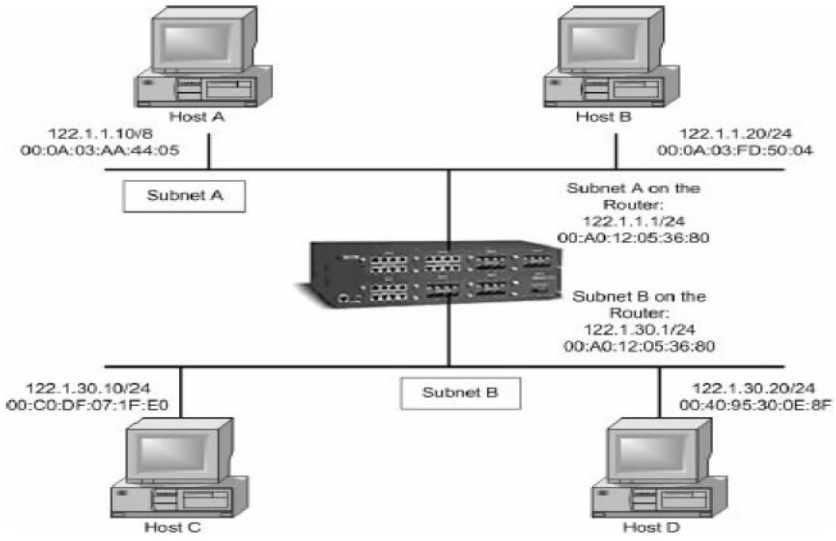


Figure 6.3: The RARP reverse address resolution process.

A reverse address resolution protocol (RARP) is used for diskless computers to determine their IP address using the network. The RARP message format is very similar to the ARP format. When the booting computer sends the broadcast ARP request, it places its own hardware address in both the sending and receiving fields in the encapsulated ARP data packet. The RARP server will fill in the correct sending and receiving IP addresses in its response to the message. This way, the booting computer will know its IP address when it gets the message from the RARP server. RARP request packet is usually generated during the booting sequence of a host. A host must determine its IP address during the booting sequence. The IP address is needed to communicate with other hosts in the network. When a RARP server receives a RARP request packet, it performs the following steps:

1. The MAC address in the request packet is looked up in the configuration file and mapped to the corresponding IP address.
2. If the mapping is not found, the packet is discarded.
3. If the mapping is found, a RARP reply packet is generated with the MAC and IP address. This packet is sent to the host, which originated the RARP request.

When a host receives a RARP reply packet, it gets its IP address from the packet and completes the booting process. This IP address is used for communicating with other hosts, till it is rebooted. The length of a RARP request or a RARP reply packet is 28 bytes.

The ‘operation’ field in the RARP packet is used to differentiate between a RARP request and a RARP reply packet. In an RARP request packet, the source and destination IP address values are undefined. In a RARP reply packet, the source IP address is the IP address of the RARP server responding to the RARP request and the destination IP address is the IP address of the host that sent the said RARP request.
Since a RARP request packet is a broadcast packet, it is received by all the hosts in the network. But only a RARP server processes a RARP request packet, all the other hosts discard the packet. The RARP reply packet is not broadcast, it is sent directly to the host, which sent the RARP request. If more than one RARP server responds to a RARP request, then only the first RARP reply received is used. All other replies are discarded. If a RARP reply is not received within a reasonable amount of time, the host, which sent the RARP request, will not be able to complete its booting sequence. Usually, the host will again retry sending the RARP request after a timeout period.
The BOOTP and DHCP protocols can be used instead of RARP to get the IP address from the MAC address.
Protocol Structure - RARP (Reverse Address Resolution Protocol)RARP and ARP has the same structure:

RARP packet:
Table 6.5: RARP packet

00	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
Hardware type																Protocol type															
Hardware address length								Protocol address length								Opcode															
Source hardware address :::																															
Source protocol address :::																															
Destination hardware address :::																															
Destination protocol address :::																															

- Hardware type - Specifies a hardware interface type for which the sender requires a response.
- Protocol type - Specifies the type of high-level protocol address the sender has supplied.
- Hlen - Hardware address length.
- Plen - Protocol address length.
- Operation - The values are as follows:
- ARP request.
- ARP response.
- RARP request.
- RARP response.
- Dynamic RARP request.
- Dynamic RARP reply.
- Dynamic RARP error.
- InARP request.
- InARP reply.
- Sender hardware address -HLen bytes in length.
- Sender protocol address - PLen bytes in length.
- Target hardware address - HLen bytes in length.
- Target protocol address - PLen bytes in length.

Exercise: Check Your Progress 2
Note: i) Use the space below for your answer.
Ex1: List the advantages and disadvantages of RARP.

……..……

……..……

……...

6.8 Simple Mail Transfer Protocol (SMTP)

SMTP (Simple Mail Transfer Protocol) is a TCP/IP protocol used in sending and receiving e-mail. However, since it is limited in its ability to queue messages at the receiving end, it is usually used with one of two other protocols, POP3 or IMAP that let the user save messages in a server mailbox and download them periodically from the server. In other words, users typically use a program that uses SMTP for sending e-mail and either POP3 or IMAP for receiving e-mail. On Unix-based systems, sendmail is the most widely-used SMTP server for e-mail. A commercial package, Sendmail, includes a POP3 server. Microsoft Exchange includes an SMTP server and can also be set up to include POP3 support.

The SMTP is used as the common mechanism for transporting electronic mail among different hosts within the Department of Defence Internet protocol.

Under SMTP, a user SMTP process opens a TCP connection to a server SMTP process on a remote host and attempts to send mail across the connection. The server SMTP listens for a TCP connection on a well-known port (25), and the user SMTP process initiates a connection on that port. When the TCP connection is successful, the two processes execute a simple request/response dialogue, defined by the SMTP protocol, in which the user process transmits the mail addresses of the originator and the recipient(s) for a message. When the server process accepts these mail addresses, the user process transmits the message. The message must contain a message header and message text formatted in accordance with RFC 822.

SMTP was developed to send e-mail messages across the Internet. In the OSI model, SMPT is an application layer protocol that utilizes TCP as the transport protocol to transmit mail to a destination mail exchanger, in other words, SMTP is used to transmit mail to a mail server. Mail can be transmitted by a client to the mail exchanger server, or from mail exchanger to mail exchanger. Mail sent via SMTP is usually sent from one mail exchanger to another, directly. E-mail was never designed to be instantaneous, but that is often how it appears to us.

SMTP was designed to be a reliable message delivery system. Reliable in this case means that a message handled by SMTP is intended to get to its destination or generate an error message accordingly. This is not the same as a guaranteed delivery service, it just does its best. To guarantee delivery requires all sorts of data exchanges that would add considerable communications overhead that would be pointless for everyday purposes.

6.8.1 Mail Exchangers (MX)

Mail Exchangers are the name given to the applications that support the SMTP protocol. Mail Exchangers such as send mail or Microsoft Exchange should listen for IPdatagrams that arrive on the network interface with a TCP port number of 25 and on. This port is one of the ‘well known ports’ defined in RFC 1700. When a message is received, the mail exchanger should check to see if it is for one of its users, and then move the mail to the user’s mailbox.

To identify the mail exchangers for a domain name, DNS zone files for the domain contain an MX resource record identifying the host name and IP address of the mail exchangers.

Simple mail transfer protocol differs from Post Office Protocol version 3 (POP3). POP3 is used by e-mail client applications such as Microsoft Outlook, Mozilla Thunderbird, Eudora and other e-mail applications to retrieve mail stored in personal mailboxes at the mail server.

6.8.2 E-Mail Clients

E-mail is a client-server protocol that allows the exchange of messages and attachments in various formats. An e-mail client is a software application which seamlessly handles all the technical communications tasks to connect to and find the e-mail at the server, download the e-mail messages, organizes them and presents them to the user in a usable format. An e-mail client also provides the means to compose new messages, reply to and forward received messages, and to organize the messages for later review.
E-mail clients use POP3 or IMAP instead of SMTP. SMTP is used strictly between mail servers.

6.8.3 Mail Service Components

Mail Addresses- Mail addresses identify both the originator and recipient of a mail message. They generally take this form X@Y where X is a string, often a user or account, and Y is a string, often a host. In this, X represents the local part of the mail address and Y represents the global part of the mail address. Mail addresses appear in the to: and from: fields of the message header and in the X-from: and X-to: fields of the envelope.
Envelope- The envelope is a header containing the originator and recipient mail addresses. It is pretended to each mail message by the post office, SNDMSG, or SSMTP, and it is used by all the Cisco IOS for S/390 SMTP routines as the message travels through the system. The envelope consists of the internally-defined (in other words, to the SMTP routines) header fields, X-from: and X-to: that let the SMTP routines convey mail addresses to one another.
Message Header- an RFC 822 message consists of any number of header fields, optionally followed by message text. Typical header fields include: Date:, From:, To:, CC: (carbon copy), and Subject:. The RFC 822 message header refers to the collection of these header fields.
Domain Literal- As defined in RFC 822, a domain literal is a dotted-decimal host number enclosed in square brackets. This is an example of a mail address using a domain literal:
Multi-homed Host- Multi-homed hosts are connected to more than one network and therefore have multiple network addresses.
Host Names- These are mnemonic name strings by which hosts are known on the network. Each host has one official host name and can have optional nicknames. Although nicknames are allowed, their use is discouraged as discussed in RFC 952.
Domain Name Resolver- Cisco IOS for S/390 uses the Domain Name Resolver to map host names into the appropriate network addresses.

6.9 File Transfer Protocol

File Transfer Protocol (FTP) was one of the first efforts to create a standard means of exchanging files over a TCP/IP network, so the FTP has been around since the 1970's. The FTP was designed with as much flexibility as possible, so it could be used over networks other than TCP/IP, as well as being engineered to have the capability with exchanging files with a broad variety of machines.
File Transfer Protocol, or FTP, is a protocol used for transferring files from one computer to another typically from your computer to a Web server. FTP is the preferred method of exchanging files because it is faster than other protocols like HTTP or POP. If you need to exchange large files, you should consider FTP.
File Transfer Protocol (FTP) is a standard Internet protocol for transmitting files between computers on the Internet. Like the Hypertext Transfer Protocol (HTTP), which transfers displayable Web pages and related

files, and the Simple Mail Transfer Protocol (SMTP), which transfers e-mail, FTP is an application protocol that uses the Internet's TCP/IP protocols. FTP is commonly used to transfer Web page files from their creator to the computer that acts as their server for everyone on the Internet. It is also commonly used to download programs and other files to your computer from other servers.

FTP data is sent and received through computer port 21 and under the TCP protocol. The transfer is asynchronous, meaning not at the same time, and therefore faster than other protocols.

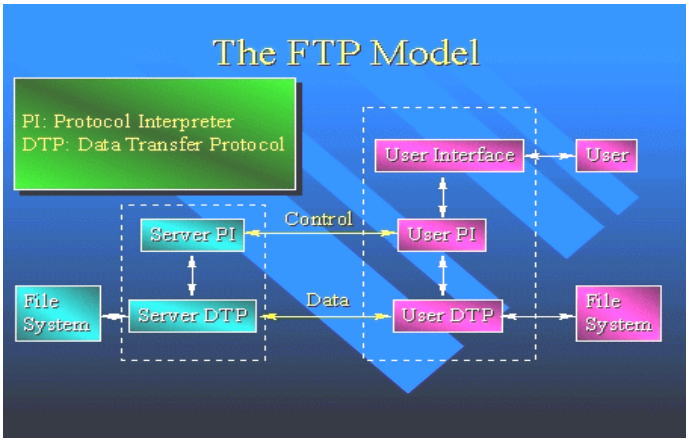


Figure 6.4: FTP model.

Self Assessment Question

1. …….. is the term used to describe features or attributes that apply to both IPv4 and IPv6.

(a) IP (b) TCP/IP (c) SMTP (d) None of these.

2. Hypertext Transfer Protocol (HTTP) was introduced

(a) 1995 (b) 1996 (c) 1997 (d) None of these.

3. File Transfer Protocol (FTP) was introduced……..

(a) 1980 (b) 1982 (c) 1973 (d) None of these.

4. ……contains a series of defined Destination Unreachable messages

(a) SMTP (b) IP (c) FTP (d) ICMP

5. ……was developed to send e-mail messages across the Internet.

(a) SMTP (b) IP (c) FTP (d) ICMP

6. ……..is a header containing the originator and recipient mail addresses.

(a) IP (b) Envelope (c) Both (a) and (b) (d) None of these

6.10 Dynamic Host Configuration Protocol (DHCP)

DHCP is a TCP/IP standard that reduces the complexity and administrative overhead of managing network client IPv4 addresses and other configuration parameters. A properly configured DHCP infrastructure eliminates the configuration problems associated with manually configuring TCP/IP.

A DHCP infrastructure consists of the following elements:

- DHCP servers

Computers that offer dynamic configuration of IPv4 addresses and related configuration parameters to DHCP

clients.

- DHCP clients

Network nodes that support the ability to communicate with a DHCP server to obtain a dynamically leased IPv4 address and related configuration parameters.

- DHCP relay agents

Network nodes, typically routers, that listen for broadcast and unicast DHCP messages and relay them between DHCP servers and DHCP clients. Without DHCP relay agents, you would have to install a DHCP server on each subnet that contains DHCP clients.

Each time a DHCP client starts, it requests IPv4 addressing information from a DHCP server, including:

- IPv4 address
- Subnet mask
- Additional configuration parameters, such as a default gateway address, Domain Name System (DNS) server addresses, a DNS domain name, and Windows Internet Name Service (WINS) server addresses.

6.10.1 Benefits of Using DHCP

To understand why DHCP is beneficial in configuring TCP/IP on client computers, it is useful to contrast the manual configuration method with the DHCP method.

Configuring TCP/IP Manually

Correct operation of TCP/IP on a host computer requires careful configuration of an IPv4 address, subnet mask, and default gateway before the client can communicate with other network nodes. If the configuration is incorrect, the following might happen:

- A user who configures a random IPv4 address, instead of getting a valid IPv4 address from a network administrator, can create network problems that are difficult to troubleshoot.
- An error in typing one of the numbers for the IPv4 address, subnet mask, or default gateway can also lead to problems. These problems can range from trouble communicating using TCP/IP (if the default gateway or subnet mask is wrong) to problems with attempting to use a duplicate IPv4 address.
- If the network node moves to another subnet, the IPv4 address, subnet mask, and default gateway are no longer valid and cannot be used for TCP/IP-based connectivity.

Correct manual configuration is especially important for wireless LANs. For example, a wireless user using a wireless-enabled laptop computer moves from one building to another in a corporate campus. When the user and their laptop change buildings, they might switch to another subnet. Without automated configuration, the user must manually type a different IPv4 address, subnet mask, and default gateway for the new subnet to restore TCP/IP connectivity.

Configuring TCP/IP Using DHCP

Using DHCP to automatically configure IPv4 address configurations means the following:

- Users no longer need to acquire IPv4 address configurations from a network administrator to properly configure TCP/IP.

When a DHCP client is started, it automatically receives an IPv4 address configuration that is correct for the attached subnet from a DHCP server. When the DHCP client moves to another subnet, it automatically obtains a new IPv4 address configuration for that subnet.

- The DHCP server supplies all of the necessary configuration information to all DHCP clients.

As long as the DHCP server has been correctly configured, all DHCP clients of the DHCP server are configured correctly.

How DHCP Works

DHCP uses the following basic process to automatically configure a DHCP client:

1. When the TCP/IP protocol initializes and has DHCP enabled on any of its interfaces, the DHCP client sends a DHCPDiscover message to find the DHCP servers on the network and to obtain a valid IPv4 address configuration.
2. All DHCP servers that receive the DHCPDiscover message and that have a valid IPv4 address configuration for the client send a DHCPOffer message back to the DHCP client.
3. The DHCP client selects an IPv4 address configuration to use from the DHCPOffer messages that it receives and sends a DHCPRequest message to all the DHCP servers, requesting the use of the selected configuration.

The DHCPRequest message identifies the server that sent the offer that the DHCP client selected. The other DHCP servers that receive the DHCPRequest message that sent offers place their offered IPv4 addresses back into the available pool of addresses.

4. The selected DHCP server assigns the IPv4 address configuration to the DHCP client and sends a DHCPAck (acknowledgment) message to the DHCP client.

The DHCP client computer finishes initializing the TCP/IP protocol on the interface. Once complete, the client can use all TCP/IP services and applications for normal network communications and connectivity to other IPv4 hosts. Figure 6.5 shows the basic DHCP process.

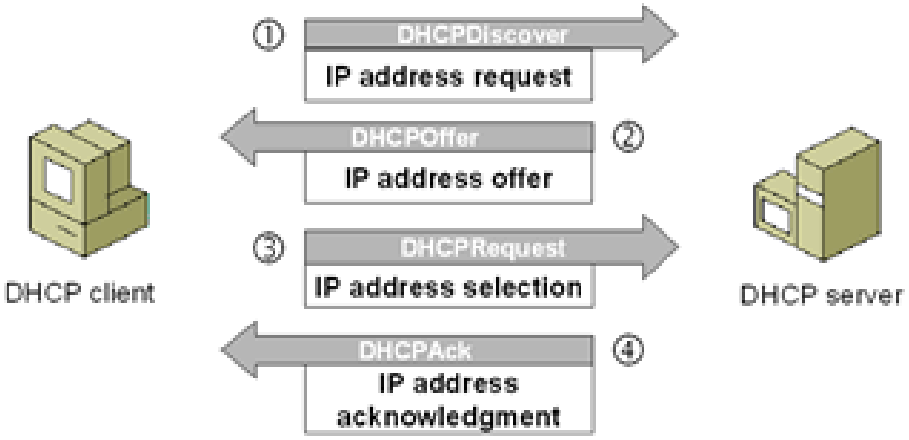


Figure 6.5: The basic DHCP process.

If a computer has multiple network adapters, the DHCP process occurs separately over each network adapter that is configured for automatic TCP/IP addressing until each network adapter in the computer has been allocated a unique IPv4 address configuration.

6.10.2 DHCP Messages and Client States

The DHCP client can go through six states in the DHCP process:

- Initializing
- Selecting
- Requesting
- Bound
- Renewing
- Rebinding

DHCP clients and servers use the following messages to communicate during the DHCP configuration process:

- DHCPDiscover (sent from client to server)
- DHCPOffer (sent from server to client)
- DHCPRequest (sent from client to server)

- DHCPAck (sent from server to client)
- DHCPNak (sent from server to client)
- DHCPDecline (sent from client to server)
- DHCPRelease (sent from client to server)

Figure 6.6 shows DHCP client states and messages, which are discussed in detail in the following sections

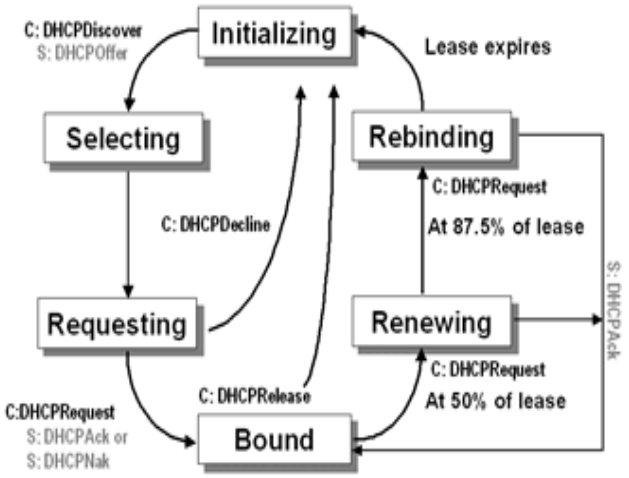


Figure 6.6: DHCP states and messages.

Computers running Windows XP or Windows Server 2003 use an additional DHCP message, the DHCP Inform message, to request and obtain information from a DHCP server for the following purposes:

- To detect authorized DHCP servers in an environment that includes the Active Directory directory service.
- To obtain updated addresses for DNS servers and WINS servers and a DNS domain name when making a remote access connection.
- To obtain additional configuration parameters.

Initializing State

In the Initializing state, the DHCP client is trying to initialize TCP/IP and it does not yet have an IPv4 address configuration. This state occurs the first time the TCP/IP protocol stack is initialized after being configured for automatic configuration and when the DHCP client cannot renew the lease on an IPv4 address configuration.

When the DHCP client is in the Initializing state, its IPv4 address is 0.0.0.0, also known as the unspecified address. The DHCP client's first task is to obtain an IPv4 address configuration by broadcasting a DHCPDiscover message from UDP port 67 to UDP port 68. Because the DHCP client does not yet have an IPv4 address and has not determined the IPv4 addresses of any DHCP servers, the source IPv4 address for the DHCPDiscover broadcast is the unspecified address, 0.0.0.0, and the destination is the limited broadcast address, 255.255.255.255. The DHCPDiscover message contains the DHCP client's media access control (MAC) address and computer name.

If a DHCP server is on the DHCP client's subnet, the server receives the broadcast DHCPDiscover message. If no DHCP server on the DHCP client's subnet (a more typical configuration), a DHCP relay agent on the DHCP client's subnet receives the broadcast DHCPDiscover message and relays it as a unicast DHCPDiscover message from the DHCP relay agent to one or more DHCP servers. Before forwarding the original DHCPDiscover message, the DHCP relay agent makes the following changes:

- Increments the Hops field in the DHCP header of the DHCPDiscover message. The Hops field, which is separate from the Time to Live (TTL) field in the IPv4 header, indicates how many DHCP relay agents

have handled this message. Typically, only one DHCP relay agent is located between any DHCP client and any DHCP server.

- If the value of the Giaddr (Gateway IP Address) field in the DHCP header of the DHCPDiscover message is 0.0.0.0 (as set by the originating DHCP client), changes the value to the IPv4 address of the interface on which the DHCP Discover message was received. The Giaddr field records the IPv4 address of an interface on the subnet of the originating DHCP client. The DHCP server uses the value of the Giaddr field to determine the address range, known as a scope, from which to allocate an IPv4 address to the DHCP client.
- Changes the source IPv4 address of the DHCP Discover message to an IPv4 address assigned to the DHCP relay agent.
- Changes the destination IPv4 address of the DHCP Discover message to the unicast IPv4 address of a DHCP server.

The DHCP relay agent sends the DHCPD is cover message as a unicast IPv4 packet rather than as an IPv4 and MAC-level broadcast. If the DHCP relay agent is configured with multiple DHCP servers, it sends each DHCP server a copy of the DHCP Discover message.
Figure 6.7 shows the sending of the DHCPD is cover message by a DHCP relay agent that is configured with two DHCP servers.

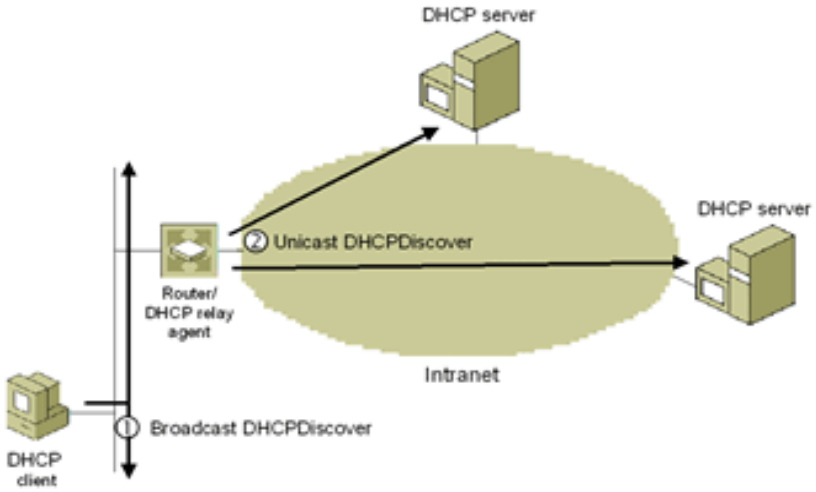


Figure 6.7: Sending the DHCP Discover message.

Selecting State

In the Initializing state, the DHCP client can select from the set of IPv4 address configurations that the DHCP servers offered. All DHCP servers that receive the DHCPDiscover message and that have a valid IPv4 address configuration for the DHCP client respond with a DHCPOffer message from UDP port 68 to UDP port 67. A DHCP server can receive the DHCPDiscover message either as a broadcast (because the DHCP server is on the same subnet as the DHCP client) or as a unicast from a DHCP relay agent.
The DHCP server uses the following process to determine the scope on the DHCP server from which an IPv4 address for the DHCP client is to be selected and included in the DHCPOffer message:

1. If the Giaddr field is set to 0.0.0.0, set the value of the Giaddr field to the IPv4 address of the interface on which the DHCPDiscover message was received.
2. For each scope on the DHCP server, perform a bit-wise logical AND of the value in the Giaddr field with the subnet mask of the scope. If the result matches the subnet prefix of the scope, the DHCP server

allocates an IPv4 address from that scope. To obtain the subnet prefix of the scope, the DHCP server performs a bit-wise logical AND of the subnet mask of the scope with any address in the scope.

If the DHCPDiscover message was received as a broadcast, the DHCP server sends the DHCPOffer message to the DHCP client using the offered IPv4 address as the destination IPv4 address and the client's MAC address as the destination MAC address. If the DHCPDiscover message was received as a unicast, the DHCP server sends the DHCPOffer message to the DHCP relay agent. The DHCP relay agent uses the Giaddr value to determine the interface to use to forward the DHCPOffer message. The DHCP relay agent then forwards the DHCPOffer message to the client using the offered IPv4 address as the destination IPv4 address and the client's MAC address as the destination MAC address.
Figure 6.8 shows the sending of the DHCPOffer message.

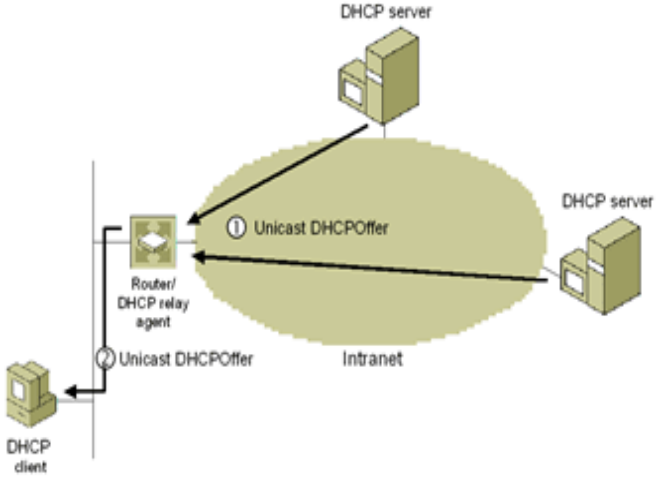


Figure 6.8: Sending of DHCPOffer message.

The DHCPOffer messages contain the DHCP client's MAC address, an offered IPv4 address, appropriate subnet mask, a server identifier (the IPv4 address of the offering DHCP server), the length of the lease, and other configuration parameters. When a DHCP server sends a DHCPOffer message offering an IPv4 address, the DHCP server reserves the IPv4 address so that it will not be offered to another DHCP client.
The DHCP client selects the IPv4 address configuration of the first DHCPOffer message it receives. If the DHCP client does not receive any DHCPOffer messages, it continues to retry sending DHCPDiscover messages for up to one minute. After one minute, a DHCP client based on Windows Server 2003 or Windows XP configures an alternate configuration, either through APIPA or an alternate configuration that has been configured manually.
Requesting State

In the Requesting state, the DHCP client requests a specific IP address configuration by broadcasting a DHCPRequest message. The client must use a broadcast because it does not yet have a confirmed IPv4 address configuration. Just as in the DHCPDiscover message, the DHCP client sends the DHCPRequest message from UDP port 67 to UDP port 68 using the source IPv4 address of 0.0.0.0 and the destination IPv4 address of 255.255.255.255.

If the DHCP client does not have a DHCP server on its subnet, a DHCP relay agent on its subnet receives the broadcast DHCPRequest message and relays it as a unicast DHCPRequest message from the DHCP relay agent to one or more DHCP servers.

The data in the DHCPRequest message varies in the following way, depending on how the requested IPv4 address was obtained:

- If the IPv4 address configuration of the DHCP client was just obtained with a DHCPDiscover/DHCPOffer message exchange, the DHCP client includes the IPv4 address of the server from which it received the offer in the DHCPRequest message. This server identifier causes the specified DHCP server to respond to the request and all other DHCP servers to retract their DHCP offers to the client. These retractions make the IPv4 addresses that the other DHCP servers offered immediately available to the next DHCP client.

- If the IPv4 address configuration of the client was previously known (for example, the computer was restarted and is trying to renew its lease on its previous address), the DHCP client does not include the IPv4 address of the server from which it received the IPv4 address configuration. This condition ensures that when restarting, the DHCP client can renew its IPv4 address configuration from any DHCP server.

Figure 6.9 shows the sending of the DHCPRequest message.

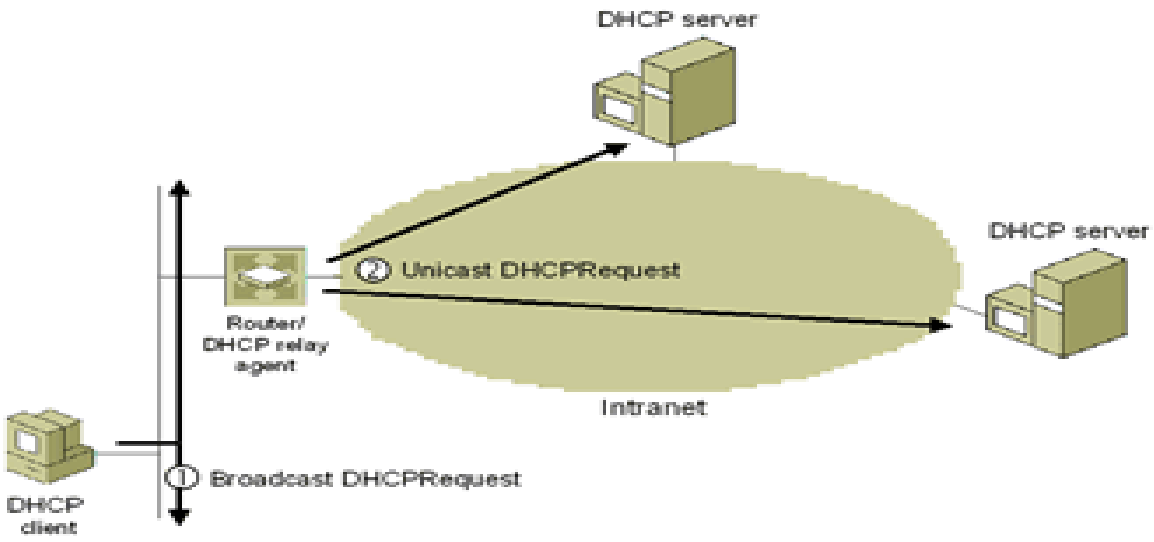


Figure 6.9: Sending the DHCPRequest message.

Bound State

In the Bound state, the DHCP client receives confirmation that DHCP server has allocated and reserved the offered IPv4 address configuration to the DHCP client. The DHCP server that leased the requested IPv4 address responds with either a successful acknowledgment (DHCPAck) or a negative acknowledgment (DHCPNak). The DHCP server sends the DHCPAck message from UDP port 68 to UDP port 67, and the message contains a lease period for the requested IPv4 address configuration as well as any additional configuration parameters.

If the DHCPRequest message was received as a broadcast, the DHCP server sends the DHCPAck message to the DHCP client using the offered IPv4 address as the destination IPv4 address and the client's MAC address as the destination MAC address. If the DHCPRequest was received as a unicast, the DHCP server sends the DHCPAck message to the DHCP relay agent. The DHCP relay agent uses the Giaddr value to determine the interface to use to forward the DHCPAck message. The DHCP relay agent then forwards the DHCPAck message to the DHCP client using the offered IPv4 address as the destination IPv4 address and the DHCP client's MAC address as the destination MAC address.

Figure 6.10 shows the sending of the DHCPAck message.



Figure 6.10: Sending the DHCPAck message.

When the DHCP client receives the DHCPAck message, it enters the Bound state. The DHCP client completes the initialization of TCP/IP, which includes verifying that the IPv4 address is unique on the subnet. If the IPv4 address is unique, the DHCP client computer can use TCP/IP to communicate. If the IPv4 address is not unique, the DHCP client broadcasts a DHCPDecline message and returns to the Initializing state. The DHCP server receives the DHCPDecline message either as a broadcast or as a unicast through a DHCP relay agent. When the DHCP server receives the DHCPDecline message, it marks the offered IPv4 address as unusable.

A DHCP server sends a DHCPNak (DHCP negative acknowledgement) message if:

- The client is trying to lease its previous IPv4 address and the IPv4 address is no longer available.
- The IPv4 address is invalid because the client has been physically moved to a different subnet.

The DHCPNak message is forwarded to the DHCP client's subnet using the same method as the DHCPAck message. When the DHCP client receives a DHCPNak, it returns to the Initializing state.

Renewing State

In the Renewing state, a DHCP client is attempting to renew the lease on its IPv4 address configuration by communicating directly with its DHCP server. By default, DHCP clients first try to renew their lease when 50% of the lease time has expired. To renew its lease, a DHCP client sends a unicast DHCPRequest message to the DHCP server from which it obtained the lease.

The DHCP server automatically renews the lease by responding with a DHCPAck message. This DHCPAck message contains the new lease and additional configuration parameters so that the DHCP client can update its settings. For example, the network administrator might have updated settings on the DHCP server since the lease was acquired or last renewed. When the DHCP client has renewed its lease, it returns to the Bound state. Figure 6.11 shows the DHCP renewing process.

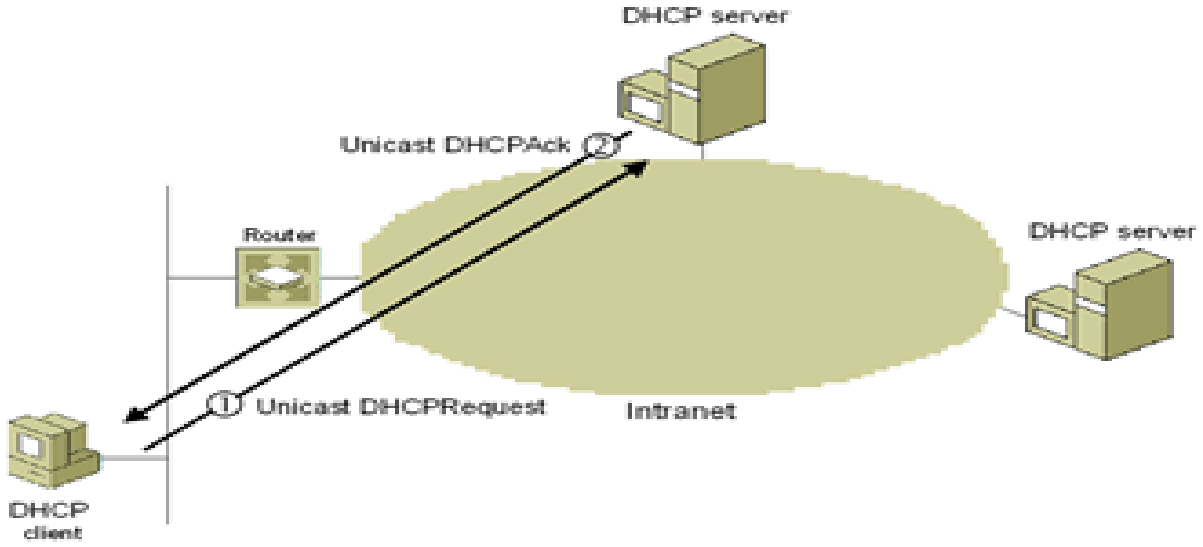


Figure 6.11: DHCP renewing process.

Restarting a Windows-based DHCP Client

The DHCP Client service in Windows XP and Windows Server 2003 uses these states when leasing an IPv4 address configuration from a DHCP server. However, when a Windows-based DHCP client is shut down, by default it does not release the IPv4 address configuration and return to the Initializing state. It does not send a DHCPRelease message and, from the perspective of the DHCP server, the client is still in the Bound state. When the Windows-based DHCP Client service is restarted, it enters the Requesting state and attempts to lease its previously allocated IPv4 address configuration through a broadcasted DHCPRequest message. The DHCPRequest is sent to the limited IPv4 broadcast address 255.255.255.255 and to the MAC-level broadcast address and contains the MAC address and the previously allocated IPv4 address of the DHCP client.

Figure 6.12 shows the DHCP states for a Windows-based DHCP client.

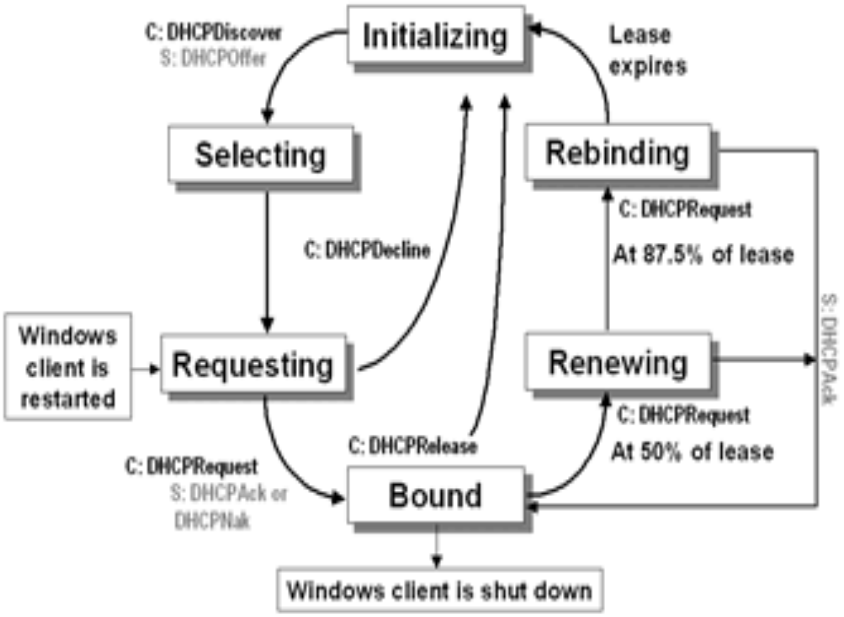


Figure 6.12: DHCP states for a Windows-based DHCP client.

When a DHCP relay agent on the subnet receives the DHCPRequest message, it makes the following changes to the message before forwarding:

- Increments the Hops field in the DHCP header.
- Records the IPv4 address of the interface on which the DHCPRequest message was received in the Giaddr field.
- Changes the source IPv4 address of the DHCPRequest message to an IPv4 address assigned to the DHCP relay agent.
- Changes the destination IPv4 address to the IPv4 address of a DHCP server.

When the DHCP server receives the DHCPRequest message, it compares the subnet prefix of client's previously allocated IPv4 address to the subnet prefix of the IPv4 address stored in the Giaddr field and does the following:

- If the two subnet prefixes are the same and the IPv4 address can be reallocated to the DHCP client, the DHCP server sends a DHCPAck to the DHCP relay agent. When the DHCP relay agent receives the DHCPAck, the agent re-addresses the message to the client's current IPv4 address and MAC address.
- If the two subnet prefixes are the same and the IPv4 address cannot be reallocated to the DHCP client, the DHCP server sends a DHCPNak to the DHCP relay agent. When the DHCP relay agent receives the DHCPNak, it sends the message to the client's current IPv4 address and MAC address. At this point, the DHCP client goes into the Initializing state.
- If the two subnet prefixes are not the same, the DHCP client has moved to a different subnet, and the DHCP server sends a DHCPNak to the DHCP relay agent. When the DHCP relay agent receives the DHCPNak, the agent sends the message to the client's current IPv4 address and MAC address. At this point, the DHCP client goes into the Initializing state.

6.10.3 DHCP Scopes

A DHCP scope is the consecutive range of possible IPv4 unicast addresses that DHCP clients on a subnet can use. Scopes typically define a single physical subnet on your network to which DHCP services are offered. Scopes are the primary way for the DHCP server to manage distribution and assignment of IPv4 addresses and any related configuration parameters to DHCP clients on the network.

Caution

Be aware about the IP address while sending the file, any mistake can cause the loss of data.

6.11 Remote Login (rlogin)

Remote Login (rlogin) is a Unix command that allows an authorized user to login to other UNIX machines (hosts) on a network and to interact as if the user were physically at the host computer. Once logged in to the host, the user can do anything that the host has given permission for, such as read, edit, or delete files.

Each remote machine may have a file named /etc/hosts.equiv containing a list of trusted hostnames with which it shares usernames. Users with the same username on both the local and remote machine may rlogin from the machines listed in the remote machine's /etc/hosts.equiv file without supplying a password. Individual users may set up a similar private equivalence list with the file .rhosts in their home directories. Each line in this file contains two names: a host- name and a username separated by a space. An entry in a remote user's .rhosts file permits the user named username who is logged into hostname to log in to the remote machine as the remote user without supplying a password. If the name of the local host is not found in the /etc/hosts.equiv file on the remote machine, and the local username and hostname are not found in the remote user's .rhosts file, then the remote machine will prompt for a password. Hostnames listed in /etc/hosts.equiv and .rhosts files must be the

official hostnames listed in the hosts database; nicknames may not be used in either of these files. For security reasons, the .rhosts file must be owned by either the remote user or by root.

6.11.1 Rlogin Operation

The protocol requires *rlogin* server software to be running on the host that is going to allow remote access; it is usually called *rlogind* (for *rlogin daemon*, the latter word being the standard UNIX term for a background server process). The server listens for incoming connection requests on TCP port 513. A user who wants to remotely log in to the server runs the *rlogin* command on his or her local host, and specifies the name of the server. The client makes a TCP connection to the server, and then sends to the server a string containing the following information:

- The login name of the user on the client machine.
- The login name that the user wants to use on the server (which is often the same as the user's login name on the client, but not always.)
- Control information such as the type and speed of the terminal.

Exercise: Check Your Progress 3

Note: i) Use the space below for your answer.

Ex1: Create a structure for remotely login to a network.

……..……

……..……

……...

6.12 Network File System (NFS)

Network File System (NFS), originally developed by Sun Microsystems in 1984 and then extended by IETF, allows file sharing over network among different types of systems. In other words, NFS was designed for remote file access and sharing over network with various types of machines, operating systems, network architecture and transport protocols.

A Network File System (NFS) allows remote hosts to mount file systems over a network and interact with those file systems as though they are mounted locally. This enables system administrators to consolidate resources onto centralized servers on the network.

NFS uses a client/server architecture and consists of a client program and a server program. The server program makes file systems available for access by other machines via a process called *exporting*. NFS clients access shared file systems*mounting* them from an NFS server machine. NFS mount protocol is used to communicate between the server and the client for the file access and sharing. NFS mount protocol also allows the server to grant remote access privileges to a restricted set of clients via export control.

NFS Version 2, the first widely implemented version of NFS, originally operated entirely over UDP and was meant to keep the protocol stateless. Several vendors had extended NFSv2 to support TCP as transport. NFS Version 3 introduced support for using TCP as transport. Using TCP as transport made using NFS over a WAN more feasible. Inheritated the good features of the previous versions, the current NFS Version 4 features the following improvements:

Improved access and performance on the Internet. The protocol is designed to transit firewalls easily, perform well where latency is high and bandwidth is low, and scale to very large numbers of clients per server.

Strong security with negotiation built into the protocol. The protocol builds on the work of the ONCRPC working group in supporting the Remote Procedure Call (RPC) RPCSEC_GSS protocol. Additionally, the

NFS version 4 provides a mechanism to allow clients and servers to negotiate security and require clients and servers to support a minimal set of security schemes.
Designed for protocol extensions. The protocol is designed to accept standard extensions that do not compromise backward compatibility.
NFS is strongly associated with UNIX systems, though it can be used on any platform such as Macintosh and Microsoft Windows operating systems. The Server Message Block (SMB) and Common Internet File System (CIFS) are a similar protocol that has equivalent implementation of a network file system under Microsoft Windows.

How It Works
Currently, there are three versions of NFS. NFS version 2 (NFSv2) is older and is widely supported. NFS version 3 (NFSv3) has more features, including 64bit file handles, Safe Async writes and more robust error handling. NFS version 4 (NFSv4) works through firewalls and on the Internet, no longer requires portmapper, supports ACLs, and utilizes stateful operations. Red Hat Enterprise Linux supports NFSv2, NFSv3, and NFSv4 clients, and when mounting a file system via NFS, Red Hat Enterprise Linux uses NFSv3 by default, if the server supports it.

All versions of NFS can use Transmission Control Protocol (TCP) running over an IP network, with NFSv4 requiring it. NFSv2 and NFSv3 can use the User Datagram Protocol (UDP) running over an IP network to provide a stateless network connection between the client and server.

6.13 Summary

- Transmission Control Protocol/Internet Protocol (TCP/IP) is an industry standard protocols that is designed for large networks consisting of network segments that are connected by routers.
- RARP is used by many diskless systems to obtain their IP address when bootstrapped. The RARP packet format is nearly identical to the ARP packet. An RARP request is broadcast, identifying the sender's hardware address, asking for anyone to respond with the sender's IP address. The reply is normally unicast.
- RARP is available for Ethernet, Fiber Distributed-Data Interface, and Token Ring LANs. ARP (Address Resolution Protocol) performs the opposite function as the RARP: mapping of an IP address to a physical machine address.
- SMTP (Simple Mail Transfer Protocol) is a TCP/IP protocol used in sending and receiving e-mail.
- File Transfer Protocol (FTP) is a standard Internet protocol for transmitting files between computers on the Internet.
- DHCP is a TCP/IP standard that reduces the complexity and administrative overhead of managing network client IPv4 addresses and other configuration parameters.

6.14 Keywords

FTP: It is a protocol used for transferring files from one computer to another typically from your computer to a Web server.
RARP: It is used for diskless computers to determine their IP address using the network.
SMTP*:* It is used as the common mechanism for transporting electronic mail among different hosts within the Department of Defense Internet protocol.
Sockets: It is a name given to the package of subroutines that provide access to TCP/IP on most systems.

TCP/IP: It is the protocol that is used on the Internet, which is the collection of thousands of networks worldwide that connect research facilities, universities, libraries, government agencies, private companies, and individuals.

UDP: It provides a connectionless datagram service that offers unreliable, best-effort delivery of data transmitted in messages.

6.15 Review Questions

1. Explain the overview of TCP/IP
2. What is the TCP/IP protocol? Explain briefly.
3. Define User Datagram Protocol
4. What is the main purpose internet control message protocol?
5. Write the short notes address resolution protocol.
6. Describes the reverse address resolution protocol.
7. Explain the simple mail transfer protocol.
8. Differentiate between mail exchangers and e-mail clients
9. What is the concept file transfer protocol?
10. Explain the dynamic host configuration protocol.
11. Understand the remote login (rlogin).
12. Discuss network file system.

Answers for Self Assessment Questions

1. (a) 2. (b) 3. (c) 4. (d) 5(a) 6. (b)

6.16 Further Studies

1. The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference by Charles M. Kozierok
2. Computer Networks: A Systems Approach by Larry L. Peterson, Bruce S. Davie

7

IP Addressing and Subnet

STRUCTURE

7.0 Objectives

7.1 Introduction

7.2 Introduction to IP

7.3 Domain Name System (DNS)

7.4 Uniform Resource Locator (URL)

7.5 Electronic Mail

7.6 Electronic Mail

7.7 Subnet and Subnet Masks

7.8 Summary

7.9 Keywords

7.10 Review Questions

7.11 Further Studies

7.0 Objectives

After studying this chapter, you will be able to:

- Discuss about the IP.
- Explain the domain name system
- Describes the uniform resource locator
- Define electronic mail

7.1 Introduction

The following gives an introduction to IP addresses and subnetting on local area networks. If you want to find out about the advantages of using private network IP addresses on your local area network, or what subnetting can do for you, the explanation is here. You can also find the recipe for how you calculate a subnet mask, a network address and broadcast address. An IP address is an address used to uniquely identify a device on an IP network. The address is made up of 32 binary bits which can be divisible into a network portion and host portion with the help of a subnet mask. The 32 binary bits are broken into four octets (1 octet = 8 bits). Each octet is converted to decimal and separated by a period (dot). For this reason, an IP address is said to be expressed in dotted decimal format (for example, 172.16.81.100). The value in each octet ranges from 0 to 255 decimal, or 00000000 − 11111111 binary.

7.2 Introduction to IP

Networks provide communication between computing devices. To communicate properly, all computers (hosts) on a network need to use the same communication protocols. An Internet Protocol network is a network of computer using Internet Protocol for their communication protocol.

All computers within an IP network must have an IP address that uniquely identifies that individual host. An Internet Protocol-based network (an IP Network) is a group of *hosts* that share a common physical connection and that use Internet Protocol for network layer communication. The IP addresses in an IP network are contiguous, that is, one address follows right after the other with no gaps.

Even without subnetting, hosts on the Internet or any other IP network are assigned a network number. Network numbering allows a group of hosts (peers) to communicate efficiently with each other. Hosts on the same network may be computers located in the same facility or all computers used by a workgroup, for example. Multi-homed hosts, that contain multiple network adapters, can belong to multiple networks, but each adapter is assigned exactly one network number.

Network numbers look very much like IP addresses, but the two should not be confused. Consider for example the host IP address 10.0.0.1, an address commonly used on private networks. Because it is a Class A address, with no subnetting employed, its leftmost byte (eight bits) by default refers to the network address and all other bits remain set at zero. Thus, 10.0.0.0 is the network number corresponding to IP address 10.0.0.1.

The portion of the IP address that does not refer to the network refers instead to the host address- literally, the unique identifier of the host on that network. In the above example, the host address becomes '0.0.0.1' or simply '1'. Also note that a network address becomes a reserved address that should not be assigned to any actual host. Configuring a live host at 10.0.0.0 in the example above could impact communications for all hosts on that network.

The Table 7.1 illustrates the default numbering scheme for Class A, B, and C networks.

Table 7.1: Class A, B, and C networks

Class	Host address range	Network address	Default mask
A	0.0.0.0 – 127.255.255.255	x.0.0.0	255.0.0.0
B	128.0.0.0 – 191.255.255.255	x.x.0.0	255.255.0.0
C	192.0.0.0 – 223.255.255.255	x.x.x.0	255.255.255.0

In general, a network address uses the leftmost byte of its hosts addressing if the hosts fall within the Class A range, the leftmost two bytes for hosts in Class B, and the leftmost three bytes for hosts in Class C. This algorithm is applied in practice through the use of a network mask. The Table 7.1 shows the decimal representation of the default network masks that is commonly used by network operating systems. Note that the decimal value '255' corresponds to one byte that has all bits set to one (11111111).

7.2.1 Address Types in an IP Network

Within a given range of IP addresses used in every IP network are special addresses reserved for

- Host Addresses
- Network Addresses
- Broadcast Addresses

Host address

A host's IP address is the address of a specific host on an IP network. All hosts on a network must have a unique IP address. This IP address is usually not the first or the last IP address in the range of network IP addresses as the first IP address and last IP address in the range of IP addresses are reserved for special functions. The host addresses are all the addresses in the IP network range of IP addresses except the first and last IP addresses. Host IP addresses allow network hostst to establish one-to-one direct communication. This one-to-one communication is referred to as *unicast* communication.

All host IP addresses can be split into two parts: a *network* part and a *host* part. The *network* part of the IP addresses identifies the IP Network the host is a member of. The *host* part uniquely identifies an individual host.

Network Address

The network address is the first IP address in the range of IP addresses. To be more precise, the network address is the address in which all binary bits in the host portion of the IP address are set to zero. The purpose of the Network Address is to allow hosts that provide special network services to communicate. In practice, the network address is rarely used for communication.

Broadcast Address

The broadcast IP address is the last IP address in the range of IP addresses. To be more precise, the broadcast address is the IP address in which all binary bits in the host portion of the IP address are set to one. The broadcast address is reserved and allows a single host to make an announcement to all hosts on the network. This is called *broadcast* communication and the last address in a network is used for broadcasting to all hosts because it is the address where the host portion is all ones. This special address also sometimes called the *all hosts* address. Some vendors allow you to set an address other than the last address as the *broadcast* address.

Note that each network has its own network and broadcast addresses. Network addresses do not necessarily have a zero in the decimal representation of the host portion of their address.

Example:

Table 7.2 Address Types in an IP Network

	network portion								*host portion*	
Network Address (decimal)	192	**.**	168	**.**	1	**.**	**0**			All zeroes in the host portion
Network Address (binary)	11000000	**.**	10011000	**.**	0000001	**.**	**000000**			
First **host** address (decimal)	192	**.**	168	**.**	1	**.**	**1**			00000001 00000010 00000011 ... 11111100 11111101 11111110
First **host** address (binary)	11000000	**.**	10011000	**.**	00000001	**.**	**00000001**			
...and all the addresses in between.										
Last **host** address (decimal)	192	**.**	168	**.**	1	**.**	**254**			
Last **host** address (binary)	11000000	**.**	10011000	**.**	00000001	**.**	**11111110**			
Broadcast Address (decimal)	192	**.**	168	**.**	1	**.**	**255**			All ones in the host portion
Broadcast Address (binary)	11000000	**.**	10011000	**.**	00000001	**.**	**11111111**			

Example:

In example, we borrow 2 bits from what would normally be the host portion and use it as bits that indicate the network portion. This makes a smaller network of just 64 addresses, of which 62 are usable for hosts. Remember, the first address in a range of IP addresses is reserved for the network address. The last address is reserved for the broadcast address.

Table 7.3 Address Types in an IP Network

	network portion								*host portion*	
Network Address (decimal)	192	.	168	.	1	.	64	All **zeroes** in the host portion
Network Address (binary)	11000000	.	10011000	.	0000001	.	01**000000**	
First host address (decimal)	192	.	168	.	1	.	1	
First host address (binary)	11000000	.	10011000	.	00000001	.	01**000001**	
...and all the addresses in between.								
Last host address (decimal)	192	.	168	.	1	.	254	
Last host address (binary)	11000000	.	10011000	.	00000001	.	01**111110**	
Broadcast Address (decimal)	192	.	168	.	1	.	127	All **ones** in the host portion
Broadcast Address (binary)	11000000	.	10011000	.	00000001	.	01**111111**	

Caution

The same IP address in a network can cause of a problem during Internet accessing.

7.2.2 How It Works

Your computer uses a mask in its network connection that allows it to determine whether the computer it wants to talk to over the local network is actually *on* the local network or on a network outside the local network. If it is a computer that is local, it will use a local protocol to find the other computer's hardware address. If the other computer is not local (it is outside the local network), then it will send the data to the local gateway (usually called the default gateway). Your computer will try to determine the hardware address of the local default gateway and will use the hardware address in the Ethernet header (or FDDI header or Token Ring header etc.). The Ethernet frame will contain a payload which will be filled with IP data. The IP address of the remote computer will be in this payload as part of the IP header. Since the IP address in the IP datagram is the remote computer, but the Ethernet frame is addressed to the local default gateway, the default gateway will accept the Ethernet frame, read the IP data, verify the IP address is indeed outside the local network and then forward the IP datagram over whatever other network connections the computer has that will allow it to reach all the other outside networks.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: What is IP and how it works?

……..……

……..……

……...

7.3 Domain Name System (DNS)

The DNS is a widely used naming service on the Internet and other TCP/IP networks. The network protocols, data and file formats, and other aspects of the DNS are Internet Standards, specified in a number of RFC documents. The DNS has a distributed, client-server architecture. There are reference implementations for the server and client, but these are not part of the standard. There are a number of additional implementations available for many platforms.

7.3.1 Naming Services

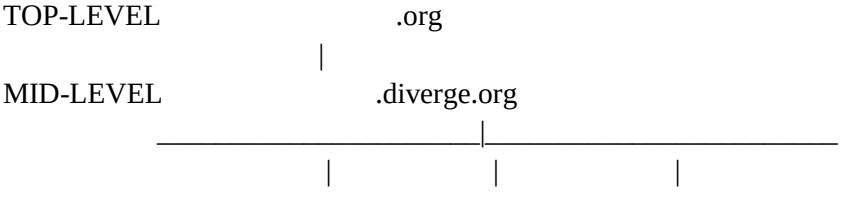
Naming services are used to provide a mapping between textual names and configuration data of some form. A name server maintains this mapping, and clients request the nameserver to resolve a name into its attached data.

The reader should have a good understanding of basic hosts to IP address mapping and IP address class specifications, "Name Service Concepts".

In the case of the DNS, the configuration data bound to a name is in the form of standard Resource Records (RR's). These textual names conform to certain structural conventions.

7.3.2 DNS namespace

The DNS presents a hierarchical name space, much like a UNIX file system, pictured as an inverted tree with the root at the top.



The system can also be logically divided even further if one wishes at different points. The example shown above shows three nodes on the diverge.org domain, but we could even divide diverge.org into sub domains such as "strider.net1.diverge.org", "samwise.net2.diverge.org" and "wormtongue.net2.diverge.org"; in this case, 2 nodes reside in "net2.diverge.org" and one in "net1.diverge.org".

There are directories of names, some of which may be sub-directories of further names. These directories are sometimes called zones. There is provision for symbolic links, redirecting requests for information on one name to the records bound to another name. Each name recognized by the DNS is called a Domain Name, whether it represents information about a specific host, or a directory of subordinate Domain Names (or both, or something else).

Unlike most file system naming schemes, however, Domain Names are written with the innermost name on the left and progressively higher-level domains to the right, all the way up to the root directory if necessary. The separator used when writing Domain Names is a period, ".".

7.3.3 Delegation

Using NS records, authority for portions of the DNS namespace below a certain point in the tree can be delegated, and further sub-parts below that delegated again. It is at this point that the distinction between a domain and a zone becomes important. Any name in the DNS is called a domain, and the term applies to that name and to any subordinate names below that one in the tree. The boundaries of a zone are narrower, and are defined by delegations. A zone starts with a delegation (or at the root), and encompasses all names in the domain below that point, excluding names below any subsequent delegations.

7.3.4 Delegation to Multiple Servers

For redundancy, it is common (and often administratively required) that there be more than one name server providing information on a zone. It is also common that at least one of these servers be located at some distance (in terms of network topology) from the others, so that knowledge of that zone does not become unavailable in case of connectivity failure. Each nameserver will be listed in an NS record bound to the name of the zone, stored in the parent zone on the server responsible for the parent domain. In this way, those searching the name hierarchy from the top down can contact any one of the servers to continue narrowing their search. This is occasionally called walking the tree.

There are a number of name servers on the Internet which are called root nameservers. These servers provide information on the very top levels of the domain namespace tree. These servers are special in that their addresses must be pre-configured into nameservers as a place to start finding other servers. Isolated networks that cannot access these servers may need to provide their own root name servers.

7.3.5 Secondary, Caching, and the SOA record

In order to maintain consistency between these servers, one is usually configured as the primary server, and all administrative changes are made on this server. The other servers are configured as secondaries, and transfer the contents of the zone from the primary. This operational model is not required, and if external considerations require it, multiple primaries can be used instead, but consistency must then be maintained by other means. DNS servers that store Resource Records for a zone, whether they be primary or secondary servers, are said to be authoritative for the zone. A DNS server can be authoritative for several zones.

7.3.6 Name Resolution

DNS clients are configured with the addresses of DNS servers. Usually, these are servers which are authoritative for the domain of which they are a member. All requests for name resolution start with a request to one of these local servers. DNS queries can be of two forms:

A recursive query asks the nameserver to resolve a name completely, and return the result. If the request cannot be satisfied directly, the nameserver looks in its configuration and caches for a server higher up the domain tree which may have more information. In the worst case, this will be a list of pre-configured servers for the root domain. These addresses are returned in a response called a referral. The local nameserver must then send its request to one of these servers.

7.3.7 Reverse Resolution

The DNS provides resolution from a textual name to a resource record, such as an A record with an IP address. It does not provide a means, other than exhaustive search, to match in the opposite direction; there is no mechanism to ask which name is bound to a particular RR. For many RR types, this is of no real consequence; however it is often useful to identify by name the host which owns a particular IP address. Rather than complicate the design and implementation of the DNS database engine by providing matching functions in both directions, the DNS utilises the existing mechanisms and creates a special namespace, populated with PTR records, for IP address to name resolution. Resolving in this manner is often called reverse resolution, despite the inaccurate implications of the term.

The manner in which this is achieved is as follows:

- A normal domain name is reserved and defined to be for the purpose of mapping IP addresses. The domain name used is “in-addr.arpa.” which shows the historical origins of the Internet in the US Government’s Defence Advanced Research Projects Agency’s funding program.
- This domain is then subdivided and delegated according to the structure of IP addresses. IP addresses are often written in decimal dotted quad notation, where each octet of the 4-octet long address is written in decimal, separated by dots. IP address ranges are usually delegated with more and more of the left-most

parts of the address in common as the delegation gets smaller. Thus, to allow delegation of the reverse lookup domain to be done easily, this is turned around when used with the hierarchical DNS namespace, which places higher level domains on the right of the name.

- Each byte of the IP address is written, as an ASCII text representation of the number expressed in decimal, with the octets in reverse order, separated by dots and appended with the in-addr.arpa. domain name. For example, to determine the hostname of a network device with IP address 11.22.33.44, this algorithm would produce the string "44.33.22.11.in-addr.arpa." which is a legal, structured Domain Name. A normal nameservice query would then be sent to the nameserver asking for a PTR record bound to the generated name.
- The PTR record, if found, will contain the FQDN of a host.

7.4 Uniform Resource Locator (URL)

URL is one type of Uniform Resource Identifier (URI); the generic term for all types of names and addresses that refer to objects on the World Wide Web. The term "Web address" is a synonym for a URL that uses the HTTP / HTTPS protocol.

A URL (Uniform Resource Locator, previously Universal Resource Locator) - usually pronounced by sounding out each letter but, in some quarters, pronounced "Earl" - is the unique address for a file that is accessible on the Internet. A common way to get to a Web site is to enter the URL of its home page file in your Web browser's address line. However, any file within that Web site can also be specified with a URL. Such a file might be any Web (HTML) page other than the home page, an image file, or a program.

The URL contains the name of the protocol to be used to access the file resource, a domain name that identifies a specific computer on the Internet, and a pathname, a hierarchical description that specifies the location of a file in that computer.

7.4.1 Types of URLs

There are many types of URLs. At last count there were nearly a hundred different types of URLs. There is supposed to be a way to specify a unique type of URL for every type of content available on the Internet. Fortunately, you will normally ever only need to worry about a few of them. Here is a brief overview of the common ones.

We will talk about:

- http:
- javascript:
- ftp:
- file:
- mailto:
- telnet:

7.5 Electronic Mail

Electronic mail (email) is the term given to an electronic message, usually a form of simple text message that a user types at a computer system and is transmitted over some form of computer network to another user.

Email once consisted of a number of proprietary email systems. Originally these email systems could only send and receive email in an office where every person was equipped with the same software. With the expansion of the Internet, some manufacturers of these proprietary email systems introduced the capability of connecting to the Internet for the transfer of messages outside of the local network. This can take the form of a software interface that converts the local messages into a recognized standard form suitable for transfer over

the Internet. These systems are more common in establishments that have used email for longer than most, and are renowned for minor problems with access to global Internet email, (e.g. problems with sending or receiving attachments) however such problems are slowly disappearing.

7.5.1 Benefits of Email

Email has become one of the driving forces behind connecting businesses to the Internet. It offers fast, economical transfer of messages anywhere in the world. As local telephone calls are free in most parts of the US, messages destined to long-distance destinations become effectively free to send. Outside of the US, local calls tend to be chargeable; therefore the email system can reduce the telephone bill considerably. The substantial cost-cutting associated with these facts have encouraged many businesses to invest in an implementation of email services.

Email has considerable benefits over traditional paper based memo's and postal systems:

Messages can be sent at any time across the world as easily as across the office, to a group of people or a single recipient, without the sender leaving their desk. Messages can be logged, ensuring some form of record is held, and messages are stored when the recipient is away from their desk. The recipient can collect their mail when they want, from wherever they are. Mobile users can collect their mail whilst out visiting customers, or at other locations. The person you are sending the message to gets it directly, without passing through any third party. Environmentally friendly email messages require no paper or resources other than storage space on a computer disk drive.

7.5.2 E-mail Client

An email client is an application that is used to read, write and send email. In simple terms it is the user interface to the email system.

The client usually consists of a combination of a simple text editor, address book, filing cabinet and communications module.

The text editor allows for the creation of the message itself, and usually includes simple spell checking and formatting facilities. The ability to allow files or documents to be attached to the message is also available. For example a diagram or schematic could be attached to an email message, offering the recipient the chance to see a project's progress, and comment on it with a reply. The address allows the users to store commonly used email addresses in an easy to get at format, reducing the chance of addressing errors. The filing cabinet allows for the storage of email messages, both sent and received, and usually gives some form of search function, allowing the easy retrieval of a desired message. The final, but most important, section of the email client is the element that deals with the actual communication of email messages to and from an email server. How this actually occurs will be described later in this FAQ. Figure 7.1 shows simple email between two offices.

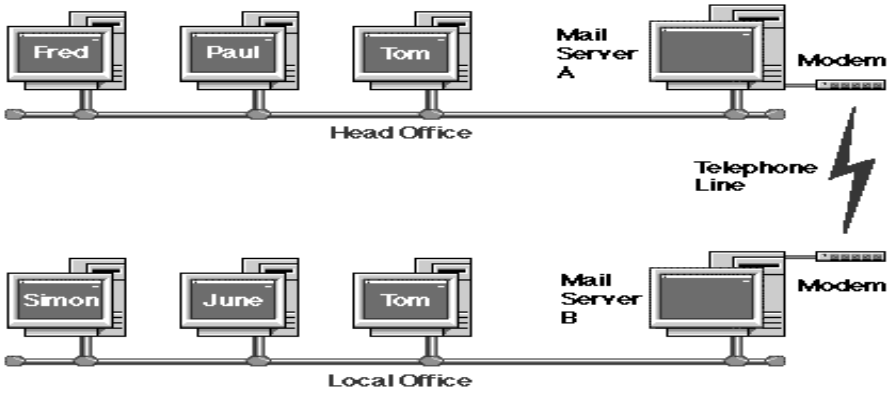


Figure 7.1: Simple email between two offices.

7.5.3 E-mail Address

Email addresses are made up of at least two parts. One part is the name of a mail domain that will ultimately translate to either the recipient's host or some host that accepts mail on behalf of the recipient. The other part is some form of unique user identification that may be the login name of that user, the real name of that user in "Firstname. Lastname" format, or an arbitrary alias that will be translated into a user or list of users. Other mail addressing schemes, like X.400, use a more general set of "attributes" that are used to look up the recipient's host in an X.500 directory server.

An example email address looks like this:



This address is made up of two parts:

Email account This is a particular user's email account name that, in this case, the vicomsoft.com mail server can deal with.

Domain name This is a name that a company has registered so that they can use it on the Internet. Other examples are: apple.com, or microsoft.com.

If a person or company has not registered their own domain name then they may be using their Internet Service Provider's (ISP) domain name, for example: netcom.com, or aol.com. This is usually a less expensive option than registering your own domain name, but does mean that you have to use your ISP's domain name all the time.

In the above example "vicomsoft.com" is the domain name that has been registered so that Vicomsoft can use it on the Internet.

Caution

Save and scan any attachments before opening them, they are also a common source of viruses and can harm the system.

Self Assessment Question

1. The address is…….made up of binary bits which can be divisible into a network portion and host portion with the help of a subnet mask.

(a) 32 (b) 22

(c) 42 (d) None of these

2. A host's IP address is the address of a specific host on an IP network. All hosts on a network must have a ……..IP address.

(a) Two (b) Unique

(c) Three (d) None of these

3. The…….is reserved and allows a single host to make an announcement to all hosts on the network.

(a) Network address (b) Host address

(c) Broadcast address (d) None of these

4. …… client is an application that is used to read, write and send email. In simple terms it is the user interface to the email system.

(a) SMTP (b) IP

(c) FTP (d) Email

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: List the limitation of emails.

……..……

……..……

……...

7.6 Subnet and Subnet Masks

A subnet allows the flow of network traffic between hosts to be segregated based on a network configuration. By organizing hosts into logical groups, subnetting can improve network security and performance.

Subnetting an IP Network can be done for a variety of reasons, including organization, use of different physical media (such as Ethernet, FDDI, WAN, etc.), preservation of address space, and security. The most common reason is to control network traffic. In an Ethernet network, all nodes on a segment see all the packets transmitted by all the other nodes on that segment. Performance can be adversely affected under heavy traffic loads, due to collisions and the resulting retransmissions. A router is used to connect P networks to minimize the amount of traffic each segment must receive.

Subnetting allows you to create multiple logical networks that exist within a single Class A, B, or C network. If you do not subnet, you are only able to use one network from your Class A, B, or C network, which is unrealistic. Each data link on a network must have a unique network ID, with every node on that link being a member of the same network. If you break a major network (Class A, B, or C) into smaller subnetworks, it allows you to create a network of interconnecting subnetworks. Each data link on this network would then have a unique network/subnetwork ID. Any device, or gateway, connecting n networks/subnetworks has n distinct IP addresses, one for each network / subnetwork that it interconnects. In order to subnet a network, extend the natural mask using some of the bits from the host ID portion of the address to create a subnetwork ID. For example, given a Class C network of 204.17.5.0 which has a natural mask of 255.255.255.0?

The network subnetting scheme allows for eight subnets (see Figure 7.2), and the network might appear as:

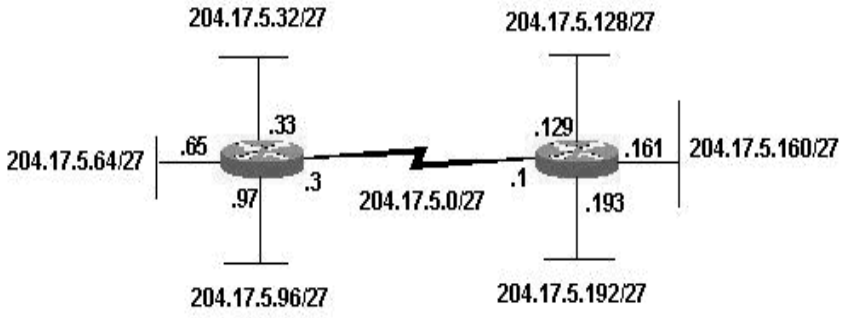


Figure 7.2: Eight subnetworks.

Subnet Masking

Subnetting is the subnet mask. Like IP addresses, a subnet mask contains four bytes (32 bits) and is often written using the same "dotted-decimal" notation. For example, a very common subnet mask in its binary representation 11111111 11111111 11111111 00000000 is typically shown in the equivalent, more readable form 255.255.255.0

Applying a subnet mask to an IP address allows you to identify the network and node parts of the address. Performing a bitwise logical AND operation between the IP address and the subnet mask results in the

Network Address or Number.
For example, using our test IP address and the default Class B subnet mask, we get:

10001100.10110011.11110000.11001000	140.179.240.200 Class B IP Address
11111111.11111111.00000000.00000000	255.255.000.000 Default Class B Subnet Mask
10001100.10110011.00000000.00000000	140.179.000.000 Network Address

Default subnet masks:
Class A - 255.0.0.0 - 11111111.00000000.00000000.00000000
Class B - 255.255.0.0 - 11111111.11111111.00000000.00000000
Class C - 255.255.255.0 - 11111111.11111111.11111111.00000000

More Restrictive Subnet Masks
Additional bits can be added to the default subnet mask for a given Class to further subnet, or break down, a network. When a bitwise logical AND operation is performed between the subnet mask and IP address, the result defines the Subnet Address. There are some restrictions on the subnet address. Node addresses of all "0"s and all "1"s are reserved for specifying the local network (when a host does not know it is network address) and all hosts on the network (broadcast address), respectively. This also applies to subnets. A subnet address cannot be all "0"s or all "1"s. This also implies that a 1 bit subnet mask is not allowed. This restriction is required because older standards enforced this restriction. Recent standards that allow use of these subnets have superseded these standards, but many 'legacy' devices do not support the newer standards. If you are operating in a controlled environment, such as a lab, you can safely use these restricted subnets. To calculate the number of subnets or nodes, use the formula (2^n - 2) where n = number of bits in either field. Multiplying the number of subnets by the number of nodes available per subnet gives you the total number of nodes available for your class and subnet mask. Also, note that although subnet masks with non-contiguous mask bits are allowed they are not recommended.
Example:

10001100.10110011.11011100.11001000	140.179.220.200 IP Address
11111111.11111111.11100000.00000000	255.255.224.000 Subnet Mask
10001100.10110011.11000000.00000000	140.179.192.000 Subnet Address
10001100.10110011.11011111.11111111	140.179.223.255 Broadcast Address

In this example a bit subnet mask was used. There are 6 subnets available with this size mask (remember that subnets with all 0's and all 1's are not allowed). Each subnet has 8190 nodes. Each subnet can have nodes assigned to any address between the Subnet address and the Broadcast address. This gives a total of 49,140 nodes for the entire class B address subnetted this way. Notice that this is less than the 65,534 nodes an unsubnetted class B address would have.
Subnetting always reduces the number of possible nodes for a given network. There are complete subnet tables available here for Class A, Class B and Class C. These tables list all the possible subnet masks for each class, along with calculations of the number of networks, nodes and total hosts for each subnet.

Exercise: Check Your Progress 3
Note: i) Use the space below for your answer.
Ex1: Calculate the range of Class B IP addresses.
……..……

……..……

……...

7.7 Summary

- Subnets are created by using a so-called subnet mask to divide a single Class A, B, or C network number into smaller pieces, thus allowing an organization to add subnets without having to obtain a new network number through an Internet service provider.
- URL contains the name of the protocol to be used to access the file resource, a domain name that identifies a specific computer on the Internet, and a pathname, a hierarchical description that specifies the location of a file in that computer.
- Electronic mail (email) is the term given to an electronic message, usually a form of simple text message that a user types at a computer system and is transmitted over some form of computer network to another user.
- An *internetwork* is a collection of individual networks, connected by intermediate networking devices, that functions as a single large network.
- A routing protocol is a protocol that specifies how routers communicate with each other, disseminating information that enables them to select routes between any two nodes on a computer network, the choice of the route being done by routing algorithms.
- The Simple Network Management Protocol (SNMP) is an application-layer protocol that facilitates the exchange of management information between network devices.
- The SNMP message format specifies which fields to include in the message and in what order. Ultimately, the message is made of several layers of nested fields.
- MIME stands for Multi-purpose Internet Mail Extensions or Multimedia Internet Mail Extensions. At first it was used as a way of sending more than just text via email. The protocol was extended to manage file typing by Web servers.
- The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web.

7.8 Keywords

Broadcast IP address: is the last IP address in the range of IP addresses. To be more precise, the broadcast address is the IP address in which all binary bits in the host portion of the IP address are set to one.

Connectionless: It is describes communication between two network end points in which a message can be sent from one end point to another without prior arrangement.

Domain name: This is a name that a company has registered so that they can use it on the Internet. Other examples are: apple.com, or microsoft.com.

IP multicast: It is a method of sending Internet Protocol (IP) datagrams to a group of interested receivers in a single transmission.

IP Network: It is a group of hosts that share a common physical connection and that use Internet Protocol for network layer communication.

Network address: It is the first IP address in the range of IP addresses. To be more precise, the network address is the address in which all binary bits in the host portion of the IP address are set to zero.

Simple Network Management Protocol (SNMP): It is an "Internet-standard protocol for managing devices on IP networks. Devices that typically support SNMP include routers, switches, servers, workstations, printers, modem racks."

Subnet: It is a segment of a network. Subnetting is a technique that allows a network administrator to divide one physical network into smaller logical networks and, thus, control the flow of traffic for security or efficiency reasons.

URL: It is one type of Uniform Resource Identifier (URI); the generic term for all types of names and addresses that refer to objects on the World Wide Web.

7.9 Review Questions

1. Define IP.
2. Explain the domain name system (DNS).
3. What is the uniform resource locator (URL)? Explain briefly.
4. Define electronic mail and also explain the electronic mail address.
5. What is the main purpose subnet and subnet masks?
6. Write the short notes principles of internetworking.
7. Describes the IPv6.
8. Explain the IP multicasting.
9. What are the routing protocols?
10. Differentiate between HTTP and MIME.

Answers for Self Assessment Questions

1. (a) 2. (b) 3. (c) 4. (d)

7.10 Further Studies

1. IP Address Management Principles and Practice By Timothy Rooney
2. Computer Networks: A Systems Approach by Larry L. Peterson, Bruce S. Davie

8

Network Security and Firewall

STRUCTURE

8.0 Objectives

8.1 Introduction

8.2 The Need for Security

8.3 Security Barriers in Network Pathways

8.4 Attacks

8.5 Approaches to Network Security

8.6 Levels of Security

8.7 Security Services

8.8 Firewall

8.9 Firewall Design Principles

8.10 Types of Firewall

8.11 Firewall Configurations

8.12 Demilitarized Zone (DMZ)

8.13 Networks VLAN

8.14 Summary

8.15 Keywords

8.16 Review Questions

8.17 Further Studies

8.0 Objectives

After studying this chapter, you will be able to:

- Discuss the need for security
- Discuss the common threats and security barriers in network pathway
- Explain the classification of attacks
- Discuss the approaches to network security and levels

8.1 Introduction

When a computer connects to a network and begins communicating with others, it is taking a risk. Network security involves the protection of a computer's internet account and files from intrusion of an unknown user. Basic

security measures involve protection by well selected passwords, change of file permissions and back up of computer's data.

Security concerns are in some ways peripheral to normal business working, but serve to highlight just how important it is that business users feel confident when using IT systems. Security will probably always be high on the IT agenda simply because cyber criminals know that a successful attack is very profitable. This means they will always strive to find new ways to circumvent IT security, and users will consequently need to be continually vigilant. Whenever decisions need to be made about how to enhance a system, security will need to be held uppermost among its requirements. The Web has become an integral part of the Internet. The Web facility on the internet is made up of collection of server and client that can exchange information.

8.2 The Need for Security

The Internet is a global collection of Interconnected Networks that facilitate information exchange and computer connectivity. The Internet is comprised of many different computers, all of which fall into two categories: servers (also known as "hosts") and clients (also known as "guests") technically, everything on the Internet can be considered a "host," but for-this discussion, we will use "hosts" and "guests." Guest machines send bursts of computer data called "packets" which are analyzed by the server belonging to the guest's Internet service provider. If the data is located locally (on the ISP's machine), the ISP's server will return the packets. If the information sought is not local (on another machine), the ISP's server hands off the packets to a router, which then sends the packets to the server containing the information. Once the information is located, it is sent back to the guest machine.

There are many different types of computers that fill these two categories: mainframes, minicomputers, PCs, Macintosh, UNIX and others. Despite the many varieties of computers that combine to form the Internet, every computer connected to the Internet needs to be able to communicate with every other computer without this ability, there is no Internet. All of these computers: are able to communicate because in a sense they can all speak the same language TCP/IP actually is not a language; it is in computer terms what is known as a "protocol." A protocol is simply a standard for transmitting and receiving bits of information. As long as all of the interconnected computers follow the same protocol, they can exchange information. Unfortunately, when data is sent from one computer to another on the. Internet, every other computer in between has an opportunity to see what's being sent. This poses an obvious security problem.

Say you decide to purchase concert tickets on the Web server. It is possible that someone could be watching the data passing through one of the computers that is in between your computer and the concert ticket server. No one knows how often this happens, but everyone concedes it is technically possible and it is also possible off the Web, too e-mail can be captured (and read, if not encrypted), as can file transfers via unsecure FTE. If someone wanted to, it would not be too difficult to connect a capture device to someone's phone line (assuming they use a modem to connect to the Internet) and steal an electronic copy of the data exchanged on the Internet. Even if you make your purchase on a secure Web site supporting the latest security features, it has been recently shown that secure sites can cause internet explorer (and other browsers) to send sensitive information to a non-secure server in plain text format.

Security is concerned with making sure that unwanted people cannot read or modify message intended for other receivers. It is also concerned that unauthorized users can not access remote services or database.

8.2.1 Common Threats

Three common factors emerges when dealing with network security, these are vulnerability, threat, and attack.

Vulnerabilities

An experienced hacker knows that every network or device has a certain degree of vulnerability or weakness, and they take advantages of each security weakness or loophole to exploit the network. A hacker or computer

criminals are continually in search of unsecured networks or devices to exploit. This includes routers, switches, desktops, servers, and even security devices.
They use variety of tools, programs and scripts to accomplish these threats. The primary network vulnerabilities or weaknesses are: Technological, Configuration and Security policy weaknesses,
Technological weaknesses: Every computer network and device has an inherent security weakness. These include TCP/IP protocol (HTTP, FTP, SMTP, SNMP) on which the Internet was designed, operating system (UNIX, Linux, Mac OS, Windows OS, and network equipment weaknesses (Routers, Firewalls, Switches etc).
Configuration weaknesses: Incorrect configuration or application of security software or firewall devices due to laxity can help to compromise a network. These includes unsecured user accounts information or passwords, system accounts information or passwords, misconfigured internet services, unsecured default settings within products, misconfigured network equipments.
Security policy weaknesses: Every organisation must have a security policy that governs and maintains how the network or company information should be used. Security risks to the network exist if users do not follow the security policy. Security weaknesses emerge when there is no clear cut or written security policy document.

8.3 Security Barriers in Network Pathways

The Pathways Network brings together many people and organizations involved in various aspects of sustainability. It is a place where ideas can be exchanged, innovations can be profiled and problems can be solved. Not only can members use typical tools such as blogs and messaging in a social networking environment, they can access a very powerful crowd sourcing tool. This tool uses the assumption that the crowd will have better answers, ideas and solutions to a problem than any one individual. New ideas can be vetted to the crowd, improvements made and viabilities assessed. Good ideas can be funded directly by the network.

8.4 Attacks

A network attack can be defined as any method, process, or means used to maliciously attempt to compromise network security.
There are a number of reasons that an individual(s) would want to attack corporate networks. The individuals performing network attacks are commonly referred to as network attackers, hackers, or crackers.

8.4.1 Classification of Attacks

Network attacks can be classified into the following four types:

1. Internal threats
2. External threats
3. Unstructured threats
4. Structured threats

Threats to the network can be initiated from a number of different sources; hence the reason for network attacks being classified as either external or internal network attacks/threats:

External Threats: Individuals carry out external threats or network attacks without assistance from internal employees or contractors. A malicious and experienced individual, a group of experienced individuals, an experienced malicious organization, or inexperienced attackers (script kiddies) carry out these attacks. Such attackers usually have a predefined plan and the technologies (tools) or techniques to carry out the attack. One of the main characteristics of external threats is that they usually involve scanning and gathering

information. Users can therefore detect an external attack by scrutinizing existing firewall logs. Users can also install an Intrusion Detection System to quickly identify external threats.

External threats can be further categorized into either structured threats or unstructured threats:

Structured External Threats: These threats originate from a malicious individual, a group of malicious individual(s), or a malicious organization. Structured threats are usually initiated from network attackers that have a premeditated thought on the actual damages and losses that they want to cause. Possible motives for structured external threats include greed, politics, terrorism, racism, and criminal payoffs. These attackers are highly skilled on network design, avoiding security measures, Intrusion Detection Systems (IDSs), access procedures, and hacking tools. They have the necessary skills to develop new network attack techniques and the ability to modify existing hacking tools for their exploitations. In certain cases, an internal authorized individual may assist the attacker.

Unstructured External Threats: These threats originate from an inexperienced attacker, typically from a script kiddie. Script kiddie refers to an inexperienced attacker who uses cracking tools or scripted tools readily available on the Internet to perform a network attack. Script kiddies are usually inadequately skilled to create the threats on their own. They can be considered bored individuals seeking some form of fame by attempting to crash Websites and other public targets on the Internet. External attacks can also occur either remotely or locally:

Remote external attacks: These attacks are usually aimed at the services that an organization offers to the public. The various forms that remote external attacks can take are. Remote attacks aimed at the services available for internal users. This remote attack usually occurs when there is no firewall solution implemented to protect these internal services. Remote attacks aimed at locating modems to access the corporate network. Denial of service (DoS) attacks to place an exceptional processing load on servers in an attempt to prevent authorized user requests from being serviced war dialling of the corporate private branch exchange (PBX). Attempts to brute force password authenticated systems.

Local External Attacks: These attacks typically originate from situations where computing facilities are shared and access to the system can be obtained.

Internal Threats: Internal attacks originate from dissatisfied or unhappy inside employees or contractors. Internal attackers have some form of access to the system and usually try to hide their attack as a normal process. For instance, internal disgruntled employees have local access to some resources on the internal network already. They could also have some administrative rights on the network. One of the best means to protect against internal attacks is to implement an Intrusion Detection System and to configure it to scan for both external and internal attacks. All forms of attacks should be logged and the logs should be reviewed and followed up. With respect to network attacks, the core components that should be included when users design network security are:

1. Network attack prevention
2. Network attack detection
3. Network attack isolation
4. Network attack recovery

8.4.2 Specific Attacks

Without security measures and controls in place, your data might be subjected to an attack. Some attacks are passive, meaning information is monitored; others are active, meaning the information is altered with intent to corrupt or destroy the data or the network itself.

Your networks and data are vulnerable to any of the following types of attacks if you do not have a security plan in place.

Eavesdropping

In general, the majority of network communications occur in an unsecured or "cleartext" format, which allows an attacker who has gained access to data paths in your network to "listen in" or interpret (read) the traffic. When an attacker is eavesdropping on your communications, it is referred to as sniffing or snooping. The ability of an eavesdropper to monitor the network is generally the biggest security problem that administrators face in an enterprise. Without strong encryption services that are based on cryptography, your data can be read by others as it traverses the network.

Data Modification

After an attacker has read your data, the next logical step is to alter it. An attacker can modify the data in the packet without the knowledge of the sender or receiver. Even if you do not require confidentiality for all communications, you do not want any of your messages to be modified in transit. For example, if you are exchanging purchase requisitions, you do not want the items, amounts, or billing information to be modified.

Identity Spoofing (IP Address Spoofing)

Most networks and operating systems use the IP address of a computer to identify a valid entity. In certain cases, it is possible for an IP address to be falsely assumed identity spoofing. An attacker might also use special programs to construct IP packets that appear to originate from valid addresses inside the corporate intranet.

After gaining access to the network with a valid IP address, the attacker can modify, reroute, or delete your data. The attacker can also conduct other types of attacks, as described as follows:

Password-Based Attacks

A common denominator of most operating system and network security plans is password-based access control. This means your access rights to a computer and network resources are determined by who you are, that is, your user name and your password.

Older applications do not always protect identity information as it is passed through the network for validation. This might allow an eavesdropper to gain access to the network by posing as a valid user.

When an attacker finds a valid user account, the attacker has the same rights as the real user. Therefore, if the user has administrator-level rights, the attacker also can create accounts for subsequent access at a later time.

After gaining access to your network with a valid account, an attacker can do any of the following:

- Obtain lists of valid user and computer names and network information.
- Modify server and network configurations, including access controls and routing tables.
- Modify, reroute, or delete your data.

Man-in-the-Middle Attack

As the name indicates, a man-in-the-middle attack occurs when someone between you and the person with whom you are communicating is actively monitoring, capturing, and controlling your communication transparently. For example, the attacker can re-route a data exchange. When computers are communicating at low levels of the network layer, the computers might not be able to determine with whom they are exchanging data.

Man-in-the-middle attacks are like someone assuming your identity in order to read your message. The person on the other end might believe it is you because the attacker might be actively replying as you to keep the exchange going and gain more information. This attack is capable of the same damage as an application-layer attack.

Compromised-Key Attack

A key is a secret code or number necessary to interpret secured information. Although obtaining a key is a difficult and resource-intensive process for an attacker, it is possible. After an attacker obtains a key, that key is referred to as a compromised key.

An attacker uses the compromised key to gain access to a secured communication without the sender or receiver being aware of the attack. With the compromised key, the attacker can decrypt or modify data, and try

to use the compromised key to compute additional keys, which might allow the attacker access to other secured communications.

Sniffer Attack

A sniffer is an application or device that can read, monitor, and capture network data exchanges and read network packets. If the packets are not encrypted, a sniffer provides a full view of the data inside the packet. Even encapsulated (tunneled) packets can be broken open and read unless they are encrypted and the attacker does not have access to the key.

Using a sniffer, an attacker can do any of the following:

- Analyze your network and gain information to eventually cause your network to crash or to become corrupted.
- Read your communications.

Application-Layer Attack

An application-layer attack targets application servers by deliberately causing a fault in a server's operating system or applications. This results in the attacker gaining the ability to bypass normal access controls. The attacker takes advantage of this situation, gaining control of your application, system, or network, and can do any of the following:

- Read, add, delete, or modify your data or operating system.
- Introduce a virus program that uses your computers and software applications to copy viruses throughout your network.
- Introduce a sniffer program to analyze your network and gain information that can eventually be used to crash or to corrupt your systems and network.
- Abnormally terminate your data applications or operating systems.
- Disable other security controls to enable future attacks.

8.5 Approaches to Network Security

In-depth security, or defence in depth, is the principle of using a layered approach to network security to provide even better protection for your computer or network. No matter how good any single network security application is, there is someone out there smarter than the people who designed it with more time on his hands than scruples who will eventually get past it. It is for this reason that common security practice suggests multiple lines of defence, or in-depth security.

In-depth security uses layers of different types of protection from different vendors to provide substantially better protection. A hacker may develop an exploit for a vulnerability that enables them to bypass or circumvent certain types of defences, or they may learn the intricacies or techniques of a particular vendor, allowing them to effectively rendering that type of defence useless.

By establishing a layered security you will help to keep out all but the cleverest and most dedicated hackers.

As a baseline suggest implementing the following computer and network security products:

Firewall: Basically, a firewall is a protective barrier between your computer, or internal network, and the outside world. Traffic into and out of the firewall is blocked or restricted as you choose. By blocking all unnecessary traffic and restricting other traffic to those protocols or individuals that need it you can greatly improve the security of your internal network.

Antivirus: Antivirus software is a type of application you install to protect your system from viruses, worms and other malicious code. Most antivirus programs will monitor traffic while you surf the Web, scan incoming email and file attachments and periodically check all local files for the existence of any known malicious code.

Intrusion Detection System (IDS): An IDS (Intrusion Detection System) is a device or application used to inspect all network traffic and alert the user or administrator when there has been unauthorized attempts or

access. The two primary methods of monitoring are signature-based and anomaly-based. Depending on the device or application used, the IDS can either simply alert the user or administrator or it could be set up to block specific traffic or automatically respond in some way.

Self Assessment Questions

1. The internet is a……..collection of Interconnected.
(a). Global (b). Local
(c). Variable (d). None of these

2. The Internet is comprised of many different computers all of which fall into…….categories.
(a). Three (b). Two
(c). Four (d).None of these

3. There are many different types of computers that fill these……categories.
(a). Two (b). Three
(c). Five (d). None of these

4. ……is concerned with making sure that unwanted people cannot read or modify message intended for other receivers.
(a). Network (b). Server
(c). Security (d). None of these

5. Three common factors emerges when dealing with network security, these are vulnerability, threat, and attack.
(a). True (b). False

8.6 Levels of Security

There are different levels of security when it comes to securing information being sent over IP networks. The first is authentication and authorization. The user or device identifies itself to the network and the remote end by a username and password, which are then verified before the device is allowed into the system. Added security can be achieved by encrypting the data to prevent others from using or reading the data. Common methods are HTTPS (also known as SSL/ TLS), VPN and WEP or WPA in wireless networks. The use of encryption can slow down communications, depending on the kind of implementation and encryption used.

8.6.1 Username and Password Authentication

Using a username and password authentication is the most basic method of protecting data on an IP network and may be sufficient where high levels of security are not required, or where the video network is segmented off from the main network and unauthorized users would not have physical access to the video network. The passwords can be encrypted or unencrypted when they are sent; the former provides the best security. Axis network video products provide multi-level password protection. Three levels are available: Administrator (full access to all functionalities), Operator (Access to all functionalities except the configuration pages), and viewer (Access only to live video).

8.6.2 IP Address Filtering

Axis network video products provide IP address filtering, which gives or denies access rights to defined IP addresses. A typical configuration is to configure the network cameras to allow only the IP address of the server that is hosting the video management software to access the network video products.

8.6.3 IEEE 802.1X

Many axis network video products support IEEE 802.1X (see Figure 8.1), which provides authentication to devices attached to a LAN port. IEEE 802.1X establishes a point-to-point connection or prevents access from the LAN port if authentication fails. IEEE 802.1X prevents what is called "port hi-jacking"; that is, when an unauthorized computer gets access to a network by getting to a network jack inside or outside a building. IEEE 802.1X is useful in network video applications since network cameras are often located in public spaces where an openly accessible network jack can pose a security risk. In today's enterprise networks, IEEE 802.1X is becoming a basic requirement for anything that is connected to a network.

In a network video system, IEEE 802.1X can work as follows:

1. A network camera sends a request for network access to a switch or access point;
2. The switch or access points forwards the query to an authentication server; for instance, a RADIUS (remote authentication dial-in user service) server such as a Microsoft Internet Authentication Service server;
3. If authentication is successful, the server instructs the switch or access point to open the port to allow data from the network camera to pass through the switch and be sent over the network.

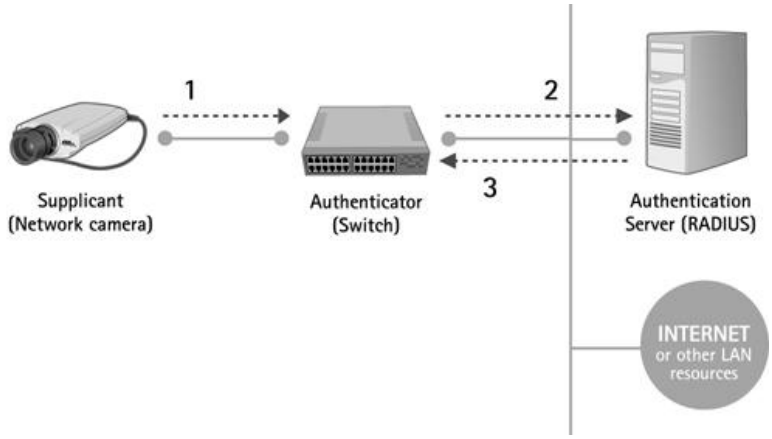


Figure 8.1: Network video system, IEEE 802.1X.

8.6.4 HTTPS or SSL/TLS

HTTPS (Hyper Text Transfer Protocol Secure) is identical to HTTP but with one key difference the data transferred is encrypted using Secure Socket Layer (SSL) or Transport Layer Security (TLS). This security method applies encryption to the data itself. Many axis network video products have built-in support for HTTPS, which makes it possible for video to be securely viewed using a Web browser. The use of HTTPS, however, can slow down the communication link and, therefore, the frame rate of the video.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: how many Security levels there are? Give a short note.

……..……

……..……

……...

8.7 Security Services

Managed Firewall

We plan, install, configure and manage firewalls in your network. Using equipment from best-in-class firewall providers, we place protection around the network perimeter to prevent access from external sources.

In addition, we identify assets that require special protection and design a defense-in-depth solution that compartmentalizes your network. This gives you additional layers of protection for your most critical assets. And it lets you restrict access to critical information only to specified groups of internal or external users and applications.

Managed Remote Access

Providing secure access to a highly distributed workforce is a challenge for IT. Employees travel or work from home. Contractors and suppliers need controlled, secure access from their locations. Through remote access services, we design, implement and manage your remote network access solution. We provide network infrastructure that lets users securely access your network from PCs, Smartphone's or PDAs. Users are authenticated via a two-factor id system rather than simple passwords, so you are protected even if passwords are lost or stolen. Network access control goes farther to let you authenticate and verify the actual device from which a user seeks access. You not only assure that it is authorized, but that its virus protection or other software is functioning and up to date.

Network Access Control

Network Access Control minimizes the risks associated with noncompliant devices regardless of system type, ownership, or access method, resulting in more resilient and secure networks. Network Access Control (NAC) restricts user access to corporate networks based on user authentication. This is done to limit a corporate network's exposure to computers, limiting these endpoints to just the resources that are required for specific business purposes. NAC helps customers maximize current investment while transitioning them to next generation network infrastructure while improving end-to-end performance, security and scalability.

Network Intrusion Prevention

Network intrusion prevention monitors your network traffic to identify and block network threats. It protects network devices and other critical infrastructure from targeted attacks and traffic anomalies. It also enables you to throttle non-mission-critical applications that hijack valuable bandwidth and IT resources. We plan, deploy, and manage intrusion prevention devices, so they are always up to date and always working with the latest attack filters. We help you incorporate alerts into your operations and security processes, so your organization can respond quickly and effectively to threats when they occur.

Managed Proxy Service and Web Content Filtering

Web proxy servers are gateways that enable your network users to securely access external Web sites while shielding your systems and network addresses from exposure to the internet. They log activity to enable reporting on internet usage. And they improve performance for Web users by storing frequently requested Web pages locally and providing them quickly when requested.

Web content filtering extends the functionality of Web proxy servers to block access to unauthorized or inappropriate Web content. This prevents unauthorized traffic from degrading network service for valid business uses. And it helps your company comply with policies concerning appropriate computer and network use. We plan, implement and manage proxy servers and Web content filtering. That provides security for your network and offers the quick, seamless access to the Web that your users need and expect.

8.8 Firewall

One of the primary security issues in computer networks is to access control and besides login and password security, it is the firewall security that firms use to protect their private networks from rest of the internet.

- A method for implementing security policies designed to keep a network secure from intruders.
- It can be a single router that filters out unwanted packets or maybe a combination of routers and servers each performing some type of firewall processing.
- Firewall is widely used to give users secure access to the internet as well as to separate a company's public Web server from its internal network.

- A firewall is either a hardware device as a software package running on a specially configured computer that sits between a secured network (your internal network) (the internet).
- Firewall performs tasks, including preventing unauthorised access to your network, limiting incoming and outgoing traffic, authenticating users, logging traffic information and producting report.
- Fundamental role is not only to monitor the traffic but also to block certain kinds of traffic completely.

Basically, a firewall is a barrier to keep destructive forces away from your property. In fact, that is why it is called a firewall. Its job is similar to a physical firewall that keeps a fire from spreading from one area to the next. There are several types of firewall techniques:

1. ***Packet filters:*** Looks at each packet entering or leaving the network and accepts or rejects it based on user-defined rules. Packet filtering is fairly effective and transparent to users, but it is difficult to configure. In addition, it is susceptible to IP spoofing.
2. ***Application gateway:*** Applies security mechanisms to specific applications, such as FTP and Telnet servers. This is very effective, but can impose performance degradation.
3. ***Circuit-level gateway:*** Applies security mechanisms when a TCP or UDP connection is established. Once the connection has been made, packets can flow between the hosts without further checking.
4. ***Proxy server:*** Intercepts all messages entering and leaving the network. The proxy server effectively hides the true network addresses.

Did You Know?

Firewall technology emerged in the late 1980s when the Internet was a fairly new technology in terms of its global use and connectivity. The predecessors to firewalls for network security were the routers used in the late 1980s.

8.9 Firewall Design Principles

Every network has a security policy that will specify what traffic is allowed to enter and leave the network. Most home computers have an open policy allowing all traffic to enter and leave. As more computers become connected to the Internet, the proliferation of viruses and hackers are a concern of even the casual Internet user. The generally accepted approach to securing your site is to take the following steps.

1. Identify what you are trying to protect.
2. Determine what you are trying to protect it from.
3. Determine how likely the threats are.
4. Implement measures that will protect your assets in a cost-effective manner.
5. Review the process continuously and make improvements each time a weakness is found.

8.9.1 Architecture

There are generally four types of firewalls: Packet Filtering Firewalls, Circuit Level Gateways, Application Level Gateways, and Stateful Multilevel Inspection Firewalls. These firewall designs are in increasing order of complexity and evolution.

Packet Filtering Firewalls

Packet filtering firewalls was the first firewall architecture and was developed by Cisco. As a packet enters the network, the IP header data is analyzed against a set of rules to determine if it should be allowed to enter the computer network. These rules cannot contain any state information and as a result are relatively simple to produce and very efficient to process. Since these rules introduce little intelligence (keeping state and using higher level protocols), complex logging and policies cannot be implemented with this technique.

Circuit Level Gateways

Circuit level gateways were the second generation of firewalls and were developed at the AT&T Bell Labs. The higher level TCP header data was used to analyze the following information session identifier, connection state (handshake, established, or closing), sequencing information, source and destination IP address, and the physical network interface. The name comes from circuit relays, which creates a direct connection between computers. The firewall will analyze a new connection and apply the security policies against it. If the connection is allowed, the remaining packets that arrive within the same connection are allowed without further tests, just as a direct connection would work. This added intelligence can help eliminate connections from only certain hosts without excessive overhead. However, protocols other than TCP have no advantages over first generation firewall architectures.

Application Level Gateways

Application level gateways were the third generation firewalls. This architecture employs a special server to act as a proxy for a host on the secured network to talk to the outside network. The proxy server will respond as though it was the actual host outside the network. Before it sends a request to the host outside the network it will apply the security policy to the request and determine if it is authorized. Since the proxy server works with the application level protocol, it can implement very complex rules. Incoming request are handled in much the same way. Instead of a direct connection, the outside computer must communicate with the proxy server in order for its packets to be passed along to the inside host. The proxy server will allow no direct connection between a trusted and outside host. Also, a proxy server has a great deal of intelligence since it can inspect application level protocols, allowing it to produce detailed logs and applying complex rules. The overhead involved in a proxy server is significant and can easily be a bottleneck for communication. Also, some applications must be configured to work with proxies and therefore complicates setup of a client computer.

Stateful Multilevel Inspection Firewalls

The latest generation is stateful multilevel inspection firewalls. These firewalls use and record the state information from the IP, transport and application level. This translates into being a very complex and powerful tool to use and offers the greatest amount of security. For example, if a hacker were to intercept packets and change the IP address to an outside host he has control over, the state of previous packets would be used to detect the inconsistency and the appropriate action (drop/ignore remaining packets in the connection and log) could be taken. This architecture will have considerable overhead since certain state information needs to be tracked and compared. Also, most operating systems either not contain a stateful firewall or have just release one (and probably require more testing).

Did You Know?

The first paper published on firewall technology was in 1988, when engineers from Digital Equipment Corporation (DEC) developed filter systems known as packet filter firewalls.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Prepare a structure firewall network security.

……..……

……..……

……...

8.10 Types of Firewall

- Several types of firewalls are there, which offer various levels of security.
- Most widely used method of firewalling is to place a computer or a router between the network and the internet, which will then control and monitor all traffic between the outside world and the local network.

Simple Traffic Logging System

It is used in Web servers; Such system record all network traffic flowing through the firewall in a file or database for auditing purposes; On most Web servers, an HTTPD log (hypertext transfer protocol daemon), also called audit log file, lists every access of files on a given Web site; It records name of file accessed, domain name that the user comes in, the exact second of access and no. of bytes transmitted by analysing the audit log of a Web site.

IP Packet Screening Routers (Packet filtering gateway)

It is the simplest firewall. This is the static traffic routing service placed between the service provider's router and the internal network. The traffic routing service may be implemented at an IP level via screening rules in a router that are programmed into the router and performed automatically.

Screening Rules

Destination application to which the packet is routed restricts access to certain applications. Block access to packets coming from entire IP address. Although well configured routers can plug many security holes, they have several disadvantages. First screening rules are difficult to specify, second screening routers are fairly inflexible.

Hardened Firewall Host

It a stripped down computer that has been configured for increased security. It requires inside or outside users to be connected to the trusted application on the firewall machine before connecting further, also it helps to prevent unauthorised users from logging onto machine on the network. It records who has logged onto a system as well as who has tried to log on but failed.

Proxy Application Gateway

Firewall can also be created through software called a proxy service. The host computer running the proxy service is referred to as application gateway. Application gateway sits; between the Internet and a company's internal network and provides middleman services to users on either site. Their primary use is to access to applications such as WWW from within a secure parameter.

Caution

A firewall can cause significant problems to users who run their PC as servers such as network file sharing or FTP service. Connections INTO your PC from outside hosts may be stopped. Running network services on PCs can be dangerous. The firewall may be a reminder that you have to reconfigure with security in mind.

8.11 Firewall Configurations

Firewalls are customizable. This means that you can add or remove filters based on several conditions. Some of these are:

IP addresses: Each machine on the Internet is assigned a unique address called an- IP address. IP addresses are 32-bit numbers, normally expressed as four "octets" in a "dotted decimal number." A typical IP address looks like this: 216.27.61.137. For example, if a certain IP address outside the company is reading too many files from a server, the firewall can block all traffic to or from that IP address.

Domain names: Because it is hard to remember the string of numbers that make up an IP address, and because IP addresses sometimes need to change, all servers on the Internet also have human-readable names, called domain names.

Protocols: The protocol is the pre-defined way that someone who wants to use a service talks with that service. The "someone" could be a person, but more often it is a computer program like a Web browser. Protocols are often text, and simply describe how the client and server will have their conversation. The HTTP in the Web's protocol. Some common protocols that you can set firewall filters for include.

- ***IP (Internet Protocol):*** The main delivery system for information over the internet.

- ***TCP (Transmission Control Protocol):*** used to break apart and rebuild information that travels over the internet.
- ***HTTP*** *(Hyper Text Transfer Protocol):* used for Web pages.
- ***FTP*** *(File Transfer Protocol):* used to download and upload files.
- ***UDP*** *(User Datagram Protocol):* used for information that requires no response, such as streaming audio and video.
- ***ICMP*** *(Internet Control Message Protocol):* used by a router to exchange the information with other routers.
- ***SMTP*** *(Simple Mail Transport Protocol):* used to send text-based information (e-mail).
- ***SNMP*** (Simple Network Management Protocol): used to collect system information from a remote computer.
- ***Telnet***: used to perform commands on a remote computer.

A company might set up only one or two machines to handle a specific protocol and ban that protocol on all other machines.

Ports: Any server machine makes its services available to the Internet using numbered ports, one for each service that is available on the server. For example, if a server machine is running a Web (HTTP) server and an FTP server, the Web server would typically be available on port 80, and the FTP server would be available on port 21. A company might block port 21 accesses on all machines but one inside the company.

Specific words and phrases: This can be anything. The firewall will sniff (search through) each packet of information for an exact match of the text listed in the filter. For example, you could instruct the firewall to block any packet with the word "X-rated" in it. The key here is that it has to be an exact match. The "X-rated" filter would not catch "X rated" (no hyphen). But you can include as many words, phrases and variations of them as you need.

8.12 Demilitarized Zone (DMZ)

In computer networking, DMZ is a firewall configuration for securing local area networks (LANs).

In a DMZ configuration, most computers on the LAN run behind a firewall connected to a public network like the Internet. One or more computers also run outside the firewall, in the DMZ. Those computers on the outside intercept traffic and broker requests for the rest of the LAN, adding an extra layer of protection for computers behind the firewall.

Traditional DMZs allow computers behind the firewall to initiate requests outbound to the DMZ. Computers in the DMZ in turn respond, forward or re-issue requests out to the Internet or other public network, as proxy servers do. (Many DMZ implementations, in fact, simply utilize a proxy server or servers as the computers within the DMZ.). The LAN firewall, though, prevents computers in the DMZ from initiating inbound requests. DMZ is a commonly-touted feature of home broadband routers. However, in most instances these features are not true DMZs. Broadband routers often implement a DMZ simply through additional firewall rules, meaning that incoming requests reach the firewall directly. In a true DMZ, incoming requests must first pass through a DMZ computer before reaching the firewall.

Caution

Do not configure a default route on your internal and DMZ interfaces. Your firewall should have exactly one default route via your ISP's Router. Wrong configuration may be the cause of harm to system.

8.13 Networks VLAN

In a traditional LAN, workstations are connected to each other by means of a hub or a repeater. These devices propagate any incoming data throughout the network. However, if two people attempt to send information at

the same time, a collision will occur and all the transmitted data will be lost. Once the collision has occurred, it will continue to be propagated throughout the network by hubs and repeaters. The original information will therefore need to be resent after waiting for the collision to be resolved, thereby incurring a significant wastage of time and resources. To prevent collisions from travelling through all the workstations in the network, a bridge or a switch can be used. These devices will not forward collisions, but will allow broadcasts (to every user in the network) and multicasts (to a pre-specified group of users) to pass through. A router may be used to prevent broadcasts and multicasts from travelling through the network.

The workstations, hubs, and repeaters together form a LAN segment. A LAN segment is also known as a collision domain since collisions remain within the segment. The area within which broadcasts and multicasts are confined is called a broadcast domain or LAN. Thus a LAN can consist of one or more LAN segments. Defining broadcast and collision domains in a LAN depends on how the workstations, hubs, switches, and routers are physically connected together. This means that everyone on a LAN must be located in the same area Figure8.2.

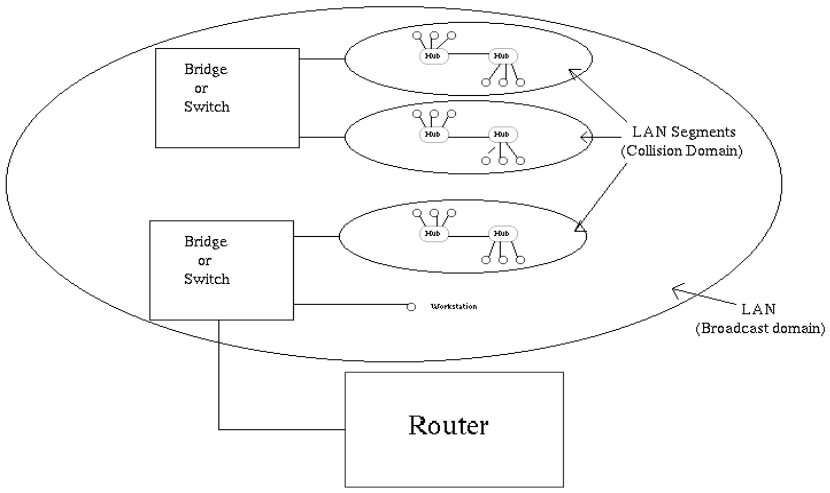
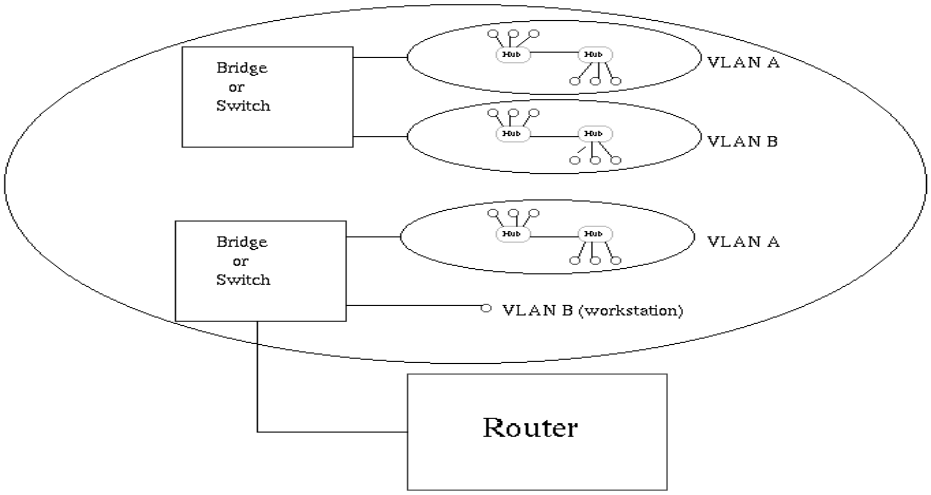
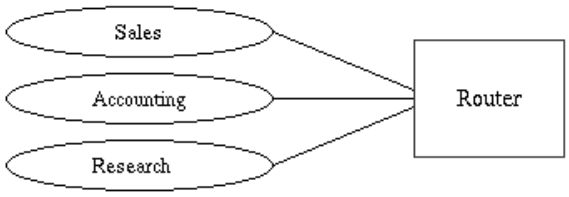


Figure 8.2: VLAN.

VLAN's allow a network manager to logically segment a LAN into different broadcast domains Figure 8.3 since this is a logical segmentation and not a physical one, workstations do not have to be physically located together. Users on different floors of the same building, or even in different buildings can now belong to the same LAN.



Physical View



Logical View

Figure 8.3: Physical and logical view of a VLAN.

VLAN's also allow broadcast domains to be defined without using routers. Bridging software is used instead to define which workstations are to be included in the broadcast domain. Routers would only have to be used to communicate between two VLAN's.

8.13.1 Types of VLAN's

VLAN membership can be classified by port, MAC address, and protocol type.

Layer 1 VLAN: Membership by Port

Membership in a VLAN can be defined based on the ports that belong to the VLAN. For example, in a bridge with four ports, ports 1, 2, and 4 belong to VLAN 1 and port 3 belongs to VLAN 2 Figure 8.4.

Port	VLAN
1	1
2	1
3	2
4	1

Figure.8.4: Assignment of ports to different VLAN's.

The main disadvantage of this method is that it does not allow for user mobility. If a user moves to a different location away from the assigned bridge, the network manager must reconfigure the VLAN.

Layer 2 VLAN: Membership by MAC Address

Here, membership in a VLAN is based on the MAC address of the workstation. The switch tracks the MAC addresses which belong to each VLAN Figure 8.5 since MAC addresses form a part of the workstation's network interface card, when a workstation is moved, no reconfiguration is needed to allow the workstation to remain in the same VLAN. This is unlike Layer 1 VLAN's where membership tables must be reconfigured.

MAC Address	VLAN
1212354145121	1
2389234873743	2

3045834758445	2
5483573475843	1

Figure 8.5: Assignment of MAC addresses to different VLAN's.

The main problem with this method is that VLAN membership must be assigned initially. In networks with thousands of users, this is no easy task. Also, in environments where notebook PC's are used, the MAC address is associated with the docking station and not with the notebook PC. Consequently, when a notebook PC is moved to a different docking station, its VLAN membership must be reconfigured.

Layer 2 VLAN: Membership by Protocol Type

VLAN membership for Layer 2 VLAN's can also be based on the protocol type field found in the Layer 2 header Figure 8.6.

Protocol	VLAN
IP	1
IPX	2

Figure 8.6: Assignment of protocols to different VLAN's.

Layer 3 VLAN: Membership by IP Subnet Address

Membership is based on the Layer 3 header. The network IP subnet address can be used to classify VLAN membership Figure 8.7.

IP Subnet	VLAN
23.2.24	1
26.21.35	2

Figure 8.7: Assignment of IP subnet addresses to different VLAN's.

Although VLAN membership is based on Layer 3 information, this has nothing to do with network routing and should not be confused with router functions. In this method, IP addresses are used only as a mapping to determine membership in VLAN's. No other processing of IP addresses is done. In Layer 3 VLAN's; users can move their workstations without reconfiguring their network addresses. The only problem is that it generally takes longer to forward packets using Layer 3 information than using MAC addresses.

Higher Layer VLAN's

It is also possible to define VLAN membership based on applications or service, or any combination thereof. For example, file transfer protocol (FTP) applications can be executed on one VLAN and telnet applications on another VLAN. The 802.1Q draft standard defines Layer 1 and Layer 2 VLAN's only. Protocol type based VLAN's and higher layer VLAN's have been allowed for, but are not defined in this standard. As a result, these VLAN's will remain proprietary.

Exercise: Check Your Progress 3

Note: i) Use the space below for your answer.

Ex1: Prepare a structure of VLAN.

……..……

……..……

……...

8.14 Summary

- Security issues for networks are visible and important, but their analysis is similar to the analysis done for other aspects of security.
- Networks usually employ many copies of the same or similar software, with a copy on each of several (or all) machines in the network.
- A network's security depends on all the cryptographic tools at our disposal, good program development processes, operating system controls, trust and evaluation and assurance methods, and inference and aggregation controls.
- Application gateway sits between the internet and a company's internal network and provides middleman services to users on either site.
- Packet filtering firewalls was the first firewall architecture and was developed by Cisco.

8.15 Keywords

Authentication: Process of establishing the identifies of the communicating parties, here the receiver must get assured that the sender is the one that it claims to be. Sender is not posing itself as somebody else.

Diversionary Tactics: Hackers may strike a set of servers in a target company and then when security administrators are busy putting out that fire, they slip in and attack another part of the network.

Firewall: One of the primary security issue in computer networks is to access control and besides login and password security, it is the firewall security that firms use to protect their private networks from rest of the internet.

Security Attack: Any action that compromise the security of information owned by the organization or individual.

Security service: A service that enhances the security of the data processing system and information transfers of an organisation.

Vulnerabilities: A hacker who worms his way into the VPN has free and easy access to the network.

8.16 Review Questions

1. What is the need for security and common threats?
2. What are the security barriers in network pathway?
3. What is the classification and specific attacks?
4. What are the approaches to network security and level of security?
5. What are the security services?
6. What are the firewall and design principles?
7. What are the types of firewall and configuration?
8. What is the demilitarized zone (DMZ)?
9. What are the networks, and virtual local area network (VLAN)?

10. What is the difference between packet filtering firewalls and Circuit Level Gateway?

Answers for Self Assessment Questions

1. (a) 2. (b) 3. (a) 4. (c) 5. (a)

8.17 Further Studies

1. http://books.google.co.in/books?id=qZgtAAAAQBAJ&pg=PA45&dq=Network+Security+and+Firewall&hl=en&sa=X&ei=Q3zCUq-_JYSWiQflv4HgDA&ved=0CD0Q6AEwAA#v=onepage&q=Network%20Security%20and%20Firewall&f=false
2. Computer Networks: A Systems Approach by Larry L. Peterson, Bruce S. Davie

9

Viruses and Security Threats

STRUCTURE

9.0 Objectives

9.1 Introduction

9.2 Virus and Threats

9.3 Malicious Programs

9.4 Types of Viruses

9.5 Virus and Countermeasures

9.6 Antivirus Approach

9.7 Advanced Antivirus Techniques

9.8 Distributed Denial of Service Attacks

9.9 Summary

9.10 Keywords

9.11 Review Questions

9.12 Further Studies

9.0 Objectives

After studying this chapter, you will be able to

- Explain the virus and threats
- Discuss the malicious programs
- Explain the types of viruses
- Discuss the virus countermeasures and antivirus approach
- Explain the advanced antivirus techniques
- Explain the distributed denial of service attacks and description

9.1 Introduction

VIRUS (Vital Information Resource under Seize). Actually, Virus is a piece of executed code that performs some unwanted or undesirable action in system that is harmful for system. This harmful code is written in different languages, like Batch Programming, C, VB etc. There are various types of viruses today, like File Virus, Boot Sector Virus etc. file virus includes all types of files and boot sector virus is a powerful piece of code that may not be found sometimes because it hides itself in boot sector of operating system; so that after each time the computer starts, the virus will execute itself and harm the system at booting time.

Today popular viruses are built in C and mostly these are in VB. Earlier viruses were built in batch programming, but these viruses are not very efficient to harm the system and quite easily for antivirus to detect and remove it, even a good computer engineer can also find the virus and remove it. But in case of VBscript viruses, it is very difficult to find and remove the virus from system.

9.2 Virus and Threats

We have all had it happen, that moment of panic when you wonder if your computer has a virus. In that situation, we can make the wrong decisions by deleting or installing files. If your system is set up properly and you back up files on a regular basis, there is no reason to panic. Hoaxes can actually make you the virus by getting you to delete needed programs or files causing the system to bog down or crash. These hoaxes usually arrive in the form of an e-mail and contain false virus warnings. When in doubt, check a list of known virus hoaxes. The best thing to do is to delete the email.

Some virus programs appear to have been sent by a friend or a company you have emailed before. Most come from unknown sources. Once opened they work by pulling names out of a computer's address book and using them to further spread the virus. You can set your email to accept plain text only. Block or remove emails that contain file attachments. Some viruses are even programmed to instruct your pc to show you only the plain text but can still infect your computer with hidden malicious code.

9.2.1 Prevent Computer Virus Threats

Help prevent computer viruses by updating windows regularly and installing virus protection programs. Virus updates should be done at least once a week. Scheduled the task time to run automatically so that it would not interfere with your work. Set up your computer to automatically download updates, but install the updates manually. If your system starts running very slowly for no reason, go online and get the system checked. A good scan of your entire operating system, memory, hard drive and boot sector is needed.

Anti-virus programs hunt for viruses and clean attachments if possible. When a file cannot be cleaned, the anti-virus program will isolate the file. The anti-virus program uses the definition list you download from the program's Website, or it matches up a general pattern of what a virus looks like. The schedule with which the anti-virus definitions are updated can vary, and you may get caught in that window of vulnerability between the virus appearing and it getting updated to the list. Software companies use patches to correct a problem or a weakness that people can take advantage of. Patching is a necessity and will be an ongoing method for computer systems weakness prevention.
The Internet is not always what you see. Scan all of your email attachments as you download them. This should be done with files you download directly from internet sites as well as music files, programs, e-books, games, etc.

Caution

Be careful sharing files with others, such as MP3, videos, programs, pictures, etc. Downloadable data can contain malicious code that you download without knowing it, and will infect your computer.

Did You Know?

The first academic work on the theory of computer viruses (although the term "computer virus" was not used at that time) was done in 1949 by John von Neumann.

9.3 Malicious Programs

Malware (for "malicious software") is any program or file that is harmful to a computer user. Thus, malware includes computer viruses, worms, Trojan horses, and also spyware, programming that gathers information about a computer user without permission. Table 9.1 describe some malicious programs.

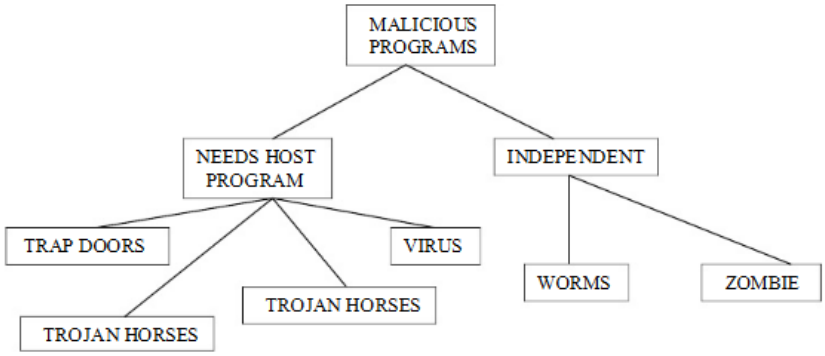


Figure 9.1: Taxonomy of Malicious Programs

9.3.1 Viruses and Worms

Viruses and worms are malicious programs that self-replicate on computers or via computer networks without the user being aware; each subsequent copy of such malicious programs is also able to self-replicate. Malicious programs which spread via networks or infect remote machines when commanded to do so by the "owner" (e.g. Backdoors) or programs that create multiple copies that are unable to self-replicate are not part of the Viruses and Worms subclass.

9.3.2 Trojans

Trojans are malicious programs that perform actions which are not authorized by the user they delete, block, modify or copy data, and they disrupt the performance of computers or computer networks. Unlike viruses and worms, the threats that fall into this category are unable to make copies of themselves or self-replicate.

Table 9.1: Malicious Programs

Malicious Programs	Explanation
	Dependent Programs
Trapdoor	This is a secret (or simply undocumented) entry point into the host program. It is used by developers to test and debug their programs by allowing remote or direct management of the program.
Trojan Horse	This seems to be an apparently useful program containing hidden codes which when it is invoked performs harmful actions such as deleting a user's files and destroying data. Like the Greek mythology's Trojan Horse, it deceives individuals into believing that it is un-harmful when it is the opposite. It allows an unauthorised user to accomplish tasks indirectly that it could not normally accomplish.

Logic Bombs	One of the oldest types of program. It embeds its code into legitimate programs and then like a bomb, executes - explodes when certain conditions are met. These condition can be either an absence or deletion of a certain file/s and a special day, time and so on. The Time bomb is the logic bomb that reacts based on time and date.
Virus	Viruses reproduce by attaching copies of themselves to existing programs. The new copy of the virus is executed when a user executes the new host program It will only cause harm after it has infected an executable file and the executable file is run. Viruses can do anything that the host program can these include destroying, modifying files, and exposing or creating vulnerabilities.
	Independent Programs
Worms	Worms is a full-featured program unlike a virus. The worm can copy itself; a virus however needs another program to start copying. It is unlike a Trojan horse in that it does not pretend to be something else. Just like an earthworm it finds its way around the network by exploiting vulnerabilities which allows it to create copies of itself on remote machines. A true worm neither deletes nor changes files on a computer all it does is to simply make multiple copies of itself and send these copies from the computer. What happens then is that it clogs disk drives with an aim to exhaust memory and clogs the Internet with multiple copies of the worm.
Bacteria/ Rabbit	Bacteria like worms are self-replicating programs and not need a host program to execute. Bacteria usually execute as separate threads/processes on multi-threaded systems; by forking off new copies the result is that they consume all the systems resources. They do not explicitly damage files but just replicate themselves which then leads to them taking up processor.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Prepare the diagram of taxonomy of malicious programs.

……..……

……..……

……...

9.4 Types of Viruses

Viruses usually consume a large amount of computer memory, resulting in system crashes. Viruses are categorized into several parts based on its features.

- Boot Sector Virus
- File Infection Virus
- Multipartite Virus
- Network Virus
- E-mail Virus
- Macro Virus

Boot Sector Viruses

Boot sector viruses directly attack the boot sector of the hard drive and infect the vital component of the boot process. The boot sector of any hard drive holds some vital information that controls the hard drive and some part of the operating system. The aim of such viruses is to load successfully into the system memory while the boot process begins. Boot sector viruses do not affect your data rather they go after the drive itself on which the virus is saved. Previously, when floppy drives were excessively used, virus spread too quickly from one computer to another through the floppy disks, but since the advent of CDs and DVDs which are not infected by viruses, it becomes impossible to infect the programs that they carry. Another major reason behind this virus has become less common is the stand guard of operating systems over the boot sector which makes it almost impossible for the virus to infect the boot sector.

File Virus

File virus is a type of virus coded in a special way that it attaches itself to the .exe files, compressed files such as zip files, and driver files. As soon as the program is started or the file it is attached to is opened, it sets into action. From there it keeps spreading to other files and programs until it starts along its intended mission for which it is codified. If you see closely, these viruses follows the two prong approach that is, first they duplicate themselves and then start its intended mission. They go through the computer program until they find the program for which they are written and when the program is run, these viruses activate. These viruses have the tendency to spread and infect the whole system or any other computer attached to the original system. They viruses contain some special codes related to the certain programs or events. When those programs are events are initiated, these codes are activated. These events can be a date or any other trigger event easily defined on any of the computer system.

Multipartite Viruses

Multipartite virus is a term used to refer to the types of computer viruses which are both a file virus and a boot sector virus. These types of viruses spread through different kind of media. Once embedded in the system memory, these viruses make its way to the hard drive and eventually infect the boot sector. For boot sector these viruses spread to the executable files and finally infect the whole system. Although, this types of computer viruses has have past their prime time for one or another reasons, these viruses were accountable for so many system crashes in the past because of their dual characteristics, i.e. of file virus and of boot sector virus.

Network Viruses

Network viruses are made particularly for networks. This virus spreads rapidly through the local area network and eventually across the Internet, but in majority of cases this virus spreads within the shared resources such as folder and drives. After getting through a computer protection system, this virus looks for vulnerable computers in the network and infect that computer in the same manner and this process keeps on going until the virus is able to find other vulnerable computers.

E-Mail Viruses

An e-mail virus is a type of computer virus that is known as a macro virus. The e-mail virus is capable of spreading itself by chasing after the other contacts in an e-mail address book by send itself on its own to those contacts hoping that they will also activate it. So, we can say that this virus has a great tendency to spread itself and at times it is activated even by viewing it in the mail client. An apt example of an e-mail virus is that of virus which was a very destructive virus cause damaged all over the world. This is why it is recommended to the users that not to open the unknown mails and the junk folder is introduced by major companies.

Macro Viruses

Macro viruses are the most common form of computer viruses which infect the files that use macros in the program. For instance, the most apt example of the program that uses macros is Microsoft Office where the files created in Word document, Excel spreadsheets, Access databases, PowerPoint presentations. Some of the other files from AmiPro, Coral Draw and so on can also be infected by macro viruses.

Macro viruses are so programmed that they use language according to the application, i.e. the language that an application understands and not the language of the operating system. This shows that these viruses aim more to the applications that they are intended to make for than the operating system and this property make them harmful for all the operating systems like Mac, Windows or Linux that supports these applications, but since then they have moved to other programs and are severe in nature. Macro viruses are found in large numbers, as the macro language is becoming more and more powerful, the macro viruses has also become a crucial type of computer viruses.

Did You Know?

Macro viruses were introduced in 1995 and that first virus infected the Microsoft Word file.

9.5 Virus and Countermeasures

All computer systems have vulnerabilities that can result in an infection with a computer virus, but not all systems have the same vulnerabilities. The key is to identify those vulnerabilities that apply to your system, and to take the countermeasures that prevent a virus from exploiting those weaknesses. Many vulnerabilities result from the software that you are using or the configuration of your computer. Countermeasures can include changing or adding software, or changing the computer system configuration.

Anti-Virus Software

The basic countermeasure to prevent infection with a computer virus is to install anti-virus software. Such software identifies and removes known virus.

Firewalls

Firewalls prevent unauthorized access of your computer through a network connection. They guard against someone installing software remotely on your computer or viewing your private data. While anti-virus software looks for viruses already in your computer, firewalls block network traffic that might contain viruses.

Passwords

One way a virus might infect your computer is through someone copying files containing a virus from a digital drive or CD onto your computer. A good countermeasure is to limit access to your computer with a log-in including a password. When only people who have the password can access your computer, you can make sure they check their files for viruses before copying them onto your machine.

Secure Wireless Networks

Even with a firewall, someone might gain access to your computer via an unsecured wireless network. When you first set up a wireless network, a password is not required for access. Anyone who can find the signal can use the wireless network, and try to gain access to your computer without your knowledge. Securing your

wireless network with a password is a countermeasure that prevents unauthorized use, and protects all the computers that are connected to the network.

Email

Email is a common vulnerability for all computers that use it. Email enters through firewalls, and attachments may masquerade as safe files when they really contain a virus. Countermeasures include not opening attachments to emails if you do not recognize the sender, and not opening attachments that have strange names or endings.

Downloading

Downloading files and installing software represents a key vulnerability for most computers. When you download and install software that contains a virus, you by-pass most of the other countermeasures. Anti-virus software may detect the virus once it is active but, if the virus is new, the software may not recognize it. Countermeasures specifically aimed at downloading malicious files include only downloading files from reputable Websites, and limiting downloading to clearly identified files for specific purposes.

Self Assessment Questions

1. ……..are different from regular botnet-based attacks.

(a). Peer-to-peer attacks (b). Degradation-of-service attacks

(b). Reflected attack (d). None of these

2. A …….involves sending mangled IP fragments with overlapping, over-sized payloads to the target machine.

(a). Teardrop attack (b). Peer-to-peer attacks

(c). Warm (d). None of these

3. Advanced Antivirus is classified as a rogue scanner, sometimes referred to as scare ware.

(a). True (b). False

4 ……and installing software represents a key vulnerability for most computers.

(a). Reload (b). Copy

(c). Downloading files (d). None of these

5 . A good countermeasure is to limit access to your computer with a log-in including a…….

(a). Log-in (b). Password.

(c). Enter (d). None of these

9.6 Antivirus Approach

There are many types of computer viruses and malware out there now, so anti-virus programs actually catch more than just "viruses." Most antivirus software can now detect computer Viruses, Worms, Spyware, Trojans, Malware, and Phishing attacks. The more complex your antivirus program is, the better protected your computer system will be:

Suspicious Behaviour Approach

There are several different methods that antivirus programs use to detect computer viruses with the suspicious. Behaviour approach, your anti-virus program will monitor the behaviour of programs on your computer. The suspicious behaviour approach is one of the better virus detection methods because it will not only detect suspicious behaviour from programs currently running on your computer, but it will also detect the behaviours of new viruses that show up on your system.

Dictionary Approach

With the dictionary approach, your anti-virus program will scan your computers files when they are created, opened, closed or emailed on your computer. The benefit of this is that it can pick up a virus as soon as it is received. The downside to the dictionary approach is that the anti-virus program can only detect viruses that are known in a "dictionary" or comprehensive database of known computer viruses. This means your anti-virus program will only pick up viruses that are "known" and can miss newer, non-registered viruses that can still infect your computer.

Other Approaches

These are the most common approaches used by antivirus software on the market today, although new techniques are constantly being developed all the time. One such new approach is called sandboxing, which runs questionable programs in a safe environment to see if they are malicious or not. Many modern antivirus programs, such as Norton Antivirus, NOD32 Antivirus, and Kaspersky Antivirus, use a combination of the suspicious behaviour and dictionary approach as well as other methods to detect computer viruses and malware.

9.7 Advanced Antivirus Techniques

Advanced antivirus (see Figure 9.2) is classified as a rogue scanner, sometimes referred to as scare ware. Rogue scanners masquerade as antivirus, antispyware, or other security software, claiming the user's system is infected in order to trick them into paying for a full version. In some cases, users who pay for the full version of a rogue scanner product will end up with erroneous charges on their credit card after purchase.

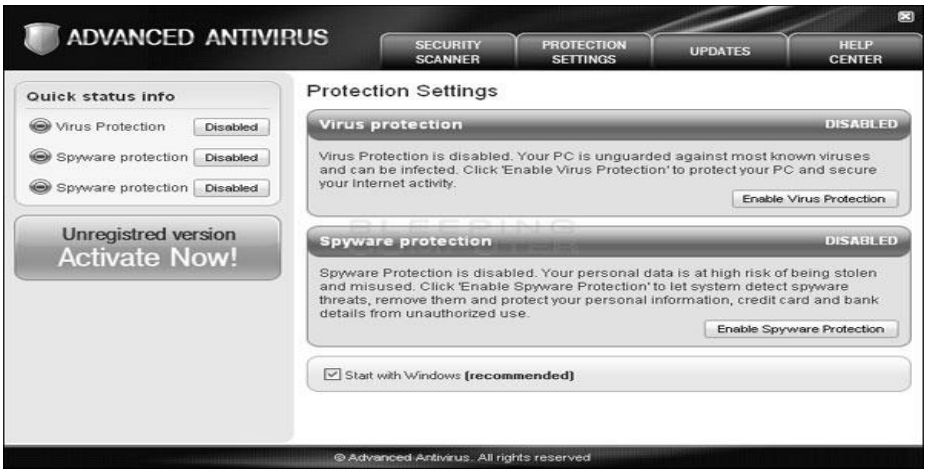


Figure 9.2: Advanced techniques.

Method of Infection

Rogue scanners may exploit security vulnerabilities to forcibly install, or they may attempt to trick users into installing by masquerading as a free security scan or system tune-up. Rogue scanners are also distributed via email scams, often masquerading as breaking news alerts.

Prevention

Follow these computer safety tips to prevent unintended installations of rogue scanners and other forms of malware. Before installing any new program, search the Web and read reviews.

Removal

The free SmitFraudFix tool is capable of detecting and removing advanced antivirus for instructions.

9.8 Distributed Denial of Service Attacks

A denial-of-service attack (DoS attack) or distributed denial-of-service attack (DDoS attack) is an attempt to make a computer or network resource unavailable to its intended users. Although the means to carry out, motives for, and targets of a DoS attack may vary, it generally consists of the concerted efforts of a person, or multiple people to prevent an Internet site or service from functioning efficiently or at all, temporarily or indefinitely. Perpetrators of DoS attacks typically target sites or services hosted on high-profile Web servers such as banks, credit card payment gateways, and even root name servers. The term is generally used relating to computer networks, but is not limited to this field; for example, it is also used in reference to CPU resource management.

One common method of attack involves saturating the target machine with external communications requests, such that it cannot respond to legitimate traffic, or responds so slowly as to be rendered effectively unavailable. Such attacks usually lead to a server overload. In general terms, DoS attacks are implemented by either forcing the targeted computers to reset, or consuming its resources so that it can no longer provide its intended service or obstructing the communication media between the intended users and the victim so that they can no longer communicate adequately.

Denial-of-service attacks are considered violations of the IAB's Internet proper use policy, and also violate the acceptable use policies of virtually all Internet service providers. They also commonly constitute violations of the laws of individual nations. When the DoS Attacker sends many packets of information and requests to a single network adapter, each computer in the network would experience effects from the DoS attack.

9.8.1 Methods of Attack

A "denial-of-service" attack is characterized by an explicit attempt by attackers to prevent legitimate users of a service from using that service. There are two general forms of DoS attacks: those that crash services and those that flood services.

A DoS attack can be perpetrated in a number of ways. The five basic types of attack are:

1. Consumption of computational resources, such as bandwidth, disk space, or processor time.
2. Disruption of configuration information, such as routing information.
3. Disruption of state information, such as unsolicited resetting of TCP sessions.
4. Disruption of physical network components.
5. Obstructing the communication media between the intended users and the victim so that they can no longer communicate adequately.

ICMP (Internet Control Message Protocol) Flood

A Smurf attack is one particular variant of a flooding DoS attack on the public Internet. It relies on misconfigured network devices that allow packets to be sent to all computer hosts on a particular network via the broadcast address of the network, rather than a specific machine. The network then serves as a smurf amplifier. In such an attack, the perpetrators will send large numbers of IP packets with the source address faked to appear to be the address of the victim. The network's bandwidth is quickly used up, preventing legitimate packets from getting through to their destination. To combat Denial of Service attacks on the internet, services like the Smurf Amplifier Registry have given network service providers the ability to identify misconfigured networks and to take appropriate action such as filtering.

Ping flood is based on sending the victim an overwhelming number of ping packets, usually using the "ping" command from Unix-like hosts (the -t flag on windows systems has a far less malignant function). It is very simple to launch, the primary requirement being access to greater bandwidth than the victim. Ping of death is based on sending the victim a malformed ping packet, which might lead to a system crash.

SYN (SYNchronize) Flood

A SYN flood occurs when a host sends a flood of TCP/SYN packets, often with a forged sender address. Each of these packets is handled like a connection request, causing the server to spawn a half-open connection, by sending back a TCP/SYN-ACK packet(Acknowledge), and waiting for a packet in response from the sender address(response to the ACK Packet). However, because the sender address is forged, the response never comes. These half-open connections saturate the number of available connections the server is able to make, keeping it from responding to legitimate requests until after the attack ends.

Teardrop Attacks

A Teardrop attack involves sending mangled IP fragments with overlapping, over-sized payloads to the target machine. This can crash various operating systems due to a bug in their TCP/IP fragmentation re-assembly code. Windows 3.1x, Windows 95 and Windows NT operating systems, as well as versions of Linux prior to versions 2.0.32 and 2.1.63 are vulnerable to this attack. But the attack targeted SMB2 which is a higher layer than the TCP packets that teardrop used.

Low-rate Denial-of-Service Attacks

The Low-rate DoS (LDoS) attack exploits TCP's slow-time-scale dynamics of retransmission time-out (RTO) mechanisms to reduce TCP throughput. Basically, an attacker can cause a TCP flow to repeatedly enter a RTO state by sending high-rate, but short-duration bursts, and repeating periodically at slower RTO time-scales. The TCP throughput at the attacked node will be significantly reduced while the attacker will have low average rate making it difficult to be detected.

Peer-to-peer Attacks

Attackers have found a way to exploit a number of bugs in peer-to-peer servers to initiate DDoS attacks. The most aggressive of these peer-to-peer-DDoS attacks exploits DC++. Peer-to-peer attacks are different from regular botnet-based attacks. With peer-to-peer there is no botnet and the attacker does not have to communicate with the clients it subverts. Instead, the attacker acts as a "puppet master," instructing clients of large peer-to-peer file sharing hubs to disconnect from their peer-to-peer network and to connect to the victim's Website instead. As a result, several thousand computers may aggressively try to connect to a target Website. While a typical Web server can handle a few hundred connections per second before performance begins to degrade, most Web servers fail almost instantly under five or six thousand connections per second. With a moderately large peer-to-peer attack, a site could potentially be hit with up to 750,000 connections in short order. The targeted Web server will be plugged up by the incoming connections.

Asymmetry of Resource Utilization in Starvation Attacks

An attack which is successful in consuming resources on the victim computer must be either carried out by an attacker with great resources, by either controlling a computer with great computation power or, more commonly, large network bandwidth controlling a large number of computers and directing them to attack as a group. A DDOS attack is the primary example of this taking advantage of a property of the operating system or applications on the victim system which enables an attack consuming vastly more of the victim's resources than the attacker's (an asymmetric attack). Smurf attack, SYN flood, Sock stress and NAPTHA are all asymmetric attacks. An attack may utilize a combination of these methods in order to magnify its power.

Permanent Denial-Of-Service Attacks

A permanent denial-of-service (PDoS), also known loosely as plashing is an attack that damages a system so badly that it requires replacement or reinstallation of hardware. Unlike the distributed denial-of-service attack, a PDoS attack exploits security flaws which allow remote administration on the management interfaces of the victim's hardware, such as routers, printers, or other networking hardware. The attacker uses these vulnerabilities to replace a device's firmware with a modified, corrupt, or defective firmware image a process which when done legitimately is known as flashing. This therefore "bricks" the device, rendering it unusable for its original purpose until it can be repaired or replaced.

The PDoS is a pure hardware targeted attack which can be much faster and requires fewer resources than using a botnet in a DDoS attack. Because of these features, and the potential and high probability of security exploits

on Network Enabled Embedded Devices (NEEDs), this technique has come to the attention of numerous hacker communities.

Application-Level Floods

Various DoS-causing exploits such as buffer overflow can cause server-running software to get confused and fill the disk space or consume all available memory or CPU time.

Other kinds of DoS rely primarily on brute force, flooding the target with an overwhelming flux of packets, over saturating its connection bandwidth or depleting the target's system resources. Bandwidth-saturating floods rely on the attacker having higher bandwidth available than the victim; a common way of achieving this today is via Distributed Denial of Service, employing a botnet. Other floods may use specific packet types or connection requests to saturate finite resources by, for example, occupying the maximum number of open connections or filling the victim's disk space with logs.

A "banana attack" is another particular type of DoS. It involves redirecting outgoing messages from the client back onto the client, preventing outside access, as well as flooding the client with the sent packets.

Caution

Use a suitable downloader to download any software to avoid the virus attack on your system.

9.8.2 DDoS Attack Description

A distributed denial of service attack (DDoS) occurs when multiple systems flood the bandwidth or resources of a targeted system, usually one or more Web servers. These systems are compromised by attackers using a variety of methods. Malware can carry DDoS attack mechanisms; one of the better-known examples of this was My Doom. Its DoS mechanism was triggered on a specific date and time. This type of DDoS involved hard coding the target IP address prior to release of the malware and no further interaction was necessary to launch the attack.

A system may also be compromised with a Trojan, allowing the attacker to download a zombie agent. Attackers can also break into systems using automated tools that exploit flaws in programs that listen for connections from remote hosts. This scenario primarily concerns systems acting as servers on the Web. Stacheldraht is a classic example of a DDoS tool. It utilizes a layered structure where the attacker uses a client program to connect to handlers, which are compromised systems that issue commands to the zombie agents, which in turn facilitate the DDoS attack. Agents are compromised via the handlers by the attacker, using automated routines to exploit vulnerabilities in programs that accept remote connections running on the targeted remote hosts. Each handler can control up to a thousand agents.

These collections of systems compromisers are known as botnets. DDoS tools like Stacheldraht still use classic DoS attack methods centered on IP spoofing and amplification like Smurf attacks and fraggle attacks (these are also known as bandwidth consumption attacks). SYN floods (also known as resource starvation attacks) may also be used.

Reflected/Spoofed Attack

A distributed reflected denial of service attack (DRDoS) involves sending forged requests of some type to a very large number of computers that will reply to the requests. Using Internet Protocol address spoofing, the source address is set to that of the targeted victim, which means all the replies will go to (and flood) the target.

ICMP Echo Request attacks (Smurf Attack) can be considered one form of reflected attack, as the flooding hosts send Echo Requests to the broadcast addresses of mis-configured networks, thereby enticing many hosts to send. Many services can be exploited to act as reflectors, some harder to block than others. DNS amplification attacks involve a new mechanism that increased the amplification effect, using a much larger list of DNS servers.

Degradation-of-service Attacks

"Pulsing" zombies are compromised computers that are directed to launch intermittent and short-lived flooding of victim Websites with the intent of merely slowing it rather than crashing it. This type of attack, referred to as "degradation-of-service" rather than "denial-of-service", can be more difficult to detect than regular zombie invasions and can disrupt and hamper connection to Websites for prolonged periods of time, potentially causing more disruption than concentrated floods. Exposure of degradation-of-service attacks is complicated further by the matter of discerning whether the attacks really are attacks or just healthy and likely desired increases in Website traffic.

Unintentional Denial of Service

This describes a situation where a Website ends up denied, not due to a deliberate attack by a single individual or group of individuals, but simply due to a sudden enormous spike in popularity. This can happen when an extremely popular Website posts a prominent link to a second, less well-prepared site, for example, as part of a news story. The result is that a significant proportion of the primary site's regular users potentially hundreds of thousands of people click that link in the space of a few hours, having the same effect on the target Website as a DDoS attack.

Denial-of-Service Level II

The goal of DoS L2 (possibly DDoS) attack is to cause a launching of a defense mechanism which blocks the network segment from which the attack originated. In case of distributed attack or IP header modification (that depends on the kind of security behaviour) it will fully block the attacked network from Internet, but without system crash.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Prepare a list of some more viruses.

……..……

……..……

……...

9.9 Summary

- Email programs are the most common means of propagation, delivering around 75% of the 50 top virus-like threats.
- Malware attacks are increasing in both frequency and sophistication, thus posing a serious threat to the internet economy and to national security.
- Anti-virus professionals use bait files to take a sample of a virus. It is more practical to store and exchange a small, infected bait file, than to exchange a large application program that has been infected by the virus.
- Application viruses spread from one application to another on the computer. Each time an infected application program is run; the virus takes control and spreads to other applications.
- Script viruses infect other script files on the computer. Script viruses, which are written in high-level script languages such as Perl or Visual Basic, gain control when a user runs an infected script file.

9.10 Keywords

Bootsector Virus: A virus which attaches itself to the first part of the hard disk that is read by the computer upon bootup. These are normally spread by floppy disks.

Malicious Tools: Malicious tools are malicious programs designed to automatically create viruses, worms, or Trojans, conduct DoS attacks on remote servers, hack other computers, etc.

Malicious Programs: Which are specifically designed to delete, block, modify, or copy data or to disrupt the performance of computers and computer networks.

Network Viruses: This virus spreads rapidly through the local area network and eventually across the Internet, but in majority of cases this virus spreads within the shared resources such as folder and drives.

Peer-To-Peer Attacks: Are different from regular botnet-based attacks. With peer-to-peer there is no botnet and the attacker does not have to communicate with the clients it subverts.

9.11 Review Questions

1. What are the virus and threats?
2. What is the malicious program?
3. How many types of viruses and methods of attack?
4. Define viruses, worms and trojans?
5. What is the virus countermeasures and antivirus approach?
6. What are the advanced antivirus techniques?
7. What are the distributed denials of service attacks? Explain briefly.
8. What is the difference between reflected and degradation-of-service attacks?
9. What is the internet control message protocol (ICMP) flood?
10. What is the difference between suspicious behaviour and dictionary approach?

Answers for Self Assessment Questions

1. (a) 2. (a) 3. (a) 4. (c) 5. (b)

9.12 Further Studies

1. Security Threat Mitigation and Response By Tesch Dale
2. Computer Networks: A Systems Approach by Larry L. Peterson, Bruce S. Davie

10

Encryption and Decryption

STRUCTURE

10.0 Objectives

10.1 Introduction

10.2 Encryption and Decryption

10.3 Cryptography Terminology

10.4 Classification of Cryptography

10.5 Security of Algorithms

10.6 Steganography

10.7 Steganography versus Cryptography

10.8 Comparison of Symmetric and Asymmetric Key Cryptography

10.9 Public-Key Cryptography

10.10 Summary

10.11 Keywords

10.12 Review Questions

10.13 Further Studies

10.0 Objectives

After studying this chapter, you will be able to:

- Explain the encryption and decryption
- Discuss the cryptography terminology
- Explain the classification of cryptography
- Discuss the security of algorithms
- Define the steganography
- Explain the steganography versus cryptography
- Explain the public key encryption
- Discuss the comparison of symmetric and asymmetric key cryptography
- Discuss the public key cryptanalysis

10.1 Introduction

A message is called either plaintext or clear text. The process of disguising a message in such a way as to hide its substance is called encryption. An encrypted message is called cipher text. The process of turning cipher text back into plaintext is called decryption. The art and science of keeping messages secure is called cryptography, and it is practiced by cryptographers. Cryptanalysts are practitioners of cryptanalysis, the art and science of breaking cipher text, i.e. seeing through the disguise. The branch of mathematics embodying both cryptography and cryptanalysis is called cryptology, and it is practitioners are called cryptologists.

PGP uses public-key encryption to protect e-mail and data files. Communicate securely with people you have never met, with no secure channels needed for prior exchange of keys. PGP is well featured and fast, with sophisticated key management, digital signatures, data compression, and good ergonomic design.

Pretty Good Privacy (PGP) is a high security cryptographic software application for MS-DOS, UNIX, VAX/VMS, and other computers. PGP allows people to exchange files or messages with privacy, authentication, and convenience. Privacy means that only those intended to receive a message can read it. Authentication means that messages that appear to be from a particular person can only have originated from that person. Convenience means that privacy and authentication are provided without the hassles of managing keys associated with conventional cryptographic software. No secure channels are needed to exchange keys between users, which makes PGP much easier to use. This is because PGP is based on a powerful new technology called "public key" cryptography.

10.2 Encryption and Decryption

Encryption is the process of transforming information so it is unintelligible to anyone but the intended recipient. Decryption is the process of decoding encrypted information. A cryptographic algorithm, also called a cipher, is a mathematical function used for encryption or decryption. Usually, two related functions are used, one for encryption and the other for decryption.

With most modern cryptography, the ability to keep encrypted information secret is based not on the cryptographic algorithm, which is widely known, but on a number called a key that must be used with the algorithm to produce an encrypted result or to decrypt previously encrypted information. Decryption with the correct key is simple. Decryption without the correct key is very difficult, if not impossible**.**

10.2.1 Symmetric-key Encryption

With symmetric-key encryption, the encryption key can be calculated from the decryption key and vice versa. With most symmetric algorithms, the same key is used for both encryption and decryption, as shown in figure 10.1.

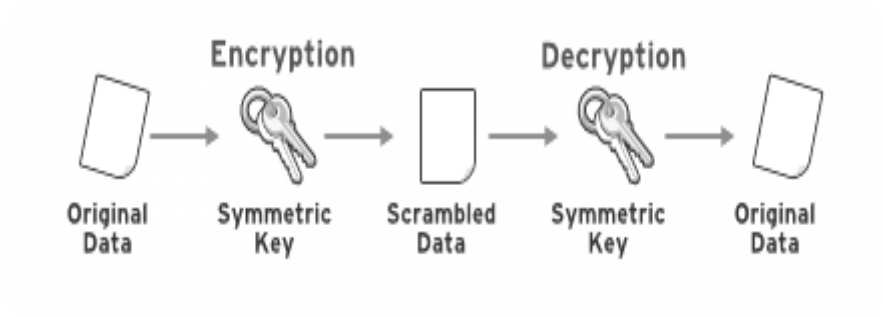


Figure 10.1: Symmetric-key encryption.

Implementations of symmetric-key encryption can be highly efficient, so that users do not experience any significant time delay as a result of the encryption and decryption. Symmetric-key encryption also provides a

degree of authentication, since information encrypted with one symmetric key cannot be decrypted with any other symmetric key. Thus, as long as the symmetric key is kept secret by the two parties using it to encrypt communications, each party can be sure that it is communicating with the other as long as the decrypted messages continue to make sense.

Symmetric-key encryption is effective only if the symmetric key is kept secret by the two parties involved. If anyone else discovers the key, it affects both confidentiality and authentication. A person with an unauthorized symmetric key not only can decrypt messages sent with that key, but can encrypt new messages and send them as if they came from one of the legitimate parties using the key.

Symmetric-key encryption plays an important role in SSL communication, which is widely used for authentication, tamper detection, and encryption over TCP/IP networks. SSL also uses techniques of public-key encryption.

10.2.2 Public-Key Encryption

Public-key encryption (also called asymmetric encryption) involves a pair of keys, a public key and a private key, associated with an entity. Each public key is published, and the corresponding private key is kept secret. Data encrypted with a public key can be decrypted only with the corresponding private key.

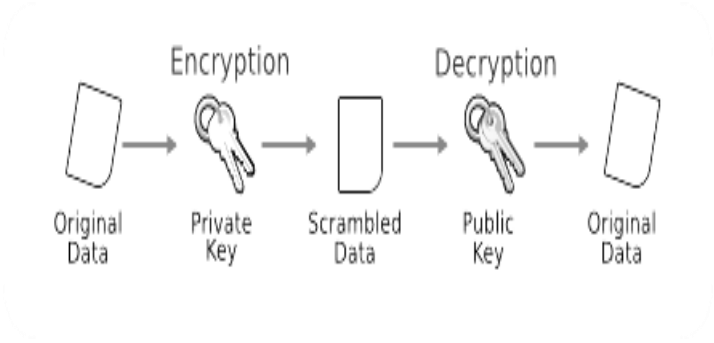


Figure 10.2: Public-key encryption.

The scheme shown in figure 10.2 “Public-Key Encryption” allows public keys to be freely distributed, while only authorized people are able to read data encrypted using this key. In general, to send encrypted data, the data is encrypted with that person’s public key, and the person receiving the encrypted data decrypts it with the corresponding private key. Compared with symmetric-key encryption, public-key encryption requires more processing and may not be feasible for encrypting and decrypting large amounts of data. However, it is possible to use public-key encryption to send a symmetric key, which can then be used to encrypt additional data. This is the approach used by the SSL/TLS protocols.

The reverse of the scheme shown in figure 10.2, “Public-Key Encryption” also works data encrypted with a private key can be decrypted only with the corresponding public key. This is not a recommended practice to encrypt sensitive data, however, because it means that anyone with the public key, which is by definition published, could decrypt the data. Nevertheless, private-key encryption is useful because it means the private key can be used to sign data with a digital signature, an important requirement for electronic commerce and other commercial applications of cryptography.

10.2.3 Key Length and Encryption Strength

Breaking an encryption algorithm is basically finding the key to the access the encrypted data in plain text. For symmetric algorithms, breaking the algorithm usually means trying to determine the key used to encrypt the text. For a public key algorithm, breaking the algorithm usually means acquiring the shared secret information between two recipients. One method of breaking a symmetric algorithm is to simply try every key within the full algorithm until the right key is found. For public key algorithms, since half of the key pair is publicly

known, the other half (private key) can be derived using published, though complex, mathematical calculations. Manually finding the key to break an algorithm is called a brute force attack.
Breaking an algorithm introduces the risk of intercepting, or even impersonating and fraudulently verifying, private information. The key strength of an algorithm is determined by finding the fastest method to break the algorithm and comparing it to a brute force attack.
For symmetric keys, encryption strength is often described in terms of the size or length of the keys used to perform the encryption longer keys generally provides stronger encryption. Key length is measured in bits. An encryption key is considered full strength if the best known attack to break the key is no faster than a brute force attempt to test every key possibility. Different types of algorithms particularly public key algorithms may require different key lengths to achieve the same level of encryption strength as a symmetric-key cipher. The RSA cipher can use only a subset of all possible values for a key of a given length, due to the nature of the mathematical problem on which it is based. Other ciphers, such as those used for symmetric-key encryption, can use all possible values for a key of a given length.

Caution
Changing the encryption algorithm may cause disruptions to the system.

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: Prepare a flowchart for the Public key encryption scheme.

……..……

……..……

……...

10.3 Cryptography Terminology

1. Encryption is the process of turning a clear-text message (Plaintext) into a data stream which looks like a meaningless and random sequence of bits (Cipher text). The process of turning ciphertext back into plaintext is called decryption.
2. Cryptography deals with making communications secure. Cryptoanalysis deals with breaking cipher text that is, recovering plaintext without knowing the key. Cryptology is a branch of mathematics which deals with both cryptography and cryptoanalysis.
3. A cryptographic algorithm, also known as a cipher, is a mathematical function which uses plaintext as the input and produces cipher text as the output and vice versa. All modern ciphers use keys together with plaintext as the input to produce cipher text. The same or a different key is supplied to the decryption function to recover plaintext from cipher text. The details of a cryptographic algorithm are usually made public. It is the key that the security of a modern cipher lies in, not the details of the cipher.
4. Symmetric algorithms use the same key for encryption and decryption. These algorithms require that both the sender and receiver agree on a key before they can exchange messages securely. Some symmetric algorithms operate on 1 bit (or sometimes 1 byte) of plaintext at a time. They are called stream ciphers. Other algorithms operate on blocks of bits at a time. They are called block ciphers. Most modern block ciphers use the block size of 64 bits.
5. Public-key algorithms (also known as asymmetric algorithms) use two different keys (a key pair) for encryption and decryption. The keys in a key pair are mathematically related, but it is computationally infeasible to deduce one key from the other. These algorithms are called “public-key” because the

encryption key can be made public. Anyone can use the public key to encrypt a message, but only the owner of the corresponding private key can decrypt it.

Some public-key algorithms such as RSA allow the process to work in the opposite direction as well: a message can be encrypted with a private key and decrypted with the corresponding public key.

Self Assessment Questions

1. Asymmetric encryption (Public Key Encryption) uses different keys for encryption and decryption.

(a). True (b). False

2. Encryption is the process used to convert .....into a form that is not readable without knowledge of the rules and key to decrypt.

(a). Cipher text (b). Plaintext

(c). Encrypt (d). None of these

3. In symmetric cryptography, the same key is used for both encryption and decryption by.....

(a). Sender (b). Receiver

(c). E-mail (d). None of these

4. Asymmetric cryptography or public-key cryptography is cryptography in which a pair of keys is used to .....a message so that it arrives securely.

(a). Encrypt (b). Decrypt

(c). Both of these (d). None of these

5. ……..allows the recipient of information to determine its origin by confirming the sender's identity

(a). Authentication (b). Authorized

(c). Recipient (d). None of these

10.4 Classification of Cryptography

There are two main classification of cryptography:

- Secret key cryptography
- Public key cryptography

In cryptographic systems, the term key refers to a numerical value used by an algorithm to alter information, making that information secure and visible only to individuals who have the corresponding key to recover the information.

Secret key cryptography is also known as symmetric key cryptography. With this type of cryptography, both the sender and the receiver know the same secret code, called the key. Messages are encrypted by the sender using the key and decrypted by the receiver using the same key. This method works well if you are communicating with only a limited number of people, but it becomes impractical to exchange secret keys with large numbers of people. In addition, there is also the problem of how you communicate the secret key securely. Public key cryptography, also called asymmetric encryption, uses a pair of keys for encryption and decryption. With public key cryptography, keys work in pairs of matched public and private keys.

The public key can be freely distributed without compromising the private key, which must be kept secret by its owner. Because these keys work only as a pair, encryption initiated with the public key can be decrypted only with the corresponding private key.

The major advantage asymmetric encryption offers over symmetric key cryptography is that senders and receivers do not have to communicate keys up front. Provided the private key is kept secret, confidential communication is possible using the public keys.

10.4.1 Types of Cipher

There are two basic types of encryption ciphers: substitution and transposition (permutation). The substitution cipher replaces bits, characters, or blocks of characters with different bits, characters, or blocks. The transposition cipher does not replace the original text with different text, but moves the original text around. It re arranges the bits, characters, or blocks of characters to hide the original meaning.

Substitution Cipher

A substitution cipher uses a key to know how the substitution should be carried out. In the caesar cipher, each letter is replaced with the letter three places beyond it in the alphabet. This is referred to as a shift alphabet.

If the caesar cipher is used with the english alphabet, when George wants to encrypt a message of "FBI," the encrypted message would be "IEL." Substitution is used in today's algorithms, but it is extremely complex compared to this example. Many different types of substitutions take place usually with more than one alphabet. This example is only meant to show you the concept of how a substitution cipher works in its most simplistic form.

Transposition Cipher

In a transposition cipher, permutation is used, meaning that letters are scrambled. The key determines the positions that the characters are moved to, as illustrated in figure 10.3. This is a simplistic example of a transposition cipher and only shows one way of performing transposition. When introduced with complex mathematical functions, transpositions can become quite sophisticated and difficult to break. Most ciphers used today use long sequences of complicated substitutions and permutations together on messages. The key value is inputted into the algorithm and the result is the sequence of operations (substitutions and permutations) that are performed on the plaintext.

Simple substitution and transposition ciphers are vulnerable to attacks that perform frequency analysis. In every language, there are words and patterns that are used more often than others. For instance, in the english language, the words "the," "and," "that," and "is" are very frequent patterns of letters used in messages and conversation. The beginning of messages usually starts "Hello" or "Dear" and ends with "Sincerely" or "Goodbye." These patterns help attackers figure out the transformation between plaintext to cipher text, which enables them to figure out the key that was used to perform the transformation. It is important for cryptosystems to not reveal these patterns.

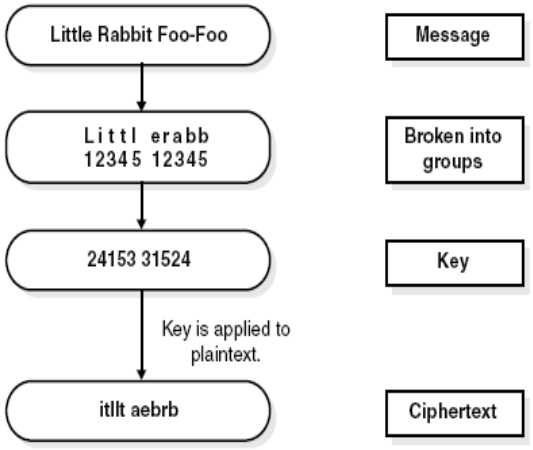


Figure 10.3: Transposition cipher.

More complex algorithms usually use more than one alphabet for substitution and permutation, which reduces the vulnerability to frequency analysis. The more complicated the algorithm, the more the resulting text (cipher text) differs from the plaintext; thus, the matching of these types of patterns becomes more difficult.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Prepare the diagram of transposition cipher.

……..……

……..……

……...

10.5 Security of Algorithms

Simply speaking, secure communication happens in two parts authentication and encryption. In authentication, two parties try to identify each other using digital certificates. The certificate is made of a digital signature generated using DSA and MD5 algorithms. While, RSA and DES are commonly used algorithms for encryption.

10.5.1 Authentication

DSA

This digital signature algorithm (DSA) is used for generating digital signatures in digital certificates. Only someone who has a public-private key pair can generate a digital signature.

A digital signature consists of two integers, called‘s’ (signature) and ‘r’ (verification), which are sent to the client for authentication. These integers are generated from several random integers.

First two prime integer numbers ‘p’ and ‘q’ are taken. Then two random integers ‘h’ and ‘k’ are selected from these. Here ‘h’ is in the range of 1 and p-1, while ‘k’ is a value greater than 0 and less than ‘q’. Subsequently, another value ‘g’ is calculated using ‘h’, ‘p’ and ‘q’. Finally, ‘r’ is calculated using ‘g’, ‘p’ and ‘q’.

For gene rating‘s’, first a random message ‘m’ is created. Then its hash is calculated using a hashing algorithm like MD5. Finally, ‘s’ is generated using ‘k’, the hashed message, private key, ‘r’ and ‘q’.

The digital signature along with ‘p’, ‘q’ and ‘g’ is sent to the client for verifying its identity. The hashing algorithm used, the message ‘m’ and the public key are also sent. On the client side the message ‘m’ is first subjected to the hashing algorithm. Then a value ‘v’ (called verifier) is calculated from this hashed message, ‘s’, ‘p’, ‘q’, and the public key. Now if ‘v’ is equal to ‘r’, then the digital signature is verified.

MD5

MD5 (Message Digest) is a hashing algorithm used in generating digital signatures. The output of MD5 is a message digest, which can be used to authenticate the owner of a private key.

The MD5 algorithm takes a message and checks whether it is size is 448-bits. If it is not, then it pads it with extra bits. Then it again takes the original message and converts it to 64 bits. These are then added to the 448 bits to give a block of 512 bits. This block is then broken into 32, 16-bit message blocks. A loop is started in which each of the 32 blocks is processed. Outside this loop, four separate 32-bit variables A, B, C, and D of standard values are taken. Then the values of these four variables A, B, C, D are copied to four different variables say a, b, c, and d. Next, within the loop new values are calculated for a, b, c, and d using the 16-bit blocks and the, b, c, and d values themselves. A different equation is used for each of these four variables. Now A, B, C, and D are incremented with the new values of a, b, c, and d.

Finally A, B, C, and D totaling to 128 bits (32x4) is the hash calculated, which is also called a message digest.

10.5.2 Encryption

Broadly speaking there are two encryption techniques symmetric and asymmetric used for secure communication? In symmetric encryption, the same key is used for both encryption as well as decryption. This is known as the private key. Consider two parties, A and B, wanting to engage in an encrypted communication. Party A generates a private key and sends its copy to party B. Hence both parties use this key to encrypt as well as decrypt messages.

In asymmetric encryption, party A generates a public-private key pair, and sends just the public key to party B. When B wants to send a secret message to A, it encrypts the message using A's public key. When A receives this encrypted message, it can only decrypt it with its corresponding private key. Similarly, the reverse can also happen. This procedure is also known as PKI or Public Key Infrastructure.

RSA

RSA, which is named after its developers (Rivest, Shamir, Adleman), is an asymmetric or public key algorithm. In this, the public-private key pair has a fixed length in bits, which can be decided at the time of their generation like 512, 768, 1,024, 2,096, with higher numbers corresponding to stronger encryption. When the public key is generated, it consists of the key size and a positive integer called public exponent, which has some typical standard values. The private key when generated includes these two along with a private exponent and two prime numbers. The two prime numbers are derived such that their product is equal to the key size. In RSA, key size is the same for both keys. The private exponent in the private key is calculated from the public exponent and the two prime numbers.

Once the keys have been generated, they are ready for encrypting or decrypting data or message. The number of bits in the message being encrypted must be less than or equal to the key size. If not, the message is broken into separate blocks and then encrypted. If the message size is smaller than the key size then some extra bits are padded to the message.

The encrypted message is created using the original message itself, public exponent, and the key size information in the public key. When the encrypted message is received on the other end, the private exponent and the key size is used to decrypt it. Since the private exponent is calculated using the public exponent, only the correct private key can decrypt the message. The encryption and decryption of the message requires a lot of exponential calculations. So RSA or as such public key encryption is slow.

DES

DES (Data Encryption Standard) is developed by IBM. It is a symmetric key encryption technique that encrypts messages in 64-bit chunks. Though the actual key size is 64-bits, it only uses 56 bits for encryption/decryption. The remaining 8 bits are used for checking whether the key has changed during its transmission either accidentally or intentionally.

Both the 56-bit key and the 64-bit data go through a process of permutation and transformation. The objective is to create sixteen; 48-bits sub keys using the 56-bit key and the 64-bit data in 16 loops. The following is the explanation of one loop.

The 56-bit key is first changed according to a key permutation table. Permutation tables change the bit positions. Then the changed key is divided into two 28-bit halves. The bits in each half are then shifted by two places. The shifting is done in all the rounds except the first, second, ninth, and the sixteenth round. Then from these two halves, a 48-bit key is chosen using a compression permutation table.

The 64-bit data is divided into two 32-bit halves called a left L and a right-half R. Now R is subjected to another permutation table called the expansion permutation table where each 32-bit block is expanded to 48 bits (by padding and repeating some bits).

After this, R is XORed (pronounced Exclusive OR, which is a digital gate function) with the 48-bit sub key generated from the 56-bit key. The result of this is fed to 8 permutation tables known as S-boxes. Each S-box accepts 6 bits (8x6=48) and generates a 4-bit output. The total output from the eight S-boxes is then combined, resulting in a 32-bit chunk. This 32-bit chunk is then fed to another permutation table called a P-box. The P-

box also produces a 32-bit chunk, which is then XORed with L. Finally if this is not the sixteenth round, L becomes R and vice versa. This swapping is called transformation. The 64-bit data undergoes 15 more such rounds for encryption. During decryption the opposite process is repeated. Since the algorithm involves just XORing and changes in bit positions, DES is relatively faster.

Caution
Be sure to remember the encryption code you entered. If you need to enter the encryption code again for some reason and you do not enter the same encryption code, all the data stored on the hard disk will be overwritten as a security precaution.

10.6 Steganography

The objective of steganography is to hide a secret message within a cover-media in such a way that others cannot discern the presence of the hidden message. Technically in simple words "steganography means hiding one piece of data within another". Modern steganography uses the opportunity of hiding information into digital multimedia files and also at the network packet level.
Hiding information into a media requires following elements.

- The cover media(C) that will hold the hidden data
- The secret message (M), may be plain text, cipher text or any type of data
- The stego function (Fe) and its inverse (Fe^{-1})
- An optional stego-key (K) or password may be used to hide and unhide the Message

The stego function operates over cover media and the message (to be hidden) along with a stego-key (optionally) to produce a stego media (S). The schematic of steganographic operation is shown in Figure 10.4. Steganography and cryptography are great partners in spite of functional difference. It is common practice to use cryptography with steganography.

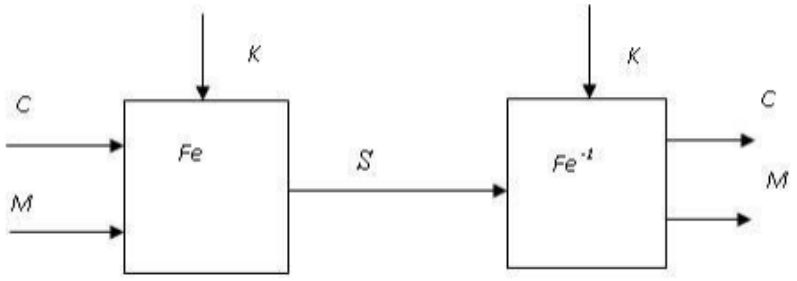


Figure 10.4: The Steganographic Operation.

Modern Techniques of Steganography
The common modern technique of steganography exploits the property of the media itself to convey a message. The following media are the candidate for digitally embedding message.

- Plaintext
- Still imagery
- Audio and Video
- IP datagram.

Did You Know?
Modern steganography entered the world in 1985 with the advent of the personal computer being applied to classical steganography problems.

10.7 Steganography versus Cryptography

Basically, the purpose of cryptography and steganography is to provide secret communication. However, steganography is not the same as cryptography. Cryptography hides the contents of a secret message from a malicious people, whereas steganography even conceals the existence of the message. Steganography must not be confused with cryptography, where we transform the message so as to make it meaning obscure to a malicious people who intercept it. Therefore, the definition of breaking the system is different. In cryptography, the system is broken when the attacker can read the secret message. Breaking a steganographic system need the attacker to detect that steganography has been used and he is able to read the embedded message. In cryptography, the structure of a message is scrambled to make it meaningless and unintelligible unless the decryption key is available. It makes no attempt to disguise or hide the encoded message. Basically, cryptography offers the ability of transmitting information between persons in a way that prevents a third party from reading it. Cryptography can also provide authentication for verifying the identity of someone or something in contras, steganography does not alter the structure of the secret message, but hides it inside a cover-image so it cannot be seen. A message in cipher text, for instance, might arouse suspicion on the part of the recipient while and "invisible" message created with steganographic methods will not. In other word, steganography prevents an unintended recipient from suspecting that the data exists. In addition, the security of classical steganography system relies on secrecy of the data encoding system. Once the encoding system is known, the steganography system is defeated.

It is possible to combine the techniques by encrypting message using cryptography and then hiding the encrypted message using steganography. The resulting stego-image can be transmitted without revealing that secret information is being exchanged. Furthermore, even if an attacker were to defeat the steganographic technique and detect the message from the stego-object he would still require the cryptographic decoding key to decipher the encrypted message.

Table 10.1: Advantages and Disadvantages comparison

Steganography	Cryptography
Unknown message passing. Little known technology. Technology still being developed for certain formats. Once detected message is known Many carrier formats.	Known message passing. Common technology. Most algorithms known to government departments. Strong algorithm are currently resistant to brute force attack. Large expensive computing power required for cracking technology increase reduces strength.

10.8 Comparison of Symmetric and Asymmetric Key Cryptography

Symmetric key encryption is the older method of encryption and asymmetric key encryption was developed to make up for its shortcomings. Yet, each of them still has advantages and disadvantages that make them appeal to different users and uses.

Methods of encrypting messages include the

- Symmetric Key Encryption and
- Asymmetric or Public Key Encryption methods.

10.8.1 Symmetric Key Encryption

Symmetric key encryption is also known as shared-key, single-key, secret-key, and private-key or one-key encryption. In this type of message encryption, both sender and receiver share the same key which is used to both encrypt and decrypt messages. Sender and receiver only have to specify the shared key in the beginning and then they can begin to encrypt and decrypt messages between them using that key. Examples include AES (Advanced Encryption Standard) and Triple DES (Data Encryption Standard).

Advantages

Simple: This type of encryption is easy to carry out. All users have to do is specify and share the secret key and then begin to encrypt and decrypt messages.

Encrypt and Decrypt your own Files: If you use encryption for messages or files which you alone intend to access, there is no need to create different keys. Single-key encryption is best for this.

Fast: Symmetric key encryption is much faster than asymmetric key encryption.

Uses less Computer Resources: Single-key encryption does not require a lot of computer resources when compared to public key encryption.

Prevents Widespread Message Security Compromise: A different secret key is used for communication with every different party. If a key is compromised, only the messages between a particular pair of sender and receiver are affected. Communications with other people are still secure.

Disadvantages

Need for Secure Channel for Secret Key Exchange: Sharing the secret key in the beginning is a problem in symmetric key encryption. It has to be exchanged in a way that ensures it remains secret.

Too Many Keys: A new shared key has to be generated for communication with every different party. This creates a problem with managing and ensuring the security of all these keys.

Origin and Authenticity of Message Cannot be Guaranteed: Since both sender and receiver use the same key, messages cannot be verified to have come from a particular user. This may be a problem if there is a dispute.

10.8.2 Asymmetric/Public Key Encryption

Also known as public key encryption, this method of encrypting messages makes use of two keys a public key and a private key. The public key is made publicly available and is used to encrypt messages by anyone who wishes to send a message to the person that the key belongs to. The private key is kept secret and is used to decrypt received messages. An example of asymmetric key encryption system is RSA.

Advantages

Convenience: It solves the problem of distributing the key for encryption. Everyone publishes their public keys and private keys are kept secret.

Provides for Message Authentication: Public key encryption allows the use of digital signatures which enables the recipient of a message to verify that the message is truly from a particular sender.

Detection of Tampering: The use of digital signatures in public key encryption allows the receiver to detect if the message was altered in transit. A digitally signed message cannot be modified without invalidating the signature.

Provide for Non-Repudiation: Digitally signing a message is akin to physically signing a document. It is an acknowledgement of the message and thus, the sender cannot deny it.

Disadvantages

Public Keys Should/Must be Authenticated: No one can be absolutely sure that a public key belongs to the person it specifies and so everyone must verify that their public keys belong to them.

Slow: Public key encryption is slow compared to symmetric encryption. Not feasible for use in decrypting bulk messages.

Uses up More Computer Resources: It requires a lot more computer supplies compared to single-key encryption.

Widespread Security Compromise is Possible: If an attacker determines a person's private key, his or her entire messages can be read.
Loss of Private Key Maybe Irreparable: The loss of a private key means that all received messages cannot be decrypted. As for which of them is more secure, there are differing views. Some experts consider symmetric key encryption to be more secure and others think the public key encryption method is better.

Did You Know?
An asymmetric-key cryptosystem was developed in 1976 by Whitfield Diffie and Martin Hellman.

10.9 Public-Key Cryptography

Public-key cryptography and related standards underlie the security features of many products such as signed and encrypted email, single sign-on, and Secure Sockets Layer (SSL) communications.
Internet traffic, which passes information through intermediate computers, can be intercepted by a third party.
Eavesdropping: Information remains intact, but its privacy is compromised. For example, someone could gather credit card numbers, record a sensitive conversation, or intercept classified information.
Tampering: Information in transit is changed or replaced and then sent to the recipient. For example, someone could alter an order for goods or change a person's resume.
Impersonation: Information passes to a person who poses as the intended recipient. Impersonation can take two forms.

1. Spoofing a person can pretend to be someone else. For example, a person can pretend to have the email address. Example.net or a computer can falsely identify itself as a site called www.example.net.
2. Misrepresentation a person or organization can misrepresent itself. For example, a site called www.example.net can purport to be an on-line furniture store when it really receives credit-card payments but never sends any goods.

The two main branches of public key cryptography are.

1. ***Public key encryption:*** A message encrypted with a recipient's public key cannot be decrypted by anyone except a possessor of the matching private key it is presumed that this will be the owner of that key and the person associated with the public key used.
2. Digital signatures a message signed with a sender's private key can be verified by anyone who has access to the sender's public key, thereby proving that the sender had access to the private key (and therefore is likely to be the person associated with the public key used), and the part of the message that has not been tampered with.

10.10 Summary

- Encryption is the process used to convert plaintext into a form that is not readable without knowledge of the rules and key to decrypt.
- Decryption is the process used to convert from an encrypted form back to plaintext, using a key.
- A code book uses a series of code values to use as replacements. In traditional code book models, a message is converted by mapping from alphabet letters to numbers or other letters, according to a pattern in the code book.
- Symmetric encryption (private key encryption) is a type of encryption where the same secret key is used to encrypt and decrypt information or there is a simple transform between the two keys.
- Asymmetric encryption (Public Key Encryption) uses different keys for encryption and decryption. The encryption key is public so that anyone can encrypt a message.

10.11 Keywords

Digital Signatures: A message signed with a sender's private key can be verified by anyone who has access to the sender's public key, thereby proving that the sender had access to the private key, and the part of the message that has not been tampered with.

Public-key: Systems that use two keys, a public key known to everyone and a private key that only the recipient of messages uses.

Public-key Cryptography: Refers to a cryptographic system requiring two separate keys, one to lock or encrypt the plaintext, and one to unlock or decrypt the cyphertext.

Recipient decrypts: The received message using their own secret key, identifies the sender from their now-cleartext signature, and then decrypts the result using the sender's public key.

Symmetric-key Algorithms: A public key algorithm does not require a secure initial exchange of one, or more, secret keys between the sender and receiver.

10.12 Review Questions

6. What is encryption and decryption?
7. Explain the symmetric and asymmetric cryptography in detail.
8. What is the cryptography terminology?
9. Differentiate between symmetric-key and asymmetric-key encryption.
10. What is the classification of cryptography?
11. What are the substitution ciphers?
12. What is the security of algorithms? Explain with example.
13. What is the difference between steganography and cryptography?
14. What is the public key cryptanalysis?
15. Define steganography.

Answers for Self Assessment Questions

1. (a) 2. (b) 3. (d) 4. (c) 5. (a)

10.13 Further Studies

1. http://books.google.co.in/books?id=f24wFELSzkoC&pg=PA31&dq=Encryption+and+Decryption&hl=en&sa=X&ei=LX3CUsbMDIqjigeuioHYDA&ved=0CC8Q6AEwAA#v=onepage&q=Encryption%20and%20Decryption&f=false
2. Computer Networks: A Systems Approach by Larry L. Peterson, Bruce S. Davie

11

Digital Signature

STRUCTURE

11.0 Objectives

11.1 Introduction

11.2 Concept of Digital Signature

11.3 Requirements of Digital Signature

11.4 Types of Digital Signature

11.5 Authentication Protocols

11.6 Symmetric Encryption Approach

11.7 Public-Key Encryption Approach

11.8 Digital Signature Standard

11.9 RSA and Digital Signature

11.10 DSS Approach

11.11 Summary

11.12 Keywords

11.13 Review Questions

11.14 Further Studies

11.0 Objectives

After studying this chapter, you will be able to:

- Understand the concept of digital signature
- Discuss the requirements of digital signature
- Explain the types of digital signature
- Discuss the authentication protocol
- Explain the symmetric encryption approach
- Discuss the public-key encryption approach

11.1 Introduction

A digital signature authenticates electronic documents in a similar manner a handwritten signature authenticates printed documents. This signature cannot be forged and it asserts that a named person wrote or

otherwise agreed to the document to which the signature is attached. The recipient of a digitally signed message can verify that the message originated from the person whose signature is attached to the document and that the message has not been altered either intentionally or accidentally since it was signed. Also, the signer of a document cannot later disown it by claiming that the signature was forged. In other words, digital signatures enable the "authentication" and "non-repudiation" of digital messages, assuring the recipient of a digital message of both the identity of the sender and the integrity of the message.

11.2 Concept of Digital Signature

A digital signature is basically a way to ensure that an electronic document (e-mail, spreadsheet, text file, etc.) is authentic. Authentic means that you know who created the document and you know that it has not been altered in any way since that person created it.

Digital signatures rely on certain types of encryption to ensure authentication. Encryption is the process of taking all the data that one computer is sending to another and encoding it into a form that only the other computer will be able to decode. Authentication is the process of verifying that information is coming from a trusted source. These two processes work hand in hand for digital signatures.

There are several ways to authenticate a person or information on a computer:

Password: The use of a user name and password provide the most common form of authentication. You enter your name and password when prompted by the computer. It checks the pair against a secure file to confirm. If either the name or password does not match, then you are not allowed further access.

Checksum: Probably one of the oldest methods of ensuring that data is correct, checksums also provide a form of authentication since an invalid checksum suggests that the data has been compromised in some fashion. A checksum is determined in one of two ways. Let's say the checksum of a packet is 1 byte long, which means it can have a maximum value of 255. If the sum of the other bytes in the packet is 255 or less, then the checksum contains that exact value. However, if the sum of the other bytes is more than 255, then the checksum is the remainder of the total value after it has been divided by 256. Figure 11.1 shows the example:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Total	Checksum
212	232	54	135	244	15	179	80	1151	127

Figure 11.1: 1151 divided by 256 equals 4.496 (round to 4) multiply 4 X 256 which equals 10241151 minus 1024 equals 127

Did You Know?

In 1976, Whitfield Diffie and Martin Hellman first described the notion of a digital signature scheme, although they only conjectured that such schemes existed.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Prepare the structure for the working of digital signature for data security.

……..……

……..……

……...

11.3 Requirements of Digital Signature

These requirements involve the following:

- The relationship of the digital signature system to the PKI.
- Cryptographic modules, security and cost optimization.
- The signing and verifying process.
- Storage of required digital signature data.
- Performance.

The Relationship of the Digital Signature System to the PKI

- The digital signature system must be interoperable with PKI-generated certificates and certificate revocation status methods.
- The digital signature system must be interoperable with the PKI directory service.
- The digital signature system must be able to use cryptographic tokens and PKCS12 files initialized by the PKI.

Cryptographic Modules, Security and Cost Optimization

- The digital signature system must be able to use both software and hardware cryptographic tokens.
- The digital signature system must be able to enforce the use of higher security cryptographic modules for higher risk transactions while allowing less expensive cryptographic modules to be used for lower risk transactions.
- The digital signature system must provide an interface that allows the use of third-party security products.

The Signing and Verifying Process

- The digital signature system must provide a method for specifying which data to include in the data to be signed.
- The digital signature system must provide a method for modifying the data to include in the data to be signed without violating the integrity of existing signatures.
- The digital signature system must protect against database object spoofing.
- The digital signature system must allow the representation of data element to be specified.

Storage of Required Digital Signature Information

- The digital signature system should store information required to verify a signature in a normalized fashion that does not require the redundant storage of any required data elements.
- The digital signature system should store only one copy of the user's certificate in the database and link this one copy to each signed record that requires it.
- When a digital signature is verified, the data in the database must be used as a basis for this verification.

Performance

- The digital signature system should minimize network traffic.

11.4 Types of Digital Signature

11.4.1 Direct Digital Signature

Direct digital signing is when only the sender and recipient are involved. Before communication can start we have to presume that the recipient knows the senders public key. An easy digital signature can be set by encrypting the message with sender's private key or encrypting the message hash code with the sender's private key.

If the parties want to be confidential then the sender can also encrypt the whole message plus previous signature with receiver's public key or some shared secret key. These two kinds of additional measures are called public-key encryption and symmetric encryption respectively.
Direct digital signing has its weakness because there are no third parties that can arbitrate over a dispute. For example a user P can always say his private key was stolen and that he did not send that message. By introducing a third authoritative party P would be obliged to timely inform of a private key theft. Therefore we continue to arbitrate digital signing.

11.4.2 Arbitrated Digital Signature

By using an arbiter many problems from direct digital signing can be solved. The main requirement for the arbitrated digital signing is that both parties have high trust in the arbiter. The process of signing using the arbiter goes like this:

$$P \longrightarrow A \quad M||E_{K_{pa}}[ID_p||H(M)]$$

Where
P = our first partie
A = Arbiter
M = Message
Here P sends a message containing: the message, [the hash code of the message and his identifier] all encrypted with a secret key K_{pa} that only P and A share (the part in the brackets is the signature). This way the arbiter can decrypt the signature with the secret key that they share and confirm that the message (checks the hash value and compares) and the user P (if he was able to decrypt the signature with the key provided for that user) are correct and not have been tampered with.

$$A \longrightarrow Q \; E_{K_{aq}}[ID_p||M||E_{K_{pa}}[ID_p||H(M)]||T]$$

Here A sends a message to Q that is encrypted with K_{aq} that only Q and A know that includes: identifier of P, the message, the signature ($E_{K_{pa}}[ID_p||H(M)]$) and a timestamp. Timestamp is here to inform Q of a timely transfer and not an old reply. Now Q can store the message and in any further dispute it can use it to prove it came from P by sending back the same message just without the timestamp. Then A can decrypt it by using K_{aq} and check the signature if it really came from P.

In this dialogue both parties as we said need to have high trust in the arbiter in order to function. If this is true then both sides can be safe in knowing that their signature cannot be forged or from the other party.

11.5 Authentication Protocols

Authentication refers to verifying the identity of a user or computer. When a user logs on to the network, whether on a LAN or through a remote access connection, it will need to provide a username and password, a smartcard, certificate, or other means of proving that it is who it claims it is. Several authentication protocols are developed to allow a secure exchange of authentication information over network connections and are described in the following paragraphs.

11.5.1 CHAP

The Challenge Handshake Authentication Protocol (CHAP) is an authentication protocol that is primarily used for remote access PPP connections. CHAP is the successor of the Plain Authentication Protocol (PAP), which transmits the username and password in clear text over the network media. CHAP uses a more secure method; when a client logs on, the server sends a challenge request to the client, the client replies with a challenge

response that is a hashed (one-way encrypted) value based on the username/password-combination and a random number. The server performs the same encryption and if the resulting value matches the response from the client, the client is authenticated. The actual password is not transmitted across the network.

11.5.2 MS-CHAP

MS-CHAP is the Microsoft version of CHAP and provides the technology from CHAP combined with Microsoft authentication and encryption. MS-CHAP is available on Microsoft Windows 95, NT, 2000 and later versions. Windows 2003 includes MS-CHAP v2, which provides stronger security for the handshake process and mutual authentication. The latter means the client authenticates itself to the server, and the server authenticates itself to the client. While CHAP requires the password to be stored in plain text on the authentication server, an MS-CHAP password can be the user's Windows password stored on a domain controller. This allows centralized management of the username and password and offers a 'single sign-on' to connect to the remote access server and access resources in the remote network.

11.5.3 EAP (Extensible Authentication Protocol)

The Extensible Authentication Protocol (EAP) was created as an extension to PPP to provide an interface for different authentication methods. Nowadays EAP is also commonly used with the 802.1x Data Link layer authentication protocol. Instead of choosing PAP, CHAP, or MS-CHAP for example, the client and server agree to use EAP as the authentication protocol. The actual authentication method used by EAP varies a lot, and new methods can be developed and implemented without focusing on the underlying remote access technology.

11.5.4 RADIUS (Remote Authentication Dial-In User Service)

The Remote Authentication Dial-In User Service (RADIUS) provides authentication to clients that connect to a remote access server by using a SLIP or PPP dialup connection and an authentication protocol such as PAP, CHAP, or EAP. It allows a Network Access Server (NAS), which can be a remote access server, router, or wireless access point for example, to delegate the task of authenticating clients to a centralized RADIUS server. When a user dials in to a remote access server, the remote access server acts as a RADIUS client and forwards the access request to the RADIUS server. The RADIUS server is usually a service running on a Windows or UNIX server and uses its local user database or contacts another server, such as a Windows domain controller or LDAP directory, to authenticate the client's logon information. If the remote client is successfully authenticated, the RADIUS server replies to the RADIUS client (the NAS), which in turn accepts the connection of the remote client. Figure 11.2 shows the RADIUS server.

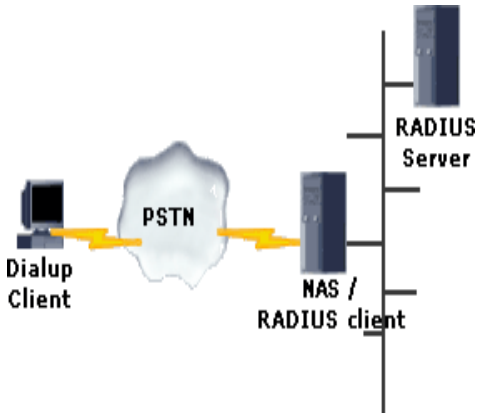


Figure 11.2: RADIUS

11.5.5 Kerberos

Kerberos is a very popular and advanced authentication protocol developed by MIT. Version 4 still runs in many networks, but V5 is considered to be standard Kerberos. It is the default authentication protocol in Windows 2000/2003 environments. In a Kerberos environment, a centralized authorization server called the Key Distribution Center (KDC) issues a ticket to a client when it successfully logs on to the network. This ticket is used to grant the client (system or user) access to network resources such as shares, printers, intranet applications, databases; anything that support Kerberos. The main advantage is that Kerberos provides single sign-on functionality for users in large heterogeneous network environments. Once users are authenticated by the KDC, a Windows 2003 domain controller for example, they will automatically be authorized when they try to access another network resource, without having to enter a username and password again and again.

Self Assessment Questions

1. Hashed message is signed by a sender using ……

(a) His public key (b) His private key

(c) Receiver's public key (d) Receiver's private key

2. A hashing function for digital signature

(i) Must give a hashed message which is shorter than the original message

(ii) Must be hardware implementable

(iii) Two different messages should not give the same hashed message

(iv) Is not essential for implementing digital signature

(a) i and ii (b) ii and iii

(c) i and ii1 (d) iii and iv

3. A digital signature is required

(i) To tie an electronic message to the sender's identity

(ii) For non repudiation of communication by a sender

(iii) To prove that a message was sent by the sender in a court of law

(iv) In all e-mail transactions

(a) i and ii (b) i, ii, iii

(c) i, ii, iii, iv (d) . ii, iii, iv

4. A digital signature is .....

(a) Bit string giving identity of a correspondent.

(b) A unique identification of a sender.

(c) An authentication of an electronic record by tying it uniquely to a key only a sender knows.

(d) An encrypted signature of a sender.

11.6 Symmetric Encryption Approach

Symmetric encryption is the oldest cryptographic technique used to transmit or store digital data securely.

Data that has been encrypted with symmetric encryption is considered to be confidential, in the sense that only those entities (persons or systems) who have the key can understand what the data means.

Cryptography is a security-related technology, but it works differently to others. Rather than preventing unauthorized entities from accessing the information, cryptography prevents them from understanding what the data means. Notice that encrypted data is still data, but without meaning in relation to its purpose. So, the encryption process transforms data into other data that has no meaning.

With symmetric encryption, confidentiality is guaranteed by the use of a secret key. With this key, and a mathematical algorithm, data is converted to an unintelligible form. By using the same key, and applying the mathematical algorithm in the reverse way, the original data can be reconstituted. The algorithm is said to be a two-way algorithm.

Symmetric algorithms can be divided into two main classes: stream algorithms and block algorithms. Stream Algorithms operate directly on a stream of bytes whereas block algorithms operate by transforming fixed-length groups of bits, a block, at a time Block algorithms are most commonly used in the IT world today.

At this stage, it is important to notice that identical blocks of data are transformed in the same manner inside the blocks sequence. This can be a security issue. Think about an image file. This is composed of blocks of data that, depending on the image format, might set the colour of a pixel on the screen. Suppose you want to encrypt this image. Suppose, also, that you happen to set, as block unit for the image, the same length of block that specifies a pixel colour. By encrypting it, the "pixel-block" will be encrypted into a different block of data. Remember what we say in the introduction. Encrypted data is again data, but with a different meaning. The encryption transforms a block of bytes into a different block of bytes and so a colour would, in those circumstances, become a different colour. If the identical blocks are encrypted in the same manner, we obtain an image with colours changed. The result is that the image is still intelligible to the user that sees it.

Caution

When data is not available block by block (as data that comes from a network stream) and the encryption system should retain data in memory until all blocks arrive. This could be dangerous from a security point of view.

11.7 Public-Key Encryption Approach

Public key encryption relies on key pairs. Digital signatures rely on the property of public key encryption that allows data encrypted with an entity's secret key to be decrypted with the public key of the pair. The sender calculates a secure hash on the data to be signed and then encrypts the result using a secret key. The recipient calculates the same hash and then decrypts the encrypted value attached by the sender. If the two values match, the recipient knows that the owner of the public key was the entity that signed the message and that the message was not modified during transmission.

The RSA public key encryption algorithm can be used for digital signatures: The signing entity creates a hash of the data to be signed and then encrypts that hash with its own secret key. The certifying entity then calculates the same hash on the data being received, decrypts the signature using the signing entity's public key, and compares the two values. If the hash is the same as the decrypted signature, then the data is certified. Digital signatures carry with them several implications.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Prepare the structure of the working of public key for encryption.

……..……

……..……

……...

11.8 Digital Signature Standard

DSS was put forth by the National Institute of Standards and Technology (NIST) in 1994, and has become the United States government standard for authentication of electronic documents. DSS is specified in Federal Information Processing Standard (FIPS) 186.

DSA is a pair of large numbers that are computed according to the specified algorithm within parameters that enable the authentication of the signatory, and as a consequence, the integrity of the data attached. Digital signatures are generated through DSA, as well as verified. Signatures are generated in conjunction with the use of a private key; verification takes place in reference to a corresponding public key. Each signatory has their own paired public (assumed to be known to the general public) and private (known only to the user) keys. Because a signature can only be generated by an authorized person using their private key, the corresponding public key can be used by anyone to verify the signature.

A data summary of the information (called a message digest) is created through the use of a hash function (called the Secure Hash Standard, or SHS, and specified in FIPS 180). The data summary is used in conjunction with the DSA algorithm to create the digital signature that is sent with the message. Signature verification involves the use of the same hash function.

11.8.1 The Digital Signature Algorithm

This Standard specifies a Digital Signature Algorithm (DSA) appropriate for applications requiring a digital rather than written signature. The DSA digital signature is a pair of large numbers represented in a computer as strings of binary digits. The digital signature is computed using a set of rules (i.e., the DSA) and a set of parameters such that the identity of the signatory and integrity of the data can be verified. The DSA provides the capability to generate and verify signatures. Signature generation makes use of a private key to generate a digital signature. Signature verification makes use of a public key which corresponds to, but is not the same as, the private key. Each user possesses a private and public key pair. Public keys are assumed to be known to the public in general. Private keys are never shared. Anyone can verify the signature of a user by employing that user's public key. Signature generation can be performed only by the possessor of the user's private key.

A hash function is used in the signature generation process to obtain a condensed version of data, called a message digest the message digest is then input to the DSA to generate the digital signature. The digital signature is sent to the intended verifier along with the signed data (often called the message). The verifier of the message and signature verifies the signature by using the sender's public key. The same hash function must also be used in the verification process. The hash function is specified in a separate standard, the Secure Hash Standard (SHS), FIPS 180. Similar procedures may be used to generate and verify signatures for stored as well as transmitted data.

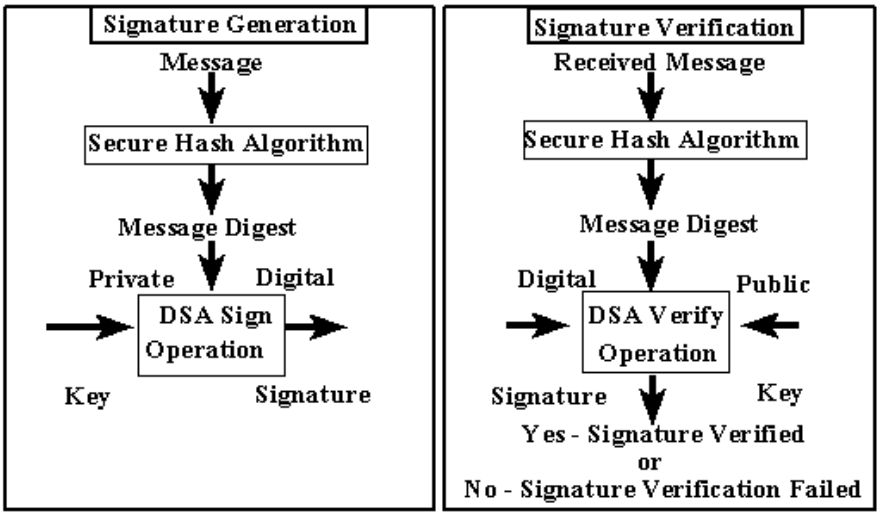


Figure 11.3: Using the SHA with the DSA

11.8.2 Use of the DSA Algorithm

The DSA is used by a signatory to generate a digital signature on data and by a verifier to verify the authenticity of the signature. Each signatory has a public and private key. The private key is used in the signature generation process and the public key is used in the signature verification process. For signature generation and verification, the data which is referred to as a message, M, is reduced by means of the Secure Hash Algorithm (SHA) specified in FIPS YY. An adversary, who does not know the private key of the signatory, cannot generate the correct signature of the signatory. In other words, signatures cannot be forged. However, by using the signatory's public key, anyone can verify a correctly signed message.

A means of associating public and private key pairs to the corresponding users is required. That is, there must be a binding of a user's identity and the user's public key. This binding may be certified by a mutually trusted party. For example, a certifying authority could sign credentials containing a user's public key and identity to form a certificate. Systems for certifying credentials and distributing certificates are beyond the scope of this standard. NIST intends to publish separate document(s) on certifying credentials and distributing certificates.

11.8.3 DSA Parameters

The DSA makes use of the following parameters:

1. p = a prime modulus, where $2^{L-1} < p < 2^{L}$ for 512 = < L = <1024 and L a multiple of 64
2. q = a prime divisor of p - 1, where $2^{159} < q < 2^{160}$
3. $g = h^{(p-1)/q}$ mod p, where h is any integer with $1 < h < p - 1$ such that $h^{(p-1)/q}$ mod $p > 1$(g has order q mod p)
4. x = a randomly or pseudo randomly generated integer with $0 < x < q$
5. $y = g^{x}$mod p
6. k = a randomly or pseudo randomly generated integer with $0 < k < q$

The integer's p, q, and g can be public and can be common to a group of users. A user's private and public keys are x and y, respectively. They are normally fixed for a period of time. Parameters x and k are used for signature generation only, and must be kept secret. Parameter k must be regenerated for each signature.
Parameters p and q shall be generated, or using other FIPS approved security methods. Parameters x and k shall be generated, or using other FIPS approved security methods.

Did You Know?

Digital Signature Standard (DSS) is the digital signature algorithm (DSA) developed by the U.S. National Security Agency (NSA) to generate a digital signature for the authentication of electronic documents.

11.9 RSA and Digital Signature

The notion of digital signatures goes back to the beginning of public-key cryptography. Whitfield Diffie and Martin Hellman introduced the idea that someone could form a digital signature using public-key cryptography that anyone else could verify but which no one else could generate.

While Diffie and Hellman provided a general model for digital signatures of any kind, the method developed by Rivest, Shamir, and Adleman in 1977, known as "RSA," has become the most proven and most popular, and achieved the widest adoption by standards bodies and in practice. Two other methods, discrete logarithm cryptography (including the Digital Signature Algorithm and the Diffie-Hellman key agreement method) and elliptic curve cryptography have also been embodied in several standards, but neither has yet been as widely adopted in practice as RSA.

The RSA digital signature scheme applies the sender's private key to a message to generate a signature. The signature can then be verified by applying the corresponding public key to the message and the signature through the verification process, providing either a valid or invalid result. These two operations — *sign* and v*erify* — comprise the RSA digital signature scheme.
Any signature generated by the first operation will always verify correctly with the second operation if the corresponding public key is used. If the signature was generated differently or if the message was altered after being signed, then the chances of the second operation verifying correctly are extremely small; with typical parameters, the chance is roughly 1 in 2160 or essentially zero. Although there are better ways to forge a signature than just guessing, the use of a sufficiently large key ensures security by making it computationally impractical to do so. For instance, it has been estimated to take thousands or even millions of years to break a given 1024-bit key (find the private key, given the public key), depending on the amount of computing power applied.

Taking a closer look at the signature generation portion of the process, the first step in generating an RSA signature is applying a cryptographic hash function to the message. The hash function is specifically designed to reduce a message of any length to a short number, called the "hash value" (typically 160 bits long), and to do it in a way such that two conditions are satisfied:

- It is difficult to find a message with a specific hash value.
- It is difficult to find two messages with the same hash value (an easier problem to solve).

Caution
The care should be taken while generating encryption/decryption key, it must be unique due to security reason.

11.10 DSS Approach

The authenticity of many legal, financial, and other documents is determined by the presence or absence of an authorized handwritten signature and photocopies do not count. For computerized message systems to replace the physical transport of paper and ink documents, a solution must be found to these problems.
The problem of devising a replacement for handwritten signatures is a difficult one. Basically, what is needed is a system by which one party can send a "signed". Message to another party in such a way that

1. The receiver can verify the claimed identity of the sender.
2. The sender- cannot later repudiate the contents of the message.
3. The receiver cannot possibly have concocted the message himself.

The first requirement is needed, for example, in financial systems. When a customer's computer orders a bank's computer to buy a ton of gold, the bank's Computer needs to be able to make sure that the computer giving the order really belongs to the company whose account is to be debited.

11.10.1 Secret-Key Signatures

One approach to digital signatures is to have a central authority that knows everything and whom everyone trusts, say Big Brother (BB). Each user then chooses a secret key and carries it by hand to BB's office. Thus only Alice and BB know Alice's secret, K_A, and so on.
When Alice wants to send a signed plaintext message, P, to her banker, Bob, she generates K_A (B, R_A, t, P) and sends it as depicted in Figure 11.4. BB sees that the message is from Alice, decrypts it, and sends a message to Bob as shown. The message to Bob contains the plaintext of Alice's message and also the signed message K_{BB}(A, t, P), where t is a timestamp. Bob now carries out Alice's request.

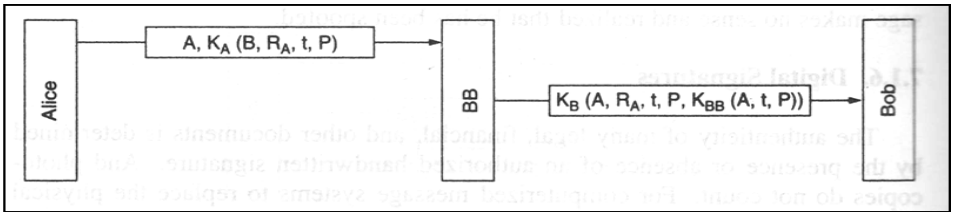


Figure 11.4: Digital signatures with Big Brother.

What happens if Alice later denies sending the message? Step 1 is that everyone sues everyone. Finally, when the case comes so court and Alice vigorously- denies sending Bob the disputed message, the judge will ask Bob how he can be sure that the disputed message came from Alice and not from Trudy. Bob first points out that BB will not accept a message from Alice unless it is encrypted with K_A, so there is no possibility of Trudy sending BB a false message from Alice.

Bob then dramatically produces Exhibit A, K_{BB} (A, t, P). Bob says that this is a message signed by BB which proves Alice sent P to Bob. The judge then asks BB (whom everyone trusts) to decrypt Exhibit A. When BB testifies that Bob is telling the truth, the judge decides in favour of Bob. Case dismissed.

One potential problem with the signature protocol of Figure11.4 is Trudy replaying either message. To minimize this problem, timestamps are used throughout. Furthermore, Bob can check all recent messages to see if RA was used in any of them. If so, the message is discarded as a replay. Note that Bob will reject very old messages based on the timestamp. To guard against instant replay attacks, Bob just checks the RA of every incoming message to see if such a message has been received from Alice in the past hour: If not, Bob can safely assume this is a new request.

11.10.2 Public-Key Signatures

A structural problem with using secret key cryptography for digital signatures is that everyone has to agree to trust Big Brother. Furthermore, Big Brother gets to read all signed messages. The most logical candidates for running the Big Brother server are the government, the banks, or the lawyers. These organizations do not inspire total confidence in all citizens: Hence, it would be nice if signing documents did not require a trusted authority.

Fortunately, public-key cryptography can make an important contribution here. Let us assume that the public-key encryption and decryption algorithms have the property that E(D(P)) = P in addition to the usual property that D(E(P)) = P. (RSA has this property, so the assumption is not unreasonable.) Assuming that this is the case, Alice can send a signed plaintext message, P, to Bob by transmitting E_B(DA(P)). Note carefully that Alice know her own (private) decryption key, D_A, as well as Bob's public key, E_B, so constructing this message is something Alice can do.

When Bob receives the message, he transforms it using his private key, as usual, yielding D_A(P), as shown in Figure11.5. He stores this text in a safe place and then decrypts it using E_A to get the original plaintext.

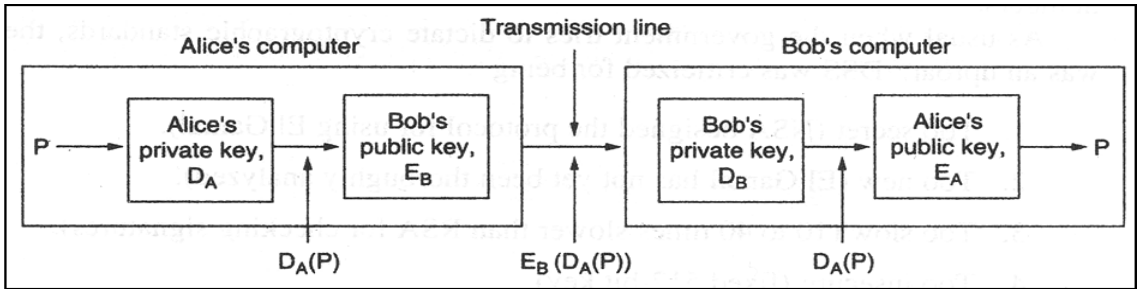


Figure11.5: Digital signatures using public-key cryptography.

To see how the signature property works, suppose that Alice subsequently denies having sent the message P to Bob when the case comes up in court Bob can produce both P and D_A (P). The judge can easily verify that Bob indeed valid message encrypted by D_A by simply applying E_A to it. Since Bob does not know what Alice's private key is the only way Bob could have acquired a sage encrypted by it is if Alice did indeed send it. While in jail for perjury and fraud, Alice will have plenty of time to devise interesting new public-key algorithms.

In principle, any public-key algorithm can be used for digital signatures. The de facto industry standard is the RSA algorithm. Many security products use it However, in 1991, NIST (National Institute of Standards and Technology} posed using a variant of the El Gamal public-key algorithm for their new Digital Signature Standard (DSS). El Gamal gets its security from the difficulty of computing discrete logarithms, rather than the difficulty of factoring numbers.

As usual when the government tries to dictate cryptographic standards, was uproar. DSS was criticized for being

1. Too secret (NSA designed the protocol for using El Gamal).
2. Too new (El Gamal has not yet been thoroughly analyzed).
3. Too slow (10 to 40 times slower than RSA for checking signatures).
4. Too insecure (fixed 512-bit key).

Self Assessment Questions

5. The responsibility of a certification authority for digital signature is to authenticate the

(a) Hash function used
(b) Private keys of subscribers
(c) Public keys of subscribers
(d) Key used in DES

6. Certification of Digital signature by an independent authority is needed because .....

(a) It is safe.
(b) It gives confidence to a business.
(c) The authority checks and assures customers that the public key indeed belongs to the business which claims its ownership.
(d) Private Key claimed by a sender may not be actually his.

7. Public key encryption method is a system

(a).which uses a set of public keys one for each participant in e-commerce.
(b).in which each person who wants to communicate has two keys; a private key known to him only and a public key which is publicized to enable others to send message to him.
(c).which uses the RSA coding system.
(d).which is a standard for use in e-Commerce.

8 Public key system is useful because.....

(a) It uses two keys.
(b) There is no key distribution problem as public key can be kept in a commonly accessible database.
(c) Private key can be kept secret.
(d) It is a symmetric key system.

9 A digital signature authenticates ……..

(a) Paper documents
(b) electronic documents
(c) Both of (a) and (b)
(d) None of these.

10 Symmetric encryption is the oldest cryptographic technique used to transmit or store digital data securely
(a) True (b) False

Exercise: Check Your Progress 3

Case Study-Digital Ink Signatures

The Tablet PC is the premier tool for digital forms-based data collection processes. In addition to providing the means to collect, validate, and store data, digital forms often must support some degree of authentication. One of the oldest methods of authentication is pen-based signatures—from the feather quill pen of old to the digital stylus of today, people have been using written signatures to indicate their intentions or approval for thousands of years. This case discusses concepts and practices related to pen-based signatures on the Tablet PC. The collection, preservation, and protection of electronic signatures created with a digital pen in Tablet PC applications is a new frontier. As such, the body of code, components, and case law for this topic area is not extensive, and this paper does not present a single or even suggested approach to the subject. Instead, the goal of this paper is to provide solution architects, systems and business analysts, software designers, and technical team leaders with a foundation upon which to build as they design and implement Tablet PC applications where pen-based signatures provide useful functionality.

Definitions

The vocabulary of electronic signatures can be a bit confusing. The term "digital signatures" can be used interchangeably to refer to digital certificates administered by trusted third parties ("certificate authorities"), and similar technologies, as well as digital ink signatures created on the Tablet PC. For the purposes of this paper, we will use the term "e-signature" when referring to a digitized signature created with a stylus and digital ink on a Tablet PC computer.

Basics

An e-signature collected on a Tablet PC is a collection of Stroke objects that are captured in a fully digital, vector zed format by a custom component or pen-enabled form:

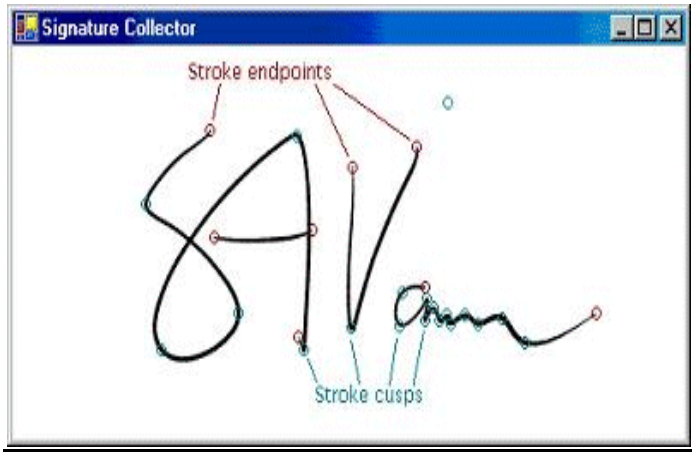


Figure: Composition of an e-signature on a Tablet PC

How the Stroke objects that constitute a signature should be collected, stored, and authenticated are topic areas that are not discussed in the Tablet PC Platform SDK or other Microsoft software documentation. This case provides some initial context for your own research and discussion around these tasks.

The four core discussion and policy areas you should consider with respect to e-signatures are:

- Collection—can your e-signatures be collected with a simple Ink Picture control, or with a component that you have built to match your business needs, or with a third-party product?
- Protection—how can the expropriation and misuse of the e-signature's Strokes be prevented?
- Storage—how and where should the e-signature's Strokes be saved, if at all?

- Validation—how can your application perform signature recognition (validation) of the collected Strokes?

Questions

1. What is e-signature?
2. Explain digital ink signature?

11.11 Summary

- Authentication is the process of verifying that information is coming from a trusted source.
- CHAP is the successor of the Plain Authentication Protocol (PAP), which transmits the username and password in clear text over the network media.
- Digital Signature Standard (DSS) is the digital signature algorithm (DSA) developed by the U.S. National Security Agency (NSA) to generate a digital signature for the authentication of electronic documents.
- A digital signature is basically a way to ensure that an electronic document (e-mail, spreadsheet, text file, etc.) is authentic.
- Cryptography is a security-related technology.

11.12 Keywords

Cipher text: It is encrypted form. And unreadable until it has been converted into plain text (decrypted).
Hash function: It is any algorithm or subroutine that maps large data sets, called keys, to smaller data sets.
Plain test: It is clear test.
Private Key: It is a secret key, used in Asymmetric Encryption.
Public key: It is a publically distributed key, used in Asymmetric Encryption.
Radius: It is authentication protocol

11.13 Review Questions

1. What is a Digital Signature?
2. Differentiate between Direct Digital Signatures And Arbitrated Digital Signature?
3. What is an Authentication?
4. What is the relationship between plaintext and cipher text?
5. What is Authentication protocol?
6. What is Cryptography?
7. Explain a Symmetric Encryption Approach?
8. Why are you using digital signature in business?
9. Differentiate between Symmetric Encryption Approach and public key Encryption Approach?
10. What is DSA?

Answers for Self Assessment Questions

1. (b)	2. (c)	3. (b)	4. (c)	5. (c)
6. (c)	7. (b)	8. (b)	9. (b)	10. (a)

11.14 Further Studies

1. Digital Signature: Network Security Practices By Kailash N. Gupta, Kamalesh N. Agarwala, Prateek Amar Agarwala
2. Computer Networks: A Systems Approach by Larry L. Peterson, Bruce S. Davie

12

Applications of Computers and Information Technology

STRUCTURE

12.0 Objectives

12.1 Introduction

12.2 Business and Computer

12.3 E-Mail

12.4 E-Commerce

12.5 Computers in Personnel Administration

12.6 Project Management

12.7 Accounting

12.8 Computers in Cost and Budget Control

12.9 Use of Computer Technology in Marketing

12.10 Role of Computers and Information Technology in Manufacturing

12.11 Insurance and Stock broking

12.12 Computers in Warehousing

12.13 Summary

12.14 Keywords

12.15 Review Questions

12.16 Further Studies

12.0 Objectives

After studying this chapter, you will be able to:

- Explain the business and computer
- Discuss the e-mail and e-commerce
- Explain the project management
- Discuss the computers in personnel administration
- Discuss the accounting and marketing
- Explain the computers in cost and budget control

- Discuss the manufacturing and materials management
- Explain the banking and insurance and stock broking
- Explain the purchasing and computers in warehousing

12.1 Introduction

Using a computer and the internet, you will learn about information technology and online conferencing by interacting with other students and your course teacher. The first module in the course teaches you the basics of information technology, the business role of computers, and how to understand computers. In the second module of the course, you will learn about the various computer operating systems, applications software, and hardware add-ons to make computers more effective. The third module of the course deals with communication systems, computer networks, and the Internet. The final module discusses workplace implications and issues associated with the World Wide Web.

12.2 Business and Computer

Almost everyone is aware that Information Technology (IT) has played a very significant role in taking businesses to new heights. Before the advent of computers and relevant technology, businesses were totally done using manual resources. As a result, the time taken to complete a task was more, quality of work was not up to the mark, and the procedures also tended to be more complicated. However, as computers started to be used in businesses establishments, the processing of work got more stabilized.

Corporate computing is a concept that is concentrated on the involvement of information technology in business concerns. If you are a working professional, you surely might have easily known how much computer technologies are used in businesses. These technologies are used in almost all sectors such as accounts and payroll management, inventory management and control, shipping functions, data and database management, financial analysis, software development, security control and many other essential fields. The end result of corporate computing is increased productivity and quality. Let us now focus on the use of computers in business world.

12.2.1 Use of Computers in Business World

Following are only a few major fields in business where computing is used largely.

Inventory Control and Management

Inventory control and management is a crucial process, especially in establishments related to retail and production. Computers are used for recording all aspects of the goods coming in, details of goods and services, distribution of stock, and storage details. Note, that in small retail and production firms, simple computer software are generally used. Where as in large corporations, Enterprise Resource Planning (ERPs) are employed.

Accounts and Payroll Management

Accounting and payroll management is also believed to be an important part of the overall system in a company. Be it any kind of industry; computers are largely used for the purpose of managing accounts of administration, sales, purchases, invoices, and also payroll management, which includes recording financial details of employees. These are just some components of the accounts and payroll management system where computing is used.

Database Management

Database management is associated with filing and recording, managing, and retrieval of data whenever required. For smooth running of businesses, it is very important that they have all procedures and details stored. This storage of data is done with the help of large databases and servers which have to be maintained on a regular basis. These information databases and servers are controlled by computers by appropriate authorities in a company.

Software Development

It can be said that for every computing need, software has to be used. Software can only be made using computers for the purpose of helping businesses to combine processes and carry out their work properly. Nowadays, ERPs are largely used in business to blend all their processes together and execute the output as expected. There are many other software and application packages that a business may need to use according to the nature of work.

This is some general information on the use of computers in businesses. There are many other fields such as security control, communication, research, budgeting and forecasting, Web management, where computers are essential. The impact of information technology on business has certainly changed the way businesses operate and have coordinated different practices of the firm to function collectively. Computer use is not only present in businesses, but computers are even used in sectors like medical and defense.

12.3 E-Mail

Electronic mail, commonly known as email or e-mail, is a method of exchanging digital messages from an author to one or more recipients. Modern email operates across the Internet or other computer networks. Some early email systems required that the author and the recipient both be online at the same time, in common with instant messaging. Today's email systems are based on a store-and-forward model. Email servers accept, forward, deliver and store messages. Neither the users nor their computers are required to be online simultaneously; they need connect only briefly, typically to an email server, for as long as it takes to send or receive messages.

An email message consists of three components, the message envelope, the message header, and the message body. The message header contains control information, including, minimally, an originator's email address and one or more recipient addresses. Usually descriptive information is also added, such as a subject header field and a message submission date/time stamp.

Originally a text-only (7-bit ASCII and others) communications medium, email was extended to carry multi-media content attachments, a process standardized in RFC 2045 through 2049. Collectively, these RFCs have come to be called Multipurpose Internet Mail Extensions (MIME).

Three types of email marketing:

Direct Email

Direct email involves sending a promotional message in the form of an email. It might be an announcement of a special offer, for example. Just as you might have a list of customer or prospect postal addresses to send your promotions too, so you can collect a list of customer or prospect email addresses.

Retention Email

Instead of promotional email designed only to encourage the recipient to take action (buy something, sign-up for something, etc.), you might send out retention emails. These usually take the form of regular emails known as newsletters. A newsletter may carry promotional messages or advertisements, but will aim at developing a long-term impact on the readers. It should provide the readers with value, which means more than just sales messages. It should contain information which informs, entertains or otherwise benefits the readers.

Advertising in Other People's Emails

Instead of producing your own newsletter, you can find newsletters published by others and pay them to put your advertisement in the emails they send their subscribers. Indeed, there are many email newsletters that are created for just this purpose to sell advertising space to others.

Did You Know?
The first email sent in the early 1970s looks quite similar to a basic text message sent on the Internet today.

Caution
Be aware while forwarding mail. Forwarding email is so simple that viruses can quickly infect many machines. Most viruses do not even require users to forward the email they scan a users' computer for email addresses and automatically send the infected message to all of the addresses they find.

12.4 E-Commerce

Simplest form ecommerce is the buying and selling of products and services by businesses and consumers over the Internet. People use the term "ecommerce" to describe encrypted payments on the internet.
Sometimes these transactions include the real-time transfer of funds from buyer to seller and sometimes this is handled manually through an eft-pos terminal once a secure order is received by the merchant.
Internet sales are increasing rapidly as consumers take advantage of lower prices offer by wholesalers retailing their products. This trend is set to strengthen as Web sites address consumer security and privacy concerns.

12.4.1 Benefits of E-Commerce

E-commerce can provide the following benefits over non-electronic commerce:

- Reduced costs by reducing labour, reduced paper work, reduced errors in keying in data, reduce post costs.
- Reduced time shorter lead times for payment and return on investment in advertising, faster delivery of product.
- Flexibility with efficiency the ability to handle complex situations, product ranges and customer profiles without the situation becoming unmanageable.
- Improve relationships with trading partners improved communication between trading partners leads to enhanced long-term relationships.
- Lock in customers the closer you are to your customer and the more you work with them to change from normal business practices to best practice e-commerce the harder it is for a competitor to upset your customer relationship.
- New markets the internet has the potential to expand your business into wider geographical locations.

B2C - Business to Consumer

In the context B2C (business to consumer) trading activity has been slow to take off as at first consumers had doubts about the security of credit card transactions.
Initial B2C trading focused on music CDs, software and books items which were compact and easily shipped and where prices could be slashed once the retailer's cut was taken out of the margin. The Amazon book store would be a good example of this. These products pushed the perimeters of the market out for goods bought on-line.
Books and CDs are relatively generic products. A CD bought in the US will have the same music and quality as one bought locally (the exception is the cover art) and so there is no doubt in the consumers mind exactly what the product is. This is not the case with clothing, where sizes can confuse the purchase decision and where tactile senses figure strongly in the purchasing decision.

Ebay has really transform purchasing behaviour on the Web. Many people have made their first ecommerce transaction on Ebay. Many people sell on Ebay too, given raise to the work-from-home/drop shipping model of ecommerce.

Interestingly through B2C transactions of previously localised or hard to find products can be extremely strong. If you have a unique product that is highly relevant to a niche audience, you are likely to do very well on the Web.

Although sales are increasing rapidly on the internet, the volumes of turnover figures continue to fail short of industry estimates. But as retail Web sites become more navigable and privacy policies are displayed, more people will be drawn to Net-based purchasing by lower prices and convenience.

B2B - business to business

On the Internet, B2B (business to business) is the exchange of products or services between businesses rather than between businesses and consumers.

Although early interest centred on the growth of retailing on the internet, forecasts are that B2B revenue will far exceed B2C revenue in the near future.

Payment Gateways

Both Paypal and pay mate offer credit card to bank account payments. Using one of these services you can invoice a customer, they can pay on Paymate and the funds will be deposited in your bank account less a transaction fee.

Unlike a credit card merchant facility you will not have ongoing, minimum monthly fees and the transaction fee is better than what most card companies offer small merchants. Additional these service are being backed into other ecommerce sites and shopping carts. Ebay for example uses Paypal to process some payments.

The Rise and Rise of Auction Sites

Auction sites such as Ebay and TradeMe have done an enormous amount to get ordinary people involved in online trading. Today many Ebay merchants are establishing their own Web sites to avoid Ebay and Pay Pal fees. They have learnt about how to present their product in their Ebay store and what issues are important to their customers in purchasing their product and now they are ready to start their own Web site.

Did You Know?

Electronic data interchange (EDI) is used to transfer electronic documents or business data from one computer system to another computer system, was first introduced in 1960s and used in 1965.

12.5 Computers in Personnel Administration

A personal computer (PC) is any general-purpose computer whose size, capabilities, and original sales price make it useful for individuals, and which is intended to be operated directly by an end-user with no intervening computer operator. In contrast, the batch processing or time-sharing models allowed larger, more expensive minicomputer and mainframe systems to be used by many people, usually at the same time. Large data processing systems require a full-time staff to operate efficiently.

This study of computer use and privacy issues in personnel administration was conducted through an examination of pre-computer patterns of employer information collection and record-keeping about job applicants, employees, and former employees, noting the prevailing rules of organizational policy and law that governed the use of such data before computerization began; an examination of where and how (electronic data processing) EDP systems have been used during the past two decades for personnel administration in government, business, and the non-profit sector; an identification of the impact of EDP uses on employee rights of privacy, confidentiality, and individual access; an analysis of current policy issues involving the use of employee data; and a presentation of alternative policy options for organizational leaders, computer managers, regulatory bodies, legislators, judges, and public interest groups concerned with this field. The

study's methodology included an analysis of published literature and reports; interviews with a wide range of organizational managers, systems developers, public-interest groups, and others; a detailed on-site examination of three major organizations.

12.6 Project Management

Project Management have the opportunity to engage in coursework and projects involving:

Mobile Computing

Mobile Computing involves the development and deployment of specialized software and technologies that enable mobile and hand-held computing devices to function, such as smart phones, PDAs, and pocket PCs. In this knowledge domain, students apply their software development and deployment knowledge to many different business and scientific areas such as the internet, desktop computing and enterprise-scale communications.

Students in this area learn the theory and practice of creating highly available and secure voice and data networks. Areas of emphasis include routing and switching, system and network administration, and system management.

Wireless Networking

In Wireless Networking, students have the opportunity to apply their computing fundamentals to the growing world of wireless communications and technologies. This includes all aspects of radio-frequency-based communications, the technologies used for multi-point digital communications, and the application of these technologies to corporate, IT project and process management.

In project and process management, students will augment their fundamental skills in computing and information technology through advanced courses and ongoing research in the skills needed for successful project management. These include human resource management, change control, risk management, and the tools and techniques required for scope definition and management.

In this knowledge domain, students extend their skills in database systems to develop specialized data management solutions for both business management efforts and discovery support systems in the life sciences.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Prepare a structure to develop a project for management of a company.

……..……

……..……

……...

12.7 Accounting

The systematic recording, reporting, and analysis of financial transactions of a business. The person in charge of accounting is known as an accountant, and this individual is typically required to follow a set of rules and regulations, such as the generally accepted accounting principles. Accounting allows a company to analyze the financial performance of the business, and look at statistics such as net profit.

12.7.1 Purpose of Accounting

1. Economists may define it as the practical application of economic theory in that it measures income and values assets.

2. Corporate managers may define it as a set of timely gauges that helps them actually manage the organisation.
3. Labour unions may see it as a monitor of an organisations activities and performance, particularly in relation to the benefits secured by employees versus owners.
4. A Board of directors or a Chief Executive Officer (CEO) may see accounting as a data process and reporting system that provide the information needed for sound financial or economic decision making for their organisation.
5. Banks and other providers of loan funds may see it as a process of providing reports showing the financial position of an organisation in relation to the assets owned, amounts owed to others and monies invested as well as the profitability of the organisation's operations in relation to repaying the loan with interest.
6. Governments may see it as a way of making organisations accountable to the general community by way of taxation contributions and transparency in the outcomes from their decision making.
7. Potential investors may see it as a method of evaluating an organisation's effectiveness in relation to industry benchmarks and the investor's required returns.
8. Investors in some failed enterprises may sadly call it a method of fooling some of the people, some of the time with what has been dubbed.

Financial Accounting

Financial accounting is focused on producing a limited set of specific prescribed financial statements in accordance with generally accepted accounting principles. The central outputs from financial accounting are audited financial statements such as the balance sheet and income statement that provides a scorecard by which a company's overall past performance can be judged by outsiders.

This branch of accounting targets those external stakeholders that have an interest in the reporting enterprise, but that are not involved in the day-to-day operations. The reports produced by this branch are used for so many different purposes that it is often called "general-purpose accounting". In addition to the financial statements, external stakeholders also have access to financial reporting via press releases that are sent directly to investors and creditors or via the open communications of the internet.

The emphasis in financial accounting is on summaries of financial consequences of past activities and decisions. So, only summarized data is prepared, that covers the entire organization. The data prepared must be objective, precise and be verifiable, usually by an outside 'auditor'. This style of reporting must follow the generally accepted accounting principles that are set by peak accounting bodies in conjunction with government agencies. The numbers used in financial accounting are historical in nature. Now whilst appearing set in stone, financial statements are actually based on estimates, judgments, and assumptions. This is why financial statements usually include 'notes to the accounts' which are the explanations from management that help explain and interpret the numerical information. A more specialised area of financial accounting is tax accounting.

Management Accounting

Managerial accounting deals with information that is not made public and is used for internal decision making only. These reports are far more detailed than financial accounting and can cover performances and activities by departments, products, customers, and employees. It is an accounting system that helps management achieve the goals and objectives of the organisation with an emphasis on the measurement, analysis, communication and the control of financial and non-financial information. This branch of accounting is primarily interested in assisting the organisation's department heads, division managers, and supervisors make better decisions about the day-to-day operations of the business and in particular, those relating to the planning and control decisions. The essential data is conveyed in a wide variety of reports and is specifically targeted at those who direct and control the organisation. These reports help to promote more efficient and effective plan making, resources organizing, personnel directing, motivating and performance evaluation, and operations

control. Unlike financial accounting, there are no external rules governing management accounting. The emphasis in this branch is on making decisions that affect the future with results being compared to budgets, activity-based costing, and financial planning or to industry benchmarks. These reports are delivered frequently and in a timely way according to the requirements of management. Most reports are analytical in nature with a heavy emphasis on variances in the key indicators that monitor the financial performance of the business unit. A more specialised area of management accounting is cost accounting.

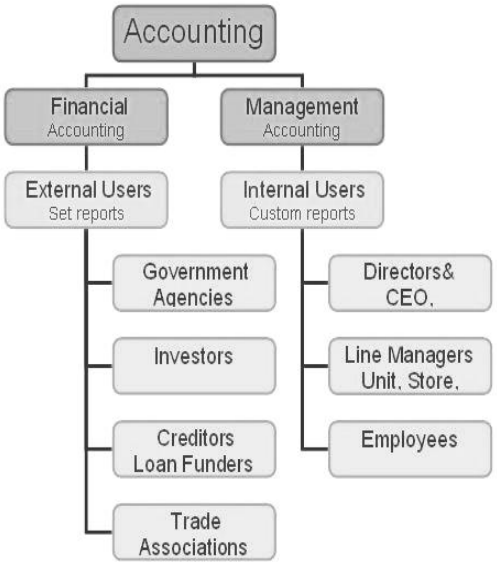


Figure 12.1: Accounting structure.

Exercise: Check Your Progress 2
Note: i) Use the space below for your answer.
Ex1: Prepare the diagram of accounting.

……..……

……..……

……...

Caution
The transaction should be so secure to avoid the hacking of the account by the hackers.

12.8 Computers in Cost and Budget Control

A budget is a plan expressed in quantitative, usually monetary term, covering a specific period of time, usually one year. In other words a budget is a systematic plan for the utilization of manpower and material resources.
In a business organization, a budget represents an estimate of future costs and revenues. Budgets may be divided into two basic classes: Capital Budgets and Operating Budgets.
Capital budgets are directed towards proposed expenditures for new projects and often require special financing. The operating budgets are directed towards achieving short-term operational goals of the organization, for instance, production or profit goals in a business firm. Operating budgets may be sub-divided into various departmental of functional budgets.
The main characteristics of a budget are:

1. It is prepared in advance and is derived from the long-term strategy of the organization.

2. It relates to future period for which objectives or goals have already been laid down.

It is expressed in quantitative form, physical or monetary units, or both.
Different types of budgets are prepared for different purposed e.g. sales budget, production budget, administrative expense budget, Raw-material budget etc. All these sectional budgets are afterwards integrated into a master budget, which represents an overall plan of the organization.

12.8.1 Advantages of Budgets

A budget helps us in the following ways:

1. It brings about efficiency and improvement in the working of the organization.
2. It is a way of communicating the plans to various units of the organization. By establishing the divisional, departmental, sectional budgets, exact responsibilities are assigned. It thus minimizes the possibilities of buck passing if the budget figures are not met.
3. It is a way or motivating managers to achieve the goals set for the units.
4. It serves as a benchmark for controlling on-going operations.
5. It helps in developing a team spirit where participation in budgeting is encouraged.
6. It helps in reducing wastage and losses by revealing them in time for corrective action.
7. It serves as a basis for evaluating the performance of managers.
8. It serves as a means of educating the managers.

Budgetary Control

No system of planning can be successful without having an effective and efficient system of control. Budgeting is closely connected with control. The exercise of control in the organization with the help of budgets is known as budgetary control. The process of budgetary control includes:

1. Preparation of various budgets.
2. Continuous comparison of actual performance with budgetary performance.
3. Revision of budgets in the light of changed circumstances.

A system of budgetary control should not become rigid. There should be enough scope of flexibility to provide for individual initiative and drive. Budgetary control is an important device for making the organization. More efficient on all fronts. It is an important tool for controlling costs and achieving the overall objectives.

Organization for budgeting

The setting up of a definite plan of organization is the first step towards installing budgetary control system in an organization a budget manual should be prepared giving details of the powers, duties, responsibilities and areas of operation of each executive in the organization.

1. Budget Manual a document which setout, inter alias, the responsibilities of the persons engaged in, the routine of, and the forms and records required for, budgetary control.
2. Web for obtaining the necessary approval of budgets, the authority of granting approval should be stated in explicit terms. Whether one, two or more signatures are to be required on each document should also be clearly stated.
3. Timetable for all stages of budgeting.
4. Reports, statements, forms and other records to be maintained.
5. The accounts classification to be employed. It is necessary that the framework within which the costs, revenues and other financial amount are classified must be identical both in accounts and the budget department.

Self Assessment Questions

1. A key purpose of accounting is to promote informed judgments and decisions by users of the financial information

(a). True (b). False

2. Which one of the following is NOT a benefit for the entrepreneur from the accounting process

(a). It is a reality check (b). It helps calculate the tax assessment

(c). It only has to report the good results (d). It helps monitor key indicators.

3. All bookkeepers can also perform the functions of an accountant.

(a). True (b). False

4. Which one of the following is NOT a function usually performed by the accountant.....

(a). Cash flow and profit forecasting

(b). Recording the day to day financial transactions

(c). Resolving complex financial reporting issues

(d). Tax planning

5. Computer based control systems can be combined with manufacturing technology, such as robots, machine tools, automated guided vehicles, to improve manufacturing operations

(a). True (b). False

12.9 Use of Computer Technology in Marketing

The computer has changed society in many different ways since its introduction after the Second World War. There are now many national advertising agencies and campaigns that focuses exclusively on computer marketing. The use of the computer in advertising has yet to be fully played out as there are many new evolutions of approach appearing every day.

Spam

The type of email known derogatorily as "spam" was perhaps the first form of widespread internet marketing. Most often unsolicited, and often including questionable content in the form of advertising for products that are less than respected, spam has been complained about since email first became popular. Many email accounts now include a "spam filter" just to deal with and filter out the large glut of advertising emails that flood the in-boxes of computer users.

Viral Marketing

Due to the social nature of the internet and its media, it has become popular among many national brands and companies to attempt viral marketing. Viral marketing campaigns involve the creation of content advertising that it is hoped consumers will share by themselves with other viewers through links and email. Many successful viral marketing campaigns have helped to promote movies and music, as well as already established brand products such as deodorant products and cars.

Automated Marketing

A new trend as computers become more capable of self-regulating tasks is the use of automated marketing programs. These programs use computers to automatically create spam content to send out to email boxes at a rate faster than human labor could manage. There are thought to be numerous illegal spam networks of automated computers now that send out spam on a regular basis to email accounts and that even compete with each other.

Advertising Networks

Large internet companies have created advertising networks that businesses may pay money to in order to have their advertising placed across a range of internet Websites based upon page views and relevance to content. These networks work through automated processes that scan pages for key words and then pay a certain amount of revenue to the creators of the Web pages. This has become a major funding source for bloggers and other independent internet operators.

12.10 Role of Computers and Information Technology in Manufacturing

Computer based control systems can be combined with manufacturing technology, such as robots, machine tools, automated guided vehicles, to improve manufacturing operations. In this role, the computer can assist integrating these technologies into a lean and efficient factory capable of competing in world markets. This combination of information technology and factory automation is often called computer- integrated manufacturing.

Computer-integrated manufacturing (CIM) blends development in manufacturing with information technology to achieve competitive advantage. When properly organized, CIM offers the opportunity to automate design, manufacturing and production planning and control. Each component is described briefly here:

- Engineering design through computer aided design (CAD) allows an organization to make high quality specialized designs rapidly. The design can be tailored to meet individual customer needs.
- Flexibility manufacturing systems (FMSs) can quickly produce a variety of high quality product efficiently. An (FMSs) also allow an organization to produce high specialized designs.
- Computer based production planning and control systems allow an organization to cope with the complexity of managing facilities that produce a wide variety of specialized products without losing efficiency.
- When properly combined, these components can yield synergetic results. An organization can have more flexible and integrated operations, be better equipped to manage complex operations, and exercise better controls then can a company that operates without CIM. To merge these components into one coordinated whole, staff from the information systems functions needs to integrate engineering, manufacturing, and business databases into a cross functional decision support system. Once accomplished, the flexibility to respond to customer demands with low cost, high quality specialized products becomes a powerful competitive advantage.

12.10.1 Material Management

In modern management, time plays a great role. Right decision at right time is the need of hour. All the decisions are based upon the past data or information and as such with the change in technology and other developments, it is necessary that the decision be based on the latest information, for which a sound data processing system is called for. The materials manager who wishes to utilize a computer in the operation of his department does not have to understand its electronic intricacies. He should, however, know generally what a computer can do, what it cannot do and how his departmental procedures must be designated so as to be compatible with computer operation.

Effective use of a computer offers a manager several significant advantages:

- Computing ability to process huge volumes of data rapidly.
- Freeing the personnel from routine clerical work and repetitive tasks.
- Enable departmental personnel to do more creative work.
- Immediate availability of much more complete data for use in decision making.

Integrated Management Information System

The system using computers can generate various types of information / reports depending on the requirements. Following list gives a list of few reports used in materials department. The list is only illustrative and not exhaustive.

- Long-term production schedule.
- Short-term schedule.
- Materials manual.
- Requirement of non-stock items from user departments.
- Information regarding lead time, supply position (shortages), price trends, anticipated price changes, etc.
- Production Schedule handed over to production department.
- Materials requisition from the production to stores.
- Materials supplied from stores to production.
- Information regarding receipts from the stores.
- Date regarding issues from stores.
- Due dates of supply from purchase department.
- Information to purchase department for follow up of supplies.
- Purchase requisition to purchase department.
- Purchase order.
- Materials from suppliers to stores.
- Previous year's consumption data.

Objectives of Materials Management

- Accurate and speedy remittance of information to customers and suppliers, bills, statements, reminders, etc.
- Information for management for control purposes e.g. stock levels, inventory position, customer's acceptance etc.
- Improvement of customer relations, vendor and employee relations, by timely accurate and relevant information and through vendor analysis etc.
- Reduction of inventories to free the working capital.
- To increase profits at the same time giving better service by optimum scheduling, forecasting and optimizing utilization of traffic facilities warehouses, machines, etc.
- To increase the productivity of materials and facilities.

12.10.2 Banking

Computers are getting more sophisticated. They have given banks a potential they could only dream about and have given bank customers high expectations. The changes that new technologies have brought to banking are enormous in their impact on officers, employees, and customers of banks. Advances in technology are allowing for delivery of banking products and services more conveniently and effectively than ever before thus creating new bases of competition. Rapid access to critical information and the ability to act quickly and effectively will distinguish the successful banks of the future. The bank gains a vital competitive advantage by having a direct marketing and accountable customer service environment and new, streamlined business processes. Consistent management and decision support systems provide the bank that competitive edge to forge ahead in the banking marketplace.

Major Applications

The advantages accruing from computerization are three-directional to the customer, to the bank and to the employee.

For the Customer

Banks are aware of customer's need for new services and plan to make them available. IT has increased the level of competition and forced them to integrate the new technologies in order to satisfy their customers. They have already developed and implemented a certain number of solutions among them:

- ***Self-inquiry Facility:*** Facility for logging into specified self-inquiry terminals at the branch to inquire and view the transactions in the account.
- ***Remote Banking:*** Remote terminals at the customer site connected to the respective branch through a modem, enabling the customer to make inquiries regarding his accounts, on-line, without having to move from his office.
- ***Anytime Banking- anywhere Banking:*** Installation of ATMs which offer non-stop cash withdrawal, remittances and inquiry facilities. Networking of computerized branches inter-city and intra-city will permit customers of these branches, when interconnected, to transact from any of these branches.
- ***Tele Banking:*** A 24-hour service through which inquiries regarding balances and transactions in the account can be made over the phone.
- ***Electronic Banking:*** This enables the bank to provide corporate or high value customers with Graphical User Interface (GUI) software on a PC, to inquire about their financial transactions and accounts, cash transfers, cheque book issue and inquiry on rates without visiting the bank. Moreover, LC text and details on bills can be sent by the customer, and the bank can download the same. The technology used to provide this service is called electronic data interchange (EDI). It is used to transmit business transactions in computer-readble form between organizations and individuals in a standard format.

12.11 Insurance and Stock broking

Insurance companies, finance houses and stock broking firms also make use of computers. Here conditions and requirements are similar to those in banking. Large files of information have to be retained and updated, interest rates and bonuses have to be calculated, policy statements and renewal notice have to be prepared and payments made. In the buying and selling of stocks and shares various calculations have to be made, contract notes drawn up and files consulted and amended.

Stockbrokers are individuals or firms who act as middlemen between investors and stock sellers. They accept money from customers and invest the funds into the stock market on behalf of their customers. Stockbrokers may also provide assistance to customers in deciding which stocks to invest their money in.

General Liability Insurance for Stockbrokers: Slip/trip/fall hazards for stockbrokers can be reduced by a number of common sense practices. Floor coverings inside the building should be clean, dry and free of defects and worn spots. All wiring from equipment and computers should be properly managed to prevent trip and fall accidents. Walk spaces between objects such as furniture, displays, and walls should be adequate for easy passage.

Workers' Compensation Insurance for Stockbrokers: Stockbrokers and their employees may suffer from repetitive motion injuries to fingers, hands, arms, shoulders, neck, and back as a result of using computers on a regular basis. Eyestrain is also a concern when looking at computer screens. Periodic breaks may help to reduce repetitive motion injuries and eyestrain.

Business Auto Insurance for Stockbrokers: Stockbrokers and their employees often travel to meet with clients, therefore increasing their risks of automobile accidents. Employees using their own vehicles for business related travel should be required to carry personal auto liability insurance. Their MVRs should also be checked and verified for acceptability.

Property Insurance for Stockbrokers: The age and construction type of the stockbrokers' buildings are primary risk factors for fire. Buildings over 20 years of age should be updated with new electrical and wiring systems in order to reduce fire risks. Other risks of fire include potentially large amounts of paper files that are stored inside the office building. Stockbrokers that use electronic imaging and store their paper documents in a

separate location will have lower fire loads. Firewalls and anti-virus protection should be installed on all servers and PCs.

Crime Insurance for Stockbrokers: The crime exposures for stockbrokerage firms may result from dishonest employees, theft of money and securities by outsiders, or electronic theft by outsiders. Pre-employment hiring screening should include detailed applications and complete background and reference checks. The use of passwords with upper and lower case letters, encryption, anti-virus software, firewalls, and SSL will help prevent electronic theft by outsiders.

12.11.1 Purchasing

The activity of acquiring goods or services to accomplish the goals of an organization.

The major objectives of purchasing are to:

1. Maintain the quality and value of a company's products.
2. Minimize cash tied-up in inventory.
3. Maintain the flow of inputs to maintain the flow of outputs.
4. Strengthen the organization's competitive position.

Purchasing may also involve

- Development and review of the product specifications
- Receipt and processing of requisitions
- Advertising for bids
- Bid evaluation
- Award of supply contracts
- Inspection of good received
- Their appropriate storage and release

12.12 Computers in Warehousing

The concept of data warehousing is not hard to understand. The notion is to create a permanent storage space for the data needed to support reporting, analysis, and other BI functions. On the surface, it may seem wasteful to store data in more than one place. The advantages, however more than justify the effort and cost of doing so.

Data Warehouses Typically

- Reside on computers dedicated to this function.
- Run on a database management system (DBMS) such as those from Oracle, Microsoft, or IBM.
- Retain data for long periods of time.
- Consolidate data obtained from many sources.
- Are built around a carefully designed data model that transforms production data from a high speed data entry design to one that supports high speed retrieval.

12.13 Summary

- Computers are used to help design products using computer generated models and 3D drawings. Reduces the need to build physical models to test certain conditions, known as prototypes.
- Manufacturing operational control focuses on day-to-day operations, and the central idea of this process is effectiveness and efficiency.

- Maintains a thorough knowledge of all (Electronic Banking) EB policies and procedures, protocols, authorizations, interfaces with external EB systems, as well as emerging EB technology and applications.
- Managerial accounting deals with information that is not made public and is used for internal decision making only.
- Inventory control and management is a crucial process, especially in establishments related to retail and production.

12.14 Keywords

Banks: Are aware of customer's need for new services and plan to make them available. IT has increased the level of competition and forced them to integrate the new technologies in order to satisfy their customers.
Computer-integrated Manufacturing (CIM): Computers control the whole production line. Best example is in car production where robots undertake much of the work, reducing the need for labour to perform boring, routine tasks.
Data Warehouses: Contain a wide variety of data that present a coherent picture of business conditions at a single point in time.
Information Technology: It concerned with improvements in a variety of human and organizational problem-solving endeavors through the design, development.
Internal Storage: It allows transferring metadata together with the data it describes; thus, metadata is always at hand and can be manipulated easily.

12.15 Review Questions

1. What is the use of computers in business?
2. What is the difference between direct email and retention email?
3. Define the e-commerce and discuss its benefits.
4. What is the difference between B2C and B2B?
5. What is the purpose of accounting?
6. What are the computers in cost and budget control?
7. What is the use of computer technology in marketing?
8. Discuss the major application of banking?
9. What is the insurance and stock broking?
10. What is the computer warehousing?

Answers for Self Assessment Questions

1. (a) 2. (c) 3. (b) 4. (b) 5. (a)

12.16 Further Studies

1. Computer Networks and Information Technologies: Second International edited by Vinu V. Das, Janahanlal Stephen, Yogesh Chaba
2. Computer Networks: A Systems Approach by Larry L. Peterson, Bruce S. Davie

System Analysis & Design

(MCA/PGDCA - 107)



Jaipur National University
Directorate of Distance Education

Established by Government of Rajasthan
Approved by UGC under Sec 2(f) of UGC ACT 1956
(**Recognised by Joint Committee of UGC-AICTE-DEC, Govt. of India**)

1

System Concept and System Environment

STRUCTURE

1.0 Objectives

1.1 Introduction

1.2 System Concepts

1.3 Characteristics of a System

1.4 Types of System

1.5 Management Information System (MIS)

1.6 Decision Support System (DSS)

1.7 Enterprise Resource Planning (ERP) Systems

1.8 Summary

1.9 Keywords

1.10 Review Questions

1.11 Further Studies

1.0 Objectives

After studying this chapter, you will be able to:

- Understand system concepts
- Discuss the characteristics of system
- Explain types of system
- Discuss management information system (MIS)
- Discuss decision support system (DSS)
- Discuss enterprise resource planning (ERP) systems

1.1 Introduction

The concept "system" refers to wholeness, interrelationships between parts or elements and self-regulation. It is a systematic organisation of the elements that operate in a unique way if we can take an example of our motorbike. It has different parts viz. brake, handle, gear, battery etc.

All these parts have their own functions. These are all like elements. If any part does not function, the other parts are also affected and the bike cannot functions this all discussion shows that the elements are interrelated and interdependent, functioning towards the bike effective operation. With all these characteristics, the bike becomes a system. Hence a system has a number of elements functioning together in an interrelated and interdependent manner towards the attainment of certain functions of the system as a whole. A system is also dependent to the surroundings. Man is a social animal who lives in a more or less organised group of people which is known as society. If we apply the concept of system as described above, society can be considered a system, with a set of goals to be achieved, different sections with different functions, working towards these common goals of the society. A society has a certain set of elements working toward the goal of managing funds for the welfare of the people, another set for taking care of the health of the people, another set for education of the people and for employment of the people, and so on and so forth. Unless all these different sections of the society work effectively in a coordinated tradition, the goal of the society i.e. successful proportion of the society cannot be achieved.

1.2 System Concepts

A system is a set of elements (often called 'components' instead) and relationships which are different from relationships of the set or its elements to other elements or sets.

Environment and Boundaries

Systems theory views the world as a complex system of interconnected parts. We scope a system by defining its boundary; this means choosing which entities are inside the system and which are outside part of the environment. We then make simplified representations (models) of the system in order to understand it and to predict or impact its future behaviour. These models may define the structure and/or the behaviour of the system. In science and engineering, a system is the part of the universe that is being studied, while the environment is the remainder of the universe that lies outside the boundaries of the system. It is also known as the surroundings, and in thermodynamics, as the reservoir. Depending on the type of system, it may interact with the environment by exchanging mass, energy (including heat and work), linear momentum, angular momentum, electric charge, or other conserved properties. In some disciplines, such as information theory, information may also be exchanged. The environment is ignored in analysis of the system, except in regards to these interactions. The Environmental systems are vital to the human race and to all living organisms. Without the systems in place and working, we would all cease to exist. Rivers and streams are an example, if the system of this were to not work, then the whole system would collapse.

Natural and Human-made Systems

There are natural and human-made (designed) systems. Natural systems may not have an apparent objective but their outputs can be interpreted as purposes. Human-made systems are made with purposes that are achieved by the delivery of outputs. Their parts must be related; they must be "designed to work as a coherent entity" else they would be two or more distinct systems.

Theoretical Framework

An open system exchanges matter and energy with its surroundings. Most systems are open systems; like a car, coffeemaker, or computer. A closed system exchanges energy, but not matter, with its environment; like Earth or the project Biosphere 2 or 3. An isolated system exchanges neither matter nor energy with its environment. A theoretical example of such system is the Universe.

Process and Transformation Process

A system can also be viewed as a bounded transformation process, that is, a process or collection of processes that transforms inputs into outputs. Inputs are consumed; outputs are produced. The concept of input and output here is very broad. For example, an output of a passenger ship is the movement of people from departure to destination.

Subsystem

A *subsystem* is a set of elements, which is a system itself, and a component of a larger system.

System Model

A system comprises multiple views. For the man-made systems it may be such views as planning, requirement (analysis), design, implementation, deployment, structure, behaviour, input data, and output data views. A system model is required to describe and represent all these multiple views.

System Architecture

System architecture, using one single integrated model for the description of multiple views such as planning, requirement (analysis), design, implementation, deployment, structure, behaviour, input data, and output data views, is a kind of system model. Scholars in various disciplines who are concerned about the tendency toward the fragmentation of knowledge and the increasing complexity of phenomena have sought a unifying approach to knowledge. Some biologist, developed a general systems thereby that applied to any arrangement of elements such as cells, people, societies or even planets. Norbert Wiener, a mathematician observed that information and communications provides connecting links for unifying fragments or elements, His systems concept of information theory, which shows the parallel between the functioning of human beings and electronic systems, laid the foundation for today's computer systems.

Systems analysis and information systems were founded in general systems theory, which emphasizes a close look at all, parts of a system. Too often analysts focus on only one component and over look other equally important component. General systems theory is concerned with "developing a systematic, the theoretical framework upon which to make decisions'. It discourages thinking in a vacuum and encourages consideration of all the activities of the organization and its external environment. Pioneering work in general systems theory emphasized that organizations be viewed as total systems. The idea of systems has become most practical and necessary in conceptualizing the interrelationships and integration of operations, especially when using computers. Thus a system is a way of thinking about organizations and their problems. It also involves a set of techniques that helps in solving problems.

Did You Know?

The term system is derived from the Greek word "systema", which means an organized relationship among functioning units or components.

1.3 Characteristics of a System

Here are some important characteristics that are present in all systems: organization, interaction, inters dependences, interaction and central objective.

- ***Organization***: Organization implies structure and order. It is the arrangement of components that helps to achieve objectives. In the design of a business system, for example, the hierarchical relationship starting with president on top and leading downward to the blue-collar workers

represents the organization structure. Such an arrangement portrays a system subsystem relationship, defines the authority structure, specifies the formal flow of communication and formalizes the chain of command. Likewise a computer system is designed around an input device, a central processing unit, and output device and one or more storage units.

- ***Interaction****:* Interaction refers to the manner in which each component functions with other components of the system. In an organization, for example, purchasing must interact with production, advertising with sales, and payroll with personnel. In a computer system the central processing unit must interact with input device to solve a problem. In turn, the main memory holds programs and data that the arithmetic unit uses for computation. The interrelationship between these components enables the computer to perform.
- ***Interdependence:*** Interdependence means that parts of the organization or computer system depend on one another. They are coordinated and linked together according to a plan. Our subsystem depends on the input of another subsystem for proper functioning i.e. the output of one subsystem is the required input for another subsystem. This interdependence is crucial in systems work.
- ***Integration:*** Integration refers to the holism of systems. Synthesis follows analysis to achieve the central objective of the organization. Integration is concerned with how a system is tied together. It is more than sharing a physical part or location. It means that parts of the system work together within the system even though each part performs a unique function.
- ***Central Objective****:* The last characteristics of a system are its central objective. Objective may be real or stated. Although a stated objective may be the real objective, it is not uncommon for an organization to state one objective and operates to achieve another.

Caution

The important point is that users must know the central objective of a computer application early in the analysis for a successful design and conversion.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Draw the working flow diagram of your college system.

……

……

……

1.4 Types of system

The frame of reference within which one views a system is related to the use of the systems approach for analysis. Systems have been classified in different ways.

Common classifications are: (1) physical or abstract, (2) open or closed, and (3) manmade information systems. (4) Computer Based Information systems

1.5 Management Information System (MIS)

1.5.1 MIS Definition

The Management Information System (MIS) is a concept of the last decade or two. It has been understood and described in a number ways. It is also known as the Information System, the Information and Decision System, the Computer based information System.
The MIS has more than one definition, some of which are give below.

1. The MIS is defined as a system which provides information support for decision making in the organization.
2. The MIS is defined as an integrated system of man and machine for providing the information to support the operations, the management and the decision making function in the organization.
3. The MIS is defined as a system based on the database of the organization evolved for the purpose of providing information to the people in the organization.
4. The MIS is defined as a Computer based Information System.

MIS structure

The concept of the MIS has evolved over a period of time comprising many different facets of the organizational function. MIS is a necessity of all the organizations.
The initial concept of MIS was to process data from the organization and presents it in the form of reports at regular intervals. The system was largely capable of handling the data from collection to processing. It was more impersonal, requiring each individual to pick and choose the processed data and use it for his requirements. This concept was further modified when a distinction was made between data and information. The information is a product of an analysis of data. This concept is similar to a raw material and the finished product. What are needed are information and not a mass of data. However, the data can be analyzed in a number of ways, producing different shades and specifications of the information as a product. It was, therefore, demanded that the system concept be an individual- oriented, as each individual may have a different orientation towards the information. This concept was further modified, that the system should present information in such a form and format that it creates an impact on its user, provoking a decision or an investigation. It was later realized then even though such an impact was a welcome modification, some sort of selective approach was necessary in the analysis and reporting. Hence, the concept of exception reporting was imbibed in MIS. The norm for an exception was necessary to evolve in the organization. The concept remained valid till and to the extent that the norm for an exception remained true and effective. Since the environment turns competitive and is ever changing, fixation of the norm for an exception becomes ka futile exercise at least for the people in the higher echelons of the organization. The concept was then evolved that the system should be capable of handling a need based exception reporting. This need maybe either of an individual or a group of people. This called for keeping all data together in such a form that it can be accessed by anybody and can be processed to suit his needs. The concept is that the data is one but it can be viewed by different individuals in different ways. This gave rise to the concept of database, and the MIS based on the database proved much more effective.

Over a period of time, when these conceptual developments were taking place, the concept of the end user computing using multiple databases emerged. This concept brought a fundamental charge in MIS. The change was decentralization of the system and the user of the information becoming independent of computer professionals. When this becomes a reality, the concept of MIS changed to a decision making system. The job in a computer department is to manage the information resource and leave the task of information processing to the user. The concept of MIS in today's world is a system which handles the databases, databases, provides com-putting facilities to the end user and gives a variety of decision making tools to the user of the system.

The concept of MIS gives high regard to the individual and his ability to use information. An MIS gives information through data analysis. While analyzing the data, it relies on many academic disciplines. These include the theories, principles and concepts from the Management Science, Psychology and Human Behaviour, making the MIS more effective and useful. These academic disciplines are used in designing the MIS, evolving the decision support tools for modelling and decision making.

The foundation of MIS is the principles of management and if its practices. MIS uses the concept of management Information System can be evolved for a specific objective if it is evolved after systematic planning and design. It calls for an analysis of a business, management views and policies, organization culture and the culture and the management style. This is possible only when it in conceptualized as system with an appropriate design. The MIS, therefore, relies heavily on the systems theory offers solutions to handle the complex situations of the input and output flows. It uses theories of communication which helps to evolve a system design capable of handling data inputs, process, and outputs with the least possible noise or distortion in transmitting the information form a source to a destination. It uses the principles of system Design, viz., an ability of continuous adjustment or correction in the system in line with the environmental change in which the MIS operates. Such a design helps to keep the MIS tuned with the business management's needs of the organization.

The concept, therefore, is a blend of principle, theories and practices of the Management, Information and System giving rise to single product known as Management Information System (MIS). The conceptual view of the MIS is shown as a pyramid in Figure.1.1.

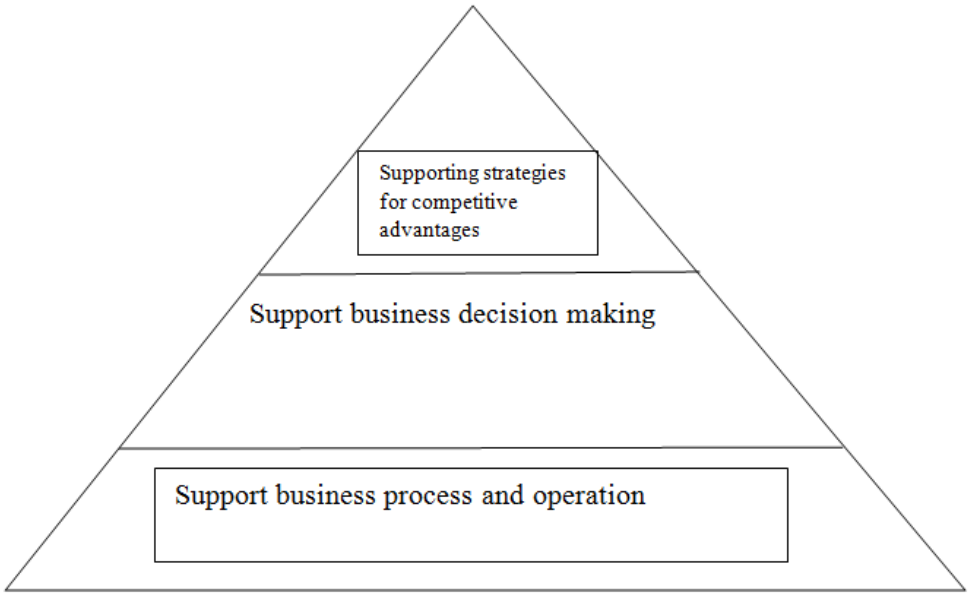


Figure.1.1: The conceptual view of the MIS.

The Physical view of the MIS can be seen as assembly of several subsystems based on the databases in the organization. These subsystems range from data collection, transaction processing and validating, processing, analyzing and storing the information in databases. The subsystem could be at a functional level or a corporate level. The information is evolved through them for a functional or a department management and it provides the information for the management of business at the corporate level. The physical view of the MIS can be shown as in Figure.1.2.

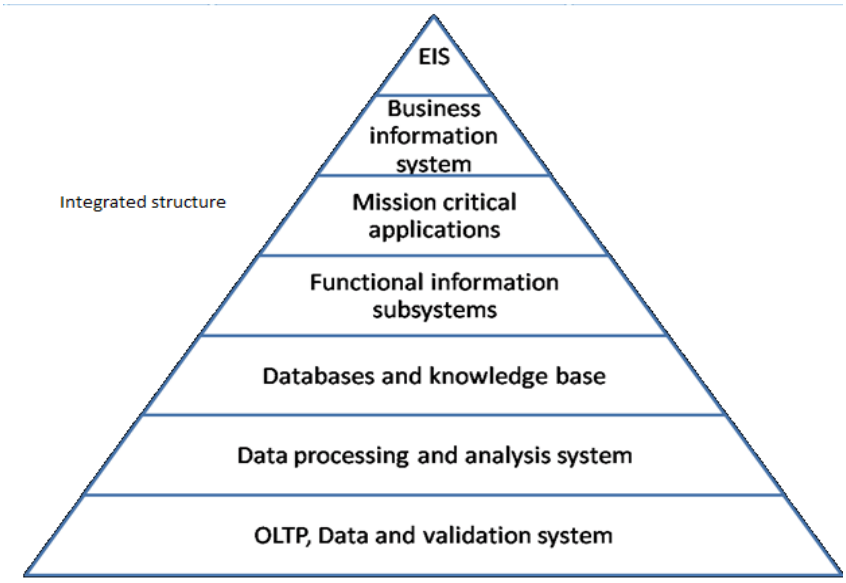


Figure.1.2: The physical view of the MIS.

The MIS is a product of a multi–disciplinary approach to the business management. The MIS differs since the people in two organizations involved in the same business. The MIS is for the people in the organization. The MIS model may be the same but it differs greatly in the contents.

The MIS, therefore, is a dynamic concept subject to change, time and again, with a change in the business management process. It continuously interacts with the internal and the external environment of the business and provides a corrective mechanism in the system so that the change needs of information are with effectively. The MIS, therefore, is a dynamic design, the primary objectively. The MIS, therefore, is a dynamic design the primary objective of which is to the information the information for decision making and it is developed considering the organizational fabric, giving due regard to the people in the organizational the management functions and the managerial and the managerial control.

The MIS model of the organization changes over a time as the business passes through several phases of developmental growth cycle. It supports the management of the business in each phase by giving the information which is crucial in that phase. Every phase has critical success factors in each phase of growth cycle and the MIS model gives more information on the critical success factors for decision making.

Caution

MIS needs to be kept under a constant review and modification to meet the corporate needs of the information, prescribed in product design for the organization.

1.5.2 Role of the Management Information System

The role of the MIS in an organization can be compared to the role of heart in the body. The information is the blood and MIS is the heart. In the body the heart plays the role of supplying pure blood to all the elements of the body including the brain.

The heart works faster and supplies more blood when needed. It regulates and controls the incoming impure blood, processes it and sends it to the destination in the quantity needed. It fulfils the needs of blood supply to human body in normal course and also in crisis.

The MIS plays exactly the same role in the organization. The system ensures that an appropriate data is collected from the various sources, processed, and sent further to all the needy destinations. The

system is expected to fulfil the information needs of an individual, a group of individuals, the management functionaries: the managers and the top management. The MIS satisfies the diverse needs through a variety of systems such as Query Systems, Analysis Systems, Modelling Systems and Decision Support Systems the MIS helps in Strategic Planning, Management Control, Operational Control and Transaction Processing. The MIS helps the clerical personnel in the transaction processing and answers their queries on the data pertaining to the transaction, the status of a particular record and references on a variety of documents. The MIS helps the junior management personnel by providing the operational data for planning, scheduling and control, and helps them further in decision making at the operations level to correct an out of control situation. The MIS helps the middle management in short them planning, target setting and controlling the business functions. It is supported by the use of the management tools of planning and control. The MIS helps the top management in goal setting, strategic planning and evolving the business plans and their implementation. The MIS plays the role of information generation, communication, problem identification and helps in the process of decision making. The MIS, therefore, plays a vital role in the management, administration and operations of an organization.

Self Assessment Questions

1. A society has a certain set of elements working toward the .....of managing funds for the welfare of the people.

(a).concept (b) event

(c) role (d) goal

2. The concept 'system' refers to wholeness, interrelationships between parts or .....and self-regulation.

(a).object (b) system

(c) elements (d) function

3. Successful functioning of each section is determinant for maintaining the continuity of the society.

(a) True (b) False

4. In computer science and information science, system is a software system which has components as its .....inter-process communications as its behavior.

(a).structure (b) observable

(c) structure and observable (d) None of these

Did You Know?

In 1980 to the present, there has been an explosion of technology in the field of information systems.

1.6 Decision Support System (DSS)

Abbreviated DSS, the term refers to an interactive computerized system that gathers and presents data from a wide range of sources, typically for business purposes. DSS applications are systems and subsystems that help people make decisions based on data that is culled from a wide range of sources. ***For example:*** a national on-line book seller wants to begin selling its products internationally but first needs to determine if that will be a wise business decision. The vendor can use a DSS to gather information from its own resources (using a tool such as OLAP) to determine if the company has the ability or potential ability to expand its business and also from external resources, such as industry

data, to determine if there is indeed a demand to meet. The DSS will collect and analyze the data and then present it in a way that can be interpreted by humans. Some decision support systems come very close to acting as artificial intelligence agents.

1.6.1 Benefits of DSS

Decision Support Systems are specifically designed to improve decision-making processes. The benefits of establishing a man1 machine relationship with a Decision Support System are many. Some of the immediate benefits you can expect to receive are:

Direct access to information: **A** DSS is designed to be used directly by managers and analysts. There is no data processing intermediary between you and your data. You simply access the system and issue simple commands that tell the computer what you want it to do.

Access to the kind of information you need: A DSS incorporates internal operating data with external industry data and economic indicators to provide you with the scope of information you need to make strategic decisions. New data is entered easily into the system, and changes are made with minimum disruption to the system.

Information when you want it: A DSS gives you on-the-spot access to the information you need. Eliminated are the usual delays of waiting for information to be compiled, a report to be scheduled, or a graph to be drawn. Results are immediate, enabling decisions to be made more quickly.

Results the way you want to see them: A DSS lets you control how your information is displayed without doing any work in advance. You access the system and, through simple commands, instruct the computer in the way you want a table formatted, which figures you want subtotalled, where you want percentages used, and SO on.

Ability to make comparisons: A DSS enables you to examine the impact alternative courses of action will have on your organization. It lets you design your own modelling relationships which, when applied to your information, indicate the consequence of proposed decisions. You can judge the effects of decisions before they are even made.

Freedom from time- consuming tasks: A DSS takes over the computational burden inherent in analysis. It pro- videos you with a wide selection of pre-programmed statistical and arithmetic tools, such as ratios and regressions, and performs complex mathematical computations in minutes.

1.6.2 Disadvantages of Decision Support System

Decision Support System can create advantages for organizations and can have positive benefits, however building and using Decision Support System can create negative outcomes in some situations.

(1) ***Monetary cost:*** The decision support system requires investing in information system to collect data from many sources and analyze them to support the decision making. Some analysis for Decision Support System needs the advance of data analysis, statistics, econometrics and information system, so it is the high cost to hire the specialists to set up the system.

(2) ***Overemphasize decision making:*** Clearly the focus of those of us interested in computerized decision support is on decisions and decision making. Implementing Decision Support System may reinforce the rational perspective and overemphasize decision processes and decision making. It is important to educate managers about the broader context of decision making and the social, political and emotional factors that impact organizational success. It is especially important to continue examining when and under what circumstances Decision Support System should be built and used. We must continue asking if the decision situation is appropriate for using any type of Decision

Support System and if a specific Decision Support System is or remains appropriate to use for making or informing a specific decision.

(3) ***Assumption of relevance:*** According to researcher "Once a computer system has been installed it is difficult to avoid the assumption that the things it can deal with are the most relevant things for the manager's concern." The danger is that once Decision Support System becomes common in organizations, that managers will use them inappropriately. There is limited evidence that this occurs. Again training is the only way to avoid this potential problem.

(4) ***Transfer of power:*** Building Decision Support System, especially knowledge-driven Decision Support System, may be perceived as transferring decision authority to a software program. This is more a concern with decision automation systems than with Decision Support System. We advocate building computerized decision support systems because we want to improve decision making while keeping a human decision maker in the "decision loop". In general, we value the "need for human discretion and innovation" in the decision making process.

(**5)** ***Unanticipated effects*****:** Implementing decision support technologies may have unanticipated consequences. It is conceivable and it has been demonstrated that some Decision Support System reduce the skill needed to perform a decision task. Some Decision Support System overloads decision makers with information and actually reduces decision making effectiveness. We are sure that other such unintended consequences have been documented. Nevertheless, most of the examples seem correctable, avoidable or subject to remedy if and when they occur.

(6) ***Obscuring responsibility*****:** The computer does not make a "bad" decision, people do. Unfortunately some people may deflect personal responsibility to a Decision Support System. Managers need to be continually reminded that the computerized decision support system is an intermediary between the people who built the system and the people who use the system. The entire responsibility associated with making a decision using a Decision Support System resides with people who built and use the system.

(7) ***False belief in objectivity:*** Managers who use Decision Support System may or may not be more objective in their decision making. Computer software can encourage more rational action, but managers can also use decision support technologies to rationalize their actions. It is an overstatement to suggest that people using a Decision Support System are more objective and rational than managers who are not using computerized decision support.

(8) ***Status reduction:*** Some managers argue using a Decision Support System will diminish their status and force them to do clerical work. This perceptual problem can be a disadvantage of implementing a Decision Support System. Managers and IS staff who advocate building and using computerized decision support need to deal with any status issues that may arise. This perception may, or should be less common now that computer usage is common and accepted in organizations.

(9) ***Information overload:*** Too much information is a major problem for people and many Decision Support System increase the information load. Although this can be a problem, Decision Support System can help managers organize and use information. Decision Support System can actually reduce and manage the information load of a user. Decision Support System developers need to try to measure the information load created by the system and Decision Support System users need to monitor their perceptions of how much information they are receiving. The increasing ubiquity of handheld, wireless computing devices may exacerbate this problem and disadvantage.

Did You Know?

The history of the implementation of Decision Support System (DSS) systems begins in the mid-1960s.

Self Assessment Questions

5. System can also be used referring to a framework, be it software or hardware, designed to allow software programs to run.

(a) True (b) False

6. A system is a set of .....which are different from relationships of the set or its elements to other elements or sets.

(a).elements (b) relationships

(c) Both (a) and (b) (d) None of these

7. Natural systems may not have an apparent objective but their outputs can be interpreted as purposes.

(a) True (b) False

8. Human-made systems are made with purposes that are achieved by the delivery of.....

(a).inputs (b) inputs and outputs

(c) process (d) outputs

1.7 Enterprise Resource Planning (ERP) Systems

ERP is an acronym for Enterprise Resource Planning; a term that is used for business management systems which are designed to integrate the data sources and processes of an entire organization into a unified system. A key element is the use of a single database to store data for the various system modules.

ERP systems utilize components of both computer software and hardware. Traditionally the software is installed at the customer site, but many companies now offer hosted or 'cloud' ERP solutions to reduce the up-front and technical costs.

Enterprise Resource Planning (ERP) has evolved from Material Requirements Planning (MRP) as a means for covering all of the basic functions of an enterprise, in addition to production and inventory.

Components of ERP:

Customer Relationship Management (CRM) - Sales Force Automation, Quoting & Estimating, Order Entry

Manufacturing - Forecasting, Material & Production Planning (MPP), Shop Floor Control, Routings, Capacity Planning & Scheduling, Purchasing, Lot/Serial Control, Inventory, Workflow

Supply Chain - Demand Planning, Purchasing, Supplier Management, Purchasing to Jobs/Projects

Financials - Costing, Accounts Receivable (AR), Accounts Payable (AP), General Ledger (GL)

Human Resources - Labor Collection, Payroll, Benefits

Business Performance Management (BPM) - Business Intelligence (BI), Multi-Entity Consolidation, Corporate Governance, Reporting

Engineering & Product Lifecycle Management (EPLM) - Parts & Bill of Materials (BOM's), CAD Interface, Routings, Parts & Product Attributes, Change Management

Business Benefits:

- Syncronization - End-to-end business function integration
- Accessibility - All key business information in one place
- Responsiveness - Real-time workflow and reporting tools

- Decisiveness - Informed decision making
- Consistency - Everyone in the company is on the same page
- Efficiency - Elimination of most or all side systems and manual calculations

Exercise: Check Your Progress 3
Note: i) Use the space below for your answer.
Ex1: Prepare the ERP software for the small organization.

……

……

……

Did You Know?
In 1990 Gartner Group first employed the acronym ERP as an extension of material requirements planning (MRP).

1.8 Summary

- Systems analysis and information systems were founded in general systems theory, which emphasizes a close look at all, parts of a system.
- Open system has many interfaces with its environment i.e. system that interacts freely with its environment, taking input and returning output.
- Closed system does not interact with the environment; changes in the environment and adaptability are not issues for closed system.
- Management information system provides information that is needed to manage organizations efficiently and effectively. Management information systems involve three primary resources: people, technology, and information or decision making.
- Initial concept of MIS was to process data from the organization and presents it in the form of reports at regular intervals.
- Computerized decision support systems became practical with the development of minicomputers, timeshare operating systems and distributed computing.

1.9 Keywords

Closed System: A system that cannot exchange matter with its surroundings.
Decision Support System: It is a collection of integrated software applications and hardware that form the backbone of an organization's decision making process.
Enterprise Resource Planning: is business management software that allows an organization to use a system of integrated applications to manage the business.
Management Information System: It is a product of a multi- disciplinary approach to the business management.
Physical systems: It is tangible entities that may be static or dynamic in operation.
Subsystem: It is a set of elements, which is a system itself, and a component of a larger system.

1.10 Review Questions

1. What are the system concepts and natural and human-made systems?
2. Define system and system model.
3. Example the characteristics of system.
4. Define the types of system.
5. What is the management information system?
6. Discuss the role of the management information system.
7. Define the impact of the management information system.
8. What is the decision support system?
9. Example the disadvantages of decision support system.
10. What are the enterprise resource planning systems?

Answers for Self Assessment Questions

1. (d)	2. (c)	3. (a)	4. (c)
5. (a)	6. (c)	7. (a)	8. (d)

1.11 Further Studies

1. Systems Analysis and Design: Eighth Edition by Gary B. Shelly, Thomas J. Cashman, Harry J. Rosenblatt.

2

System Development Life Cycle

STRUCTURE

2.0 Objectives

2.1 Introduction

2.2 Defining a System

2.3 Concept of System Development Life Cycle

2.4 Phases of System Development Life Cycle

2.5 Considerations for Candidate Systems

2.6 Summary

2.7 Keywords

2.8 Review Questions

2.9 Further Studies

2.0 Objectives

After studying this chapter, you will be able to:

- Define system
- Explain the concept of system development life cycle
- Describes the phases of system development life cycle
- Explain the considerations for candidate systems

2.1 Introduction

The systems development life cycle (SDLC) is the process of understanding how an information system (IS) can support business needs, designing the system, building it, and delivering it to users. If you have taken a programming class or have programmed on your own, this probably sounds pretty simple.

Most of us would like to think that these problems only occur to "other" people or "other" organizations, but they happen in most companies. Sampling of significant IT project failures. Even Microsoft has a history of failures and overdue projects (e.g., Windows 1.0, Windows 95). Although we would like to promote this chapter as a "silver bullet" that will keep you from experiencing failed IS projects, we must admit that a silver bullet guaranteeing IS development success does not exist.

Instead, this chapter will provide you with several fundamental concepts and many practical techniques that you can use to improve the probability of success.
The key person in the SDLC is the systems analyst who analyzes the business situation, identifies opportunities for improvements, and designs an information system to implement them. Being a systems analyst is one of the most interesting, exciting, and challenging jobs around. As a systems analyst, you will work with a variety of people and learn how they conduct business. Specifically, you will work with a team of systems analysts, programmers, and others on a common mission.

2.2 Defining a System

A collection of components that work together to realize some objectives forms a system. Basically there are three major components in every system, namely input, processing and output (see Figure 2.1)

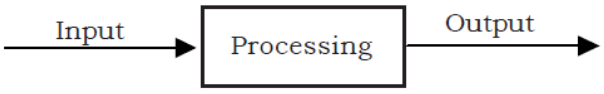


Figure 2.1: Basic system components.

In a system the different components are connected with each other and they are interdependent. For example, human body represents a complete natural system. We are also bound by many national systems such as political system, economic system, educational system and so forth. The objective of the system demands that some output is produced as a result of processing the suitable inputs. A well-designed system also includes an additional element referred to as 'control' that provides a feedback to achieve desired objectives of the system.

2.2.1 System Life Cycle

System life cycle is an organizational process of developing and maintaining systems. It helps in establishing a system project plan, because it gives overall list of processes and sub-processes required for developing a system. System development life cycle means combination of various activities. In other words we can say that various activities put together are referred as system development life cycle. In the System Analysis and Design terminology, the system development life cycle also means software development life cycle.

2.3 Concept of System Development Life Cycle

The software development life cycle **(**SDLC**)** sometimes referred to as the system development life cycle) is the process of creating or altering software systems, and the models and methodologies that people use to develop these systems. In any such lifecycle, people, process and technology all play a role in success. PTC is committed to developing solutions for engineering (Software System Lifecycle Management) and IT (Application Lifecycle Management) organizations along with world class technology (Integrity) to accelerate innovation in software intensive products and systems (see Figure 2.2) (SDLC is also an abbreviation for Synchronous Data Link Control.).

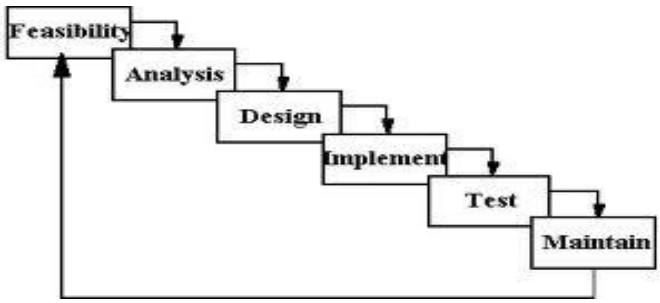


Figure 2.2: System development life cycle.

The systems development life cycle (SDLC) is a conceptual model used in project management that describes the stages involved in an information system development project, from an initial feasibility study through maintenance of the completed application.

Various SDLC methodologies have been developed to guide the processes involved, including the waterfall model (which was the original SDLC method); rapid application development (RAD); joint application development (JAD); the fountain model; the spiral model; build and fix; and synchronize-and-stabilize. Frequently, several models are combined into some sort of hybrid methodology. Documentation is crucial regardless of the type of model chosen or devised for any application, and is usually done in parallel with the development process. Some methods work better for specific types of projects, but in the final analysis, the most important factor for the success of a project may be how closely the particular plan was followed.

In general, an SDLC methodology follows the following steps:

1. The existing system is evaluated. Deficiencies are identified. This can be done by interviewing users of the system and consulting with support personnel.
2. The new system requirements are defined. In particular, the deficiencies in the existing system must be addressed with specific proposals for improvement.
3. The proposed system is designed. Plans are laid out concerning the physical construction, hardware, operating systems, programming, communications, and security issues.
4. The new system is developed. The new components and programs must be obtained and installed. Users of the system must be trained in its use, and all aspects of performance must be tested. If necessary, adjustments must be made at this stage.
5. The system is put into use. This can be done in various ways. The new system can phased in, according to application or location, and the old system gradually replaced. In some cases, it may be more cost-effective to shut down the old system and implement the new system all at once.
6. Once the new system is up and running for a while, it should be exhaustively evaluated. Maintenance must be kept up rigorously at all times.

Caution

Users of the system should be kept up-to-date concerning the latest modifications and procedures.

Did You Know?

The history of the term “Systems Development Life Cycle” is very vague but it naturally came into being since the 1960s when developers started to create programs specific to a certain need.

Self Assessment Question

1. The three major components in every system, namely input, processing and output.

(a). True (b). False.

2. In a system the different components are connected with each other and they are.....
(a). interdependent (b). dependent
(c). inter-process (d). None of these.

3. The objective of the system demands that some output is produced as a result of processing the suitable.....
(a). output (b). inputs.
(c). process (d). None of these.

4. System life cycle is a.....process of developing and maintaining systems.
(a). organizational (b). system analysis
(c). design (d). None of these.

5. In the system analysis and design terminology, the system development life cycle also means.....
(a). design (b). software development life cycle
(c). Both (a) and (b) (d). None of these.

2.4 Phases of System Development Life Cycle

Following are the different phases of system development life cycle:

- Preliminary study
- Feasibility study
- Detailed system study
- System analysis
- System design
- Coding
- Testing
- Implementation
- Maintenance

The different phase of system development life cycle is shown in Figure 2.3.

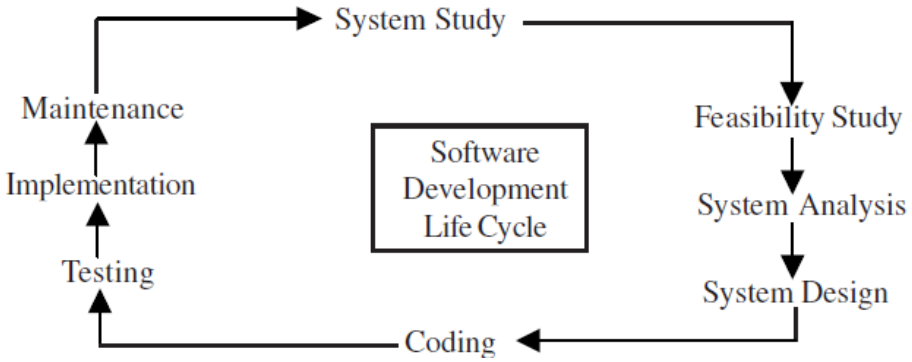


Figure 2.3: Phases of System Development Life Cycle.

2.4.1 Preliminary System Study

Preliminary system study is the first stage of system development life cycle. This is a brief investigation of the system under consideration and gives a clear picture of what actually the physical system is? In practice, the initial system study involves the preparation of a 'System Proposal' which lists the Problem Definition, Objectives of the Study, Terms of reference for Study, Constraints, and Expected benefits of the new system, etc. in the light of the user requirements.

The system proposal is prepared by the System Analyst (who studies the system) and places it before the user management. The management may accept the proposal and the cycle proceeds to the next stage. The management may also reject the proposal or request some modifications in the proposal.

In summary, we would say that system study phase passes through the following steps:

- Problem identification and project initiation
- Background analysis
- Inference or findings (system proposal)

2.4.2 Feasibility Study

In case the system proposal is acceptable to the management, the next phase is to examine the feasibility of the system. The feasibility study is basically the test of the proposed system in the light of its workability, meeting user's requirements, effective use of resources and of course, the cost effectiveness. These are categorized as technical, operational, economic and schedule feasibility. The main goal of feasibility study is not to solve the problem but to achieve the scope. In the process of feasibility study, the cost and benefits are estimated with greater accuracy to find the Return on Investment (ROI). This also defines the resources needed to complete the detailed investigation. The result is a feasibility report submitted to the management. This may be accepted or accepted with modifications or rejected. The system cycle proceeds only if the management accepts it.

2.4.3 Detailed System Study

The detailed investigation of the system is carried out in accordance with the objectives of the proposed system. This involves detailed study of various operations performed by a system and their relationships within and outside the system. During this process, data are collected on the available files, decision points and transactions handled by the present system. Interviews, on-site observation and questionnaire are the tools used for detailed system study.

Using the following steps it becomes easy to draw the exact boundary of the new system under consideration:

- Keeping in view the problems and new requirements
- Workout the pros and cons including new areas of the system

All the data and the findings must be documented in the form of detailed data flow diagrams (DFDs), data dictionary, logical data structures and miniature specification.

The main points to be discussed in this stage are:

- Specification of what the new system is to accomplish based on the user requirements.
- Functional hierarchy showing the functions to be performed by the new system and their relationship with each other.
- Functional network, which are similar to function hierarchy but they highlight the functions which are common to more than one procedure.
- List of attributes of the entities – these are the data items which need to be held about each entity (record)

2.4.4 System Analysis

Systems analysis is a process of collecting factual data, understand the processes involved, identifying problems and recommending feasible suggestions for improving the system functioning. This involves studying the business processes, gathering operational data, understand the information flow, finding out bottlenecks and evolving solutions for overcoming the weaknesses of the system so as to achieve the organizational goals. System Analysis also includes subdividing of complex process involving the entire system, identification of data store and manual processes.

The major objectives of systems analysis are to find answers for each business process: What is being done How is it being done, who is doing it, When is he doing it, Why is it being done and How can it be improved? It is more of a thinking process and involves the creative skills of the System Analyst. It attempts to give birth to a new efficient system that satisfies the current needs of the user and has scope for future growth within the organizational constraints. The result of this process is a logical system design. Systems analysis is an iterative process that continues until a preferred and acceptable solution emerges.

2.4.5 System Design

Based on the user requirements and the detailed analysis of the existing system, the new system must be designed. This is the phase of system designing. It is the most crucial phase in the developments of a system. The logical system design arrived at as a result of systems analysis is converted into physical system design.

Normally, the design proceeds in two stages:

- Preliminary or General Design
- Structured or Detailed Design

Preliminary or General Design: In the preliminary or general design, the features of the new system are specified. The costs of implementing these features and the benefits to be derived are estimated. If the project is still considered to be feasible, we move to the detailed design stage.

Structured or Detailed Design: In the detailed design stage, computer oriented work begins in earnest. At this stage, the design of the system becomes more structured. Structure design is a blue print of a computer system solution to a given problem having the same components and inter-relationships among the same components as the original problem. Input, output, databases, forms, codification schemes and processing specifications are drawn up in detail.

In the design stage, the programming language and the hardware and software platform in which the new system will run are also decided. There are several tools and techniques used for describing the system design of the system.

These tools and techniques are:

- Flowchart
- Data flow diagram (DFD)
- Data dictionary
- Structured English
- Decision table
- Decision tree

The system design involves:

- Defining precisely the required system output
- Determining the data requirement for producing the output
- Determining the medium and format of files and databases

- Devising processing methods and use of software to produce output
- Determine the methods of data capture and data input
- Designing Input forms
- Designing Codification Schemes
- Detailed manual procedures
- Documenting the Design

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: List the all phase of software development life cycle.

……

……

……..

2.4.6 Coding
The system design needs to be implemented to make it a workable system. This demands the coding of design into computer understandable language, i.e., programming language. This is also called the programming phase in which the programmer converts the program specifications into computer instructions, which we refer to as programs. It is an important stage where the defined procedures are transformed into control specifications by the help of a computer language. The programs coordinate the data movements and control the entire process in a system. It is generally felt that the programs must be modular in nature. This helps in fast development, maintenance and future changes, if required.

2.4.7 Testing
Before actually implementing the new system into operation, a test run of the system is done for removing the bugs, if any. It is an important phase of a successful system. After codifying the whole programs of the system, a test plan should be developed and run on a given set of test data. The output of the test run should match the expected results. Sometimes, system testing is considered a part of implementation process.
Using the test data following test run are carried out:
- Program test
- System test

Program test: When the programs have been coded, compiled and brought to working conditions, they must be individually tested with the prepared test data. Any undesirable happening must be noted and debugged (error corrections)
System Test: After carrying out the program test for each of the programs of the system and errors removed, then system test is done. At this stage the test is done on actual data. The complete system is executed on the actual data. At each stage of the execution, the results or output of the system is analysed. During the result analysis, it may be found that the outputs are not matching the expected output of the system. In such case, the errors in the particular programs are identified and are fixed and further tested for the expected output.

When it is ensured that the system is running error-free, the users are called with their own actual data so that the system could be shown running as per their requirements.

Did You Know?
The separation of debugging from testing was initially introduced by Glenford J. Myers in 1979.

2.4.8 Implementation

After having the user acceptance of the new system developed, the implementation phase begins. Implementation is the stage of a project during which theory is turned into practice.
The major steps involved in this phase are:

- Acquisition and Installation of Hardware and Software
- Conversion
- User Training
- Documentation

The hardware and the relevant software required for running the system must be made fully operational before implementation. The conversion is also one of the most critical and expensive activities in the system development life cycle. The data from the old system needs to be converted to operate in the new format of the new system.
The database needs to be setup with security and recovery procedures fully defined. During this phase, all the programs of the system are loaded onto the user's computer. After loading the system, training of the user starts.
Main topics of such type of training are:

- How to execute the package
- How to enter the data
- How to process the data (processing details)
- How to take out the reports

After the users are trained about the computerized system, working has to shift from manual to computerized working. The process is called 'Changeover'.

The following strategies are followed for changeover of the system.
(i) Direct Changeover: This is the complete replacement of the old system by the new system. It is a risky approach and requires comprehensive system testing and training.
(ii) Parallel run: In parallel run both the systems, i.e., computerized and manual, are executed simultaneously for certain defined period. The same data is processed by both the systems.
This strategy is less risky but more expensive because of the following:

- Manual results can be compared with the results of the computerized system.
- The operational work is doubled.
- Failure of the computerized system at the early stage does not affect the working of the organization, because the manual system continues to work, as it used to do.

(iii) Pilot run: In this type of run, the new system is run with the data from one or more of the previous periods for the whole or part of the system. The results are compared with the old system results. It is less expensive and risky than parallel run approach. This strategy builds the confidence and the errors are traced easily without affecting the operations. The documentation of the system is also one of the most important activities in the system development life cycle. This ensures the continuity of the system. There are generally two types of documentation prepared for any system.

These are:

- User or Operator Documentation
- System Documentation

The user documentation is a complete description of the system from the user's point of view detailing how to use or operate the system. It also includes the major error messages likely to be encountered by the users. The system documentation contains the details of system design, programs, their coding, system flow, data dictionary, process description, etc. This helps to understand the system and permit changes to be made in the existing system to satisfy new user needs.

2.4.9 Maintenance

Maintenance is necessary to eliminate errors in the system during its working life and to tune the system to any variations in its working environments. It has been seen that there are always some errors found in the systems that must be noted and corrected. It also means the review of the system from time to time.

The review of the system is done for:

- knowing the full capabilities of the system
- knowing the required changes or the additional requirements
- Studying the performance.

If a major change to a system is needed, a new project may have to be set up to carry out the change. The new project will then proceed through all the above life cycle phases.

2.5 Considerations for Candidate Systems

In today business there is more demand for computer services than the resources available to meet the demand.

The demand is made-up of the following:

1. Operations of existing systems.
2. Maintenance that focuses on patching programs often represents over 50% of maintenance.
3. Enhancements that involve major modifications in program structure or equipment.
4. Request for candidate system.

- All the demands require resources – human, financial and technologies.

The computer department has to provide the following:

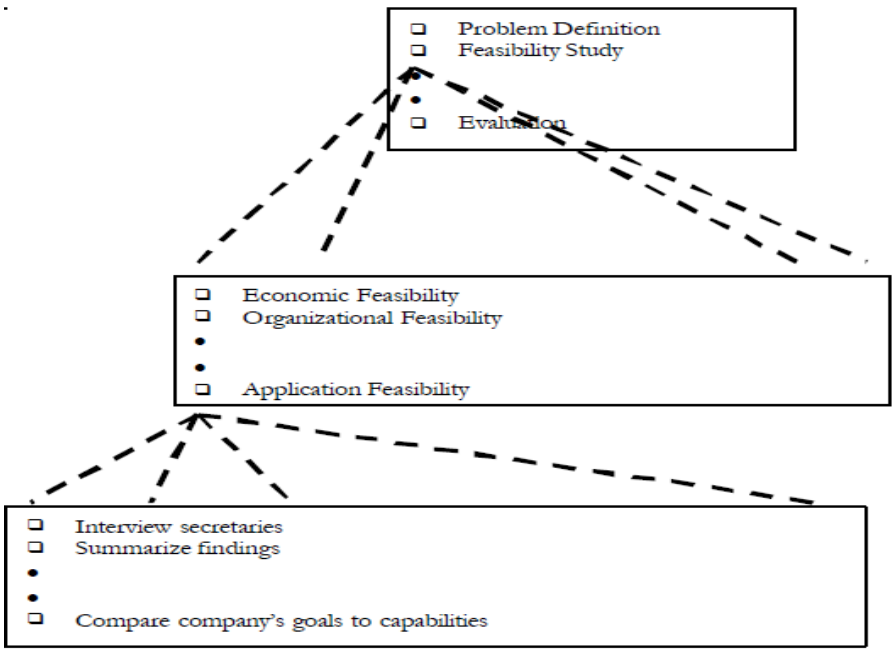
1. Computer operators to run equipment.
2. System analysts to define and design specifications.
3. Application programmers to convert system specifications to computer programs.
4. Maintenance programmers to repair errors.
5. Supervisors project leaders, and managers to coordinate the jobs with the users.

- Thus the basic problem is to match the demands for services with available resources.
- How much one project is favoured over another depends on technical, behavioural, and economic factors.
- The technical factor involves the system department's ability to handle a project.
- The behavioural factor involves
- The user's past experience with existing with an existing system.
- The success record of the analyst
- The influence the user can exert on upper management to finance a candidate system.

- The economic factor focuses on the system's potential return on investment.
- The main idea behind the system development life cycle is to formalize a means of establishing control over a complex process.
- Work units have to be structured at three major levels for effective control of the project.
- At the lowest level, work assignments are broken down into small manageable tasks.
- A task is well-defined, structured work unit that can be carried out by one individual.
- The task can be easily budgeted and scheduled and its quality is measured.

2.5.1 Benefits of SDLC

- Proven framework
- Uniformity
- Uniform method
- Uniform operation (function)
- Results (deliverables)
- Facilitates information exchange
- Defines and focuses roles and responsibilities
- Predefined Degree of Precision
- A complete solution
- A correct solution
- A predictable solution
- Identified milestones
- Enforced Planning and Control



2.5.2 Political Consideration

The growth and development of any industry, be it service or manufacturing, to a large extent is linked with government policies, cooperation and monitoring. Similarly, political environment and stability constitute the other factors for industrial growth. This becomes more pronounced in the case of tourism. This is because not only the whole society is involved in some way or the other but also

because its development or under-development is a direct outcome of the attitudes and perceptions of the politicians, legislators, bureaucrats and others involved in policy formulation. Even the nature and type of tourism and tourist attractions to be developed are linked to it.

In developed countries the government itself has many responsibilities including defiance, economic development, education, health and law and order. These particular responsibilities that rest mainly with government have been the subject of much public policy enquiry. However, tourism has only recently emerged as an obvious commitment and important consideration in the public sector.

As a result analysis of tourism plans and policies is often constrained by the:

- Lack of consensus concerning definitions of such fundamental concepts as 'tourism', 'tourists', and the 'tourism industry',
- Lack of recognition given to tourism planning and policy making processes and the consequent lack of comparative data and case-studies,
- Lack of well defined analytical and theoretical frameworks, and
- Limited amount of quantitative and qualitative data.

2.5.3 Prototyping

Prototyping is especially useful in situations where the requirements (and therefore the costs) are poorly defined or when speed is needed. However, it requires effective management to make sure that the iterations of prototyping do not continue in definitely. It is important to have tools such as 4GLs and screen generators when using this approach. If the project is large, it is probably better to establish the information requirements through prototyping and then use a more formal SDLC to complete the system.

Prototyping is the process of building a model of a system. In terms of an information system, prototypes are employed to help system designers build an information system that intuitive and easy to manipulate for end users. Prototyping is an iterative process that is part of the analysis phase of the systems development life cycle. During the requirements determination portion of the systems analysis phase, system analysts gather information about the organization's current procedures and business processes related the proposed information system. In addition, they study the current information system, if there is one, and conduct user interviews and collect documentation. This helps the analysts develop an initial set of system requirements. Prototyping can augment this process because it converts these basic, yet sometimes intangible, specifications into a tangible but limited working model of the desired information system. The user feedback gained from developing a physical system that the users can touch and see facilitates an evaluative response that the analyst can employ to modify existing requirements as well as developing new ones. Prototyping comes in many forms - from low tech sketches or paper screens (Pictive) from which users and developers can paste controls and objects, to high tech operational systems using CASE (computer-aided software engineering) or fourth generation languages and everywhere in between. Many organizations use multiple prototyping tools. For example, some will use paper in the initial analysis to facilitate concrete user feedback and then later develop an operational prototype using fourth generation languages, such as Visual Basic, during the design stage.

Some Advantages of Prototyping:

- Reduces development time.
- Reduces development costs.
- Requires user involvement.
- Developers receive quantifiable user feedback.
- Facilitates system implementation since users know what to expect.

- Results in higher user satisfaction.
- Exposes developers to potential future system enhancements.

Some Disadvantages of Prototyping

- Can lead to insufficient analysis.
- Users expect the performance of the ultimate system to be the same as the prototype.
- Developers can become too attached to their prototypes.
- Can cause systems to be left unfinished and/or implemented before they are ready.
- Sometimes leads to incomplete documentation.
- If sophisticated software prototypes (4th GL or CASE Tools) are employed, the time saving benefit of prototyping can be lost.

Self Assessment Questions

6. The systems development life cycle (SDLC) is a conceptual model used in project management that describes the stages involved in an information system development project.

(a). True (b). False.

7. The.....may also reject the proposal or request some modifications in the proposal.

(a). organization (b). management

(c). system analysis (d). None of these.

8. The feasibility study is basically the test of the proposed system in the light of its workability, meeting user's requirements, effective use of resources and the cost effectiveness.

(a). True (b). False

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Benefits if SDLC?

……

……

……

2.6 Summary

- The systems development life cycle (SDLC) is a conceptual model used in project management that describes the stages involved in an information system development project, from an initial feasibility study through maintenance of the completed application.
- Preliminary system study is the first stage of system development life cycle. This is a brief investigation of the system under consideration and gives a clear picture of what actually the physical system is?
- The main goal of feasibility study is not to solve the problem but to achieve the scope. In the process of feasibility study, the cost and benefits are estimated with greater accuracy to find the Return on Investment (ROI).

- Maintenance is necessary to eliminate errors in the system during its working life and to tune the system to any variations in its working environments.
- Prototyping is especially useful in situations where the requirements (and therefore the costs) are poorly defined or when speed is needed.

2.7 Keywords

Feasibility Study: It is basically the test of the proposed system in the light of its workability, meeting user's requirements, effective use of resources and of course, the cost effectiveness.

Program test: When the programs have been coded, compiled and brought to working conditions, they must be individually tested with the prepared test data**.**

Prototyping: It is the process of building a model of a system. In terms of an information system, prototypes are employed to help system designers build an information system that intuitive and easy to manipulate for end users.

Structure Design: It is a blue print of a computer system solution to a given problem having the same components and inter-relationships among the same components as the original problem.

System Life Cycle: It is an organizational process of developing and maintaining systems. It helps in establishing a system project plan, because it gives overall list of processes and sub-processes required for developing a system.

Systems Development Life Cycle (SDLC): It is the process of understanding how an information system (IS) can support business needs, designing the system, building it, and delivering it to users.

2.8 Review Questions

1. Define a system. Explain the components of a system.
2. How do you explain system development life cycle?
3. What is the concept of system development life cycle? Explain the benefits of SDLC.
4. Discuss the importance of system analysis and design in the development of a system.
5. Explain the phases of system development life cycle.
6. Write short notes on:

(a) Feasibility study (b) Coding
(c) Testing (d) Maintenance.

7. Differences between detailed system and feasibility study.
8. Understand the considerations for candidate systems.
9. What is the political consideration in SDLC?
10. Explain the prototyping in SDLC.

Answers of Self Assessment Questions

1. (a)	2. (a)	3. (b)	4. (a)
5. (b)	6. (a)	7. (b)	8. (a)

2.9 Further Studies

1. The Information Systems Development Life Cycle: A First Course in Information Systems by David E. Avison, Hanifa U. Shah

3

Role of System Analyst

STRUCTURE

3.0 Objectives

3.1 Introduction

3.2 Historical Perspective of System Analyst

3.3 Who Are Called Systems Analyst?

3.4 What Does a System Analyst Do?

3.5 Who Can Be a System Analyst?

3.6 System Analysis and Designing Skills

3.7 Personal Qualification

3.8 Educational Background and Work Experience

3.9 Career Prospects in System Analysis

3.10 Summary

3.11 Keywords

3.12 Review Questions

3.13 Further Studies

3.0 Objectives

After studying this chapter, you will be able to:

- Explain the concept of system analyst
- Discuss the historical perspective of system analyst
- Understand the role and working of systems analyst
- Explain systems analysis works and feasibility

3.1 Introduction

System development can generally be thought of having two major components: systems analysis and systems design. In System Analysis more emphasis is given to understanding the details of an existing system or a proposed one and then deciding whether the proposed system is desirable or not and whether the existing system needs improvements. Thus, system analysis is the process of

investigating a system, identifying problems, and using the information to recommend improvements to the system.

The system analyst is the person (or persons) who guides through the development of an information system. In performing these tasks the analyst must always match the information system objectives with the goals of the organization. Role of System Analyst differs from organization to organization. Most common responsibilities of System Analyst are following

3.2 Historical Perspective of System Analyst

It always finds it amusing when it tells a young person in this industry that it worked with punch cards and plastic templates years ago. It is kind of the same dumbfounded look It get from my kids when It tell them we used to watch black and white television with three channels, no remote control, and station signoffs at midnight. It has been my observation that our younger workers do not have a sense of history; this is particularly apparent in the systems world. If they do not have an appreciation of whence we came, it doubts they will have an appreciation of where we should be going. Consequently, it has assembled the following chronology of events in the hopes this will provide some insight as to how the systems industry has evolved to its current state.

3.2.1 Pre-1950's - "Systems and Procedures"

Perhaps the biggest revelation to our younger readers regarding this period will be that there was any form of systems prior to the advent of the computer. In fact, "Systems and Procedures" Departments predated the computer by several years. Such departments would be concerned with the design of major business processes using "work measurement" and "work simplification" techniques as derived from Industrial Engineering. Such processes were carefully designed using grid diagrams and flowcharts. There was great precision in the design of forms to record data, filing systems to manage paperwork, and the use of summary reports to act as control points in systems. For example, spreadsheets have been extensively used for many years prior to the introduction of Lotus 1-2-3 or MS Excel. There was also considerable attention given to human behaviour during the business process (the precursor to "ergonomics").

3.2.2 1950's - Introduction of the Computer

The ENIAC was developed for the military at the end of World War II. More importantly, the UNIVAC It (UNIVversal Automatic Computer) was introduced in 1951 by J. Presper Eckert and John Mauchly. The UNIVAC It was a mammoth machine that was originally developed for the U.S. Bureau of the Census. Corporate America took notice of the computer and companies such as DuPont in Wilmington, Delaware began to line-up to experiment with it for commercial purposes. The Remington Rand Corporation sponsored the project, but the company's focus and name eventually changed to "UNIVAC" (today it is referred to as "UNISYS," representing a merger of UNIVAC with Burroughs).

3.2.3 1960's - Management Information Systems

Competition between computer manufacturers heated up during this decade, resulting in improvements in speed, capacity, and capabilities. Of importance here was the introduction of the much touted IBM 360 (the number was selected to denote it was a comprehensive solution - 360 degrees). Other computer vendors offered products with comparable performance, if not more so, but the IBM 360 was widely adopted by corporate America.

The programming of computers was still a difficult task and, consequentially, Procedural Languages were introduced (the third generation languages). In actuality, these languages got their start in the late 1950's, but the proliferation of computers in the 1960's triggered the adoption of procedural languages such as COBOL, FORTRAN, and PL/1. Interestingly, these languages were patterned after Les Matthies' "Playscript" technique which made active use of verbs, nouns, and "if" statements.

3.2.4 1970's - Awakening

Although the MIS movement was noble and ambitious in intent, it floundered due to the size and complexity of the task at hand. Many MIS projects suffered from false starts and botched implementations. This resulted in a period where a series of new methods, tools and techniques were introduced to reign in these huge development efforts.

The first was the introduction of the "methodology" which provided a road map or handbook on how to successfully implement systems development projects. This was pioneered by MBA with its "PRIDE" methodology in 1971. Although the forte of "PRIDE" was how to build systems, it was initially used for nothing more than documentation and as a means to manage projects. Following "PRIDE" was John Toellner's Spectrum It methodology and SDM/70 from Atlantic Software. Several CPA based methodologies followed thereafter.

3.2.5 1980's - The Tool-Oriented Approach

As big iron grew during the 1960's and 1970's, computer manufacturers identified the need for smaller computers to be used by small to medium-sized businesses. In the 1970's, people were sceptical of their usefulness but by the 1980's their power and sophistication caused the "mini" computer to gain in popularity as either a general purpose business machine or dedicated to a specific system.

3.2.6 1990's - Rediscovery

As the PC gained in stature, networking became very important to companies so that workers could collaborate and communicate on a common level. Local Area Networks (LAN) and Wide Area Networks (WAN) seemed to spring-up overnight. As the PC's power and capacity grew, it became obvious that companies no longer needed the burden of mainframes and minis. Instead, dedicated machines were developed to control and share computer files, hence the birth of "client/server computing" where client computers on a network interacted with file servers. This did not completely negate the need for mainframes and minis (which were also used as file servers), but it did have a noticeable impact on sales. Companies still needed mainframes to process voluminous transactions and extensive number-crunching, but the trend was to move away from big iron.

3.2.7 2000's - Gadgets

We are now past the halfway point in this decade and there is nothing of substance to report in terms of computer hardware, other than our machines have gotten faster, smaller, with even more capacity. Perhaps the biggest innovation in this regard is the wide variety of "gadgets" that have been introduced, all of which interface with the PC, including: Personal Digital Assistants (PDA's), iPods, MP3 players, digital cameras, portable CD/DVD players (and burners), cell phones, PS2 and XBox game players. These devices are aimed at either communications or entertainment, giving us greater mobility, yet making us a bit dysfunctional socially. All of this means the computer has become an integral part of our lives, not just at work but at home as well.

3.3 Who Are Called Systems Analyst?

A system analyst is the person who selects and configures computer systems for an organization or business. His or her job typically begins with determining the intended purpose of the computers. This means the analyst must understand the general objectives of the business, as well as what each individual user's job requires. Once the system analyst has determined the general and specific needs of the business, he can choose appropriate systems that will help accomplish the goals of the business.

When configuring computer systems for a business, the analyst must select both hardware and software. The hardware aspect includes customizing each computer's configuration, such as the processor speed, amount of RAM, hard drive space, video card, and monitor size. It may also involve choosing networking equipment that will link the computers together. The software side includes the operating system and applications that are installed on each system. The software programs each person requires may differ greatly between users, which are why it is important that the system analyst knows the specific needs of each user.

To summarize, the system analyst's job is to choose the most efficient computer solutions for a business, while making sure the systems meet all the company's needs. He must also be willing to listen to the constant needs and complaints of the users he builds systems for.

Caution

The system analyst must have a solid understanding of computer hardware and software and should keep up-to-date knowledge on all the latest technologies.

3.4 What Does a System Analyst Do?

A systems analyst is responsible for researching, planning, coordinating and recommending software and system choices to meet an organization's business requirements. The systems analyst plays a vital role in the systems development process. A successful systems analyst must acquire four skills: analytical, technical, managerial, and interpersonal. Analytical skills enable systems analysts to understand the organization and its functions, which helps him/her to identify opportunities and to analyze and solve problems. Technical skills help systems analysts understand the potential and the limitations of information technology. The systems analyst must be able to work with various programming languages, operating systems, and computer hardware platforms. Management skills help systems analysts manage projects, resources, risk, and change. Interpersonal skills help systems analysts work with end users as well as with analysts, programmers, and other systems professionals.

Because they must write user requests into technical specifications, the systems analysts are the liaisons between vendors and the IT professionals of the organization they represent. They may be responsible for developing cost analysis, design considerations, and implementation time-lines. They may also be responsible for feasibility studies of a computer system before making recommendations to senior management.

A systems analyst performs the following tasks:

- Interact with the customers to know their requirements
- Interact with designers to convey the possible interface of the software
- Interact/guide the coders/developers to keep track of system development
- Perform system testing with sample/live data with the help of testers
- Implement the new system
- Prepare High quality Documentation

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: Draw the flow diagram for system analysis.

……

……

……

Exercise: Check Your Progress 2
Note: i) Use the space below for your answer.
Ex1: Prepare high quality software a system analysis.

……

……

……

3.5 Who Can Be a System Analyst?

Any system has to be designed and developed according to the requirements. More specifically, a system has to serve the intended use in the right way. Having a plan and clear-cut idea upfront is thus vital to the entire process of design and development. The required analysis is done by the System analyst. He is the one responsible for handling the overall project from a higher level of view, managing within it the specifics of and integrity with the lower programming level of perspective. With sufficient knowledge about the dynamics of every aspect of the system, interactions with programmers, customers and other relevant people System analyst's job is to get the right solution in the most efficient way. Let us see more about the role he plays.

3.5.1 Get the Requirements

Well, a solution to a problem can be really good only if the problem statement is taken in completely. Customer's words are the key to figuring out the exact requirement set. As a System analyst, one has to be the customer's voice when he drafts out the requirements clearly. Apart from the customer interactions, it is his job to translate the practical needs of customer into a more technical requirement. Acting as a bridge between the customer and the developers the System analyst has to give the right translation of customer's terms into a programmer's idea.

So, where does the analysis come in?

Stating the requirement gathering as just a "translation" can make it seem simple! The truth is that the technical documents that the System analysts prepare from the "Use cases" derived from the customer interactions or marketing documents should take in all considerations of technical feasibility and programming aspects/technologies. To do this a reasonable knowledge from the programming/development side is required. With interactions with the programmers he has to make sure of the feasibility of solution. Cooperation with the development team right through the development phase makes periodic validation/verification possible. Checking for conformance to use case requirements, standards of development and guiding the entire team together forward towards

the goal makes the analyst's job a very critical one for overall success of the software system under development.

Self Assessment Questions

1. The analyst must provide alternate solutions to the management and should an in dept study of the system to avoid.....

(a).Future update (b) Future analyst (c).Future problems (d) None of these

2. The most important role of a systems analyst in business is.....

(a).technical understanding of information systems

(b) Problem solving

(c).knowing what data needs to be stored and used?

(d).special programming skills

3. Time to time, the users update the analyst with the necessary information for developing the system.

(a) True (b) False

4. .....refers to the division of a system into processes or subsystems.

(a).System design (b) Programming

(c).Data management (d).Functional decomposition

3.6 System Analysis and Designing Skills

Systems analysts need not only to demonstrate great communications skills, calm and self-motivation, but also to have well-developed technical skills. Strong technical skills will see analysts through the most difficult problems. College courses and certifications all provide great backgrounds for analysts, and job experience comes only with time. The more programming languages, computer programs and operating systems they learn, the better adapted they are for the next adventure

3.6.1 Performance Tuning

Computers, routers, programs and networks are configured to provide the best working conditions for employees. Analysts need to understand how everything at the site works together and be able to fine-tune and enhance areas that require attention. Monitoring bottlenecks that slow down company productivity and then adjusting system programs to alleviate the problems are all in a day's work.

3.6.2 Programming Languages

Knowledge and familiarity with common programming languages like C++ and Java are useful in the workplace. Systems analysts do not necessarily need to do the coding, but they should be able to read it. That way they can discuss it with the programmers who are responsible for modifying it.

3.6.3 Computer Platforms

Familiarity with operating systems such as AIX, Windows, Mac and LINUX are useful in environments that use many technologies or if a job requires lots of travel.

3.6.4 Hardware Capabilities

In-depth knowledge of the types of hardware in use at each job site is necessary for the systems analyst to ascertain the extent of computer problems and possible equipment damage. Knowledge of

the storage and computing capacity of computers, routing equipment, printers and external and Bluetooth devices can help employees continue to work while repairs are made.

3.6.5 Network Structure and Function

Analysts deal with computers on networks, so it is only logical that they are familiar with network topologies, or structures, and also how to administer a network. Skills of this sort are generally gained on the job and also through classes taught by companies that produce networking software.

3.6.6 Data Structures

Data is everywhere in the workplace, and databases are common at all companies, big and small. Proprietary company databases and more common ones such as Oracle and FoxPro used across industry are based on similar data structures.

3.6.7 Computer Security

Security breaches are more common than ever, and analysts help keep computers free of viruses and other invasive nuisances. They help develop strategies to ensure on-site data integrity. Analysts help define company policies regarding computer security such as not allowing employees to download software onto networks.

3.6.8 Skills Needed By a Systems Analyst

This usually involves meeting with managers and users to better understand what their needs and expectations are.

Systems analysts then use a variety of tools--including structured analysis, data modelling and information engineering--to create the system needed. This part of the job often involves a cost-benefit study to determine if the proposed investment is worth the expense.

Caution

Systems analysts must be familiar with an organization's needs from top to bottom in order to set up a computer network that performs all the jobs a company requires.

3.7 Personal Qualification

Systems Analyst must have following qualifications:

1. Bachelors degree in an IT or engineer field
2. Must have a good understanding of IT architecture, IT systems, and programming and development skills.

3.7.1 Business Analyst

1. Bachelors degree in an IT or business field. A combination of the two is advantageous.
2. Some employers may require an MBA although this is not common place for Business Analysts and is normally associated more with Consultants.
3. Understanding of IT strategy applied to business, excellent communication skills must be obtained as a Business Analyst is often the interface between technical and business teams.

A systems Analyst may be involved with avionics systems, where independent "Black boxes" perform functions, and a controller coordinates the information and system controls.

Systems analysts are now referred to mostly as business analysts. So you, have to be clear about the usage of the terms.

Did You Know?
In 1980 Central Computer and Telecommunications Agency evaluate analysis and design methods.

3.8 Educational Background and Work Experience

Most employers want their systems analysts to at least have a bachelor's degree. It is also helpful if a job candidate for a systems analyst job has practical experience in the organization's area. For instance, businesses hiring system analysts might prefer candidates who have a master in business administration, in addition to formal training in computer science. Hospitals might like to see some training in health-care related issues.

Complex, highly technical jobs might require a graduate degree. This would be true for systems analyst jobs for science-oriented businesses or colleges or universities.

What is more, computer science is changing so rapidly that systems analysts all need to continually study in order to remain competitive. This kind of supplemental training may come from private institutions, colleges, and hardware and software vendors.

Personality traits are also often a factor in finding a job as a systems analyst. Systems analysts need to be able to communicate effectively with people who do not have technical training in computers and wound not is able to understand technical jargon. A good systems analyst can explain computer procedures in simple, easy-to-understand terms users will remember.

Systems analysts are needed in nearly all types of industries to set up computer systems and make existing systems more efficient. A bachelor's degree is usually the minimum requirement, but the career also entails advanced expertise in computer systems design and maintenance. While professionals in this field tend to benefit from high job prospects and earnings, it is important to weigh the pros and cons before making your career decisions.

3.8.1 Essential Career Info

As a systems analyst, your goal is to identify the information technology needs of a company and design computer systems to meet those needs. This may involve developing new systems or configuring existing systems to increase productivity. You may perform tasks like data modelling and information engineering to determine the specifications of your design plan. After determining a system design strategy, you will implement the necessary hardware and software technologies. You will then perform quality assurance tests to resolve any design flaws and to ensure things are operating as expected.

Some other responsibilities that are common for this position include linking computers via networks to maximize information sharing, making recommendations to improve systems and providing training to end users. You might also troubleshoot for computer program and system malfunctions. Additionally, the scope of your job may be limited to certain types of computer systems, such as those dealing with accounting, engineering or science.

These professionals often work for computer systems design firms, but many also work directly for companies and organizations in a wide range of industries or as self-employed consultants. You will generally work a full-time schedule, though the job often requires you to work overtime and may entail travel to clients' places of business.

3.8.2 Job Outlook

Compared to other occupations, employment for systems analysts was projected to grow rapidly at a rate of 22% from 2010-2020. Factors fuelling this job growth include an increased demand for more sophisticated technology, such as mobile and wireless networks. The healthcare industry, in

particular, will have a wealth of job openings due to growing prevalence of e-prescriptions and electronic medical records. Many other positions will be available in IT consulting firms, who hire analysts to go from company to company updating and designing systems.

3.8.3 What Do Employers Look For?

Most employers prefer to hire systems analysts with technology-related bachelor's degree, such as in computer science or engineering. Some employers prefer candidates who have completed industry-related coursework. For example, a financial institution may favour an applicant who incorporated finance into his or her college curriculum. However, having a bachelor's degree in field not related to technology would not automatically disqualify you for a systems analyst position. You can still be a desirable job candidate if you have relevant work experience or expertise in the field you are applying for.

In addition, employers are looking for applicants that are detailed-oriented, can be effective in managing multiple projects simultaneously and have strong analytical, interpersonal and problem-solving skills.

Exercise: Check Your Progress 3

Note: i) Use the space below for your answer.

Ex1: Design computer solutions system to improve organisational efficiency and productivity.

……

……

……

Did You Know?

In 1993 structured systems analysis and design method V4 standard and tools conformance scheme launched.

3.9 Career Prospects in System Analysis

Someone on your left speaks French and someone on your right speaks English; both individuals need to speak to each other. The systems analyst is the middleman, assessing the needs of the end-user and translating them into programming or turning over the programming responsibility to the development department. What are the business requirements? Who will comprise the user community? How large is the application going to be? Will it be internal or external? These are all questions facing the systems analyst, who spends much of the day in front of the computer poring over these issues. With a new product, other elements come into play, such as network location, user community, type of machine, and portability. If the analyst is reviewing an established product, the user community will dictate its changes and enhancements. "One of the biggest surprises in my 25 years of technology work is that people who have a creative background as opposed to a degree in computer science tend to make better systems analysts," says one seasoned professional. "The best analysts it has come across came from backgrounds in theatre, art, and filmmaking.

But they were all able to see and grasp big-picture concepts very quickly, and break them down into subcomponents. People who have a computer science or math background tend to be very technical, and sometimes that can be a hindrance." Systems analysts need to be independent thinkers-people who can "think out of the box" by grasping concepts quickly and seeing the big picture as opposed to

the small details. “It also looks for someone who is self-motivated. Here take the ball and run with it and come back if you have any issues,” says one employer who heads up a technology group.

3.9.1 Paying Your Dues

Few companies are willing to spend money on someone who does not have some kind of programming background. There is not much difference between an analyst and a programmer, though the programmer needs to be versed in a programming language. As far as dealing with the functional requirements, these are the same position. There are junior-level analyst positions, which is almost like being a junior programmer. Any of the Java applets and the basic visual C++ programs is very applicable to today’s market, while COBOL and the older programs such as Assembler are considered dinosaurs. Without experience, a support role at the help desk with internal training is a good way to start out.

3.9.2 Associated Careers

Financial companies and most of the Fortune 100 companies have systems analysts who may also have programming responsibilities. But today, many young analysts are flocking to Web companies where there’s money to be made. A small starting salary combined with options could make you a multimillionaire within a short period of time, or your company could go bust within the year. Internet ventures attract risk-takers, so it all depends on how much of a gambler you are.

Many systems analysts come from creative backgrounds; some return to those fields, while others combine their artistic passions with Internet opportunities. “If it left my position and was able to do anything, it would go back to photography or painting or apply those talents to Web design,” says one systems analyst.

3.9.3 How Can It Stand Out? Earn a Graduate Degree

If you really want to impress employers, consider earning a master’s degree, jobs that require more complex duties often entail a master’s degree in computer science. To better qualify yourself for upper-level management positions, you might earn a Master of Business Administration degree with an information technology concentration, which focuses on instruction in both business and IT management.

3.9.4 Get Certified

You can also set yourself apart from the competition right out of college by obtaining the Information Systems Analyst (ISA) certification from the Institute for the Certification of Computing Professionals (ICCP). If you are a senior graduating with a bachelor’s degree in information systems, you may be able to take the ISA exam at your college or university. A passing score between 50-70% qualifies you for the ISA-Practitioner credential, while a score of 70% or higher qualifies you for the ISA-Mastery level credential. You are then awarded the credential once you graduate from college.

3.9.5 Other Career Paths Systems Software Developer

Similar to systems analysts, systems software developers evaluate companies’ needs and design software to meet business objectives. They also configure existing software to work with new hardware, test software programs and make suggestions on improving technology. This job usually entails a bachelor’s degree in software engineering or computer science. These professionals, however, tend to bring in higher salaries and have better job prospects;

3.9.6 Information Security Analyst

These days, companies need to keep their networks secure to avoid cyber attacks more than ever before. While systems analysts design systems to help companies better run their businesses, information security analysts are needed to help keep those systems secure. In this position, you will perform tasks like encrypting files, setting up firewalls, monitoring networks and updating software programs to protect against viruses. These professionals typically hold bachelor's degrees related to computer science.

3.9.7 Courses and Careers

Systems analysts are Information and Communications Technology (ICT) specialists, who design tailor-made computer systems for larger users such as banks, supermarkets, insurance companies, utilities, government departments and defence systems. It is their job to investigate and analyse a business problem in detail, in both a commercial and technical context, and specify one or more designs for computer systems to provide possible solutions.

Systems analysts begin by discussing the nature of the ICT problem with managers and other users in order to establish exactly what is required. They then plan the structure of the new system, setting out in detail the data to be used, input and output files needed and the mathematical and logical operations to be performed. They may also prepare a cost-benefit analysis to help management decide whether the proposed programming project is worth undertaking.

Once the project is approved, they specify the particular files and records to be used by the program, determine the sequence of processing, and design how the final output will look. After the program is written, they oversee the testing and evaluation process and make sure that any problems are corrected. The final stage is to draw up a detailed implementation plan, write relevant user manuals and set up staff training programmes.

3.9.8 Entry Level

The computer industry is flexible about qualifications and many employers find that relevant experience can be a greater asset than formal academic qualifications. Many people develop computer skills, for example, when working in other fields such as financial services, insurance, banking or accounting. A computer-related degree could undoubtedly prove useful - and there are many to choose from - but would be by no means essential.

An alternative entry route for young people is to enter ICT work directly through a Modern Apprenticeship (Skill seeker in Scotland). There are no set entry requirements, but employers would want to be sure that you are motivated and enthusiastic and can benefit from the training.

3.9.9 Making the Grade

Many employers provide training both in systems analysis and in software applications. Sometimes this is part of introductory ICT training, along with programming training, but it is more commonly given when you move into systems analysis work. This training can lead to professional qualifications, including those of the British Computer Society, the Institute for the Management of Information Systems and the National Computing Centre. Companies such as Microsoft also offer their own professional certification.

3.9.10 Personal Qualities

You would need good overall awareness of computer systems, hardware, programming and software. You must also fully understand the needs of business and industry. Good communication skills would be essential to liaise with programmers and hardware providers as well as end users. You would have

to develop good investigative and presentation skills, the ability to communicate well with staff at all levels, report writing, planning and negotiating skills. A logical and analytical mind is essential, together with a systematic approach to problem solving. Accuracy and attention to detail are equally vital.

3.9.11 Systems Analyst Job Description

A systems analyst designs new IT solutions to improve business efficiency and productivity. The work might be for an external client or an internal client (such as a department within the same organisation).

Working closely with the client, analysts examine existing business models and flows of data, discuss their findings with the client, and design an appropriate improved IT solution.

They produce outline designs and castings of new IT systems, specifying the operations the system will perform, and the way data will be viewed by the user, present their design to the client and, once it is approved, work closely with the client team to implement the solution.

3.9.12 Typical Work Activities

Most systems analysts work with a specific type of IT system, which varies with the type of organisation.

Work activities also depend on the size and nature of the organisation, but typically involve:

- Liaising extensively with external or internal clients
- Analysing clients' existing systems
- Translating client requirements into highly specified project briefs
- Identifying options for potential solutions and assessing them for both technical and business suitability
- Drawing up specific proposals for modified or replacement systems
- Producing project feasibility reports
- Presenting proposals to clients
- Working closely with developers and a variety of end users to ensure technical compatibility and user satisfaction
- Ensuring that budgets are adhered to and deadlines met
- Drawing up a testing schedule for the complete system
- Overseeing the implementation of a new system
- Planning and working flexibly to a deadline
- Writing user manuals
- Providing training to users of a new system
- Keeping up to date with technical and industry sector developments

Did You Know?

In 1983 structured systems analysis and design method made mandatory for all new information system developments.

Self Assessment Questions

5. .....is basically the Information access point for the customer and the marketers.

(a).Planning (b) Analyst

(c). Both (a) and (b) (d) None of these

6. A .....mind is essential, together with a systematic approach to problem solving.

(a).analytical (b) logical

(c).logical and analytical (d).None of these

7. A systems analyst is responsible for researching, planning, coordinating and recommending .....and system choices to meet an organization's business requirements.

(a).Hardware (b) Software

(c).Software and hardware (d) None of these

3.10 Summary

- Systems analyst researches problem, plans solutions, recommends software and systems, and coordinates development to meet business or other requirements.
- The system analyst is the person (or persons) who guides through the development of an information system.
- Successful systems analyst must acquire four skills analytical, technical, managerial, and interpersonal.
- The system analyst role leads and coordinates requirements elicitation and use-case modelling by outlining the system's functionality and delimiting the system.
- The primary objective of any system analyst is to identify the need of the organization by acquiring information by various means and methods.

3.11 Keywords

End User: It is the person that a software program or hardware device is designed.

Computer Platforms: It is includes a hardware architecture and a software framework (including application frameworks), where the combination allows software, particularly application software, to run.

Interpersonal Skills: Such skills are required at various stages of development process for interacting with the users and extracting the requirements out of them.

Problem Solving Skills*:* A system analyst has enough problem solving skills for defining the alternate solutions to the system and also for the problems occurring at the various stages of the development process.

System Analyst: It is the person who selects and configures computer systems for an organization or business.

3.12 Review Questions

1. What is the system analyst?
2. What is the role of the system analyst?
3. Define the skills needed by a systems analyst.
4. What does a system analyst do?
5. How systems analysis works and feasibility?
6. Explain the gathering facts, data and opinions of users in system analyst.
7. What is the system analysis and designing skills?
8. Define the personal qualification for system analyst.
9. Explain the educational background and work experience in system analyst.

10. Explain the career prospects in system analysis.

Answers for Self Assessment Questions

1. (c) 2. (b) 3. (a) 4. (d) 6. (b) 7. (c) 8. (b)

3.13 Further Studies

1. Structured System Analysis and Design By J. B. Dixit

4

System Planning and Initial Investigation

STRUCTURE

4.0 Objectives

4.1 Introduction

4.2 System Planning

4.3 Why Systems Planning?

4.4 Strategic MIS planning

4.5 Managerial and operational MIS planning

4.6 Determining the user's requirements

4.7 Strategies for Information Requirements Determination

4.8 Getting information from the existing information system

4.9 Prototyping

4.10 Initial investigation

4.11 Summary

4.12 Keywords

4.13 Review Questions

4.14 Further Studies

4.0 Objectives

After studying this chapter, you will able to:

- Understand system planning
- Explain that why system planning is necessary
- Define strategic MIS planning
- Discuss managerial and operational MIS planning
- Determine the user's requirements
- Explain strategies for determining information requirements

4.1 Introduction

Planning is fundamental to the way our cities, towns and villages look, the way they work, and the way they relate to each other. Getting right planning is that our goals for society are easier to achieve. Good planning can have a huge beneficial effect on the way we live our lives. It must have a vision of how physical development can improve a community. We need a simpler, faster, more accessible system that serves both business and the community.

4.2 System Planning

The system planning phase is the starting point for the systems analysis and design process. Figure 4.1 shows the planning phase in the process. The planning phase is essentially a screening process for new systems requests. If the systems request is deemed to be a high priority item and is feasible, it will be passed onto the analysis phase.

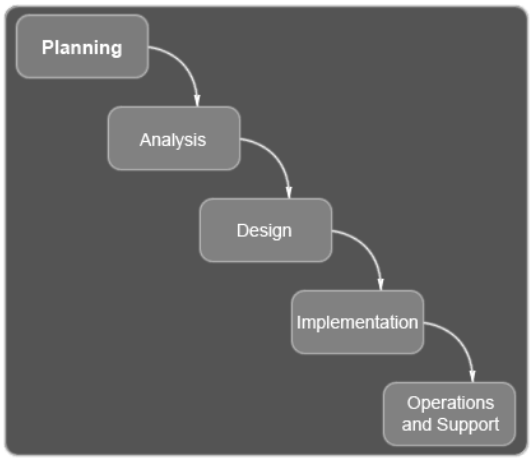


Figure 4.1: Planning Phase.

Figure 4.2 shows the basic process for this phase of the Systems Development Life Cycle (SDLC).

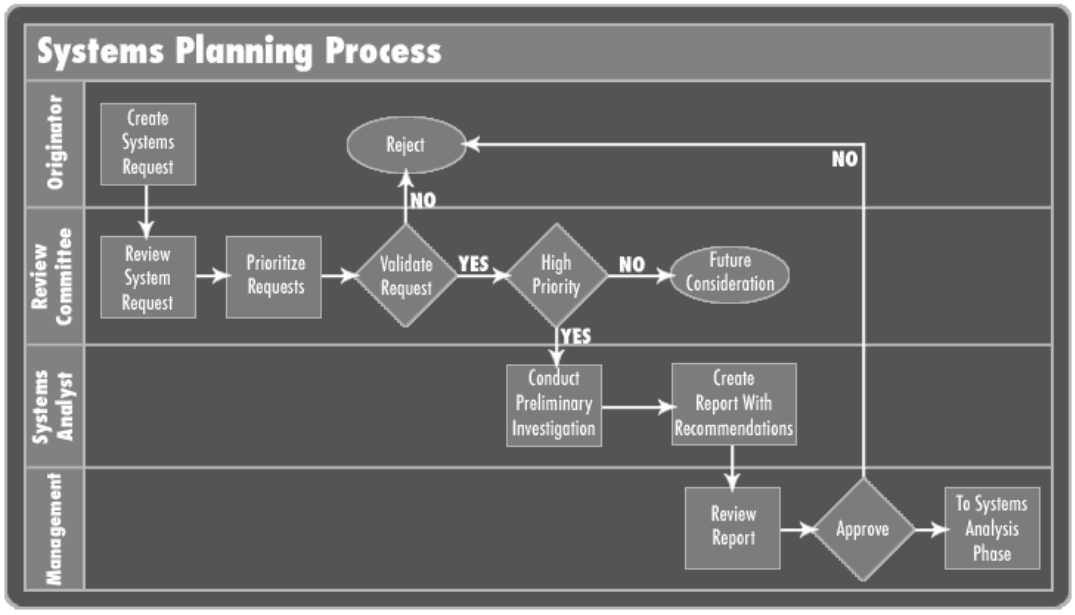


Figure 4.2: Systems Planning Process.

The phase has two major tasks. The first task is to review and prioritize systems requests and select a project or projects for further assessment. Figure 4.3 shows this first task.

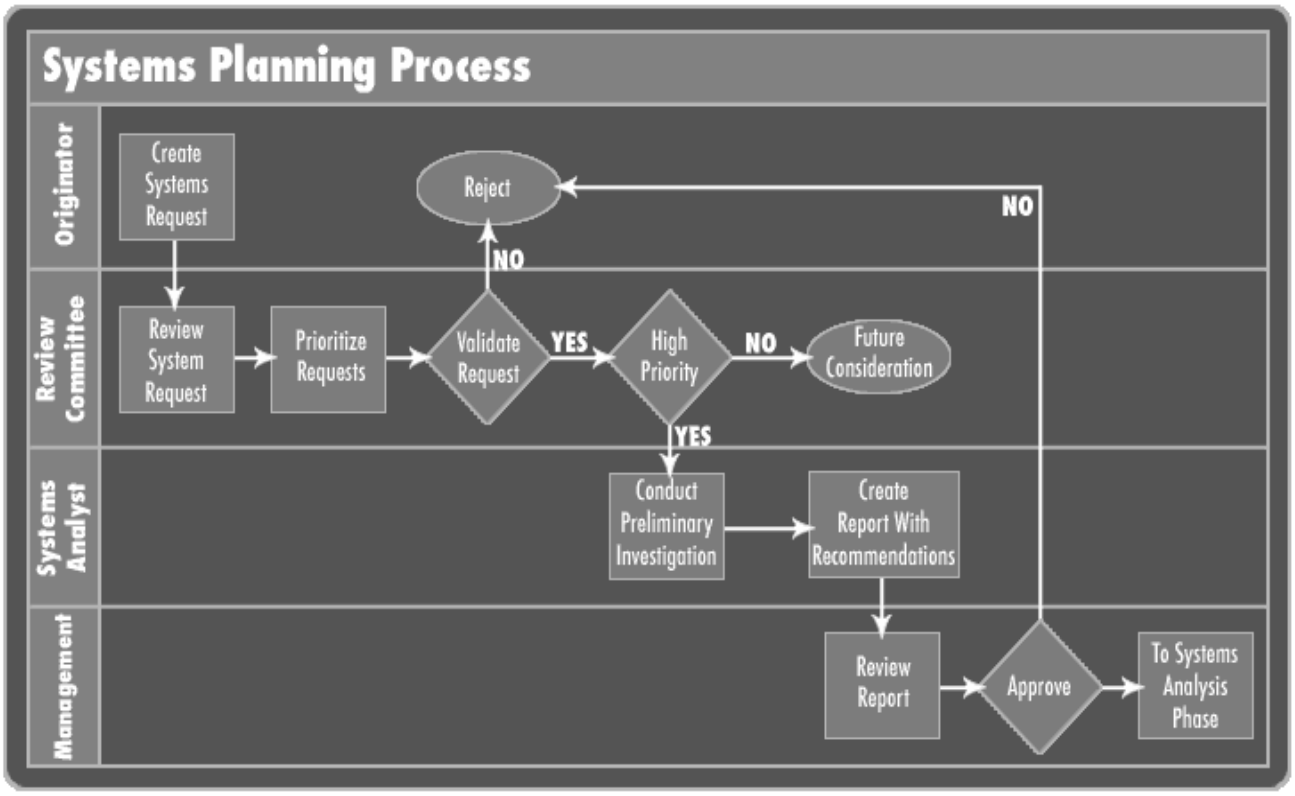


Figure 4.3: Review System Request in Systems Planning Process.

The second major task is to perform a preliminary investigation of the selected project or projects. Figure 4.4 shows this second task. The review process is typically initiated by someone submitting a systems request proposal. The proposal is then evaluated.

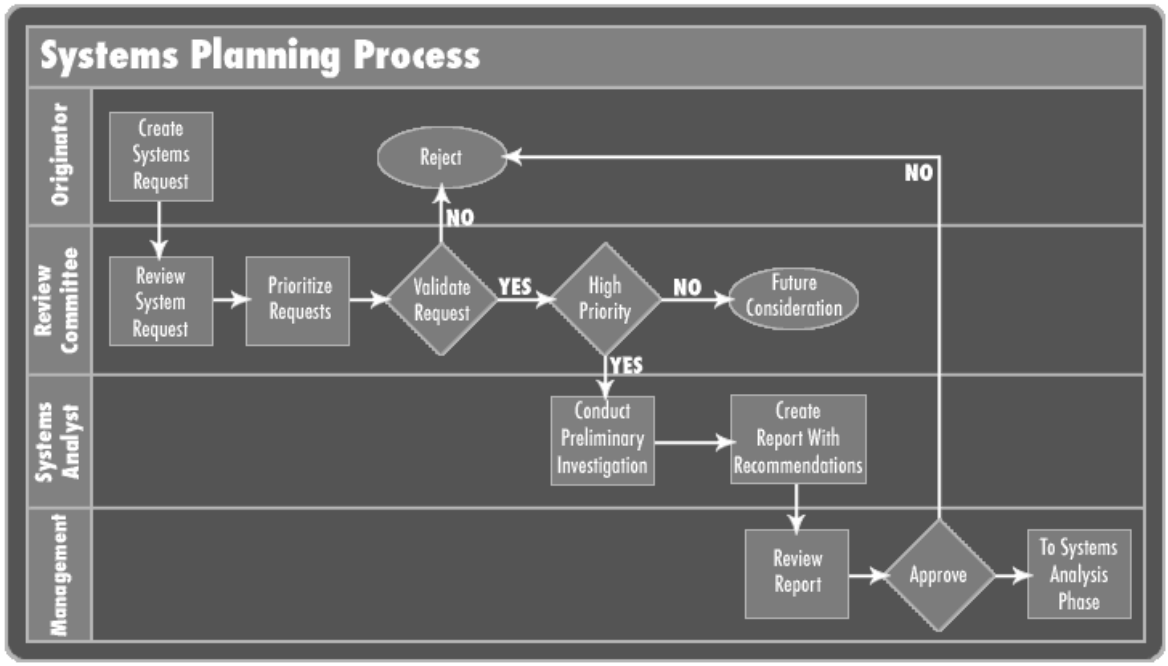


Figure 4.4: Conduct Preliminary Investigation in Systems Planning Process.

Depending on the size of the organization, this proposal review process may be formal or informal. For a small organization, there may be no formal process for reviewing potential or proposed projects. In small organizations, an employee may discuss an idea directly with the president or CEO. For larger organizations, the process is usually much more formal. There may be a form that employees are required to complete to start the formal review process. One of the major reasons for the form is to have a tracking mechanism in place in the case of very large number of requests. A committee or an individual may review the requests.

For significant business changes or for the implementation of large information systems, the request for a system may be dictated by strategic management or an executive committee. Whatever the process may be, the highest priority project or projects are selected for further study. A preliminary investigation is then performed where the facts surrounding the project are researched. This may include interviews, document reviews and other fact-finding techniques. Once sufficient facts are uncovered, the feasibility of the project is determined. The proposal, a summary of the facts, the feasibility analysis, and a recommendation is documented and presented. As with all phases, the recommendation is typically whether or not to proceed to the next phase. In this case, if the proposal is desirable and feasible, the recommendation would be to proceed to the analysis phase.

Did You Know?

The planning system we have in England today originated from the industrial and agricultural revolutions that began in the 1700s.

4.2.1 Objectives of Systems Planning

System planning is always performed with an organization's managers; not for them. System planning is never undertaken as an end in itself: its objective is to create action by being implemented.

Recognizing these facts:

- Describe a step by step approach to planning that will help ensure that the right information is collected at the right time
- Present methods and techniques for summarizing and analyzing the information
- Present guidelines for resource deployment and utilization that will optimize productivity
- Point out pitfalls that may be encountered, with suggestions for avoiding them.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Draw the diagram of any system planning in organization.

……

……

……

4.3 Why Systems Planning?

These posts are applicable to any organization. Depending on organization structure, this could correspond to a branch, department, division, company or group of companies.

The tools and techniques described in the following posts can be used to plan successfully the integration of a single, very large system, or to plan the integration of all systems in an organization.

In the first case, systems planning leads to specification of the hardware, communications, etc. needed to run the system. In the second case, system planning is necessarily broader. Hardware, communications, and so forth are examined at a conceptual level only, and the recommended solution is refined in later phases of the Systems Integration Life Cycle.
These posts will describe the key activities and procedures related to systems planning. Many of these (business process definition, data modelling, capacity analysis, configuration design, etc.) are also carried out in subsequent phases of the Systems Integration Life Cycle.

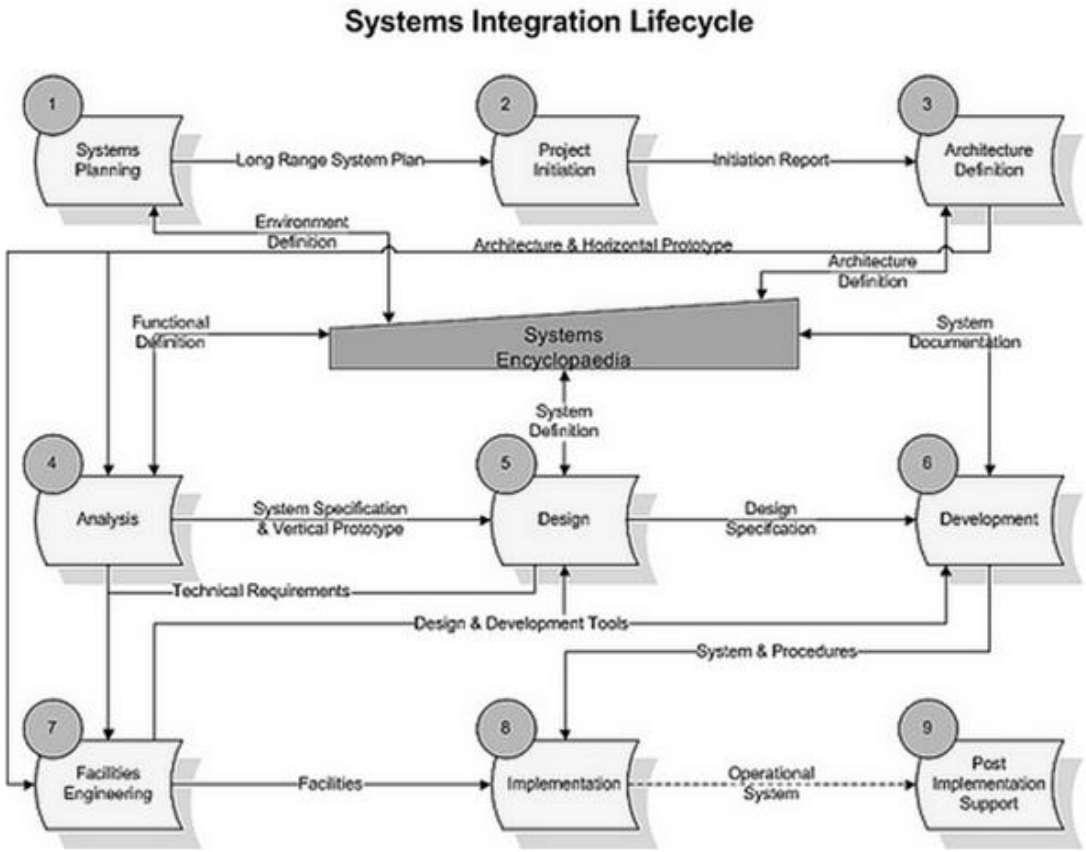


Figure 4.5: System Integration Lifecycle.

Their treatment in my posts will be focused on how they are applied during systems planning.
Each planning activity is described in terms of:

- Objectives
- Responsibilities
- Inputs
- Method
- Working documents
- Tools
- Deliverables
- Variations

Non-analytical duties that form part of the planner's responsibilities within a project management administrative framework, such as progress reporting and budgetary control, are covered in Project Management posts.

4.3.1 Relationship to Systems Integration Life Cycle

System planning is the first phase of the life cycle, and when it is completed, enough work will have been done to initiate the individual projects identified in the plan. These projects will, themselves, involve additional analysis and design, leading to development, implementation and operation of systems. Systems integration, in other words, is a process of progressively more detailed analysis and design that leads to working system solutions.

The relationship between the depth of analysis carried out in systems planning and in subsequent phases of the life cycle is shown in the Figure 4.6.

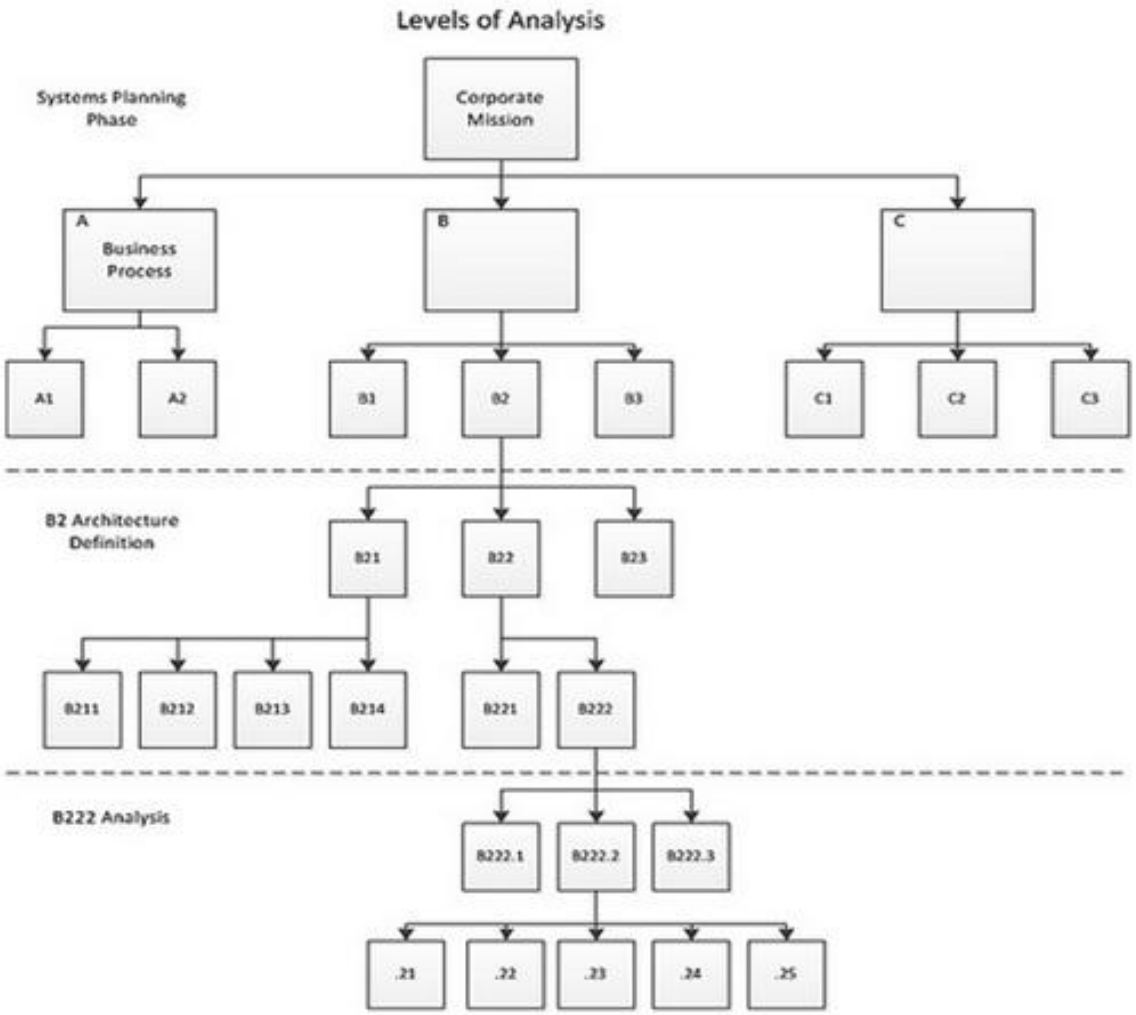


Figure 4.6: Levels of Analyses.

The Figure 4.6 illustrates the relationship for business processes, but a similar relationship holds true for all aspects of the systems plan (e.g., data, hardware and communications technologies, and applications, organization of human resources to implement the plan, and so forth).

4.4 Strategic MIS planning

Figure t abstractly shows the overall process for performing MIS strategic planning. This figure shows an "MIS Strategic Planning" process which transforms an "Organizational Strategy Set," made up of organizational mission, objectives, strategy, and other strategic organizational attributes, into an "MISS strategy Set," made up of system objectives, constraints, and design principles.

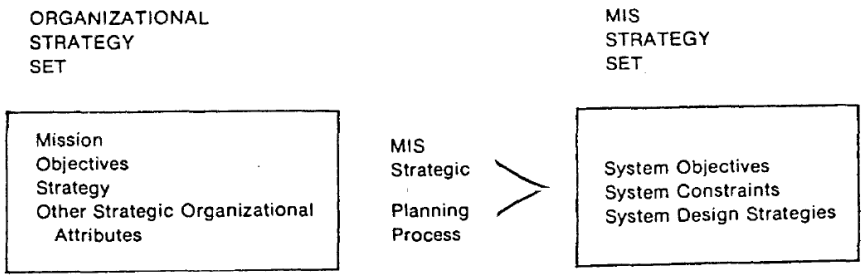


FIGURE 1. Overall MIS Strategic Planning Process

Figure-1 describes an information-based approach to strategic planning for the MIS in that it identifies an information set -- the "MIS Strategy Set" -- which will guide the design and development of the MIS. While the elements of this MIS Strategy Set -- system objectives, constraints, and design principles -- are not usually thought of in this context, they are generally recognized to be the guiding considerations in developing the MIS design.

However well recognized the elements of the MIS Strategy Set are, Figure 1 shows the MIS Strategy Set as emanating directly from information set, the "Organizational Strategy Set." This direct relationship between the two information sets is neither well recognized nor operational zed. It is this linkage which is the province of MIS Strategic Planning and it is on the operational zing of the transformation process between these two information sets that this article focuses.

It will prove useful to describe both the "Organizational Strategy Set" and the "'MIS Strategy Set" in some detail, before describing an operational process for accomplishing the MIS Strategic Planning function which is described conceptually in Figure 1.

Did You Know?

Kenneth and Jane Laudon identify five eras of MIS evolution corresponding to five phases in the development of computing technology: 1) mainframe and minicomputer computing, 2) personal computers, 3) client/server networks, 4) enterprise computing, and 5) cloud computing.

4.5 Managerial and operational MIS planning

There are several business reasons for developing ongoing information management plans. These include:

Service Delivery

Information assets are core to the business of our ministry and the government. Better information management can improve the delivery of services to clients, stakeholders and the public and support the service excellence.

Limited Resources

Information management covers a wide range of activities. It is impossible to address all areas of information management at the same time on a continuous basis. Developing a plan will help you set priorities and improve the management of information to support your ministry's business objectives.

Different Levels of Readiness

Within your organization, different business units will be at various stages of readiness. Attempting to change information management practices prematurely can lead to failure. Planning can help you assess "where to begin."

Linking Information Management Planning to Other Planning Activities

Management planning has a strong commitment to business planning, operational planning and other planning activities such as human resources management, information technology, business continuity as well as knowledge management. Information management planning should be integrated with these other planning activities.

Coordination

Developing ministry information management plans can lead to greater coordination across the ministry. This coordination can help you identify needs as well as implement and leverage best practices across the ministry.

Life Cycle Management

Planning can help assure you that you are addressing issues of information management throughout the life cycle of information assets.

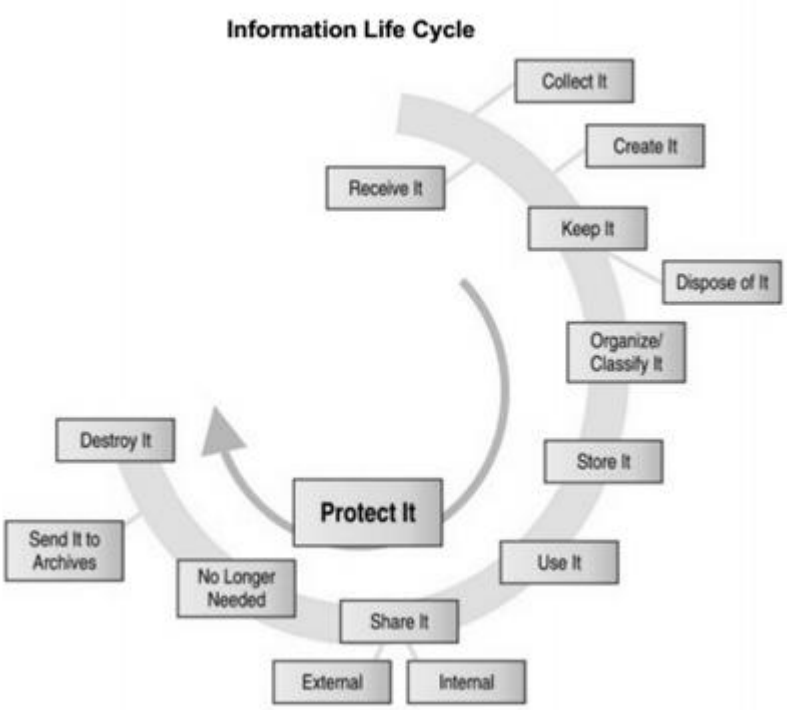


Figure 4.7: Information Life Cycle.

Caution

Attempting to change information management practices prematurely can lead to failure in an organization.

4.5.1 Developing the Plan

Establish project team. Members should represent a mix of information practitioner communities (e.g., information management, records management, web management, libraries, knowledge management and IT) as well as the business units of the organization.

Define vision and future state. Before developing the plan, the team should define the vision and future state that you are trying to achieve. The vision and guiding principles of the Information Management Framework can set the stage for this discussion. However, it is important to define the future state in terms of the ministry's business objectives and operating environment.

Assess current state. It is important to understand current practices related to information management. This will help you identify your ministry's information volumes and growth rates, its strengths (i.e., good practices that are in place), any gaps and will help you to recognize where you are starting from.

Identify gaps and set priorities. By analyzing the results of the interviews, current state assessment, the project team can then begin to identify gaps between current practices and the planned future state. For each gap, you will need to assess "what needs to be done," "what is the level of effort required," and "what the likely impact on the organization is."

Develop the action plan. Based on your assessment of priorities, the next stage is to develop the action plan. The action plan should address all gaps. Typically, the action plan will be a three to five year strategic and tactical plan to improve information management practices in your ministry with the outcome of moving closer to the future state identified earlier in the process. The action plan should include an estimate of resources – human and financial – required to carry out the plan.

Validate the plan. Because information management occurs in a distributed manner in most organizations, it is important to validate your plan with staff. This will allow you the opportunity to gain "buy-in" and to show how the plan will improve productivity and help the ministry achieve its business objectives. Based on the validation, you may end up making adjustments to the plan.

Executive endorsement and communication: Involve your executive management. Endorsement of the plan by executives will help integrate information planning into the strategic direction of the ministry. With executive endorsement, you can then implement your communications strategy for the plan.

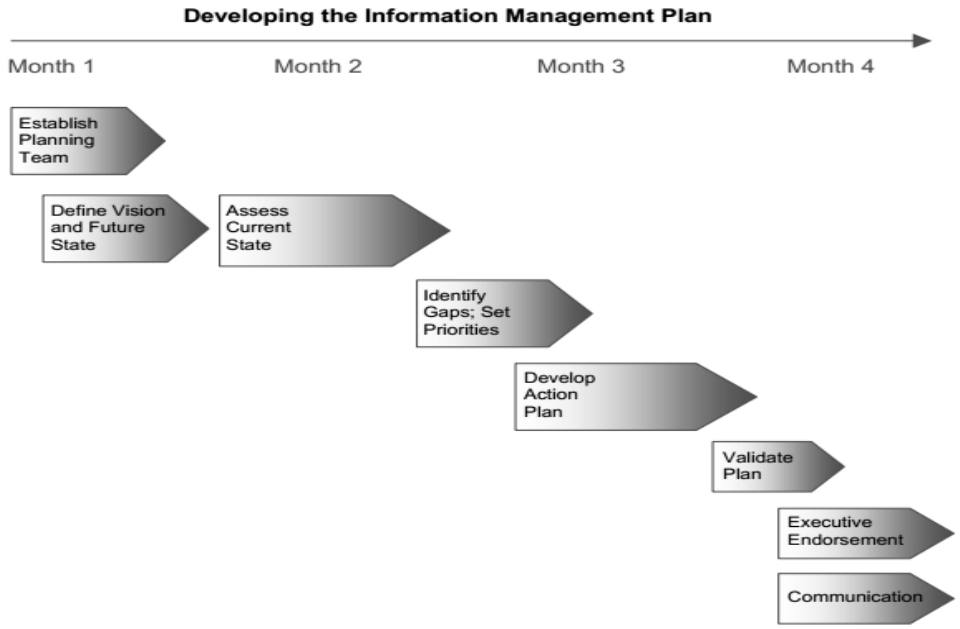


Figure 4.8: Developing the Information Management Plan.

Exercise: Check Your Progress 2
Note: i) Use the space below for your answer.
Ex1: List the some important point of strategic MIS planning.

……

……

……

4.6 Determining the user's requirements

A method is defined as an orderly or systematic procedure; a methodology is a set of methods and techniques. The terms are frequently used interchangeably.

Based on human limitations, an information requirements determination methodology should meet certain needs:

1. Assist an analyst to constrain and structure the problem space. It is estimated that analysts spend 75% of their time on this activity.

2. Assist in searching efficiently within the problem space. It should aid in discovering requirements that are not obtained by anchoring and adjustment and in overcoming short-term memory limitations in human information processing.

3. Assist in overcoming biasing factors such as regency, concreteness, and small samples.

4. Provide assurance that requirements are complete and correct. Methodologies differ in the amount of structure provided. Some provide conceptual structure but little process and documentation structure; others provide detailed structure for all tasks and all documentation. The importance of detailed structure may vary with different circumstances. For example, analysts and users with little experience and expertise may find detailed structure very useful; analysts and users experienced in the application area and able to define requirements may find detailed structure in a methodology to be inhibiting and frustrating.

Self Assessment Questions

1. The .....phase is the starting point for the systems analysis and design process.

(a) Information system (b) information management

(c) System planning (d) None of these

2. One of the major reasons for the form is to have a .....in place in the case of very large number of requests.

(a) information system (b) tracking mechanism

(c) system mechanism (d) None of these

3. .....for information systems begins with the identification of needs.

(a) Planning (b) Scheduling

(c) Forecasting (d) None of these

4.7 Strategies for Information Requirements Determination

A strategy was defined earlier as an approach for achieving an objective. Strategies are general approaches; methods and methodologies are the detailed means for doing it.

There are four strategies for determining information requirements:

(1) Asking,

(2) Deriving from an existing information system,

(3) Synthesis from characteristics of the utilizing system, and

(4) Discovering from experimentation with an evolving information system.

In a specific case, one of the strategies may be used as the primary strategy; others may be used as supplementary strategies. The set of four strategies is applicable both to organizational information requirements determination and to application requirements. For each strategy, there are a number of methods and methodologies that are in use (or have been proposed). In the discussion of strategies, some methods or methodologies will be used as illustrations; no attempt will be made to provide a comprehensive list. In addition to strategies and methods for eliciting requirements, there are also strategies and methods for obtaining assurance that requirements are complete and correct and those systems as implemented meet that requirement. A complete strategy for information system analysis, design, and implementation should include both an eliciting strategy and a quality assurance strategy. The selection of an assurance strategy has been described elsewhere; this paper focuses only on the strategy for eliciting or determining the information requirements. It is not directed at life cycle or other methodologies for assurance.

4.8 Getting information from the existing information system

Existing information systems that have been implemented and have an operational history can be used to derive requirements for a proposed information system for the same type of organization or for the same type of application.
The types of existing information systems that are useful in deriving requirements are:
1. Existing system that will be replaced by the new system
2. Existing system in another, similar organization
3. Proprietary system or package
4. Descriptions in textbooks, handbooks, industry studies, etc.

4.9 Prototyping

Prototyping is a method used by designers to acquire feedback from users about future designs. Prototypes are similar to mock-ups, but are usually not as low-fidelity as mock-ups and appear slightly later in the design process.Prototypes may be horizontal or vertical: A horizontal prototype appears to have a very broad range of the intended future features, but only very little of the actual functionality of the features is implemented. For example, a horizontal prototype of a computer application may have a very well-developed and broad user interface (the horizontal dimension) but not much of the underlying functionality is implemented (the vertical dimension, i.e. the deeper layers of the software). Correspondingly, a vertical prototype only has very few features, which on the other hand are almost fully implemented or at least so-called “walking skeletons”.

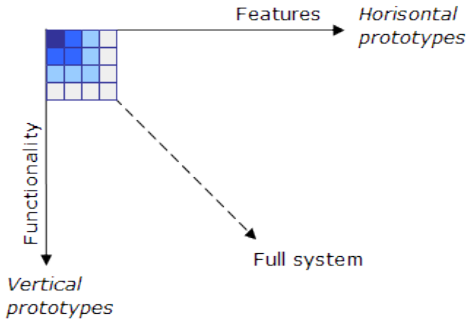


Figure 4.8: Vertical and Horizontal Prototypes.

In other words:

- ***Horizontal prototype:*** A 'dumb' or 'shallow' user interface with not much implemented functionality.
- ***Vertical prototype:*** Only a subset of the user interface is implemented but this subset is implemented in-depth.

4.9.1 Process of Prototyping

The process of prototyping, on the other hand, can be characterised as explorative, experimental, or evolutionary.

- An explorative prototype is used to explore system requirements in cooperation with users and can, as such, be seen as a communication medium and facilitator between user and designer in the same way as a mock-up.
- An experimental prototype is the type of prototype that is closest to the classical concept of prototype ("first of its kind"). It is experimental in the sense that it is built to try to determine whether the planned system will be adequate and acceptable when finished. Experimental prototypes can be used as requirements specification.
- Lastly, prototypes can be evolutionary, meaning that a system evolves through multiple generations/prototypes succeeding each other. Thus, each prototype is an early version of the system that is further worked upon until the prototype has evolved into a finished system

4.9.2 Advantages of Prototypes

The advantages of prototypes are numerous. For example:

- Prototypes incite criticism from users because they may be low-cost and low-fidelity. If a user is presented with an early version of a system that has required substantial work, he/she is likely to be more reluctant (as well as able) to criticise it.
- Prototypes comply with the wish to show fast results to the client
- Prototypes have the advantage of 'rounding' the discussion during a user session; making the sure the session does not get too much off track.
- Not only can the prototype function as a discussion medium between designer and user but also between the members of the design team. Thus, prototypes may help facilitate work across disciplinary borders, bringing together a disparate team.
- Prototypes make it possible to do usability testing early in the development process.
- Prototypes incite and legalise experimentation as they are inexpensive to alter.
- Prototypes focus on content and functionality and turn attention away from details of graphic design.
- Prototypes make it possible to get a formal approval of the design from both programmers and the client before you proceed to the development stage.

4.9.3 Example of a Prototype

Figure 4.13 shows a horizontal prototype of a calendar application for school kids developed in Macromedia Flash. The prototype was used to do early usability testing and as a proof-of-concept.

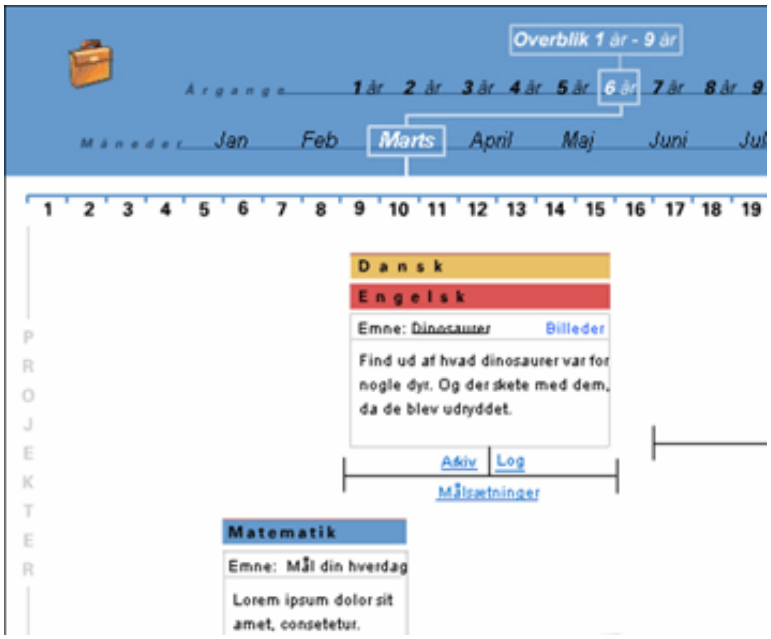


Figure 4.9: Prototype developed in Macromedia Flash of a calendar application.

Self Assessment Questions

4. .....give guidance for database design.

(a) Clustering reports (b) Affinity reports

(c) Financial report (d) All of these

5. Analysts spend 75% of their time on structure the problem space.

(a) True (b) False

6. A .....appears to have a very broad range of the intended future features.

(a) clustering reports (b) vertical prototype

(c) horizontal prototype (d) All of these

7. A .....only has very few features and is almost fully implemented or at least so-called "walking skeletons".

(a) clustering reports (b) vertical prototype

(c) horizontal prototype (d) All of these

4.10 Initial investigation

The initial investigation is conducted in order to gather more information about the problem or opportunity and to decide if the current situation warrants a solution that would require the development of an information system. The two major areas for investigation are the environment and the direct problem.

4.10.1 The Environment

There are a number of aspects that will affect whether and how the project will be conducted.

1. Information System Policy Alternatives: The systems analyst must be aware of policies such as centralization versus decentralization, charge back, vendor selection, outsourcing, and information systems development standards. All of these policies may indirectly affect a project.

2. The Steering Committee: This ongoing committee consists of top management who make business-type decisions about which information systems development projects to initiate and whether or not a project should proceed from one stage to another. Also this committee may give direction to the project team.

3. The Project Team: This ad hoc team will consist of representatives from the user area, systems analyst(s), and any necessary technical resources that may be required as the information systems project proceeds through the SDLC stages.

4.10.2 Investigation of the Problem

When conducting investigations into this initial problem, systems analysts must be aware that they are outsiders and may be treated with distrust. It is important to be able to deal with people and to develop trust and commitment to the task at this early stage of interaction with the user.

The systems analyst may conduct a direct investigation through interviews, questionnaires, or observation. A thorough knowledge of these fact finding techniques will help the systems analyst obtain a detailed understanding of the problem situation. The systems analyst may also carry out indirect investigation. The systems analyst may refer to existing documentation such as procedural flowcharts in order to trace information flows. Also records may be reviewed to determine what data are captured and used throughout the information system. The information gathered in this initial investigation will be used now to determine the likelihood that solving the problem will be beneficial to the organization. Project feasibility will be tested in four ways.

Operational feasibility investigates whether there will be sufficient support for the project from management and users.

Technical feasibility determines if the necessary technology exists and is capable of providing adequate service.

Economic feasibility represents a general determination of whether the resulting benefits will exceed the anticipated costs of the solution.

Schedule feasibility determines if the resulting solution can be implemented within a time frame that will prove beneficial to the organization.

An initial investigation report is prepared at the end of this stage to document the specific problem and what has been determined through the initial investigation. This report represents closure to a reported problem and allows management, through the steering committee, to make a decision about the allocation of scarce resources to the resolution of a business problem. The result of this initial investigation will be a recommendation to either take no action, resolve the problem through personnel changes (either reassignment or training), or continue with a subsequent stage of the SDLC.

4.11 Summary

- The system planning phase is the starting point for the systems analysis and design process.
- Depending on the size of the organization, this proposal review process may be formal or informal.
- A preliminary investigation is performed where the facts surrounding the project are researched.
- A business functional model is defined by analyzing major functional areas of a business.
- Data architecture is derived from the business function model by combining information requirements into generic data entities and subject databases.

4.12 Keywords

Alignment Methodologies: The methodologies in the "alignment" category align IS an objective with organizational goals.
Impact Methodologies: It helps to create and justify new uses of IT.
Life Cycle: Planning can help assure you that you are addressing issues of information management throughout the life cycle of information assets.
Prototyping: It is a method used by designers to acquire feedback from users about future designs.
System Planning: It is the first phase of the life cycle, and when it is completed, enough work will have been done to initiate the individual projects identified in the plan.

4.13 Review Questions

1. What do you mean by system planning?
2. Why system planning is necessary?
3. What are the characteristics of strategic IS planning?
4. Explain business systems planning (BSP).
5. Discuss the concepts of managerial and operational MIS planning.
6. How can we determine the user's requirements?
7. Describe the strategies for information requirements determination.
8. Explain the process of prototyping model.
9. Discuss the advantages of prototypes.
10. What do you mean by initial investigation?

Answers for Self Assessment Questions

1 (c) 2 (b) 3 (a) 6 (a) 7 (a) 8 (c) 9 (b)

4.14 Further Studies

1. http://www.bcanotes.com/Download/sad/System%20Analysis%20And%20Design/Systems%20Planning%20and%20the%20Initial%20Investigation.pdf

5

Information Gathering

STRUCTURE

5.0 Objectives

5.1 Introduction

5.2 Nature of Information

5.3 Source of Information

5.4 Information Gathering Technique

5.5 Samples of Existing Documents

5.6 Forms and Database

5.7 Research and Site Inspection

5.8 Site Observation

5.9 Questionnaires

5.10 Interview

5.11 Types of Interviews

5.12 Conducting an Interview

5.13 Summary

5.14 Keywords

5.15 Review Questions

5.16 Further Studies

5.0 Objectives

After studying this chapter, you will be able to:

- Understand the nature of information
- Discuss about information gathering technique
- Explain about samples of existing document
- Understand the research and site inspection
- Explain about site observation
- Explain about interviews
- Define types of interviews

- Understand the conducting an interview

5.1 Introduction

Information gathering is a very key part of the feasibility analysis process. Information gathering is both an art and a science. It is a science because it requires a proper methodology and tools in order to be effective. It is an art too, because it requires a sort of mental dexterity to achieve the best results.

Gathering is therefore the process of gathering information about the present system. We must know what information to gather, where to find it, how to collect it, and ultimately how to process the collected information.

5.2 Nature of Information

Information as a resource has been a topic of discussion of academics and practitioners in various subject fields especially in the field of economics. Ideas of information economy with information as the transforming resource for post-industrial society. These authors helped develop both the idea of information as a resource and the idea of information-resources management. In the wake of information and communication technologies, confirmed that information has become a commodity. Yet requirements for economic enhancement are not necessarily requirements for development in rural communities when it comes to the basic survival of people.

5.2.1 Attributes of Information

While investigating the validity of information as a resource, the natural approach would be to compare attributes of information with those of other resources, in order to find some commonality. A comparison of this nature necessitates a closer look at the attributes of the different resources. The nine basic similarities between information and other traditional resources to fit into a resource-management framework, namely:

- Information is acquired at a definite, measurable cost.
- Information has a definite value, which may be quantified and treated as ac-countable asset.
- Information consumption can be quantified.
- Cost-accounting techniques can be applied to help control the costs of information.
- Information has a clear life cycle: definition of requirements, collection, transmission, processing, storage, dissemination, use, and disposal.
- Information may be processed and refined, so that raw materials (e.g., databases) are converted into finished products (e.g., published directories).
- Substitutes for any specific item or collection of information are available, and may be quantified as more expensive, or less expensive.
- Choices are available to management in making trade-offs between different grades, types and costs of information.

Value of information

Unlike other tangible resources, information is not readily quantifiable that is, it is impossible to predict the ultimate value of information to its users. Also, over time, there is no predictable change in the value of information.

Multiplicative quality of information
The results produced by the use of information differ greatly from those produced by the use of other resources for instance, information is not lost when given to others, and does not decrease when 'consumed': sharing information will almost always because it to increase that is, information has a self-multiplicative quality.

Dynamics of information
Information cannot be regarded as a static resource to be accumulated and stored within the confines of a static system. It is a dynamic force for change to the system within which it operates. It adds value to an organization through encouraging innovation and change without being tangible.

Life cycle of information
Information seems to have an unpredictable life cycle. Ideas come into, go out of, and finally come back into, fashion.

Individuality of information
Information comes in many different forms, and is expressed in many different ways. Information can take on any value in the context of an individual situation. This proves that, as a resource, information is different from most other resources. The very fact that information is characterized as a dynamic force, "constantly altering and extending a store of knowledge", corresponds with situations in development in which outside information is offered to focus groups to alter their understanding of certain practices, which in turn can help them solve problems (such as improving food security or standards of living).
Apart from the attributes identified the following, also containing elements of intangibility, may be added to the list:

Alleviation of uncertainty
Information is the resolution of uncertainty. This is perhaps one of the intangible attributes best known among a variety of researchers.

Interdependency
Information almost always forms part of technology - it is the "soft" part. Without its information component, technology has little value as a resource for potential users who are not familiar with its workings or its background. With regard to developing rural communities, one should bear in mind that it is not necessarily new technology that brings about these achievements. All outside technology applied for the first time could be viewed as new to the user group or that particular situation, and could have similar effects.

Enhancement of economic growth
A frequent complaint is that information is often denied its role as a resource yet, when looking at the effect of information in development situations, there seems to be an underlying awareness of its importance. The impact of new technology (including information as the hidden component) in agriculture on the basis of such aspects as increased farm income, reduced risk, resource conservation, improved health, better (food) security, and overall economic growth.

Extension of the knowledge base

From a development point of view, there is more emphasis, first, on improving peoples' lives socially and only secondly on economic improvement. In development, outside technology is often introduced with the help of education, training and visual demonstrations. Training helps people in rural communities to expand horizons, increase perceptions, enhance competencies, enlarge sense of perspective, and enhance self-esteem. The above seem to emphasise the impact of the dynamic force of information, where the "extension or altering of people's stores of knowledge" positively affected their social well-being. Thus, although information is an intangible entity, it has the ability to bring about change for the better; which is the ultimate goal of development.

Context dependency

The value of information as a resource in rural development depends largely on situation-specific issues: for example, one could argue that agriculture-related information is mostly technical in nature. However, people with little exposure to modern society have many related issues they need to know about. for example, identified certain types of basic information needed for the development of crop production by traditional farmers; inter alia, information about agricultural input (seeds, fertilizer, etc.), extension, technology (farming equipment, etc.), implementation techniques (sloughing, sowing, pest and weed control), soil, water and climatic conditions, conservation, credit, marketing and infrastructure.

Culture dependency

Another attribute of information that can influence its usefulness as a development resource is that it is culture dependent - involving conceptual and cognitive differentiation. Information is culture specific, it is incommunicable unless acculturated that is, adapted for the cultural environment or the cultural mind-set of the recipient group. Here, Shields and information is not totally value-free, but is socially conditioned and shaped by the social structures that apply it. This aspect has serious implications for developers' efforts to transfer information to the rural communities of developing countries.

Medium dependency

Information is not only culture dependent, but also medium dependent. Once information is concretized outside the human memory it should be packaged in some or other format (i.e., print, images, sound, electronic digits, etc.) to be communicated to someone else. Unless receivers know how to use that particular format, the information will remain inaccessible and rendered useless; for example, an electronic medium directed at users who are unfamiliar with such facilities can impede access to available information. Thus, medium dependency of information can have serious implications for quite a number of rural people who are dependent on oral communication, owing to their oral tradition and the fact that many of them are not literate. This attribute could cause information to be a less useful resource when compared with other resources needed for development purposes.

Conversion dependency

It is a well-known fact that information is not used in the original form offered by its creator alone often, it needs to be adapted to suit a particular situation or specific circumstances. It can also happen that only a small chunk of the original information is used together with other chunks of information to form a new information package needed for a particular situation. In this way, more value can be added to the appropriateness of information. Particularly in a situation where outside information

from the industrialised world is used to improve a practice in rural development, the information content needs to be adapted to bring it to the level of understanding of potential recipients.

5.2.2 Suitability of Attributes

From the above, it is evident that information differs phenomenally from other resources needed for development especially because it is intangible in nature. Although some of its attributes make it a suitable development resource (such as its ability to act as a dynamic force), others render it less suitable (such as its inaccessibility owing to its dependency on culture and media).

Caution

The view of information as a corporate resource, which like other resources such as people, money, raw materials, equipment and energy, should be managed to give a competitive edge.

5.3 Source of Information

Sources of information can be people, letters, books, files, films, tapes in fact, anything which journalists use to put news stories together. Sources are very important if you want to report on events or issues and explain the world to your audience. Journalists try to work as much as possible from their own observations, but this is often not possible. Some events or issues are finished before the journalist gets there. Others are like plants which only show their stem and leave above the ground the all-important roots are hidden from sight. Journalists who only report what they see can miss much of the news unless they have sources to tell them of more details or other aspects which are out of sight.

5.3.1 Types of Sources

Journalists should deal in reliable facts, so it is important that the sources you use for writing stories can give you accurate information about what happened or what was said. But just as there are lots of different news events, so there are many different sources of information. Some of them will give you very accurate information and we call these sources reliable (because we can rely on what they say). Others are less reliable, but still useful, while some can hardly be trusted at all. The main way of judging sources of information is on their reliability.

5.3.2 Reporters

One of the most reliable sources of information (although not completely reliable) is other journalists. They may be your colleagues or reporters from a news agency which supplies your organisation. If they are well trained, experienced and objective, their reports will usually be accurate and can be trusted. However, if there are any essential facts missing from their reports, these will have to be provided. Either they will have to provide them or you will have to find the missing facts yourself. Mistakes can happen. This is why news organisations should have a system for checking facts. In small newsrooms, where the reporter may also be the editor or newsreader, the reporter must be especially careful in checking facts.

There is also the danger that reporters misinterpret what they think they see and then present that as a fact. This often happens when reporting such things as the size of a crowd. Unable to count every person in it, they make an estimate, often sharing their guesses with other journalists on the scene. This is just an estimate and any report which says “there were 40,000 people present” should be treated with caution, unless the reporter knows the exact number who came through the gate.

All sources, including reporters, are said to be reliable if we think they can be believed consistently. If a source is always correct in the information they provide, we will believe them next time. If they make a mistake, we may doubt what they say. Reliability is built up over time.

Your personal reliability as a journalist is important. If you have a good record for fair and accurate reporting, you will be believed. If you get a reputation for being careless in your work or biased in your interpretation, your colleagues, readers or listeners will not be able to rely upon you. In all cases it is better only to report what you know and make it clear in your report that everything else is either an estimate, an opinion or the word of someone else, perhaps a witness. You must always try to give precise facts and attributed opinion. If you cannot do that, you can use phrases like "it is believed that ..." or "it appears that ...". It is better to do this than to leave your readers or listeners believing that what you have said is a proven fact.

Caution

A reporter's story should be checked by the news editor then the sub-editor.

5.3.3 Primary Sources

Often the source is someone at the centre of the event or issue. We call such people primary sources. It might be a man who fell 1,000 meters from an aircraft and lived to tell the tale; or a union leader who is leading wage negotiations. They are usually the best sources of information about their part of what happened. They should be able to give you accurate details and also supply strong comments. For example the fall survivor might say: "I saw the ground rushing up towards me and I kept thinking 'So this is death'." The union leader might warn: "If the employers want blood on their hands, we are ready to supply it."

Of course, just because a person was present at an event does not mean that they are either accurate or fair. The fall survivor may have injured his head after landing and so be confused. The union leader will want to present his side in the best light. It is vital to double-check and cross-check facts with other sources. A word of warning here: If any of your sources, however reliable, gives you information which is defamatory, you can still be taken to court for using it. You are responsible for deciding whether or not to publish the defamatory material.

5.4 Information Gathering Technique

A strategy should be evolved by the analyst to gather information. The strategy consists of identifying information sources, evolving a method of obtaining information from the identified sources and using an information flow model of organization

5.4.1 Information Sources

The main sources of information are users of the system, forms and documents used in the organization, procedure manuals, rule books etc, reports used by the organization and existing computer programs. Information is available from a variety of sources through a variety of means.

The many different sources available to you at university and beyond will include the following:

- Journals
- Text books
- Theses
- Newspapers
- CD ROM
- Internet (on-line information)

- Audio/video tapes
- Microfilms and microfiche
- Slides
- Your own notes and experiences
- Your academic staff and other departmental resources
- The work of friends and other students.

5.4.2 Searching for Information

Information can be gathered by interviewing top-level management, middle level management and operational staff. Besides Interviews group discussions also help the analyst to gather information. It is not possible to obtain all information in a single interview; more than one interview is thus required.

5.4.3 Planning an Interview

Before starting the interview the analyst must make a list of people to be interviewed and in what order, plan and note down a list of questions to be asked, plan several interviews with same person-mainly to clarify doubts and interview groups as appropriate.

5.4.4 Interviewing Technique

There are some guidelines to ensure a successful interview:
Make a prior appointment with the person to be interviewed and meet him at the allotted time. Read background material and go prepared with the checklist. State purpose of interview. Be punctual and pay attention to what user says. Do not use computer jargon. Obtain both quantitative and qualitative Information. Discriminate between essential and desirable requirements. State what you understand and get it confirmed. Do not prolong interview and summarize the information gathered by you during the interview and verify this with the user.

5.4.5 Use of Questionnaires

Questionnaires are useful for collecting statistical data. Sometimes the questionnaires are not promptly replied and several follow-ups/personal interviews may be required to get questionnaires back from respondents. But if the questionnaires are short the probability of getting the reply is high when data has to be collected from large number of people questionnaires are useful.

5.5 Samples of Existing Documents

Sampling is the process of collecting sample documents, forms, and records.

5.5.1 Collecting Facts from Existing Documentation

- The first document the analyst should seek out is the organizational chart.
- Next, the analyst may want to trace the history that led to the project. To accomplish this, the analyst may want to collect and review documents that describe the problem. These include:
  - Interoffice memoranda, studies, minutes, suggestion box notes, customer complaints, and reports that document the problem area.
  - Accounting records, performance reviews, work measurement reviews, and other scheduled operating reports.
  - Information systems project requests – past and present.
- There are usually documents that describe the business function being studied or designed. These documents may include:

- The company's mission statement and strategic plan.
- Formal objectives for the organization sub-units being studied.
- Policy manuals that may place constraints on any proposed system.
- Standard operating procedures (SOPs), job outlines, or task instructions for specific day-to-day operations.
- Completed forms that represent actual transactions at various points in the processing cycle.
- Samples of manual and computerized databases.
- Samples of manual and computerized screens and reports.

- Do not forget to check for documentation of previous system studies and designs performed by systems analysts and consultants. This documentation may include:
  - Various types of flowcharts and diagrams.
  - Project dictionaries or repositories
  - Design documentation, such as inputs, outputs, and databases.
  - Program documentation.
  - Computer operations manuals and training manuals.
- All documentation collected should be analyzed to determine currency of the information.
- Do not discard outdated documentation.
  - Just keep in mind that additional fact-finding will be needed to verify or update the facts collected.
- As you review existing documents, take notes, draw pictures, and use systems analysis and design tools to model what you are learning or proposing for the system.

5.5.2 Document and File Sampling Techniques

Two commonly used sampling techniques are randomization and stratification.

Randomization: Randomization is a sampling technique characterized as having no predetermined pattern or plan for selecting sample data. Therefore, we just randomly choose 25 invoices.

Stratification: Stratification is a systematic sampling technique that attempts to reduce the variance of the estimates by spreading out the sampling for example, choosing documents or records by formula -- and by avoiding very high or low estimates.

For computerized files, stratification sampling can be executed by writing a sample program.

Did You Know?

Edgar F.Codd, the inventor of the relational model, introduced the concept of normalization and what we now know as the First Normal Form (1NF) in 1970.

5.6 Forms and Database

The normal forms defined in relational database theory represent guidelines for record design. The guidelines corresponding to first through fifth normal forms are presented here, in terms that do not require an understanding of relational theory. The design guidelines are meaningful even if one is not using a relational database system. We present the guidelines without referring to the concepts of the relational model in order to emphasize their generality, and also to make them easier to understand. Our presentation conveys an intuitive sense of the intended constraints on record design, although in its informality it may be imprecise in some technical details. A comprehensive treatment of the subject is provided by date. The normalization rules are designed to prevent update anomalies and data inconsistencies. With respect to performance tradeoffs, these guidelines are biased toward the assumption that all non-key fields will be updated frequently. They tend to penalize retrieval, since

data which may have been retrievable from one record in a normalized design may have to be retrieved from several records in the normalized form. There is no obligation to fully normalize all records when actual performance requirements are taken into account.

5.6.1 First Normal Form

First normal form deals with the "shape" of a record type. Under first normal form, all occurrences of a record type must contain the same number of fields.

First normal form excludes variable repeating fields and groups. This is not so much a design guideline as a matter of definition. Relational database theory does not deal with records having a variable number of fields.

5.6.2 Second Normal Form

Second normal form is violated when a non-key field is a fact about a subset of a key. It is only relevant when the key is composite, i.e., consists of several fields. Consider the following inventory record:

PART	WAREHOUSE	QUANTITY	WAREHOUSE-ADDRESS

The key here consists of the PART and WAREHOUSE fields together, but WAREHOUSE-ADDRESS is a fact about the WAREHOUSE alone. The basic problems with this design are:

- The warehouse address is repeated in every record that refers to a part stored in that warehouse.
- If the address of the warehouse changes, every record referring to a part stored in that warehouse must be updated.
- Because of the redundancy, the data might become inconsistent, with different records showing different addresses for the same warehouse.
- If at some point in time there are no parts stored in the warehouse, there may be no record in which to keep the warehouse's address.

To satisfy second normal form, the record shown above should be decomposed into (replaced by) the two records:

PART	WAREHOUSE	QUANTITY	WAREHOUSE	WAREHOUSE-ADDRESS

5.6.3 Third Normal Form

Third normal form is violated when a non-key field is a fact about another non-key field, as in

EMPLOYEE	DEPARTMENT	LOCATION

The EMPLOYEE field is the key. If each department is located in one place, then the LOCATION field is a fact about the DEPARTMENT -- in addition to being a fact about the EMPLOYEE. The problems with this design are the same as those caused by violations of second normal form:

- The department's location is repeated in the record of every employee assigned to that department.
- If the location of the department changes, every such record must be updated.
- Because of the redundancy, the data might become inconsistent, with different records showing different locations for the same department.
- If a department has no employees, there may be no record in which to keep the department's location.

To satisfy third normal form, the record shown above should be decomposed into the two records:

EMPLOYEE	DEPARTMENT	DEPARTMENT	LOCATION

5.6.4 Functional Dependencies

In relational database theory, second and third normal forms are defined in terms of functional dependencies, which correspond approximately to our single-valued facts. A field Y is "functionally dependent" on a field (or fields) X if it is invalid to have two records with the same X-value but different Y-values. That is, a given X-value must always occur with the same Y-value. When X is a key, then all fields are by definition functionally dependent on X in a trivial way, since there cannot be two records having the same X value.

There is a slight technical difference between functional dependencies and single-valued facts as we have presented them. Functional dependencies only exist when the things involved have unique and singular identifiers (representations). For example, suppose a person's address is a single-valued fact, i.e., a person has only one address. If we do not provide unique identifiers for people, then there will not be a functional dependency in the data:

PERSON	ADDRESS
Ram Sharma	123 Main St., United Staet
Ram Sharma	321 Center St., India

Although each person has a unique address, a given name can appear with several different addresses. Hence we do not have a functional dependency corresponding to our single-valued fact.

5.6.5 Fourth Normal Form

Under fourth normal form, a record type should not contain two or more independent multi-valued facts about an entity. In addition, the record must satisfy third normal form.

The term "independent" will be discussed after considering an example.

Consider employees, skills, and languages, where an employee may have several skills and several languages. We have here too many-to-many relationships, one between employees and skills, and one between employees and languages. Under fourth normal form, these two relationships should not be represented in a single record such as-

EMPLOYEE	SKILL	LANGUAGE

Instead, they should be represented in the two records

EMPLOYEE	SKILL	EMPLOYEE	LANGUAGE

Note that other fields, not involving multi-valued facts, are permitted to occur in the record, as in the case of the QUANTITY field in the earlier PART/WAREHOUSE example.

The main problem with violating fourth normal form is that it leads to uncertainties in the maintenance policies. Several policies are possible for maintaining two independent multi-valued facts in one record:

(1) A disjoint format, in which a record contains either a skill or a language, but not both:

EMPLOYEE	SKILL	LANGUAGE

Neeraj	Copy	
Neeraj	Type	
Neeraj		French
Neeraj		German
Neeraj		Greek

This is not much different from maintaining two separate record types. (We note in passing that such a format also leads to ambiguities regarding the meanings of blank fields. A blank SKILL could mean the person has no skill, or the field is not applicable to this employee, or the data is unknown, or, as in this case, the data may be found in another record.)

(2) A random mix, with three variations:

(a) Minimal number of records, with repetitions:

EMPLOYEE	SKILL	LANGUAGE
Neeraj	Copy	French
Neeraj	Type	German
Neeraj	Type	Greek

(b) Minimal number of records, with null values:

EMPLOYEE	SKILL	LANGUAGE
Neeraj	Cook	French
Neeraj	Type	German
Neeraj		Greek

(c) Unrestricted:

EMPLOYEE	SKILL	LANGUAGE
Neeraj	cook	French
Neeraj	type	German
Neeraj	type	Greek

(3) A “cross-product” form, where for each employee, there must be a record for every possible pairing of one of his skills with one of his languages:

EMPLOYEE	SKILL	LANGUAGE
Neeraj	cook	French
Neeraj	cook	German
Neeraj	cook	Greek
Neeraj	type	French
Neeraj	type	German
Neeraj	type	Greek

Self Assessment Questions

1. …….has a definite value, which may be quantified and treated as an accountable asset.

(a). Data (b). Information

(c). Raw material (d). None of these

2. Cost-accounting techniques can be applied to help control the costs of information.
(a). True (b). False

3. ……..is the value of information as a resource in rural development depends largely on situation-specific issues.
(a). Culture dependency (b). Medium dependency
(c). Context dependency (d). None of these

4. .....of information can have serious implications for quite a number of rural people who are dependent on oral communication.
(a). Culture dependency (b). Medium dependency
(c). Context dependency (d). Conversion dependency

5. Written reports cannot make an excellent source of information for a journalist.
(a). True (b). False

5.7 Research and Site Inspection

5.7.1 Reason for Site Inspection

Member Site Inspection Panels are used as one means of ensuring that Members have sufficient information about the site and proposal to reach a decision at a subsequent Development Management Committee meeting. They are used selectively as fact finding exercises to supplement officer's reports and presentations when the impact of the proposed development is difficult to visualise and the issues finely balanced. As information gathering exercises, they are not formal decision making meetings, though the panel may reach a view on the specific issues which needed clarification on site. The decision to hold a site inspection is taken in Committee, resulting from a member request and the specific reason to hold such an inspection will be minute.

5.7.2 Composition of Inspection Panel

All Members of the Development Management Committee have the right of attendance at site inspections, however there should be at least four and normally, no need for more than eight Members on the Inspection Panel. The Panel Chair will be decided at the Committee meeting where the decision to hold a site inspection was made.
Those invited to attend are:

- The applicant and agent (to respond to questions only);
- Two representatives of the relevant District/Borough Council;
- Consulters whose comments are pertinent to the planning issues under consideration (e.g., Highway Authority/ Environmental Health)
- Two representatives of the relevant parish council(s) (one only to act as spokesperson).

5.7.3 Conduct of Site Inspection

The Site Inspection will proceed as follows:

- The Chairman will welcome those attending and explain the purpose and procedures of the site inspection, emphasising that no decision will be made on site and the application will be determined at a subsequent Committee meeting;

- The planning officer will describe the application, and outline the relevant planning considerations and policies, paying particular attention to the specific issues that warranted the need for the site inspection, using plans if necessary;
- Officers of invited consultees will present their advice and comments;
- The applicant and/or agent to respond to any questions
- The views and comments of the Parish Council and then the District/Borough Council will be invited;
- Any further comments and questions from Authority Members ;
- The Chairman will close the site inspection, thank all parties for their attendance and inform them of the date and time of the Development Management Committee at which the application will be determined;
- Site Inspection Panel Members to consider issues raised in the light of the Inspection and form a collective view to be included in the officer's report to the Development Management Committee.
- Notes of the Site Inspection are taken on the day and will be included with the officer's report to the following Committee meeting.

5.8 Site Observation

It is the process of recognizing and noting people, objects and occurrences to information. The Analyst has to be an information seeker who is unbiased. The following questions can serve as a guide for on-site observations:

- What kind of system is it? What does it do?
- Who runs the system? Who are the important people in it?
- What is the history of the system? How did it get to its present stage of development?
- Apart from its formal function, what kind of system is it in comparison with other systems in the organization?
- Is it a primary or a secondary contributor to the organization?
- Is it fast paced or is it a leisurely system that responds slowly to external crisis?

5.8.1 Observation Methods

Natural: A natural observation occurs in a setting such as the employee's place of work.

Contrived: A contrived observation is set up by the observer in a place like a laboratory.

Obtrusive: An obtrusive observation takes place when the respondent knows he/she is being observed.

Unobtrusive: An unobtrusive observation takes place in a contrived way such as behind a one way mirror.

Direct: A direct observation takes place when the analyst actually observes the subject or the system at work.

Indirect: In an indirect observation the analyst uses mechanical devices such as cameras and video tapes to capture information

Structured: In a structured observation the observer looks for and records a specific action

Unstructured: These methods place the observer in a situation to observe whatever be pertinent at that time

5.8.2 Problems in On Site Observation

- Intruding into the user's area often results in adverse reactions by the staff, therefore adequate preparation and training are important
- Attitudes and motivations cannot be readily observed
- Observations are subject to error due to the observer's misinterpretation

5.9 Questionnaires

Questionnaire is a self-administered tool that is more economical and requires less skill to administer than the interview. At any point in time, unlike the interview, feedback from many respondents can be collected at the same time. Since questionnaires are usually self-administered it is critical that questions be clear and unambiguous.

5.9.1 Types of Questionnaire

Fill-in-the-blanks Questions: They seek specific responses.

Yes / No Questions: They just seek one value either true or false or Yes or NO. There is no mixed response.

Ranking Scale Questions: The respondent need to rank the responses into a certain scale. For example to a question you might be asked to rate a service from a level 1 to 5.

Multiple-Choice Questions: They ask for a specific answer choices.

With the help of a table below we can understand the differences between the questionnaires and an interview. This is designed to give a completely unbiased viewpoint of both methods. We will be able to view them in such a way that the benefits and shortcomings of each will be easily visible right away.

Table 5.1: Comparison between Questionnaire and Interview

Questionnaire	Interview
Economical	Less Economical
Can be completed by many people at the same time	Can be administered to ONLY ONE person at a time
Chances of error or omissions are fewer	It could be error prone since it depends upon the skill of the interviewer to gauge the questions and interpret the responses.
Anonymity can be maintained. Hence user is not prevented from giving his candid opinion about an issue	Anonymity is not maintained. Hence the user might feel forced to conceal his candid opinion on an issue.
Gives time to the respondents. Hence they can think and give their regarded opinions on an issue	It may not give time to the respondents. Hence they may not get enough time to think and give their opinion on an issue

5.10 Interview

5.10.1 Meaning of an Interview

An interview can be defined as an oral tool to test a candidate's traits for employment or admission to a premiere institution of learning. Being an oral test, it calls for your skills of oral and non-verbal communication to support your performance before a panel of experts. There are different types of interviews, such as Panel interview. Each type of interview requires your attention for a careful application of a particular set of communication skills.

5.10.2 Purpose of an Interview

The interview is one of the most important phases of the job search process. Your resume and cover letter are simply tools to get you to the interviewing stage. The interview is your opportunity to convince an employer that you are the right person for the job.

As the ***interviewee***, the main purposes of the interview are to:

- Communicate information about yourself, your experience and your abilities
- Seek further information about the position and the organization
- Evaluate the match between your needs and what the job offers

The main purposes of the interview for the ***interviewer*** are to gather relevant information about the candidate's:

- Interview preparation—interest in and knowledge of the industry, the position and the organization.
- Communication skills—oral presentation skills and the ability to interact with others.
- Qualifications—academic, work, volunteer and other experience
- Leadership potential and teamwork—demonstrated ability to work with others and to get others to work together.
- Clear and realistic career goals—future plans and awareness of career paths
- Self awareness realistic appraisal of self
- Motivation and success potential—enthusiasm for the position; demonstrated patterns of accomplishment
- Work ethic—acceptance of responsibility, ability to keep commitments and attitude of the importance of work.

Did You Know?

Interviews with celebrities conducted by Interview magazine, co-founded by Andy Warhol in 1969.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Draw the table of comparison between questionnaire and interview.

……

……

……

5.11 Types of Interviews

Interviews can take a variety of formats. Familiarize yourself with all of them so that you can develop realistic expectations concerning your role and responsibilities during the interview.

5.11.1 Lunch/Dinner Interview

This type of interview conducted in an informal environment. Be careful about your body language and manners. Do not order liquor even if your boss does.

5.11.2 Preliminary Interview

As a screening test, it should be attended by you with full preparation and care. It is an essential stage to be cleared for moving to final interview. Use your communication skills to express your

confidence and ease.

5.11.3 Sequential Interview

In this form of interview you have report to several people at the same time. You should very careful in your behaviour, speech, and a manner as each member is your prospective boss.

5.11.4 Skill-based Interview

Such interview requires you demonstrate your skills. If you are seeking to be an actor' a salesperson, you can be asked to act or read a script or salesperson demonstrate your pitch of voice and salespersons' persuasive oral communication.

5.11.5 Academic Interview

Academic Interview is conducted Q-A (questions-and-answers) environment. You should able to demonstrate your skills of attentive listening, eye contact, and clarity of ideas and depth of knowledge.

5.11.6 Personality Interview

This form of interview evaluates as an individual in terms of your response to certain situations set before you. The emphasis is not on what answer, but on how you answer. The whole range of verbal communication skills are brought into play to demonstrate you as an "all round personality."

5.11.7 Screening Interview

Employers sometimes use a human resource specialist to interview and screen out applicants who do not meet the basic position requirements. These are usually short interviews used to screen for specific experience, skills and/or education. Job offers do not come as a result of this interaction.

5.11.8 Telephone Interview

After reviewing your resume and cover letter, hiring authorities may conduct phone interviews as a second screen before inviting you to a panel interview. Do not underestimate the importance of making a positive impression during the telephone interview. During a telephone interview, you may use written notes of your responses to standard interview questions. However, if you read responses, it will be obvious to the telephone interviewer that you are not being spontaneous.

This type of interview can happen unexpectedly or be planned. No matter how it occurs, you must answer the questions effectively in order to move to the next stage of the process which is the in-person interview.

Guidelines for a telephone interview:

- If it is scheduled, make it for a time when you can give 100% of your attention.
- If it is not scheduled, and the timing is not convenient, let the employer know. Ask if you can call back at a more convenient time.
- Take the call in a quiet place.
- Jot down key points you want to make and questions you want to ask ahead of time.
- Keep a copy of your resume and the job description nearby.

5.11.9 One-on-One Interview

There are only interviewee and one interviewer. You may have individual interviews with several members of an organization.

5.11.10 Sequential Interviews

You may be passed from one department or manager to another, with a one-on-one interview taking place at each step. Ask the scheduler to explain who you will be meeting with for each interview. Prepare questions that would be appropriate for each person. Your questions should stem from your in-depth research about the organization's goals, operations, culture and employees.

5.11.11 Panel Interview

You are interviewed by a committee, often consisting of a chairperson and two to six members from several different departments who have varying levels of responsibility. Typically, the panel has a list of prepared questions and each panel member takes turns asking questions. Expect the panel members to introduce themselves prior to asking the first scripted question. It is important to make eye contact with each panel member and shake each member's hand at the beginning and end of the interview.

5.11.12 Group Interview

In a group interview, you and other interviewees are interviewed simultaneously. The interviewer asks each person questions while the other group members observe. It gives the interviewer the opportunity to screen many applicants at one time and also provides an opportunity to observe group interactions. Show your listening skills by paying attention to what others say during the interview and try not to repeat their responses.

5.12 Conducting an Interview

Interviews have four stages that precede the writing of a story: arrangements, preparation, the actual interview and the reconstruction.

5.12.1 Arrangements

Spontaneous interviews, except in connection with breaking news, seldom contribute to thoroughness. Once you have decided to interview someone, call in advance to make an appointment. Identify yourself by your name and the name of your company. If you feel the need to do so or are asked to describe what the story is about, be brief and general. The shape of the story might change as you continue your reporting. If you are interviewing several persons in connection with your story, interview the principal person last, because you will be better prepared based on what you learn from the earlier interviews.

5.12.2 Preparation

Do as much research as possible in advance on the person and/or topic you are working on. Sources might include the library, public records, the internet and people you know who can provide background information. Prepare your questions in advance in writing and bring them to the interview. Refer to them but do not show them to the interviewee, because it creates too formal an atmosphere. Ask other questions as they might arise, based on what the interviewee says or something new that might come to you on the spur of the moment. Bring two pencils (or pens) and paper. A stenographer's notebook is usually easier to handle than a large pad but use whatever is comfortable. Bring a tape recorder if you can but be sure to get the permission to use it from the person you are interviewing. You also should take notes, because it will help in the reconstruction phase, and, yes, tape recorders fail occasionally.

5.12.3 The Interview

It is inadvisable to launch right into the interview unless you are only being given a few minutes. Some casual conversation to start with will relax both of you. Questions should be as short as

possible. Give the respondent time to answer. Be a good listener. If he or she prattles on, it is appropriate to move on as politely as you can. You might say something such as: “Fine, but let me ask you this”. Try to draw out specifics: How long, how many, when, etc.? Absorb the atmospherics of the locale where the interview takes place, with particular attention to what might be a reflection of the interviewee's personality and interests, such as photos of children or bowling trophies or a paper-littered desk or a clean one, etc. Note characteristics of the interviewee that might be worth mentioning in your story, such as pacing, looking out the window to think, hand gestures and the like. Invite the person to call you if she/he thinks of anything pertinent after the interview. It often happens, so be sure to provide your name, email address and phone number on a card or piece of paper before you leave. If that person has a secretary, be sure to get that person's name and telephone number, too, in case there is some detail that needs follow-up and, again, leave information as to how you may be contacted. If a photo is needed and is not taken during the interview, be sure to make arrangements then to have one taken at a later time.

5.12.4 Reconstruction

As soon as it is practical after the interview, find a quiet place to review your handwritten notes. In your haste while taking notes, you may have written abbreviations for words that would not mean anything to you a day or two later. Or some of your scribbling may need deciphering, and, again, it is more likely you will be better able to understand the scribbles soon after the interview. Underline or put stars alongside quotes that seemed most compelling. One star for a good quote, two stars for a very good one, etc. It will speed the process when you get to the writing stage. One other thing to look for in your notes: the quote you wrote down might not make a lot of sense, unless you remember what specific question it was responding to. In short, fill in whatever gaps exist in your notes that will help you better understand them when writing.

6. A .....observation occurs in a setting such as the employee’s place of work.

(a). contrived (b). unobtrusive

(c). obtrusive (d). natural

7. In a structured observation the observer looks for and records a specific action

(a). indirect (b). obtrusive

(c). structured (d). natural

8. .....is a self-administered tool that is more economical and requires less skill to administer than the interview.

(a). Questionnaire (b). Information

(c). Interview (d). None of these

5.13 Summary

- The information should be seen as something tangible, physical and concrete, while viewpoints from within the information profession emphasise the intangibility of information.
- Medium dependency of information can have serious implications for quite a number of rural people who are dependent on oral communication, owing to their oral tradition and the fact that many of them are not literate.

- The main sources of information are users of the system, forms and documents used in the organization, procedure manuals, rule books etc, reports used by the organization and existing computer programs.
- The normalization rules are designed to prevent update anomalies and data inconsistencies. With respect to performance tradeoffs, these guidelines are biased toward the assumption that all non-key fields will be updated frequently.
- Member Site Inspection Panels are used as one means of ensuring that Members have sufficient information about the site and proposal to reach a decision at a subsequent Development Management Committee meeting.

5.14 Keywords

Culture Dependency: Attribute of information that can influence its usefulness as a development resource is that it is culture dependent.

Frequent Complaint: A frequent complaint is that information is often denied its role as a resource yet, when looking at the effect of information in development situations.

Information: Information may be processed and refined, so that raw materials (e.g., databases) are converted into finished products.

Questionnaires: Questionnaires are useful for collecting statistical data. Sometimes the questionnaires are not promptly replied and several follow-ups/personal interviews may be required to get questionnaires back from respondents.

Randomization: Randomization is a sampling technique characterized as having no predetermined pattern or plan for selecting sample data.

Sampling: Sampling is the process of collecting sample documents, forms, and records.

Stratification: Stratification is a systematic sampling technique that attempts to reduce the variance of the estimates by spreading out the sampling.

5.15 Review Questions

1. What is the information? Explain the nature of information.
2. What are the sources of information? Explain it.
3. Explain the information gathering techniques.
4. What is the sampling? Explain about samples of existing documents.
5. What are the forms and databases?
6. Explain about research and site inspection.
7. Explain about site observation.
8. What are the questionnaires?
9. What is interviews and explain its types?
10. Explain the procedure of conducting interviews.

Answers of Self Assessment Questions

1. (b)	2. (a)	3. (c)	4. (b)
5. (b)	6. (d)	7. (c)	8. (a)

5.16 Further Studies

1. http://www.lboro.ac.uk/service/ltd/campus/infouser.pdf

6

Structured Analysis

STRUCTURE

6.0 Objectives

6.1 Introduction

6.2 What is Structured Analysis?

6.3 Chart

6.4 Data Flow Diagram

6.5 Guidelines for Drawing Dataflow Diagrams

6.6 The Logical and Physical Data Flow Diagram

6.7 Data Dictionary

6.8 Decision Trees

6.9 Structured English

6.10 Summary

6.11 Keywords

6.12 Review Questions

6.13 Further Studies

6.0 Objectives

After studying this chapter, you will be able to:

- Explain the structured analysis
- Discuss the types of charts
- Explain the data flow diagram
- Discuss the guidelines for drawing dataflow diagrams
- Explain the logical and physical data flow diagram
- Discuss the data dictionary
- Explain the decision trees and structured English

6.1 Introduction

The structured analysis is a set of techniques and graphical tools used by the analyst for applying a systematic approach to systems analysis. The traditional approach focuses on cost/benefit and feasibility analyses, project management, hardware and software selection, and personnel

considerations. In contrast, structured analysis uses graphical tools such as data flow diagram, data dictionary, structured English, decision tree, and decision tables. The outcome of structured analysis is a new document, called system specifications, which provides the basis for design and implementation.

6.2 What is Structured Analysis?

The term structured analysis, within the domain of software development, describes the set of techniques used in the design of computer applications. These techniques help explain the required steps within a computer application in a more humanistic manner. The results of a thorough structured analysis and design approach typically describe both the physical and logical layers of the computer application. Software engineering is a complex process that requires intricate detail on the specifics about how the software application will function. The early pioneers of software engineering realized that this complexity required a method of formality that would not only document the system, but also explain the process in terms that could be understood by the general public. Structured analysis is the process that is used for documenting this complexity. Structured analysis and design are broken into four primary domains within application architecture. These are the data flows, data models, structure charts, and state models. All of these domains are typically represented in a manner starting from a summary level and progressing into a detail level of interpretation.

6.2.1 Why Structured Analysis?

The traditional approach to analysis is focused on cost/benefit and feasibility analysis, hardware and software selection and personal considerations. It focuses more on the physical system rather than the logical system. Structured analysis is a new set of techniques and graphical tools that allows the analyst to develop a system specification that is easily understood by the individuals using the system. It focuses more on the logical system. It is a way to focus on functions rather than physical implementations.

- It encouraged more graphical data flow diagrams (DFDs) wherever possible to help communicate better with the user.
- It differentiates between logical and physical systems. It removes the physical checkpoints and introduces the logical equivalents.
- It builds a logical system with system characteristics and inter-relationships before moving to implementation.
- The primary steps in structured analysis are:
- Study the affected system, and user areas resulting in a physical data flow diagram (DFD.)
- Remove the physical checkpoints and replace them with a logical equivalent, thus resulting in a logical data flow diagram (DFD.)
- Model the new logical system
- Establish the man/machine interface. The process modifies the logical data flow diagram (DFD) and considers the hardware needed to implement the system.

Did You Know?

Structured analysis is part of a series of structured methods that represent a collection of analysis, design, and programming techniques that were developed in response to the problems facing the software world from the 1960s to the 1980s.

6.3 Chart

A chart is a graphical representation of data, in which "the data is represented by symbols, such as bars in a bar chart, lines in a line chart, or slices in a pie chart". A chart can represent tabular numeric data, functions or some kinds of qualitative structures.
The term "chart" as a graphical representation of data has multiple meanings:
A data chart is a type of diagram or graph that organizes and represents a set of numerical or qualitative data.

6.3.1 Types of Charts

These are giving following charts

- Bar Charts
- Line Chart
- Pie Charts

Bar Charts

A bar chart is a little more complex. It shows the opening and closing prices, as well as the highs and lows. The bottom of the vertical bar indicates the lowest traded price for that time period, while the top of the bar indicates the highest price paid. The vertical bar itself indicates the currency pair's trading range as a whole. The horizontal hash on the left side of the bar is the opening price, and the right-side horizontal hash is the closing price.
Here is an example of Figure 6.1 a bar chart for EUR/USD:

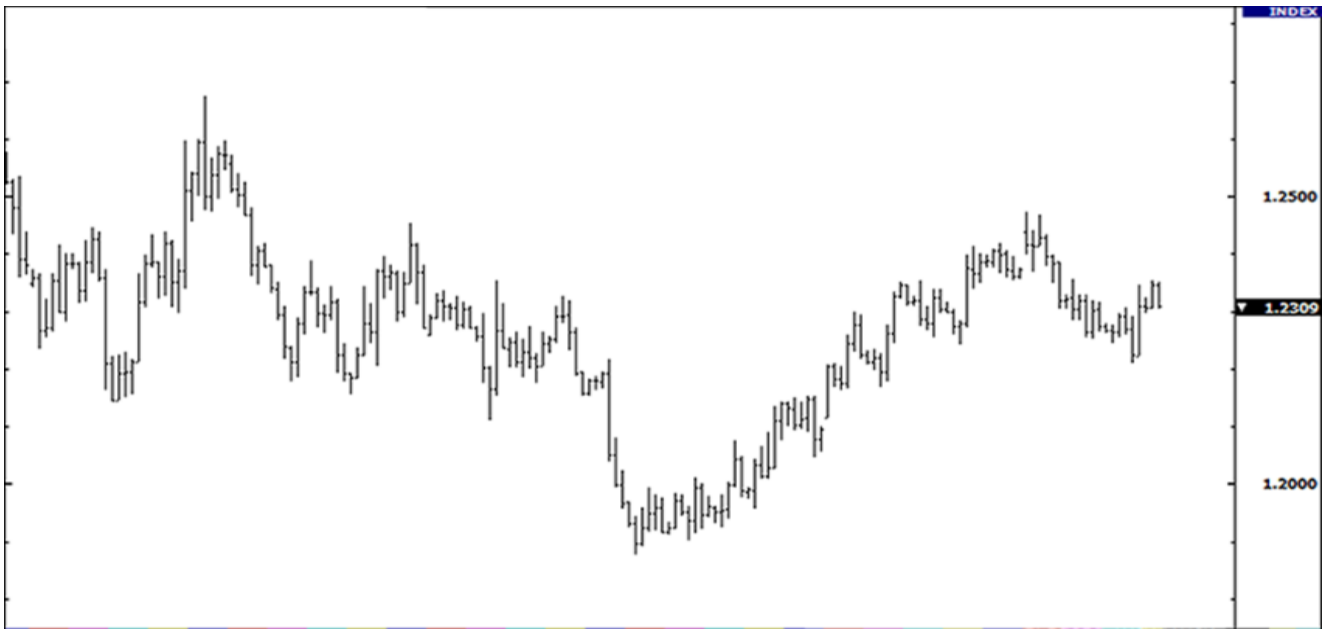


Figure 6.1: The word "bar" in reference to a single piece of data on a chart.

A bar is simply one segment of time, whether it is one day, one week, or one hour. When you see the word 'bar' going forward, be sure to understand what time frame it is referencing. Bar charts are also called "OHLC" charts, because they indicate the open, the high, the low, and the close for that particular currency.
Here's an example of Figure 6.2 a price bar:

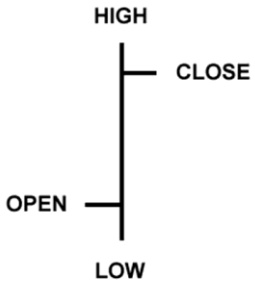


Figure 6.2: The price bar.

- *Open:* The little horizontal line on the left is the opening price
- *High:* The top of the vertical line defines the highest price of the time period
- *Low:* The bottom of the vertical line defines the lowest price of the time period
- *Close:* The little horizontal line on the right is the closing price

Line Charts

A simple line chart draws a line from one closing price to the next closing price. When strung together with a line, we can see the general price movement of a currency pair over a period of time. Here is an example of Figure 6.3 a line chart for EUR/USD:

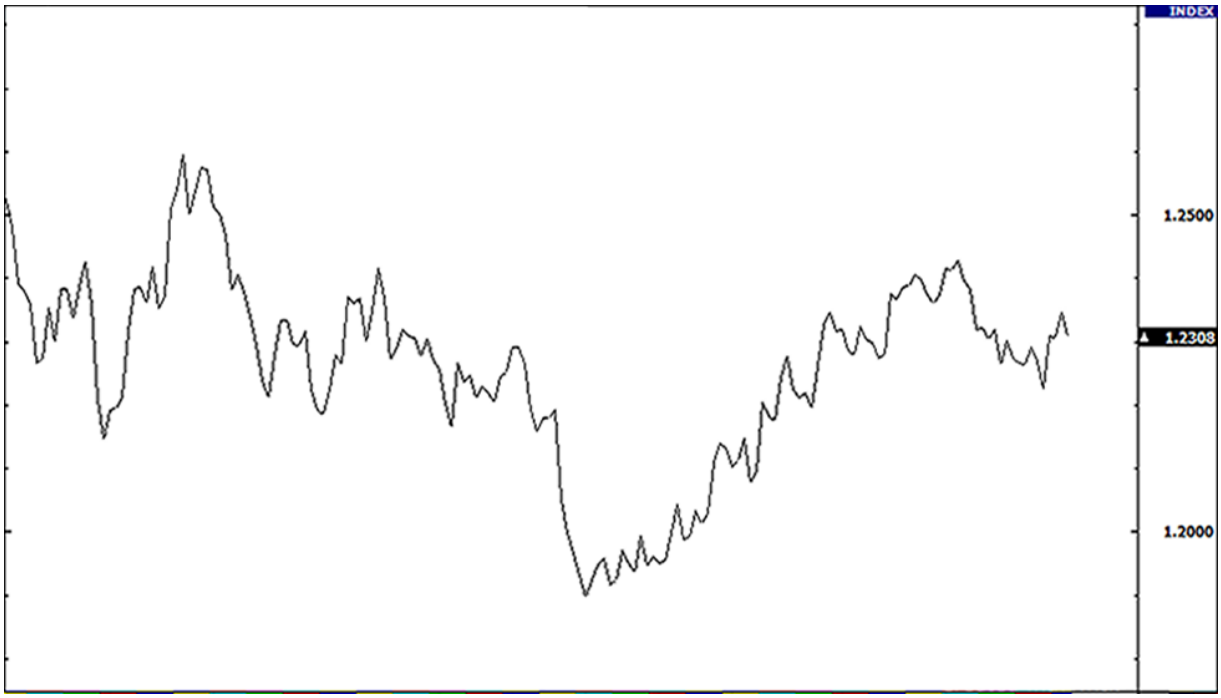


Figure 6.3: The line chart for EUR/USD.

Pie Charts

A pie chart compares parts to a whole. As such it shows a percentage distribution. The entire pie represents the total data set and each segment of the pie is a particular category within the whole. So, to use a pie chart, the data you are measuring must depict a ratio or percentage relationship. You must always use the same unit of measure within a pie chart.

The pie chart in Figure 6.4 shows where ABC enterprise's sales come from.

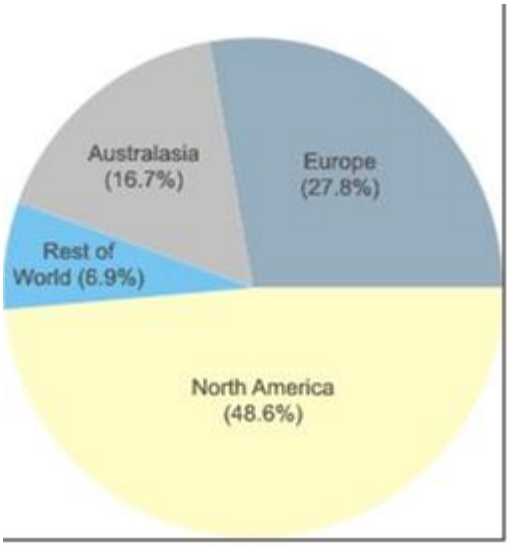


Figure 6.4: Example of a pie chart.

Note 1: Be careful not to use too many segments in your pie chart. More than about six and it gets far too crowded. Here it is better to use a bar chart instead.
Note 2: If you want to emphasize one of the segments, you can detach it a bit from the main pie. This visual separation makes it stand out.
Note 3: For all their obvious usefulness, pie charts do have limitations, and can be misleading.

6.4 Data Flow Diagram

A Data Flow Diagram (DFD) is a diagrammatic representation of the information flows within a system, showing:

- How information enters and leaves the system
- What changes the information
- Where information is stored

In SSADM a DFD model includes supporting documentation describing the information shown in the diagram. DFDs are used not only in structured system analysis and design, but also as a general process modelling tool. There are a number of commercial tools in the market today which are based on DFD modelling.
SSADM uses DFDs in three stages of the development process:

- *Current Physical DFDs*. These record the results of conventional fact finding.
- *Current Logical DFDs.* The logical information processing of the current system
- *Required Logical DFDs.* The logical information processing requirements of the proposed system.

6.4.1 Advantages of DFDs

- A simple but powerful graphic technique which is easily understood.
- Represents an information system from the viewpoint of data movements, which includes the inputs and outputs to which people can readily relate.
- The ability to represent the system at different levels of details gives added advantage.
- Helps to define the boundaries of the system.
- A useful tool to use during interviews.
- Serve to identify the information services the users require, on the basis of which the future information system will be constructed.

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: Draw the data flow diagram of library book system.

……

……

……

Did You Know?
In the 1970s the popularity of flowcharts as an own method decreased when interactive computer terminals and third-generation programming languages became the common tools of the trade, since algorithms can be expressed much more concisely as source code in such a language.

6.5 Guidelines for Drawing Dataflow Diagrams

Naming conventions:

- Processes: strong verbs
- Dataflow: nouns
- Data stores: nouns
- External entities: nouns
- No more than 7–9 processes in each DFD.
- Data flows must begin, end, or both begin and end with a process.
- Data flows must not be split.
- A process is not an analogue of a decision in a systems or programming flowchart. Hence, a dataflow should not be a control signal. Control signals are modelled separately as control flows.
- Loops are not allowed.
- A dataflow cannot be an input signal. If such a signal is necessary, then it must be a part of the description of the process, and such process must be so labelled. Input signals as well as their effect on the behaviour of the system are incorporated in the behavioural model (say, state transition graphs) of the information system.
- Decisions and iterative controls are part of process description rather than dataflow.
- If an external entity appears more than once on the same DFD, then a diagonal line is added to the north-west corner of the rectangle (representing such entity).
- Updates to data stores are represented in the textbook as double-ended arrows. This is not, however, a universal convention. It would rather not use this convention since it can be confusing. Writing to a data store implies that you have read such data store (you cannot write without reading). Therefore, data store updates should be denoted by a single-ended arrow from the updating process to the updated data store.
- ***Conservation Principles:***
  - *Data stores and Data flows*: Data stores cannot create (or destroy) any data. What comes out of a data store therefore must first have got into a data store through a process.
  - *Processes:* Processes cannot create data out of thin air. Processes can only manipulate data they have received from data flows. Data outflows from processes therefore must be derivable from the data inflows into such processes.

- ***Levelling Conventions:***
  - *Numbering:* The system under study in the context diagram is given number '0'. The processes in the top level DFD are labelled consecutively by natural numbers beginning with 1. When a process is exploded in a lower level DFD, the processes in such lower level DFD are consecutively numbered following the label of such parent process ending with a period or full-stop (for example 1.2, 1.2.3, etc.).
  - *Balancing:* The set of DFDs pertaining to a system must be balanced in the sense that corresponding to each dataflow beginning or ending at a process there should be an identical dataflow in the exploded DFD.
  - *Data stores:* Data stores may be local to a specific level in the set of DFDs. A data store is used only if it is referenced by more than one process.
  - *External entities:* Lower level DFDs cannot introduce new external entities. The context diagram must therefore show all external entities with which the system under study interacts. In order not to clutter higher level DFDs, detailed interactions of processes with external entities are often shown in lower level DFDs but not in the higher level ones. In this case, there will be some data flows at lower level DFDs that do not appear in the higher level DFDs. In order to facilitate unambiguous balancing of DFDs, such data flows are crossed out to indicate that they are not to be considered in balancing.

Self Assessment Questions

1. The term.....within the domain of software development, describes the set of techniques used in the design of computer applications.

(a). structured analysis (b). flow chart
(c). data flow diagram (d). None of these.

2. Structured analysis is the process that is used for.....this complexity.

(a). documenting (b). implement
(c). maintenance (d). None of these.

3. Structured analysis and design are broken into four .....domains within application architecture

(a). secondary (b). primary
(c). storage (d). None of these.

6.6 The Logical and Physical Data Flow Diagram

Data flow diagrams (DFDs) are categorized as either logical or physical. A logical DFD focuses on the business and how the business operates. It describes the business events that take place and the data required and produced by each event. On the other hand, a physical DFD shows how the system will be implemented. The chart shown in Figure 6.5. Contrasts the features of logical and physical models. That the logical model reflects the business, while the physical model depicts the system.

Design Feature	Logical	Physical
What the model depicts	How the business operates	How hue system will be implemented (or how the current system operates)
What the processes represent	Business activities	Programs, program modules and manual procedures
What the data stores represent	Collections of data, regardless of how the data is stored	Physical files and databases, manual files
Type of data stores	Show data stores representing permanent data collections	Master files, transaction files. Any processes that operate at two different times must be connected by a data store
System controls	Show business controls	Show controls for validating input data, for obtaining a record (record found status), for ensuring successful completion of a process and for system security (example: journal records)

Figure 6.5: Features common of logical and physical data flow diagrams.

Ideally, systems are developed by analyzing the current system (the current logical DFD), then adding features that the new system should include (the proposed logical DFD).Finally the best methods to implement the new system should be developed (the physical DFD). After the logical model for the new system has been developed, it may be used to create a physical data flow diagram for the new system. The progression of these models is illustrated in Figure 6.6.

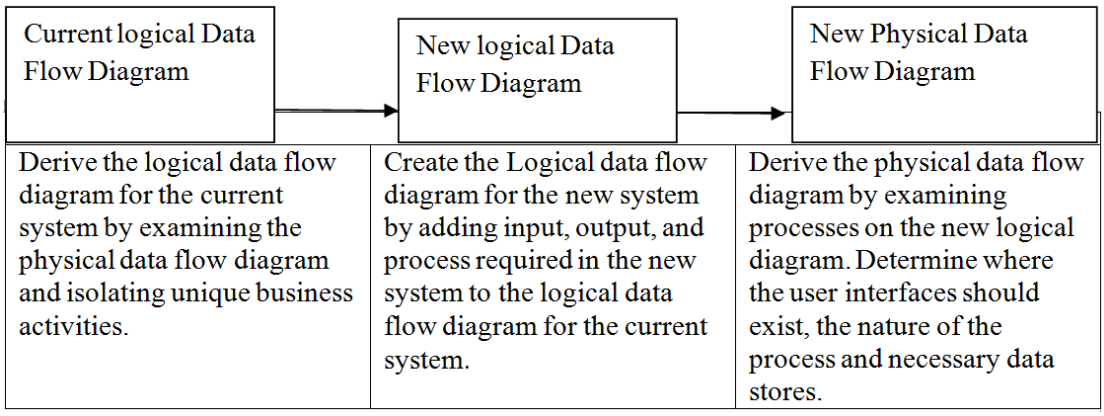


Figure 6.6: The progression of models from logical to physical.

Figure 6.7 shows a logical DFD and a physical DFD for a grocery store cashier. The customer brings the items to the register; prices for all items are looked up, and then totalled; next, payment is given to the cashier finally, the customer is given a receipt. The logical DFD illustrates the processes involved without going into detail about the physical implementation of activities.

Price code found on most grocery store items is used. In addition, the physical DFD mentions manual processes such as scanning, explains that a temporary file is used to keep a subtotal of items, and indicates that the payment could be made by cash, check, or debit card. Finally, it refers to the receipt by its name, cash register receipt.

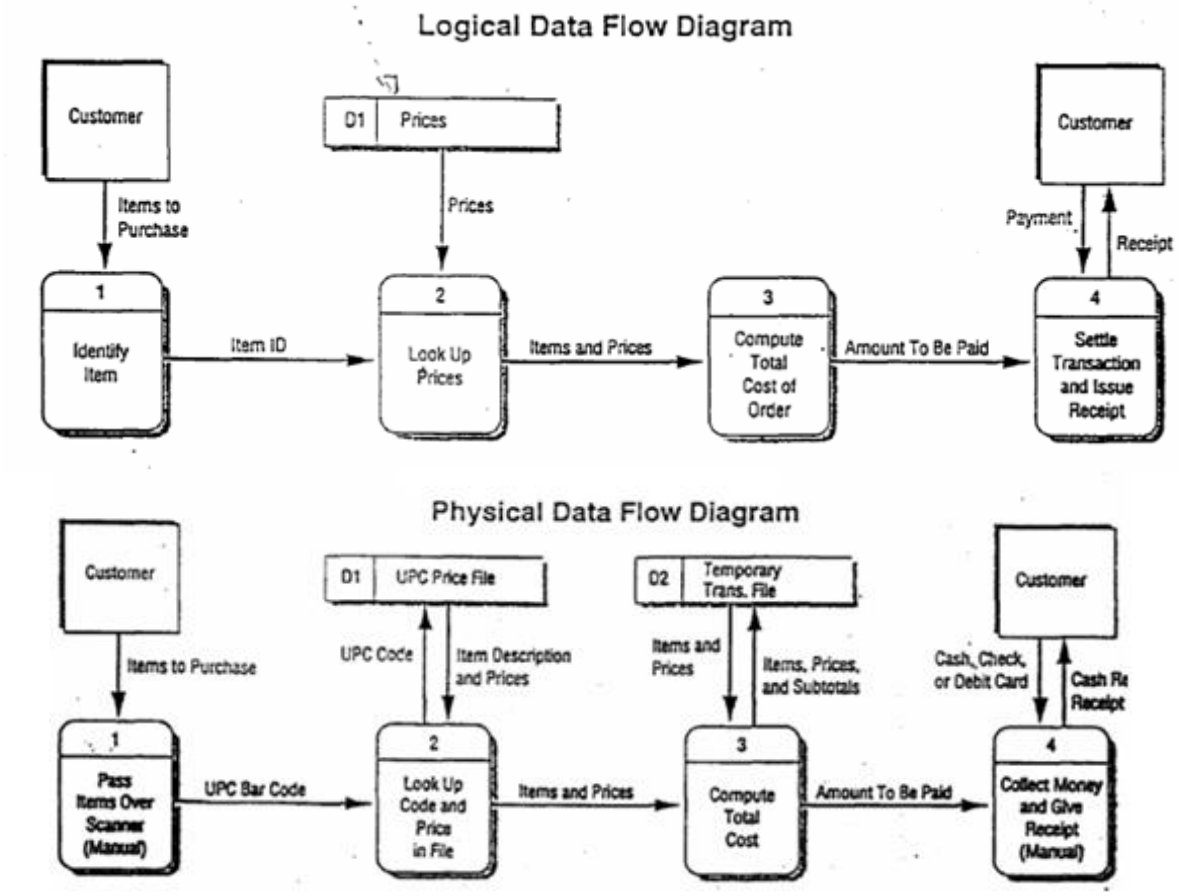


Figure 6.7: The physical data flow diagram shows certain details not found on the logical data flow diagram.

Steps of Developing DFD

The first we need to conceptualize data flows from a top-down perspective. To begin with we made a list of business activities and use it to determine various.

- External entities
- Data flows
- Process
- Data stores

6.7 Data Dictionary

One of the most important parts of an Oracle database is its data dictionary, which is a read-only set of tables that provides information about the database. A data dictionary contains:

The definitions of all schema objects in the database (tables, views, indexes, clusters, synonyms, sequences, procedures, functions, packages, triggers, and so on)

- How much space has been allocated for, and is currently used by, the schema objects
- Default values for columns
- Integrity constraint information
- The names of Oracle users
- Privileges and roles each user has been granted
- Auditing information, such as who has accessed or updated various schema objects

- Other general database information

The data dictionary is structured in tables and views, just like other database data. All the data dictionary tables and views for a given database are stored in that database's system table space.

Not only is the data dictionary central to every oracle database, it is an important tool for all users, from end users to application designers and database administrators. Use SQL statements to access the data dictionary. Because the data dictionary is read only, you can issue only queries (SELECT statements) against it is tables and views.

6.7.1 Structure of the Data Dictionary

The data dictionary consists of the following:

Base Tables

The underlying tables that store information about the associated database. Only oracle should write to and read these tables. Users rarely access them directly because they are normalized, and most of the data is stored in a cryptic format.

User-accessible Views

The views that summarize and display the information stored in the base tables of the data dictionary. These views decode the base table data into useful information, such as user or table names, using joins and where clauses to simplify the information. Most users are given access to the views rather than the base tables.

SYS Owner of the Data Dictionary

The Oracle user SYS owns all base tables and user-accessible views of the data dictionary. No Oracle user should ever alter (update, delete, or insert) any rows or schema objects contained in the SYS schema, because such activity can compromise data integrity. The security administrator must keep strict control of this central account. The data dictionary is implemented in Oracle as a set of read-only tables and views.



Figure 6.8: Hierarchical structure of the data dictionary.

6.7.2 How the Data Dictionary is used

The data dictionary has three primary uses:

- Oracle accesses the data dictionary to find information about users, schema objects, and storage structures.

- Oracle modifies the data dictionary every time that a data definition language (DDL) statement is issued.
- Any Oracle user can use the data dictionary as a read-only reference for information about the database.

Caution
Altering or manipulating the data in data dictionary tables can permanently and detrimentally affect the operation of a database.

6.8 Decision Trees

A decision tree is a graph that uses a branching method to illustrate every possible outcome of a decision. Decision trees can be drawn by hand or created with a graphics program or specialized software. Informally, decision trees are useful for focusing discussion when a group must make a decision. Programmatically, they can be used to assign monetary/time or other values to possible outcomes so that decisions can be automated. Decision tree software is used in data mining to simplify complex strategic challenges and evaluate the cost-effectiveness of research and business decisions. Variables in a decision tree are usually represented by circles.

6.8.1 The Characteristics of Decision Trees

A decision tree is a diagram that presents conditions and actions sequentially and thus shows which conditions to consider first, which second, and so on. It is also a method of showing the relationship of each condition and its permissible actions. The diagram resembles branches on a tree, hence the name. The root of the tree, on the left of the diagram, is the starting point of the decision sequence. The particular branch to be followed depends on the conditions that exist and the decision to be made. Progression from left to right along a particular branch is the result of making a series of decisions. Following each decision points is the next set of decision to be considered. The nodes of the tree thus represent conditions and indicate that a determination must be made about which condition exists before the next path can be chosen. The right side of the tree lists the actions to be taken depending on the sequence of conditions that is followed.

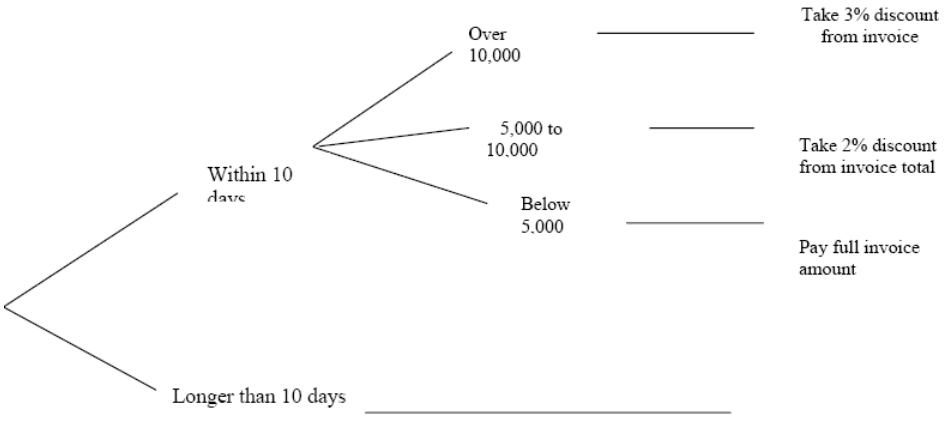


Figure 6.9: The conditions decision tree.

6.8.2 Using Decision Trees

Developing decision trees is beneficial to analysts in two ways. First of all, the need to describe conditions and actions forces analysts to formally identify the actual decision that must be made. It

becomes difficult for them to overlook and integral step in the decision process, whether it depends on quantitative or non quantitative variables. It is possible, for instance, to show what discount action to take, depending on the number of dollar spent by customers. When an organization opens accounts with dealers and suppliers, it formalizes agreements for taking discounts from the full invoice price. Two conditions are specified in this agreement: first, the invoice must always be paid within ten days of its receipt, and, second, the size of the discount will depend on the value of the invoice. It is agreed that under some conditions the organization can take the action of deducting a 3%discount; under other conditions, a 2% discount; and under all other conditions, no discount is allowed.

6.9 Structured English

Structured English is the additional method which is used for overcoming the problems of the ambiguous language in stating the actions and conditions in making the decisions and formulating the procedures. The procedure is described in the narrative format using the structured English. It does not show any decisions and rules but it states the rules. Structured English specifications require the analyst to identify the conditions which occur in a process and also identify the decisions which makes these conditions occur. It also forces the analyst to find alternative actions to be taken.

In this method the steps are listed in a specific order in which they are to be taken. No special signs, symbols or any other format are used for the displaying of the steps involved like those involved in the decisions tree of decision tables. Since only structured English statements are used it becomes easy for the analyst to state the entire procedure without wasting much time. The terminology used in structured English consists of mostly the data names of the elements and they are stored in the data dictionary.

Developing Structure Statements

The process is defined by using three types of statements: sequence structure, decision structure and iteration structure.

Sequence Structure: It is the single stepped or action included in the process and it does not depend on the existence of any other condition but if it does encounter a condition, it is taken into consideration.

Decision Structure: It occurs when two or more actions take place depending on the value of the condition. The condition is expanded and the necessary decisions are taken.

Iteration Structure: It is commonly found that certain conditions occur commonly or occur after certain conditions are executed. Iterative instructions help the analyst to describe these cases.

Self Assessment Questions

4. Structured analysis these are the data flows, data models, structure charts, and state models.

(a). True (b). False.

5. Structure charts are used to define the summary structure flow from.....

(a). not process (b). one process to another

(c). Both (a) and (b) (d). None of these

6. These models are joined with .....to define the events of an application.

(a). data flow models (b). data flow diagram

(c). structure analysis (d). None of these

7. The structured systems analysis and design method is an approach to designing.....

(a). Design
(b). analyzing information systems
(c). structure analysis
(d). None of these

6.10 Summary

- The structured analysis is the process that is used for documenting this complexity.
- Oracle reads the data dictionary to ascertain that schema objects exist and that users have proper access to them.
- Structured analysis and design are broken into four primary domains within application architecture. These are the data flows, data models, structure charts, and state models.
- The structured systems analysis and design method (SSADM) is an approach to designing and analyzing information systems.
- The traditional approach to analysis is focused on cost/benefit and feasibility analysis, hardware and software selection and personal considerations.

6.11 Keywords

Data Flow Diagram: The two-dimensional diagram that explains how data is processed and transferred in a system.

Data Stores: Data stores may be local to a specific level in the set of DFDs. A data store is used only if it is referenced by more than one process.

Graphic: It is refer to image or visual representation of an item.

Representation: A person induced into a contract on the basis of an untrue or misleading representation may sue for rescission of the contract and/or for damages.

Structured Analysis: It is a software engineering technique that uses graphical diagrams to develop and portray system specifications that are easily understood by users.

6.12 Review Questions

1. What is the structured analysis?
2. Explain the advantages and disadvantages of SSADM.
3. Explain the types of charts.
4. What are the differentiate between line charts and pie charts?
5. Explain the data flow diagram and data dictionary.
6. What are the advantages and disadvantages of DFDs?
7. Discuss the external entities.
8. What are the guidelines for drawing dataflow diagrams?
9. What is the logical and physical data flow diagram?
10. Define the decision trees and structured English.

Answers for Self Assessment Questions

1. (a) 2. (a) 3. (b) 4. (a) 5. (b) 6. (a) 7. (b)

6.13 Further Studies

1. http://www.dot.ca.gov/hq/esc/techpubs/manual/bridgemanuals/bridge-design-practice/pdf/bdp_4.pdf

7

Feasibility Study

STRUCTURE

7.0 Objectives

7.1 Introduction

7.2 Need of Feasibility Study?

7.3 Types of Feasibility

7.4 Steps of Feasibility Study

7.5 Summary

7.6 Keywords

7.7 Review Questions

7.8 Further Studies

7.0 Objectives

After studying this chapter, you will be able to:

- Discuss the need of feasibility study
- Explain the types of feasibility
- Explain the steps of feasibility study

7.1 Introduction

A feasibility studies main goal is to assess the economic viability of the proposed business. The feasibility study needs to answer the question "Does the idea make economic sense?" The study should provide a thorough analysis of the business opportunity, including a look at all the possible roadblocks that may stand in the way of the cooperative's success. The outcome of the feasibility study will indicate whether or not to proceed with the proposed venture. If the results of the feasibility study are positive, then the cooperative can proceed to develop a business plan.

If the results show that the project is not a sound business idea, then the project should not be pursued. Although it is difficult to accept a feasibility study that shows these results, it is much better to find this out sooner rather than later, when more time and money would have been invested and lost. It is tempting to overlook the need for a feasibility study. Often, the steering committee may face resistance from potential members on the need to do a feasibility study. Many people will feel that they know the proposed venture is a good idea, so why carry out a costly study just to prove what they already know? The feasibility study is important because it forces the NGC (new generation

cooperative) to put its ideas on paper and to assess whether or not those ideas are realistic. It also forces the NGC to begin formally evaluating which steps to take next.

The NGC's organizers will typically hire a consultant to conduct the feasibility study. Because the consultant is independent of the cooperative, he or she is in a better position to provide an objective analysis of the proposed venture. He or she should have previous experience in directly related work. To get an estimate of the costs of a feasibility study, prepare a rough outline of the work needed to be done.

It might be tempting to choose the lowest cost consultant or a personal acquaintance of one of the NGC's organizers, but always remember that quality work is the most important factor when choosing a consultant. Make sure that the consultant can provide an independent assessment of the business opportunity. For instance, hiring an engineering firm or an equipment manufacturer to conduct market analysis may lead to biased results in favor of proceeding with the venture. Engineering firms and equipment manufacturers may have an incentive to show positive results so they can obtain contracts with the cooperative once it chooses to start up operations. Engineering firms and equipment manufacturers are needed in order to provide information about equipment requirements and costs, but an independent consultant should conduct the overall feasibility study.

7.2 Need of Feasibility Study?

The feasibility study is the important step in any software development process. This is because it makes analysis of different aspects like cost required for developing and executing the system, the time required for each phase of the system and so on. If these important factors are not analyzed then definitely it would have impact on the organization and the development and the system would be a total failure. So for running the project and the organization successfully this step is a very important step in a software development life cycle process.

In the software development life cycle after making an analysis in the system requirement the next step is to make analysis of the software requirement. In other words feasibility study is also called as software requirement analysis. In this phase development team has to make communication with customers and make analysis of their requirement and analyze the system.

By making analysis this way it would be possible to make a report of identified area of problem. By making a detailed analysis in this area a detailed document or report is prepared in this phase which has details like project plan or schedule of the project, the cost estimated for developing and executing the system, target dates for each phase of delivery of system developed and so on. This phase is the base of software development process since further steps taken in software development life cycle would be based on the analysis made on this phase and so careful analysis has to be made in this phase.

Though the feasibility study cannot be focused on a single area some of the areas or analysis made in feasibility study is given. But all the steps given would not be followed by all system developed. The feasibility study varies based on the system that would be developed.

Feasibility study is made on the system being developed to analyze whether the system development process require training of personnel. This help in designing training sessions as required in later stage. Is the system developed has scope for expanding or scope for switching to new technology later if needed in ease. In other study is made to find the portability of the system in future.

Is the cost of developing the system high or does it meet the budgeted costs. That is a cost benefit analysis is made. In other words an analysis is made on cost feasibility of the project. This helps in identifying whether the organization would meet the budgeted costs and also helps the organization in making earlier and effective plans for meeting extra costs because of the system development.

Analysis is made on what software to use for developing the system. This study and analysis would help to choose the best implementation for system and the organization. This feasibility study includes factors like scalability, how to install, how to develop and so on. This feasibility study in short includes the analysis of technical areas. This analysis helps the efficiency of the system developed to get improved. This is because by choosing the correct technology by making analysis on the needs of system helps in improving the efficiency of the system.

The above feasibilities are analysis which helps in development of the system. But the scope of feasibility study does not end with this. Analysis or feasibility study also includes the analysis of maintenance stage. In other words feasibility study is made to analyze how one would maintain the system during maintenance stage. This helps sin planning for this stage and also helps in risk analysis. Also the analysis helps in making analysis about what training must be given and how and what all documents must be prepared to help users and developers to face maintenance phase.

Caution

The consultant should have a good understanding of the industry as well as the new generation cooperative model of business.

Advantages of making Feasibility study

There are many advantages of making feasibility study some of which are summarized:

- This study being made as the initial step of software development life cycle has all the analysis part in it which helps in analyzing the system requirements completely.
- Helps in identifying the risk factors involved in developing and deploying the system.
- The feasibility study helps in planning for risk analysis.
- Feasibility study helps in making cost/benefit analysis which helps the organization and system to run efficiently.
- Feasibility study helps in making plans for training developers for implementing the system.

So a feasibility study is a report which could be used by the senior or top persons in the organization. This is because based on the report the organization decides about cost estimation, funding and other important decisions which is very essential for an organization to run profitably and for the system to run stable. Thus before developing a product or software it is an essential step that one does feasibility study in some or all the areas mentioned which would help in developing and maintaining the software efficiently and effectively within budgeted costs.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Draw the diagram for process of feasibility study.

……

……

……

7.3 Types of Feasibility

Feasibility is the determination of whether or not a project is worth doing. The process followed in making this determination is called feasibility study. A feasibility study is carried out to select the

best system that meets performance requirements. When conducting feasibility study, an analyst can consider 7 types of feasibility:

7.3.1 Technical Feasibility

It is concerned with specifying the equipment and the computer system that will satisfy and support the proposed user requirements. Here we need to consider the configuration of the system which tells the analyst how many work stations are required, how the units are interconnected so that they can operate and communicate smoothly.

7.3.2 Operation Feasibility

It is related to human organizational aspects. The points to be considered here are what changes will be brought with the system? What new skills will be required?, do the existing staff members have these skills and can they be trained?

7.3.3 Economic Feasibility

It is the most frequently used technique for evaluating a proposed system. It is also called Cost/Benefit Analysis. It is used to determine the benefits and savings that are expected from the proposed system and compare them with the costs. If benefits are more than the cost, the proposed system is given an ok.

7.3.4 Social Feasibility

It is a determination of whether the proposed system will be acceptable to the people or not. It finds out the probability of the project being accepted by the group of people who are directly affected by the changed system.

7.3.5 Management Feasibility

It is a determination of whether the proposed system is acceptable to the management of the organization. The project may be rejected, if the management does not accept the proposed system.

7.3.6 Legal Feasibility

It is a determination of whether the proposed project is under legal obligation of known acts, Statutes, etc.

7.3.7 Time Feasibility

It is a determination of whether the project will be completed within a specified time period. If the project takes too much time, it is likely to be rejected.

Did You Know?

Several systems development frameworks have been partly based on SDLC, such as the structured systems analysis and design method (SSADM) produced for the UK government Office of Government Commerce in the 1980s.

Self Assessment Questions

1. The feasibility study is the important step in any ……..process.

(a) system development (b) design development

(c) software development (d) technological development

2……..is concerned with specifying the equipment and the computer system that will satisfy and support the proposed user requirements.

(a) Social feasibility (b) Technical feasibility
(c) Economic feasibility (d) Operation feasibility

3.……..is a determination of whether the proposed system will be acceptable to the people or not.

(a) Social feasibility (b) Legal feasibility
(c) Management feasibility (d) Time feasibility

4……..is a determination of whether the proposed system is acceptable to the management of the organization.

(a) Technical feasibility (b) Operation feasibility
(c) Management feasibility (d) Economic feasibility

5. The culmination of the feasibility study is a ……directed to management.

(a) feasibility team (b) feasibility report
(c) feasibility tests (d) None of these

7.4 Steps of Feasibility Study

A feasibility study is done by an organization in order to determine if a particular action makes sense from an economic and operational standpoint. Such a study is usually designed to provide an overview of the essential issues related to a course of action being considered. The goal is to test the feasibility of a proposed course of action and to identify any "make or break" issues that would argue against the action being taken or suggest that a successful outcome were unlikely.

Businesses find it helpful to conduct a feasibility study whenever they anticipate making an important strategic decision. For example, a company might perform a feasibility study to evaluate a proposed change in location, the acquisition of another company, a purchase of major equipment or a new computer system, the introduction of a new product or service, or the hiring of additional employees. A feasibility study is advisable as a means of fully studying an action in advance of taking the action. This allows managers a chance to fully assess the impact that any major changes they are consider may have before implementing the change.

The main objective of a feasibility study is to determine whether or not a certain plan of action is likely to produce the anticipated result that is, whether or not it will work, and whether or not it is worth doing economically. Although the primary objective of the study is dedicated to showing the outcomes of specific actions, it should begin with an evaluation of the entire operation.

A good feasibility study would review a company's strengths and weaknesses, its position in the marketplace, and it is financial situation. It would also include information on a company's major competitors, primary customers, and any relevant industry trends. This sort of overview provides small business owners and managers with an objective view of the company's current situation and opportunities. By providing information on consumer needs and how best to meet them, a feasibility study can also lead to new ideas for strategic changes.

The second part of a good feasibility study should focus on the proposed plan of action and provide a detailed estimate of its costs and benefits. In some cases, a feasibility study may lead management to determine that the company could achieve the same benefits through easier or cheaper means. For example, it may be possible to improve a manual filing system rather than purchase an expensive new

computerized database. If the proposed project is determined to be both feasible and desirable, the information provided in the feasibility study can prove valuable in implementation. It can be used to develop a strategic plan for the project, translating general ideas into measurable goals. The goals can then be broken down further to create a series of concrete steps and outline how the steps can be implemented. Throughout the process, the feasibility study will show the various consequences and impacts associated with the plan of action.

Feasibility analysis involves eight steps:

1. Form a project team and appoint a project leader.
2. Prepare system flowcharts.
3. Enumerate potential candidate systems.
4. Describe and identify characteristics of candidate systems.
5. Determine and evaluate performance and cost effectiveness of each candidate system.
6. Weight system performance and cost data.
7. Select the best candidate system.
8. Prepare and report final project directive to management.

7.4.1 Forming a System Team

The concept behind a project team is that future system users should be involved in its design and implementation. Their knowledge and experience in the operations area are essential to the success of the system. For small projects, the analyst and an assistant usually suffice; however, more complex studies require a project team. The team consists of analysts and user staff enough collective expertise to devise a solution to the problem. In many cases, an outside consultant and an information specialist join the team until the job is completed.

Projects are planned to occupy a specific time period, ranging from several weeks to months. The senior systems analyst is generally appointed as project leader. He/she is usually the most experienced analyst in the team. The appointment is temporary, lasting as long as the project. Regular meetings take place to keep up the momentum and accomplish the mission – selection of the best candidate system. A record is kept of the progress made in each meeting.

Regarding the safe deposit case, since the whole user area consists of five employees, the analyst handled most of the work.

7.4.2 Reviewing the System Data Flow Diagrams

The next step in the feasibility study is to prepare generalized system flowcharts for the system. Information – oriented charts and data flow diagrams prepared in the initial investigation are also reviewed at this time. The charts bring up the importance of inputs, outputs and data flow among key points in the existing system. All other flowcharts needed for detailed evaluation are completed at this point.

7.4.3 Developing the System Candidates

This step identifies the candidate systems that are capable of producing the outputs included in the generalized flowcharts. This requires a transformation from logical to physical system models. Another aspect of this step is consideration of the hardware that can handle the total system requirements. In the safe deposit case, it was found that virtually any microcomputer system with more than 128 KB memory and dual disk drive will do the job. It was also learned that a microcomputer can be designed to interface with the bank's mainframe. In this design, actual processing is handled by the microcomputer, whereas information such as payments and credits are transmitted to the main computer files for proper adjustment through the customer's checking

account. The question here is: which microcomputer (IBM, Apple, and Digital etc.) should be selected?

An important aspect of hardware is processing and main memory. There are a large number of computers with differing processing sizes, main memory capabilities and software support. The project team may contact vendors for information on the processing capabilities of the system available.

7.4.4 Evaluating Preliminary Evaluation of Candidates

From the candidate systems considered, the team begins a preliminary evaluation in an attempt to reduce them to a manageable number. Technical knowledge and expertise in the hardware / software area are critical for determining what each candidate system can and cannot do. In the safe deposit example, a search for the available microcomputers and safe deposit billing packages revealed the information summarized in Table 7.1.

These packages were the result of a preliminary evaluation of more than 15 other packages all purporting to meet the requirements, of the safe deposit billing system.

When the number is reduced to three key packages, the next step is to describe in some detail the characteristics of each package. For example, the first candidate system runs on an IBM PC with a minimum of 128 KB of memory. The software is written in Oracle, a relatively new language. In case of enhancements, change has to be made through the software house, since the source code is not available to the user.

Table 7.1: Safe deposit billing package and selected characteristics

Characteristics	IBM PC	HP 100	Apple III
Memory required (K bytes)	128	64	264
Source language	Oracle	VB	VB
Source Code Available	No	Yes	No
Purchase terms	Purchase (License)	Purchase (License)	Purchase (License)
Purchase price	Rs.99500	Rs.80000	Rs.1,09500
Number installed to date	200	30	50
Date of first installation	2002	2001	2000

The first package was installed in January 2002. More than 200 packages have been installed to date.

The next two candidate systems are similarly described. The information along with additional data available through the vendor highlights the positive and negative features of each system. The constraints unique to each system are also specified. For example, in the IBM PC package, the lack of an available source code means that the user has to secure a maintenance contract that costs 18% of the price of the package per year. In contrast the HP 100 package is less expensive and offers a

source code to the user. A maintenance contract (optional) is available at 18% of the price of the package.

7.4.5 Determine and Evaluate Performance and Cost Effectiveness of Each Candidate System

Each candidate system's performance is evaluated against the system performance requirements set prior to the feasibility study. Whatever the criteria, there has to be as close a match as practicable, although trade-offs are often necessary to select the best system. In the safe deposit case, the criteria chosen in advance were accuracy, growth potential, and response time less than five seconds, expandable main and auxiliary storage, and user friendly software. Often these characteristics do not lend themselves to quantitative measures. They are usually evaluated in qualitative terms (excellent, good, etc.) based on the subjective judgment of the project team.

The cost encompasses both designing and installing the system. It includes user training, updating the physical facilities and documenting. System performance criteria are evaluated against the cost of each system to determine which system is likely to be the most cost effective and also meets the performance requirements. The safe deposit problem is easy. The analyst can plot performance criteria and costs for each system to determine how each fare. Costs are more easily determined when the benefits of the system are tangible and measurable. An additional factor to consider is the cost of the study design and development. The cost estimate of each phase of the safe deposit project was determined for the candidate system (IBM PC). In many respects, the cost of the study phase is a "sunk cost" (fixed cost). Including it in the project cost estimate is optional.

7.4.6 Weight System Performance and Cost Data

In some cases, the performance and cost data for each candidate system show which system is the best choice? This outcome terminates the feasibility study. Many times, however, the situation is not so clear–cut. The performance/cost evaluation matrix at times does not clearly identify the best system, so the next step is to weight the importance of each criterion by applying a rating figure. Then the candidate system with the highest total score is selected.

The procedure for weighting candidate systems is simple:

1. Assign a weighting after factor to each evaluation criterion based on the criterion' effect on the success of the system. For example, if the usability criterion is twice as important as the accuracy factor, usability is assigned weight 4 and accuracy is assigned weight 2.
2. Assign a quantitative rating to each criterion's qualitative rating. For example, ratings (poor, fair, good, very good, excellent) may be assigned respective values (1, 2, 3, 4, 5).
3. Multiply the weight assigned to each category by the relative rating to determine the score.
4. Sum the score column for each candidate system.

Thus, the weighted candidate evaluation matrix is prepared using these steps, which in itself helps in the next step.

7.4.7 Feasibility Tests

The system with highest total score is judged the best system. This assumes the weighting factors are fair and the rating of each evaluation criterion is accurate. The criterion of growth potential is generally given the maximum weight, thus the greatest effect on the total score. Additionally, system development and user training are also given high weights.

Most feasibility studies select from more candidate systems than we have mentioned in our example. The criteria chosen and the constraints are also more complex.

In any case, management should not make the selection without having the experience to do so. Management cooperation and comments, however, are encouraged.

7.4.8 Feasibility Report

The culmination of the feasibility study is a feasibility report directed to management; it evaluates the impact of the proposed changes on the area(s) in question.

The report is a formal document for management use, brief enough and sufficiently non technical to be understandable, yet detailed enough to provide the basis for system design. There is no standard format for preparing feasibility reports. Analysts usually decide on a format that suits the particular user and system. Most reports, however, begin with a summary of findings and recommendations, followed by document details.

Starting with summary information highlights the essence of the report, giving management the option of reviewing the details later. The report contains the following sections:

1. Cover letter formally presents the report and briefly indicates to management the nature, general findings and recommendations to be considered.
2. Table of content specifies the location of the various parts of the report. Management quickly refers to the sections that concern them.
3. Overview is a narrative explanation of the purpose scope of the project, the reason for undertaking the feasibility study and the department(s) involved or affected by the candidate system. Also included are the names of the persons who conducted the study, when it began and other information that explains the circumstance surrounding the study?
4. Detailed findings outline the methods used in the present system. The system's effectiveness and efficiency as well as operating costs are emphasized. The section also provides a description of the objectives and general procedures of the candidate system. A discussion of output reports, costs and benefits gives management a feel for the pros and cons of the candidate system.
5. Economic justification details point-by-point cost comparisons and preliminary cost estimates for the development and operation of the candidate system. A return on investment (ROI) analysis of the project is also included.
6. Recommendations and conclusions suggest to management the most beneficial and cost-effective system. They are written only as a recommendation, not a command. Following the recommendations, any conclusions from the study may be included.
7. Appendixes document all memos and data compiled during the investigation. They are placed at the end of the report for reference.

Disapproval of the feasibility report is rare if it has been conducted properly. When a feasibility team has maintained good rapport with the user and his/ her staff it makes the recommendations easier to approve. Technically, the report is only a recommendation, but it is an authoritative one. Management has the final say. Its approval is required before system design is initiated.

Oral Presentation

The feasibility report is a good written presentation documenting the activities involving the candidate system. The pivotal step, however, is selling the proposed change. Invariably the project leader or analyst is expected to give an oral presentation to the end user. Although it is not as polished as the written report, the oral presentation has several important objectives. The most critical requirements for the analyst who gives the oral presentation are:

(1) Communication skills and knowledge about the candidate system that can be translated into language understandable to the user and

(2) The ability to answer questions, clarifies issues, maintain credibility and pick up on any new ideas or suggestions.

The substance and form of the presentation depend largely on the purposes sought. Table 7.2 suggests a general outline. The presentation may aim at informing, confirming, or persuading.

1. Informing. This simply means communicating the decisions already reached on system recommendations and the resulting action plans to those who will participate in the implementation.
2. **Confirming.** A presentation with this purpose verifies facts and recommendations already discussed and agreed upon. Unlike the persuading approach, no supportive evidence is presented to sell the proposed change, nor is there elaborate reasoning behind recommendations and conclusions. Although the presentation is not detailed, it should be complete. Confirming is itself part of the process of securing approval. It should reaffirm the benefits of the candidate system and provide a clear statement of results to be achieved.

Table 7.2: Oral presentation—suggested outline

1. Introduction
a. Introduce self
b. Introduce topic.
c. Briefly describe current system.
i. Explain why it is not solving the problem
ii. Highlight user dissatisfaction with it.
iii. Briefly describe scope, objectives and recommendation of the proposed system.
2. Body of presentation.
a. Highlight weaknesses of current system.
b. Describe proposed system. How is it going to solve the problem?
c. Sell proposed system.
i. Specify savings and benefits, costs and expenses.
ii. Use visual aids to justify project and explain system.
d. Summarize implementation plan and schedule.
e. Review human resources requirements to install system.
3. Conclusion.
a. Summarize proposal
b. Restate recommendations and objectives of proposal.
c. Summarize benefits and savings.
d. Ask for top-level management support. Solicit go-ahead for project.
4. Discussion period- Answer questions convincingly.

3. Persuading
This is a presentation pitched toward selling ideas- attempts to convince executives to take action on recommendations for implementing a candidate system.
Regardless of the purpose sought, the effectiveness of the oral presentation depends on how successful the project team has been in gaining the confidence of frontline personnel during the initial investigation. How the recommendations are presented also has an impact. Here are some pointers on how to give an oral presentation:

1. Rehearse and test your ideas before the presentation. Show that you are in command. Appear relaxed.

2. Final recommendations are more easily accepted if they are presented as ideas for discussion, even though they seem to be settled and final.
3. The presentation should be brief, factual and interesting Clarity and persuasiveness is critical. Skill is needed to generate enthusiasm and interest throughout the presentation.
4. Use good organization. Distribute relevant material to the user and other parties in advance.
5. Visual aids (graphs, charts) are effective if they are simple, meaningful and imaginative. An effective graph should teach or tell what is to be communicated.
6. Most important, present the report in an appropriate physical environment where the acoustics, seating pattern, visual aid technology and refreshments are available.

The most important element to consider is the length of the presentation. The duration often depends on the complexity of the project, the interest of the user group and the competence of the project team. A study that has company wide applications and took months to complete would require hours or longer presenting. The user group that was involved at the outset would likely permit a lengthy presentation, although familiarity with the project often dictates a brief presentation. Unfortunately, many oral presentations tend to be a rehash of the written document with little flare or excitement. Also, when the analyst or the project leader has a good reputation and success record from previous projects, the end user may request only a brief presentation.

Exercise: Check Your Progress 2

Case Study-Feasibility Study for your new e-Commerce Web Site

Challenge

The challenge in this case study is do the following:

Analyze the costs and benefits associated with the implementation of an ecommerce website for a membership society.

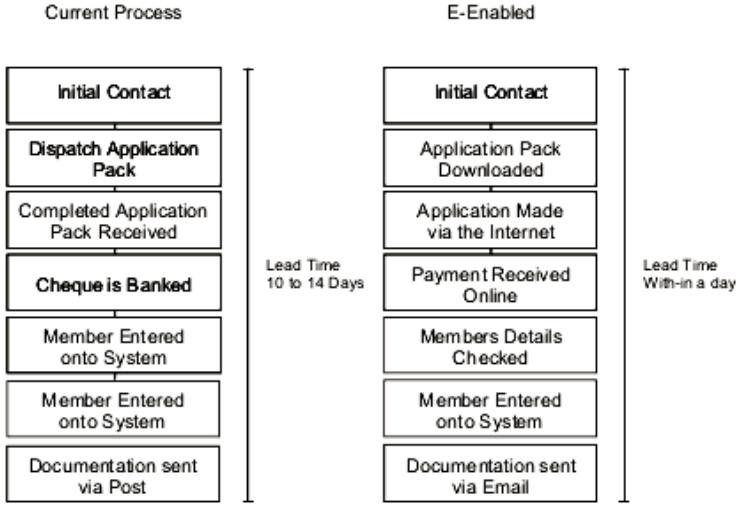


Figure 1: Membership Process.

Figure 1 illustrates typifies the processes of a generic membership system. You can see the effect e-enablement has on the current process on the right site of the illustration.

Costs for technology

To produce an ecommerce website requires a high speed connection to the Internet, a web server, and software. Other costs that are relevant is the cost of the payment system, whether it is taking online payment directly from the Societies web site or an alternative third-party like Pay pal or more expensively using an online bank.

Costs for technological development
This will involve a number of programmers who are able to interpret your functional requirements and program/create your website.

Costs for the consultancy support (design and implementation)
You would require the services of specialists in e-business design and implementation to guide you through this process.

Costs for the organization for piloting training
During the technological development of a website it is always a good idea to allow admin staff who will be using the system on a daily basis to pilot the system to as a training initiative.

Running costs
These are an upkeep of the web server and maintenance costs.
Running costs for change process
This is the cost of factoring in for your employees to train and adapting to the newly introduced technology, mainly the strategies used to make the change as smooth as possible.
Additionally being on the Internet would result in the your company having to become familiar to respond to emails, queries, and complaints that require instant or quick responses as opposed to replying to a Customer/Client via a letter. To be successful online, your company would have to address this issue of Change Management in that it would have to incorporate into its business, processes in order to guide the company to successfully maximize its effectiveness on the Internet. "Business is streamlined and service is almost instantaneous when it is done on the Web."

Financial Benefits
Improve Cash flow
Online payment would result in the lead-time to receive payments for membership within the same day of the application being made rather than the average 14 day delay. E-enabling the membership process is not just about reengineering a process so that it is quicker than before, it would result in a complete overhaul of the previous way of managing membership. Figure 1 illustrates the current membership process on the left and the E-enabled equivalent. As you can see in the E-enabled application and payment are made online. Once the application has been completed the documentation is sent via email to the member adding value by cutting the waiting time. This also saves on printing and postage for the Society. Even if the member does not have an email address the documentation will be available for download for registered members to the web site.
Increase Revenues the Internet will increase the volume of members. By going online with your business, you will generate revenue from places you never imagined.

Non-financial Benefits
Communication
Direct email marketing incurs little or no cost compared to the traditional direct mail marketing. Direct email marketing allows the flexibility of sending the companies message day or night, exactly when they want.
Transparency
The Information Management Website will allow the membership process to become transparent. For example, for the first time ever, Management will be able to know as a matter of fact:
1. The total number of members

2. Those members who need to renew their membership
3. Those members who are in arrears with their membership fees
4. Total number of members and accredited members
5. Forecast for the expected revenue that will be generated in advance and look at historic monthly generated revenues.

Exposure

The Internet means that your company will become a global Business. Thus attracting potential members from internationally. "Using the Web to sell your products removes the physical boundaries from your customer base. Customers from all over the world can learn about and purchase your products online."

Wider Considerations

When prospective members are signing up for membership they are providing personal data. This data includes the members name, full address, and credit card details. This means that the Society should register for the Data Protection Act 1998 which has a one off fee of £35.

Furthermore, your company must take measures to secure the line of communication from the visitor's computer and your web site. An SSL (system security layer) connection guarantees that this communication cannot be intercepted by any other person like a hacker.

7.5 Summary

- A feasibility studies main goal is to assess the economic viability of the proposed business.
- The feasibility study is the important step in any software development process. This is because it makes analysis of different aspects like cost required for developing and executing the system, the time required for each phase of the system and so on.
- Feasibility study is made on the system being developed to analyze whether the system development process require training of personnel.
- Feasibility is the determination of whether or not a project is worth doing.
- A good feasibility study would review a company's strengths and weaknesses, its position in the marketplace, and it is financial situation.
- Each candidate system's performance is evaluated against the system performance requirements set prior to the feasibility study.

7.6 Keywords

Economic Feasibility: It is the most frequently used technique for evaluating a proposed system. It is also called Cost/Benefit Analysis.

Legal Feasibility: It is a determination of whether the proposed project is under legal obligation of known Acts, Statutes, etc.

Management Feasibility: It is a determination of whether the proposed system is acceptable to the management of the organization.

Operation Feasibility: It is related to human organizational aspects. The points to be considered here are - what changes will be brought with the system?, what new skills will be required?, do the existing staff members have these skills and can they be trained?

Social Feasibility: It is a determination of whether the proposed system will be acceptable to the people or not. It finds out the probability of the project being accepted by the group of people who are directly affected by the changed system.

Technical Feasibility: It is concerned with specifying the equipment and the computer system that will satisfy and support the proposed user requirements.

Time Feasibility: It is a determination of whether the project will be completed within a specified time period. If the project takes too much time, it is likely to be rejected.

7.7 Review Questions

1. What is feasibility study?
2. Discuss the importance of feasibility study.
3. How many type of feasibility?
4. Discuss the eight steps of feasibility analysis.
5. How to evaluate the system candidates?
6. What is feasibility report?
7. How to feasibility tests in feasibility study?
8. What is basic requirement of form team?
9. Discuss the step of feasibility tests.
10. What is a data flow diagram for feasibility study?

Answers for Self Assessment Questions

1 (c) 2 (b) 3 (a) 4 (c) 5 (b)

7.8 Further Studies

1. Feasibility Study Preparation and Analysis by PCH Publications

8

Cost/Benefit Analysis

STRUCTURE

8.0 Objectives

8.1 Introduction

8.2 Data Analysis

8.3 Classifications of Costs and Benefits

8.4 Cost Categories

8.5 Determining Costs/Benefits

8.6 System Proposal

8.7 Summary

8.8 Keywords

8.9 Review Questions

8.10 Further Studies

8.0 Objectives

After studying this chapter, you will be able to:

- Explain the data analysis
- Describe the classifications of costs and benefits
- Describe the cost categories
- Process of determining costs/benefits
- Explain the system proposal

8.1 Introduction

A cost benefit analysis is done to determine how well, or how poorly, a planned action will turn out. Although a cost benefit analysis can be used for almost anything, it is most commonly done on financial questions. Since the cost benefit analysis relies on the addition of positive factors and the subtraction of negative ones to determine a net result, it is also known as running the numbers.

8.2 Data Analysis

Data analysis is a prerequisite to cost/ benefit analysis. System investigation and data gathering lead to an assessment of current findings. From the analysis, the system design requirements are identified, which could be:

- Better customer service
- Faster information retrieval
- Quicker reports
- Less time consuming
- Accuracy
- Reduce data redundancy
- Improved staff efficiency
- Lower processing and operating costs

Better Customer Service

Customer Service is a function of how well an organization meets the needs of its customers and Customer support is a range of customer services to assist customers in making cost effective and correct use of a product. It includes assistance in planning, installation, training, trouble shooting, maintenance, upgrading, and disposal of a product.

Faster Information Retrieval

Information retrieval (IR) is the area of study concerned with searching for documents, for information within documents, and for metadata about documents, as well as that of searching structured storage, relational databases, and the World Wide Web. There is overlap in the usage of the terms data retrieval, document retrieval, information retrieval, and text retrieval, but each also has its own body of literature, theory, praxis, and technologies. IR is interdisciplinary, based on computer science, mathematics, library science, information science, information architecture, cognitive psychology, linguistics, statistics and law.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Draw the quicker reports for proposal system.

……

……

……

8.3 Classifications of Costs and Benefits

Cost and benefit determination is to categorize costs and benefits. They may be tangible or intangible, direct or indirect, fixed or variable.

Each category is reviewed as follows:

8.3.1 Tangible or Intangible Costs and Benefits

Tangibility refers to the ease with which costs or benefits can be measured. An outlay of cash for a specific item or activity is referred to as a tangible cost. They are usually shown as disbursements on the books. The purchase of hardware or software, personnel training and employee salaries are examples of tangible costs. They are readily identified and measured.

Costs that are known to exist but whose financial value cannot be accurately measured are referred to as intangible costs. For example, employee morale problems caused by a new system or lowered company image is an intangible cost. In some cases, intangible costs may be easy to identify but

difficult to measure. For example, the cost of the breakdown of an online system during banking hours will cause the bank to lose deposits and waste human resources. The problem is by how much? In other cases, intangible costs may be difficult even to identify, such as an improvement in customer satisfaction stemming from a real-time order entry system.

Benefits are also classified as tangible or intangible. Like costs, they are often difficult to specify accurately. Tangible benefits, such as completing jobs in fewer hours or producing reports with no errors, are quantifiable. Intangible benefits, such as more satisfied customers or an improved corporate image, are not easily quantified.

Caution

Tangible and intangible costs and benefits should be considered in the evaluation process.

8.3.2 Direct or Indirect Costs and Benefits

From a cost accounting point of view, costs are handled differently depending on whether they are direct or indirect. Direct costs are those with which an exact figure can be directly associated in a project. They are applied directly to the operation. For example, the purchase of a box of diskettes is a direct cost because we can associate the diskettes with the money spent. Direct benefits also can be specifically attributable to a given project. For example, a new system that can handle 25% more transactions per day is a direct benefit.

Indirect costs are the results of operations that are not directly associated with a given system or activity. They are often referred to as overhead. A system that reduces overhead realizes a saving. If it increases overhead, it incurs an additional cost.

Insurance, maintenance, protection of the computer centre, heat, light and air conditioning are all tangible costs, but it is difficult to determine the proportion of each attributable to a specific activity such as a report. They are overhead and are allocated among users, according to a formula.

8.3.3 Fixed or Variable- Costs and Benefits

Some costs and benefits are constant, regardless of how well a system is used. Fixed costs are sunk costs. They are constant and do not change. Once encountered, they will not recur. Examples are straight line depreciation of hardware, and insurance. In contrast, variable costs are incurred on a regular (weekly, monthly) basis. They are usually proportional to work volume and continue as long as the system is in operation.

Did You Know?

Cost-budgeting analysis was first proposed in 1844 by the French engineer Jules Dupuit.

Tangible Costs

Tangible costs include the types of things a business writes checks for: salaries and wages, leases, operational inputs, employee medical benefits, transportation and commercial insurance. These costs have a clear place in the general ledger. The company cannot conduct business or produce a quality product without spending on tangible costs. They are also easy to quantify, so management tends to focus on the manipulation of tangible costs.

Intangible Costs

Intangible costs are less easily measured. Some key and common intangible costs might include a drop in employee morale, dissatisfaction with working conditions or customer disappointment with a decline in service or product quality. Intangible costs result from an identifiable source, but the costs are often not predicted. They may occur after a new practice or policy is put into effect, such as a cut

in staffing levels or in employee benefits. Managers can try to estimate intangible costs as soon as they see a pattern of loss. This estimate will be the basis of a decision to either change or continue a practice that frustrates employees or customers. If a new procedure has injured an employee, the company may need to act quickly to avoid government fines and inspections.

Direct Costs

Direct costs can be defined as costs which can be accurately traced to a cost object with little effort. Cost object may be a product, a department, a project, etc. Direct costs typically benefit a single cost object therefore the classification of any cost either as direct or indirect is done by taking the cost object into perspective. A particular cost may be direct cost for one cost object but indirect cost for another cost object. Most direct costs are variable but this may not always be the case. For example, the salary of a supervisor for a month who has only supervised the construction of a single building is a direct fixed cost incurred on the building.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Cost of gravel, sand, cement and wages incurred on production of concrete.

……

……

……

Indirect Costs

Costs which cannot be accurately attributed to specific cost objects are called indirect costs. These typically benefit multiple cost objects and it is impracticable to accurately trace them to individual products, activities or departments etc.

Exercise: Check Your Progress 3

Note: i) Use the space below for your answer.

Ex1: Cost of depreciation, insurance, power, salaries of supervisors incurred in a concrete plant.

……

……

……

Ex2: Following costs are incurred by a factory on the production of identical cupboards:

1. Laborers' wages	2. Synthetic wood
3. Power consumption	4. Glass
5. Nails and screws	6. Factory insurance
7. Handles, locks and hinges	8. Wood
9. Supervisors' salaries	10. Factory depreciation
11. Varnish, glue, paints	12. Factory manager's salary

Classify the above costs as direct or indirect.

……

……

……

Solution

1. Direct	2. Direct
3. Indirect	4. Direct
5. Indirect	6. Indirect
7. Direct	8. Direct
9. Indirect	10. Indirect
11. Indirect	12. Indirect

Fixed or Variable Costs and Benefits

Some costs and benefits are constant, regardless of how well a system is used. Fixed costs (after the {act are sunk costs. They are constant and do not change. Once encountered, they will not recur. Examples are straight-line depreciation of hardware exempt employee salaries, and insurance. In contrast, variable costs are incurred on a regular (weekly, monthly) basis. They are usually proportional to work volume and continue as long as the system is in operation. For example, the costs of computer forms vary in proportion to the amount of processing or the length of the reports required.

Self Assessment Questions

1. ……..is a prerequisite to cost/ benefit analysis.

(a) Data analysis (b) System analysis
(c) Data diagram analysis (d) None of these.

2. Customer support is a range of……..to assist customers in making cost effective and correct use of a product.

(a) vendor service (b) customer services
(c) consumer service (d) Both (a) and (c).

3. …….are examples of tangible costs.

(a) Purchase of hardware (b) Personnel training
(c) Employee salaries (d) All of these

8.4 Cost Categories

In developing cost estimates for a system, we need to consider several cost elements. Among them is hardware, personnel, facility, operating and supply costs.

Hardware Costs

It is relate to the actual purchase or lease of the computer and peripherals (for example, printer, disk drive, tape unit). Determining the actual cost of hardware is generally more difficult when the system is shared by various users than for a dedicated stand- alone system. In some cases, the best way to control for this cost is to treat it as an operating cost.

Personnel Costs

It is include EDP staff salaries and benefits (health insurance, vacation time, sick pay, etc.) as well as pay for those involved in developing the system Costs incurred during development of a system are one - time costs and are labelled developmental costs. Once the system is installed, the costs of operating and maintaining the system become recurring costs.

Facility Costs

It is expenses incurred in the preparation of the physical site where the application or the computer will be in operation. This includes wiring, flooring, acoustics, lighting and air conditioning. These costs are treated as onetime costs and are incorporated into the overall cost estimate of the candidate system.

Operating Costs

It is include all costs associated with the day-to-day operation of the system; the amount depends on the number of shifts, the nature of the applications, and the calibre of the operating staff. There are various ways of covering operating costs. One approach is to treat operating costs as overhead. Another approach is to charge each authorized user for the amount of processing they request from the system.

The amount charged is based on computer time, staff time and volume of the output produced. In any case, some accounting is necessary to determine how operating costs should be handled.

Supply Costs

It is variable costs that increase with increased use of paper, ribbons, disks, and the like. They should be estimated and included in the overall cost of the system.

A system is also expected to provide benefits. The first task is to identify each benefit and then assign a monetary value to it for cost/ benefit analysis. Benefits may be tangible and intangible, direct or indirect.

The two major benefits are improving performance and minimizing the cost of processing. The performance category emphasizes improvement in the accuracy of or access to information and easier access to the system by authorized users. Minimizing costs through an efficient system - error control or reduction of staff- is a benefit that should be measured and included in cost/benefit analysis

Did You Know?

Since the 1960s, cost–benefit analyses have been used in all aspects of government planning and budgeting, from programs that can be analyzed with a fairly high degree of accuracy.

8.5 Determining Costs/Benefits

There is a difference between expenditure and investment. We spend to get what we need, but we invest to realize a return on the investment. Building a computer – based system is an investment. Costs are incurred throughout its life cycle. Benefits are realized in the form of reduced operating costs, improved corporate image, staff efficiency, or revenues. To what extent benefits outweigh costs is the function of cost /benefit analysis.

Cost/ benefit analysis is a procedure that gives a picture of the various costs, benefits and rules associated with a system.

The determination of costs and benefits entails the following steps:

- Identify the costs and benefits pertaining to given project.
- Categorize the various costs and benefits for analysis.
- Select a method of evaluation.
- Interpret the results of the analysis.
- Take action.

Costs and Benefits Identification

Certain costs and benefits are more easily identifiable than others. For example, direct costs, such as the price of a hard disk, are easily identified form company invoice payments or cancelled checks. Direct benefits often relate one-to-one to direct costs, especially savings from reducing costs in the activity in question. Other direct costs and benefits, however, may not be well defined, since they represent estimated costs or benefits that have some uncertainty. An example of such costs is reserve for bad debt. It is a discerned real cost, although its exact amount is not so immediate.

A category of costs or benefits that is not easily discernible is opportunity costs and opportunity benefits. These are the costs or benefits forgone by selecting one alternative over another. They do not show in the organization's accounts and therefore are not easy to identify.

Self Assessment Questions

4. ..... is expenses incurred in the preparation of the physical site where the application or the computer will be in operation.

(a) Hardware costs (b) Facility costs

(c) Supply costs (d) None of these

5. Benefits may not be tangible and intangible, direct or indirect.

(a) True (b) False

6. ..... analysis is a procedure that gives a picture of the various costs, benefits and rules associated with a system.

(a) Cost (b) Benefits

(c) Both (a) and (b) (d) Data

8.6 System Proposal

The final decision following cost/benefit analysis is to select the most cost-effective and beneficial system for the user. At this time, the analyst prepares a feasibility report on the major findings and recommendations. It outlines the options and recommendations. It is presented to management for determining whether a candidate system should be designed. Effective reports follow carefully planned formats that management can understand and evaluate without having to read the entire document. The content and format of the feasibility report are summarized.

There is following terms comprising the systems proposal.

- Cover letter
- Title page of project
- Table of contents
- Executive summary
- Outline of systems study with appropriate documentation
- Detailed results of the systems study

- Systems alternatives
- Systems analysts recommendations
- Summary
- Appendices

Cover Letter: It should list the people who did the study and summarize the objectives of the study.
Title Page of Project: Name of the project, the names of the team members, date submitted.
Table of Contents: It is useful to readers of long proposals; omit if less than 10 pages.
Executive Summary: precisely provides the who, what, when, where, why, and how of the proposal.
Outline of Systems Study with Appropriate Documentation: it is provides information about all the methods used in the study and who or what was studied.
Detailed results of the systems study: It describes what was found out about human and systems needs through all the methods described in the detailed results of the systems study.
Systems Alternatives: It is two or three alternatives that directly address the problem.
Systems Analysts Recommendations: It is the recommended solution.
Summary: It is brief statement that mirrors the content of the executive summary. Conclude the proposal on a positive note.
Appendices: It can include any information that may be of interest.

Exercise: Check Your Progress 4
Note: i) Use the space below for your answer.
Ex1: Draw the structure of production proposal system.

……

……

……

8.7 Summary

- Cost benefit analysis is done to determine how well, or how poorly, a planned action will turn out. Although a cost benefit analysis can be used for almost anything, it is most commonly done on financial question
- Data analysis is a prerequisite to cost/ benefit analysis. System investigation and data gathering lead to an assessment of current findings.
- The outlay of cash for a specific item or activity is referred to as a tangible cost. They are usually shown as disbursements on the books.
- Cost/ benefit analysis is a procedure that gives a picture of the various costs, benefits and rules associated with a system.
- System proposal is presented to management for determining whether a candidate system should be designed.

8.8 Keywords

Hardware Costs: It is relate to the actual purchase or lease of the computer and peripherals.

Indirect Costs: It is refer to results of operations that are not directly associated with a given system or activity

Operating Costs: It is include all costs associated with the day-to-day operation of the system.

Supply Costs: It is variable costs that increase with increased use of paper, ribbons, disks etc.

Tangibility: It is refers to the ease with which costs or benefits can be measured.

8.9 Review Questions

1. What is the cost benefit analysis?
2. Describe the data analysis.
3. Compare the various cost methods.
4. How are tangible costs different from direct costs?
5. What are the various cost benefit categories?
6. Why is it necessary to conduct cost/benefit analysis?
7. What do you understand by system proposal?
8. What is the difference between tangible and fixed Costs and benefits?
9. Explain the determining costs/benefits.
10. Difference ate between hardware cost and operating cost.

Answers for Self Assessment Questions

1. (a) 2. (b) 3. (d) 6. (b) 7. (b) 8. (c)

8.10 Further Studies

1. http://isbm.smeal.psu.edu/library/articles/trends-study/Sample%20SAP-MaximizingValueReport.pdf/at_download/file

9

System Design

STRUCTURE

9.0 Objectives

9.1 Introduction

9.2 Design Process

9.3 Phases of Design

9.4 Methodologies of Designing

9.5 Functional Decomposition

9.6 Module Coupling and Cohesion

9.7 Prototyping Model

9.8 Information Engineering

9.9 Joint Application Development

9.10 Rapid Application Development Model

9.11 Object-oriented Design Process

9.12 Processing Controls and Data

9.13 Verification and Validation

9.14 Audit Trail and Documentation Control

9.15 Summary

9.16 Keywords

9.17 Review Questions

9.18 Further Studies

9.0 Objectives

After studying this chapter, you will be able to:

- Discuss the design process
- Explain the phases of design
- Discuss the module coupling and cohesion
- Explain the prototyping model
- Discuss the joint application development

- Discuss the object-oriented design process
- Discuss the processing controls and data

9.1 Introduction

The purpose of system design is to create a technical solution that satisfies the functional requirements for the system. At this point in the project lifecycle there should be a functional specification, written primarily in business terminology, containing a complete description of the operational needs of the various organizational entities that will use the new system. The challenge is to translate all of this information into technical specifications that accurately describe the design of the system, and that can be used as input to system construction.

The functional specification produced during system requirements analysis is transformed into a physical architecture. System components are distributed across the physical architecture, usable interfaces are designed and prototyped and technical specifications are created for the application developers, enabling them to build and test the system. Many organizations look at system design primarily as the preparation of the system component specifications; however, constructing the various system components is only one of a set of major steps in successfully building a system. The preparation of the environment needed to build the system, the testing of the system, and the migration and preparation of the data that will ultimately be used by the system are equally important. In addition to designing the technical solution, system design is the time to initiate focused planning efforts for both the testing and data preparation activities.

9.2 Design Process

To 'design' is to conceive the looks, arrangement, and workings of something before it is created.

• Design is an activity that can be approached at different levels of detail

• First the position of the process according to its volume and variety characteristics must be defined.

• Eventually the details of the process must be analysed to ensure that it fulfilsits objectives effectively.

• Process design and product/service design are interrelated

• Small changes in the design of products and services can have profound implications for the way the operation eventually has to produce them.

• Similarly, the design of a process can constrain the freedom of product and service designers to operate as they would wish

9.3 Phases of Design

The principle goal of system design is to establish a design approach that provides the functions described in the system requirements document.

System design will establish a discipline integrated engineering plan for the proposed design, understand the technical risks, explore trade-offs, and determine estimates for performance and cost to completion.

The process starts with a System Requirements Document:

- Establish system functions and required performance
- Establish user needs and features
- Complete lifecycle considered – development and operations

- Starting point for developing a solution to meet needs
- All disciplines integrated
- Basis for complete systems approach to design

The System Requirements Document drives system design, which consists of the following process steps:

- Establish functional requirements from the analysis of system requirements
- Determine design requirements – technology needs, solution space
- Iteratively consider design requirements and architecture
- Determine an initial design approach
- Analyze risks in the development and implementation of the proposed design approach
- Initial estimate of performance
- Establish an error budget
- Initial estimate of cost and schedule
- Plan for work to address research needs and reduce/mitigate risks
- Trade studies – iterate on cost/performance trade offs

Figure 9.2 is a process diagram that illustrates the system design process.

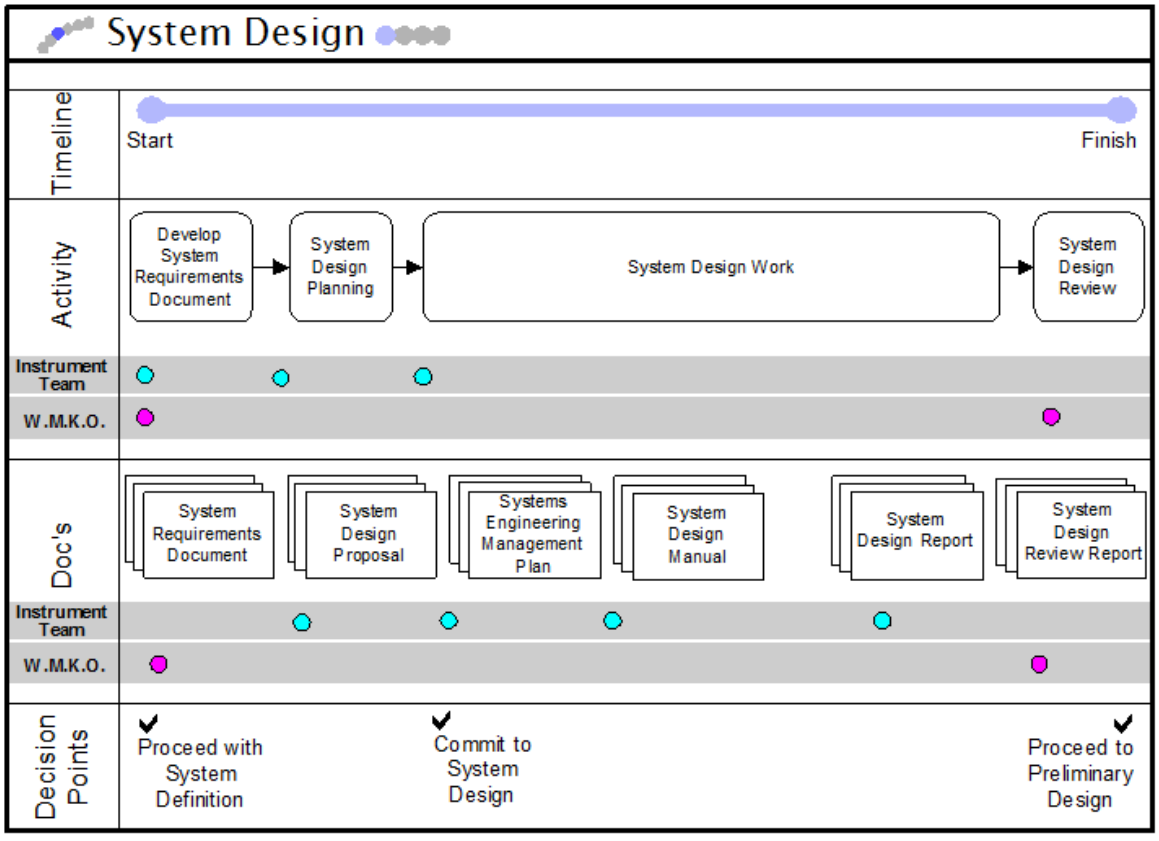


Figure 9.2: The System Design Process.

There are four major documentation products that result from the system design phase:

1. System Design Proposal

The System Design Proposal establishes the scope of work, WBS and cost estimate for the system design phase.

2. Systems Engineering Management Plan

The Systems Engineering Management Plan (SEMP) describes the management processes that will be used throughout the new instrument development project.

The SEMP includes the following:

- Staff and organization
- Roles and responsibilities
- Project work flow
- Decision making process
- Reporting and documentation requirements

3. System Design Manual

The System Design Manual establishes the scope of work for the remaining design phases and describes how the technical and scientific challenges of the instrument requirements will be met.

The system design manual includes the following:

- Describes the functional requirements and relates them to the system requirements
- Indicates the preferred design approach
- Describes the architecture
- Indicates technology drivers and research needs
- Gives an error budget
- Gives a risk analysis
- Establishes a framework for remaining design phases (preliminary and detail design)

4. System Design Report

The System Design Report summarizes the work accomplished in the system design phase and presents the initial estimate of performance for the instrument, the work plan for the preliminary design phase and an estimate of cost and schedule to completion.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Create a structural diagram for hospital management system.

……

……

……

9.4 Methodologies of Designing

9.4.1 Software Design Methodologies

A methodology can be simply defined as a set of procedure that one follows from the beginning to the completion of the software development process. The nature of the methodology is dependent on a number of factors, including the software development environment, the organization's practices, the nature or type of the software being developed, the requirements of the users, the qualification and training of the software development team, the available hardware and software resources, the availability of existing design modules, and even the budget and the time schedule. Since the 1970s,

there has been a proliferation of software design methodologies. Different methodologies have been developed to resolve different types of problems. In describing these problems, it is often possible or necessary to group problems with similar characteristics together. This is called the problem domain. Many of these methodologies have evolved from the specific milieu where the software was developed. By this it is meant that specific methodologies are often developed (to be applied) to resolve certain "classes" of problems, also called domain of application, for which it is well suited. Even though the design mechanics are different in each methodology.

Yet, each of these methods has a number of common characteristics:
(1) A mechanism for the translation of information domain representation into design representation,
(2) A notation for representing functional components and their interfaces,
(3) Heuristics for refinement and partitioning, and
(4) Guidelines for quality assessment.

There are two broad categories of design methodologies: the systematic and the formal types. As the name imply, the formal type makes extensive use of mathematical notations for the object transformations and for checking consistencies. The systematic types are less mathematical and are consisting of the procedural component, which prescribes what action or task to perform and the representation component, which prescribes how the software structure should be represented. Generally, techniques from the systematic design methodologies can be integrated and can utilize representation schemes from other techniques when and as appropriate. Due to the fact that methodologies have been developed from different milieu specifically to address certain problems or groups of problems, there is no common baseline on which to evaluate or compare the methodologies against each other.

However, the underlying principles of the methodologies can be analyzed and examined for a better understanding of the basis for each methodology. With a better understanding of the methodology, its domain of application can be more effectively applied or more accurately defined. Generally, alternative design allows for important trade off analysis before coding the software. Thus, familiarity with several methodologies makes creating competitive designs more logical and systematic with less reliance on inspiration. It is not the intention of this section to explain the detailed mechanics of each of the methodologies but to discuss specific principles of each methodology.

9.4.2 Top-down/Bottom-up Design

Top-down design directs designers to start with a top-level description of a system and then refine this view step by step. With each refinement, the system is decomposed into lower-level and smaller modules. Top-down decomposition requires identifying the major higher-level system requirements and functions, and then breaking them down in successive steps until function-specific modules can be designed. Thus, top-down design is a level-oriented design approach.

9.4.3 Stepwise Refinement

The iterative process where each system is decomposed and refined step by step is called stepwise refinement. It is a level-oriented design approach. It defines a sequence of refinement steps. In each step, the system or module is decomposed into subsystems or sub modules. Each refinement of the module should be accompanied by a refinement of the structure and relationship between the modules. In addition, each successive step in the refinement process should be a faithful extension of

the previous step. The degree of modularity derived from the refinement will determine the extensibility of the module. As the system or module is refined step by step, a notation should be used to capture the process. The nature of this notation should be such that it can capture the structure of the system and its data. It should also have a close resemblance to the programming language to be used for coding. Each refinement is based on a number of design decisions based on a certain set of design criteria. With each refinement step, design decisions are decomposed to a lower-level until it reaches an elementary level.

Stepwise refinement begins with specifications obtained from requirement analysis. The solution to the problem is first broken down into a few major modules or processes that will demonstrably solve the problem. Then through successive refinement, each module is decomposed until there are sufficient details so that implementation a programming language is straight forward. In this way, a problem is segmented into smaller, manageable units and the amount of details that have to be focused on any point in time is minimized. This allows the designer to channel his resources at a specific issue at the proper time. As stepwise refinement begins at the top-level, success is highly dependent on the designer's conceptual understanding of the complete problem and the desired solution.

Exercise: Check Your Progress 2
Note: i) Use the space below for your answer.
Ex1: Create a structural diagram for data flow in an organization using bottom-up approach.

……

……

……

9.5 Functional Decomposition

Functional decomposition is good in a procedural programming environment. It is even useful for understanding the modular nature of a larger-scale application.
Unfortunately, it does not translate directly into a class hierarchy, and this is where the problem begins. In defining this Anti Pattern, the authors started with Michael Akroyd's original thoughts on this topic. We have reformatted it to fit in with our template, and extended it somewhat with explanations and diagrams.

9.5.1 General Form

This Anti Pattern is the result of experienced, non object-oriented developers who design and implement an application in an object-oriented language. When developers are comfortable with a "main" routine that calls numerous subroutines, they may tend to make every subroutine a class, ignoring class hierarchy altogether (and pretty much ignoring object-orientation entirely).
The resulting code resembles a structural language such as Pascal or FORTRAN in class structure. It can be incredibly complex, as smart procedural developers devise very clever ways to replicate their time-tested methods in an object-oriented architecture.



Figure 9.3: General form.

You will most likely encounter this Anti Pattern in a C shop that has recently gone to C++, or has tried to incorporate CORBA interfaces, or has just implemented some kind of object tool that is supposed to help them. It is usually cheaper in the long run to spend the money on object-oriented training or just hire new programmers who think in objects.

9.5.2 Symptoms and Consequences

- Classes with "function" names such as Calculate_Interest or Display_Table may indicate the existence of this Anti Pattern.
- All class attributes are private and used only inside the class.
- Classes with a single action such as a function.
- An incredibly degenerate architecture that completely misses the point of object-oriented architecture.
- Absolutely no leveraging of object-oriented principles such as inheritance and polymorphism. This can be extremely expensive to maintain (if it ever worked in the first place; but never underestimate the ingenuity of an old programmer who's slowly losing the race to technology).
- No way to clearly document (or even explain) how the system works. Class models make absolutely no sense.
- No hope of ever obtaining software reuse.
- Frustration and hopelessness on the part of testers.

9.5.3 Typical Causes

Lack of Object-oriented Understanding

The implementers did not "get it." This is fairly common when developers switch from programming in a nonobject-oriented programming language to an object-oriented programming language. Because there are architecture, design, and implementation paradigm changes, object-orientation can take up to three years for a company to fully achieve.

Lack of Architecture Enforcement

When the implementers are clueless about object-orientation, it does not matter how well the architecture has been designed; they simply would not understand what they are doing. And without

the right supervision, they will usually find a way to fudge something using the techniques they do know.

Specified Disaster

Sometimes, those who generate specifications and requirements do not necessarily have real experience with object-oriented systems. If the system they specify makes architectural commitments prior to requirements analysis, it can and often does lead to Anti Patterns such as Functional Decomposition.

9.5.4 Known Exceptions

The Functional Decomposition Anti Pattern is fine when an object-oriented solution is not required. This exception can be extended to deal with solutions that are purely functional in nature but wrapped to provide an object-oriented interface to the implementation code.

9.5.5 Refectories Solution

If it is still possible to ascertain what the basic requirements are for the software, define an analysis model for the software, to explain the critical features of the software from the user's point of view. This is essential for discovering the underlying motivation for many of the software constructs in a particular code base, which have been lost over time. For all of the steps in the Functional Decomposition Anti Pattern solution, provide detailed documentation of the processes used as the basis for future maintenance efforts.

Next, formulate a design model that incorporates the essential pieces of the existing system. Do not focus on improving the model but on establishing a basis for explaining as much of the system as possible.

Ideally, the design model will justify, or at least rationalize, most of the software modules. Developing a design model for an existing code base is enlightening; it provides insight as to how the overall system fits together. It is reasonable to expect that several parts of the system exist for reasons no longer known and for which no reasonable speculation can be attempted.

For classes that fall outside of the design model, use the following guidelines:

1. If the class has a single method, try to better model it as part of an existing class. Frequently, classes designed as helper classes to another class are better off being combined into the base class they assist.
2. Attempt to combine several classes into a new class that satisfies a design objective. The goal is to consolidate the functionality of several types into a single class that captures a broader domain concept than the previous finer-grained classes. For example, rather than have classes to manage device access, to filter information to and from the devices, and to control the device, combine them into a single device controller object with methods that perform the activities previously spread out among several classes.
3. If the class does not contain state information of any kind, consider rewriting it as a function. Potentially, some parts of the system may be best modelled as functions that can be accessed throughout various parts of the system without restriction.

Examine the design and find similar subsystems. These are reuse candidates. As part of program maintenance, engage in refactoring of the code base to reuse code between similar subsystems.

Example

Functional Decomposition is based upon discrete functions for the purpose of data manipulation, for example, the use of Jackson Structured Programming. Functions are often methods within an object-

oriented environment. The partitioning of functions is based upon a different paradigm, which leads to a different grouping of functions and associated data.
The simple examples in Figure 9.4 see a functional version of a customer loan scenario:

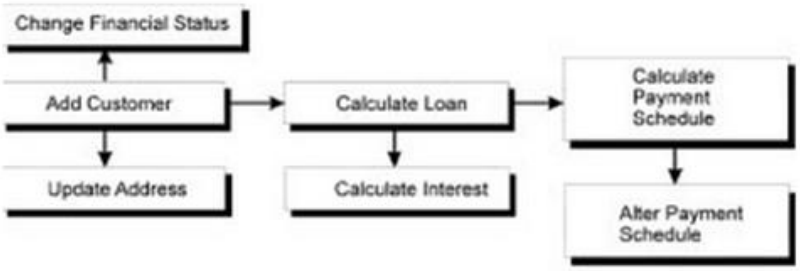


Figure 9.4: Functional version of a customer loan scenario.

- Adding a new customer.
- Updating a customer address.
- Calculating a loan to a customer.
- Calculating the interest on a loan.
- Calculating a payment schedule for a customer loan.
- Altering a payment schedule.

Next Figure 9.5 then sees the object-oriented view of a customer loan application.
The previous functions map to object methods.

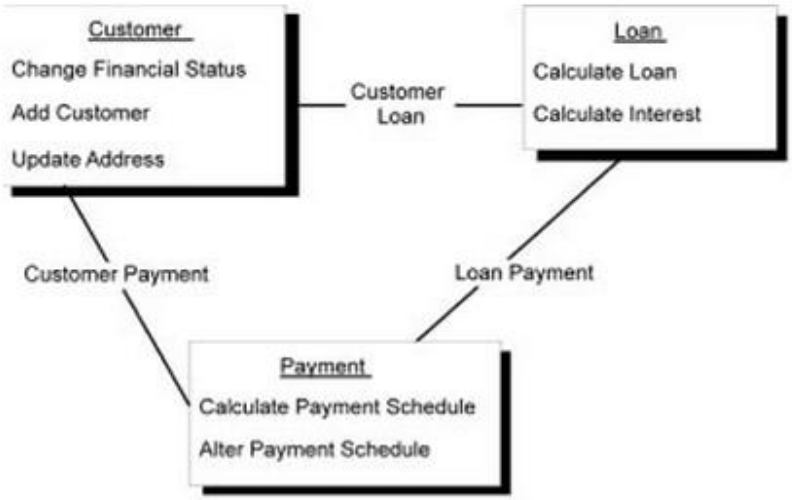


Figure 9.5: Object-oriented view of a customer loan application.

9.5.6 Related Solutions

If too much work has already been invested in a system plagued by Functional Decomposition, you may be able to salvage things by taking an approach similar to the alternative approach addressed in the Blob Anti Pattern.
Instead of a bottom-up refactoring of the whole class hierarchy, you may be able to extend the “main routine” class to a “coordinator” class that manages all or most of the system’s functionality.

Function classes can then are “massaged” into quasi-object-oriented classes by combining them and defang them up to carry out some of their own processing at the direction of the modified “coordinator” class. This process may result in a class hierarchy that is more workable.

9.5.7 Applicability to Other Viewpoints and Scales

Both architectural and managerial viewpoints play key roles in either initial prevention or ongoing policing against the Functional Decomposition AntiPattern. If a correct object-oriented architecture was initially planned and the problem occurred in the development stages, then it is a management challenge to enforce the initial architecture.

9.6 Module Coupling and Cohesion

9.6.1 Coupling

The coupling between two modules indicates the degree of interdependence between them. If two modules interchange a large amount of data, then they are highly interdependent. The degree of coupling between two modules depends on their interface complexity. The interface complexity is basically determined by the number of types of parameters that are interchanged while invoking the functions of the module.

Highly Coupled

When the modules are highly dependent on each other then they are called highly coupled.

Loosely Coupled

When the modules are dependent on each other but the interconnection among them is weak then they are called loosely coupled.

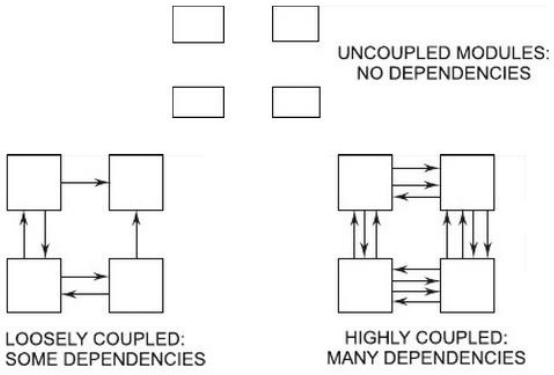


Figure 9.6: Coupling.

Uncoupled

When the different modules have no interconnection among them then they are called uncoupled modules. Factors affecting coupling between modules:

The various factors which affect the coupling between modules are depicted in Table 9.1.

Table 9.1: Factors affecting coupling

	Interface Complexity	Type of Connection	Type of Communication
Low	Simple Obvious	To module by name	Data
High	Complicated Obscure	To internal elements	ControlHybrid

Types of Couplings

Different types of couplings include content, common, external, control, stamp, and data. The strength of a coupling from the lowest coupling (best) to the highest coupling (worst) is given in Figure 9.7.

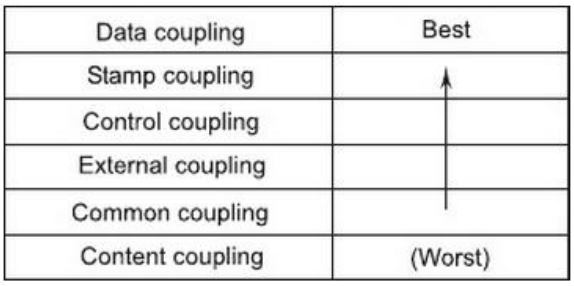


Figure 9.7: The types of module coupling.

1. Data Coupling

Two modules are data coupled if they communicate using an elementary data item that is passed as a parameter between the two; for example, an integer, a float, a character, etc. This data item should be problem related and not used for a control purpose.

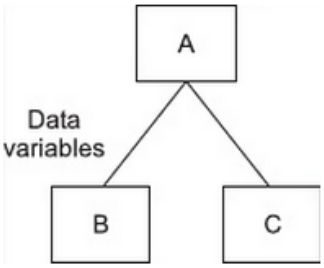


Figure 9.8: Data coupling.

When a non-global variable is passed to a module, modules are called data coupled. It is the lowest form of a coupling. For example, passing the variable from one module in C and receiving the variable by value (i.e., call by value).

2. Stamp Coupling

Two modules are stamp coupled if they communicate using a composite data item, such as a record, structure, object, etc. When a module passes a non-global data structure or an entire structure to another module, they are said to be stamp coupled.

For example, passing a record in Pascal or a structure variable in C or an object in C++ language to a module.

3. Control Coupling

Control coupling exists between two modules if data from one module is used to direct the order of instruction execution in another. An example of control coupling is a flag set in one module that is tested in another module.

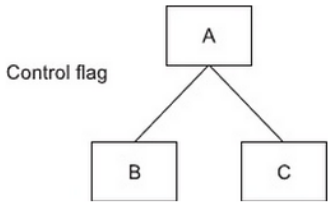


Figure 9.9: Control coupling.

The sending module must know a great deal about the inner workings of the receiving module. A variable that controls decisions in subordinate module C is set in super-ordinate module A and then passed to C.

4. External Coupling

It occurs when modules are tied to an environment external to software. External coupling is essential but should be limited to a small number of modules with structures.

5. Common Coupling

Two modules are common coupled if they share some global data items (e.g., Global variables). Diagnosing problems in structures with considerable common coupling is time-consuming and difficult. However, this does not mean that the use of global data is necessarily "bad."

Caution

A software designer must be aware of potential consequences of common couplings and take special care to guard against them.

6. Content Coupling

Content coupling exists between two modules if their code is shared; for example, a branch from one module into another module. It is when one module directly refers to the inner workings of another module. Modules are highly interdependent on each other. It is the highest form of coupling. It is also the least desirable coupling as one component actually modifies another and thereby the modified component is completely dependent on the modifying one.

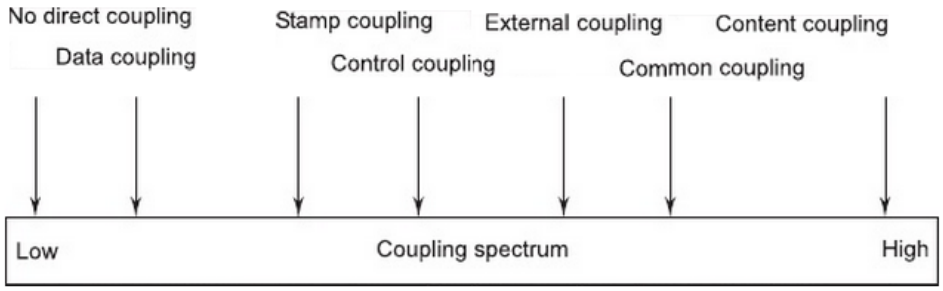


Figure 9.10: Couplings.

High coupling among modules not only makes a design difficult to understand and maintain, but it also increases development effort as the modules having high coupling cannot be developed independently by different team members. Modules having high coupling are difficult to implement and debug.

9.6.2 Cohesion

Cohesion is a measure of the relative functional strength of a module. The cohesion of a component is a measure of the closeness of the relationships between its components. A cohesive module performs a single task within a software procedure, requiring little interaction with procedures being performed in other parts of a program. A strongly cohesive module implements functionality that is related to one feature of the solution and requires little or no interaction with other modules. This is shown in Figure 9.11. Cohesion may be viewed as the glue that keeps the module together. It is a measure of the mutual off city of the components of a module.

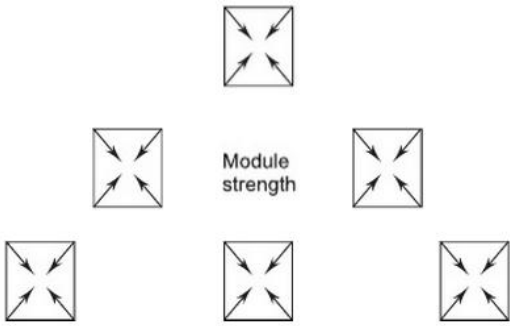


Figure 9.11: Cohesion-strength of relation within modules.

Thus, we want to maximize the interaction within a module. Hence, an important design objective is to maximize the module cohesion and minimize the module coupling.

Types of Cohesion

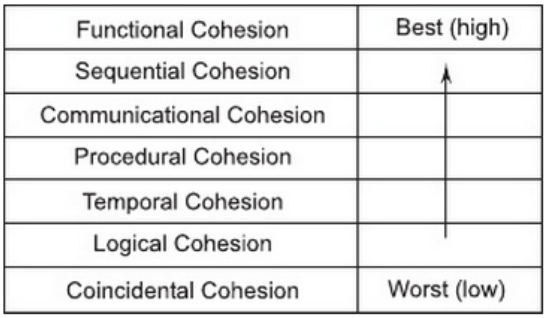


Figure 9.12: The types of module cohesion.

There are seven levels of cohesion in decreasing order of desirability, which are as follows:

1. Functional Cohesion

Functional cohesion is said to exist if different elements of a module cooperate to achieve a single function (e.g., managing an employee's payroll). When a module displays functional cohesion, and if we are asked to describe what the module does, we can describe it using a single sentence.

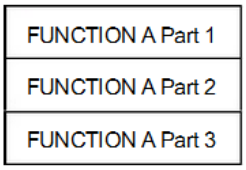


Figure 9.13: Functional cohesion, sequential with complete, related functions.

2. Sequential Cohesion

A module is said to possess sequential cohesion if the elements of a module form the parts of a sequence, where the output from one element of the sequence is input to the next.

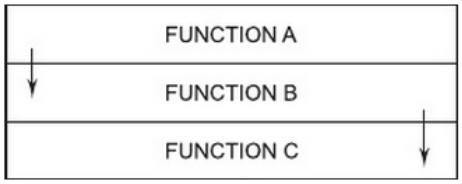


Figure 9.14: Sequential cohesion, output of one part is input to next.

3. Communicational Cohesion

A module is said to have communicational cohesion if all the functions of the module refer to or update the same data structure; for example, the set of functions defined on an array or a stack. All the modules in communicational cohesion are bound tightly because they operate on the same input or output data.

For example, the set of functions defined on an array or a stack.

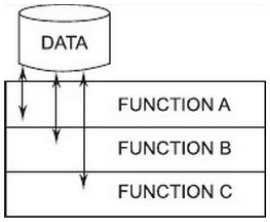


Figure 9.15: Communicational cohesion, access same data.

4. Procedural Cohesion

A module is said to possess procedural cohesion if the set of functions of the module are all part of a procedure (algorithm) in which a certain sequence of steps has to be carried out for achieving an objective; for example, the algorithm for decoding a message.

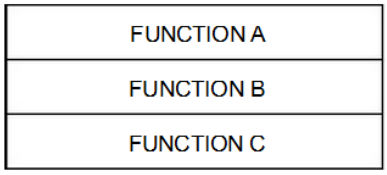


Figure 9.16: Procedural cohesion related by order of function.

5. Temporal Cohesion

When a module contains functions that are related by the fact that all the functions must be executed in the same time span, the module is said to exhibit temporal cohesion. The set of functions responsible for initialization, start up, shutdown of some process, etc., exhibit temporal cohesion.

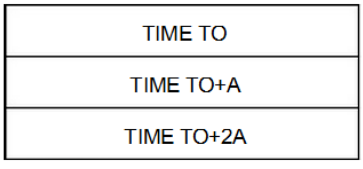


Figure 9.17: Temporal cohesion related by time.

6. Logical Cohesion

A module is said to be logically cohesive if all elements of the module perform similar operations; for example, error handling, data input, data output, etc. An example of logical cohesion is the case where a set of print functions generating different output reports are arranged into a single module.

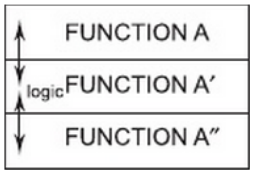


Figure 9.18: Logical cohesion similar functions.

7. Coincidental Cohesion

A module is said to have coincidental cohesion if it performs a set of tasks that relate to each other very loosely. In this case, the module contains a random collection of functions. It means that the functions have been put in the module out of pure coincidence without any thought or design. It is the worst type of cohesion.

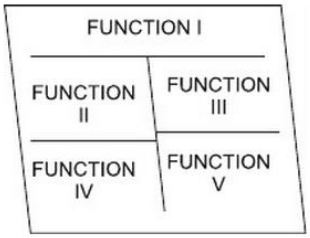


Figure 9.19: Coincidental cohesion parts unrelated.

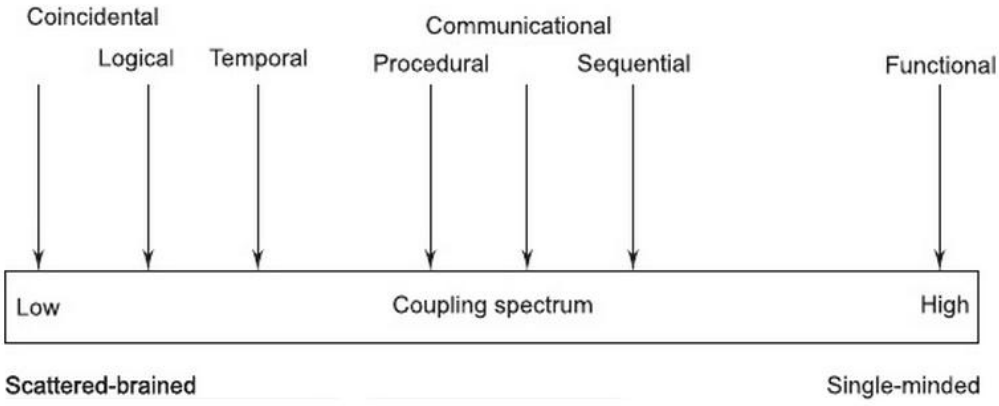


Figure 9.20: Cohesion.

9.6.3 Relationship between Coupling and Cohesion

A software engineer must design the modules with the goal of high cohesion and low coupling. A good example of a system that has high cohesion and low coupling is the “plug and play” feature of a computer system. Various slots in the motherboard of the system simply facilitate to add or remove the various services/functionalities without affecting the entire system. This is because the add-on components provide the services in a highly cohesive manner. Figure 9.11 provides a graphical review of cohesion and coupling.

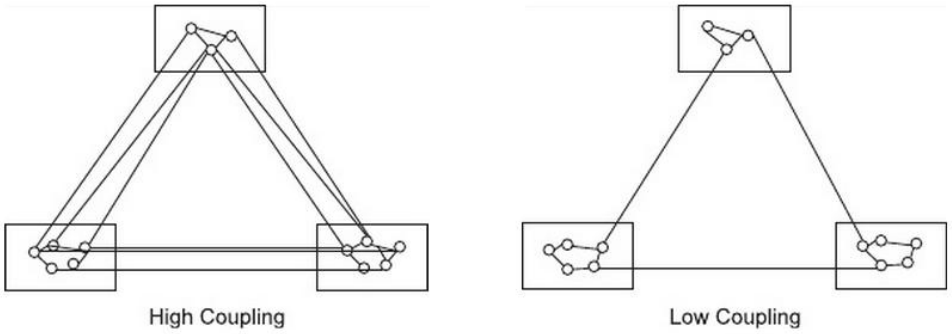


Figure 9.21: View of cohesion and coupling.

Module design with high cohesion and low coupling characterizes a modules a black box when the entire structure of the system is described. Each module can be dealt with separately when the module functionality is described.

9.7 Prototyping Model

It always happens that a customer defines a set of general objectives for software but does not identify detailed input, processing, or output requirements. In other cases, the developer may be unsure of the efficiency of an algorithm, the adaptability of an operating system, or the form that human/machine interaction should take. In these and many other situations, a prototyping paradigm may offer the best approach. Prototyping begins with communication. The developer and customer meet and define the overall objectives for the software, identify whatever requirements are known, and outline areas where further definition is mandatory. A quick design then occurs. The quick design focuses on the representation of those aspects of the software that will be visible to the customer/user (e.g., input approaches and output formats). The quick design leads to the construction of a prototype. The prototype is evaluated by the customer/user to refine requirements for the software to be developed. Iteration occurs as the prototype is tuned to satisfy the needs of the customer, while at the same time enabling the developer to better understand what needs to be done. Ideally, the prototype serves as a mechanism for identifying software requirements. If a working prototype is built, the developer attempts to use existing program fragments or applies tools that enable working programs to be generated quickly.

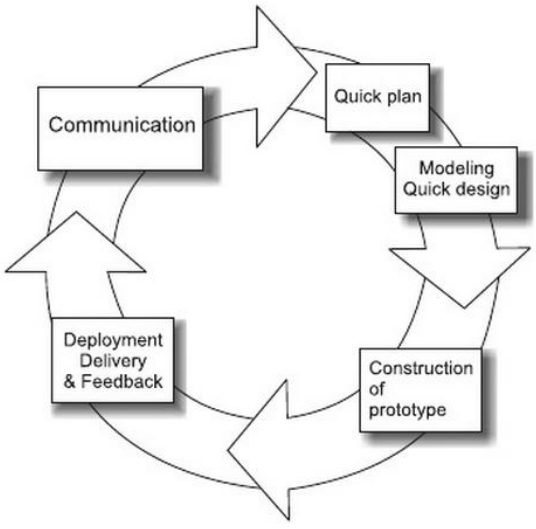


Figure 9.22: Prototyping model.

9.7.1 Reasons for Using Prototyping Model

There are several uses of a prototype. An important purpose is to illustrate the in-put data formats, messages, reports, and the interactive dialogues to the customer. This is valuable for gaining a better understanding of the customer's needs. The prototype model is very useful in developing the Graphical User Interface (GUI) part of a system. The prototyping model can be used when the technical solutions are unclear to the development team. Often, major design decisions depend on issues, such as the response time of a hardware controller or the efficiency of a sorting algorithm, etc. In such circumstances, a prototype may be the best or the only way to resolve the technical issues. The third reason for developing a prototype is that it is impossible to "get it right" the first time and one must plan to throw away the first product in order to develop a good product later, as advocated.

9.7.2 Controlling Changes during Prototyping

A major problem with prototyping is controlling changes to the prototype following suggestions by the users. One approach has been to categorize changes as belonging to one of three types:

Cosmetic (about 35% of changes)

These are simply changes to the layout of the screen.

They are:

(a) Implemented

(b) Recorded

Local (about 60% of changes)

These involve changes to the way the screen is processed but do not affect other parts of the system.

They are:

(a) Implemented

(b) Recorded

(c) Backed-up so that they can be removed at a later stage is necessary

(d) Inspected retrospectively

Global (about 5% of changes)

These are changes that affect more than one part of the processing. All changes here have to be the subject of a design review before they can be implemented.

9.7.3 Advantages of Prototyping Models

- Suitable for large systems for which there are no manual process to define the requirements
- User training to use the system
- User services determination
- System training
- Quality of software is good
- Requirements are not freeze

9.7.4 Limitations of Prototyping Model

- It is difficult to find all the requirements of the software initially
- It is very difficult to predict how the system will work after development

These two limitations are removed in the prototyping model. The first limitation is removed by unfreezing the requirements before any de-sign or coding can proceed. The second limitation is removed by making a throw away prototype to understand the requirements.

Self Assessment Questions

1..... is the complete replacement of the old system by the new system.
(a) Maintenance (b) Direct changeover (c) Parallel run (d) Pilot run

2..... consist of several functions that pass data along, for example, update and write a record.
(a) Functional (b) Clustered (c) Sequential (d) Communicational

3. The ……methodology is based on the rules for building models.
(a) SSD (b) JSD (c) DSSD (d) SADT

4. Two modules are..... if they communicate using a composite data item.
(a) data coupling (b) stamp coupling (c) control coupling (d) external coupling

5.....is essential but should be limited to a small number of modules with structures.
(a) Common coupling (b) Content coupling (c) Control coupling (d) External coupling

6.....is said to exist if different elements of a module cooperate to achieve a single function.
(a) Sequential cohesion (b) Communicational cohesion
(c) Functional cohesion (d) Procedural cohesion

9.8 Information Engineering

Information engineering is a data cantered, but process sensitive technique that is applied to the organization as a whole (or a significant part, such as a division), rather than on an ad-hoc, project-by-project basis (as in structured analysis).

The basic concept of information engineering is that information systems should he engineered like other products. Information engineering books typically use a pyramid framework to depict information systems building blocks and system development phases.

The phases are:

1. Information strategy planning (ISP) applies systems analysis methods to examine the business as a whole to define an overall plan and architecture for subsequent information systems development. No actual information systems or computer applications are developed. Instead, the project team studies the business mission and goals and defines an information systems architecture and plan to optimally align information systems to help the organization achieve its business goals.

2. Based on the strategic plan, business areas are carved out and prioritized. A business area is a collection of cross organizational business processes that should be highly integrated to achieve the information strategy plan (and business mission). A business area analysis (BAA) uses Systems analysis methods to study the business area and define the business requirements for a highly streamlined and integrated set of information Systems and computer applications to support that business area.

3. Based on the business area requirements analysis, information system applications are carved out and prioritized. These applications become projects to which other Systems analysis and design methods are applied to develop production systems. These methods may include some combination of structured analysis and design, prototyping, and object-oriented analysis and design.

9.9 Joint Application Development

As previously described, modern structured analysis and information engineering both emphasizes model-driven development. Prototyping places emphasis on construction of the working prototypes. Joint application development (JAD) complements both of these techniques by emphasizing participative development among system owners, users, designers, and builders.

JAD uses highly organized and intensive workshops to bring together system owners, users, analysts, designers, and builders to jointly define and design systems. Synonyms include joint application design and joint requirements planning. A JAD trained Systems analyst usually plays the role of facilitator for a workshop that will typically run from three to five full working days. This workshop may replace months of traditional interviews and follow-up meetings.

The JAD provides a working environment in which to accelerate methodology activities and deliverables. It promotes enhanced system owner and user participation in system development.

But it also requires a facilitator with superior mediation and negotiation skills to ensure that all parties receive appropriate opportunities to contribute to the system's development.

9.10 Rapid Application Development Model

The RAD (Rapid Application Development Model) model is proposed when requirements and solutions can be modularized as independent system or software components, each of which can be developed by different teams. After these smaller system components are developed, they are integrated to produce the large software system solution. The modularization could be on a functional, technology, or architectural basis, such as front-end-back-end, client side server side, and so on. The RAD becomes faster if the software engineer uses the component's technology such that the components are really available for reuse. Since the development is distributed into component development teams, the teams work in tandem and total development is completed in a short period (i.e., 60 to 90 days). Figure 9.23 shows the RAD model.

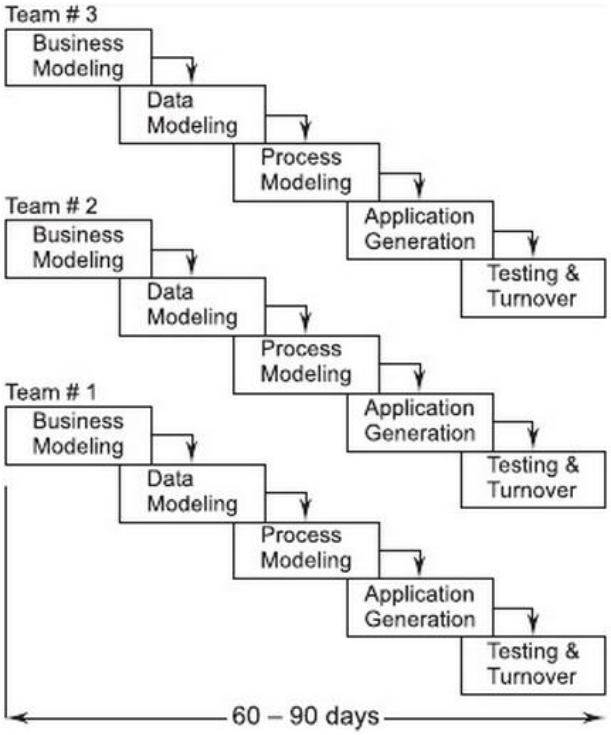


Figure 9.23: The RAD model.

1. Business Modelling

It covers the following functions:

- Where does information come from and go?
- Who processes it?
- What information drives the business process?
- What information is generated?

2. Data Modelling

The information flow defined as part of the business modelling phase is refined into a set of data objects that are needed to support the business. The characteristics of each object are identified and the relationships between these objects are defined.

3. Process Modelling

In this model, information flows from object to object to implement a business function. To add, modify, delete, or retain a data object, there is a need for description which is done in this phase.

4. Application Generation

The RAD assumes the use of fourth generation techniques. The RAD process works to reuse existing program components or create reusable components. To facilitate the construction of the software using the above cases, automated tools are used.

5. Testing and Turnover

In this phase we have to test the programs, but we use some already existing programs which are already tested, so the time involved in testing is less. Only the new programs or components must be tested.

9.10.1 Disadvantages of RAD Model

The various disadvantages of the RAD model include:

- For large, but scalable projects, RAD requires sufficient human resources to create the right number of RAD teams.
- If developers and customers are not committed to the rapid fire activities necessary to complete the system in a much abbreviated timeframe, RAD projects will fail.
- If a system cannot be properly modularized, building the components necessary for RAD will be problematic.
- If high performance is an issue, and performance is to be achieved through tuning the interfaces to system components, the RAD approach may network.
- The RAD may not be appropriate when technical risks are high (for example, when a new application makes heavy use of new technology).

9.11 Object-oriented Design Process

The process of object-oriented designed by developing an example design for the control software that is embedded in an automated weather station. There are several methods of object-oriented design with no definitive “best” method or design process. The process that covers here is a general one that incorporates activities that are common to most OOD processes. In this respect, it is comparable to the proposed UML process but you have significantly simplified this process for presentation here.

The general process that use here for object-oriented design has a number of stages:

1. Understand and define the context and the modes of use of the system
2. Design the system architecture
3. Identify the principal objects in the system
4. Develop design models
5. Specify object interfaces

A simple process diagram as that would imply that there was a neat sequence of activities in this process. In fact, all of the above activities can be thought of as inter-leaved activities that influence each other. Objects are identified and the interfaces fully or partially specified as the architecture of the system is defined. As object models are produced, these individual object definitions may be refined and this may mean changes to the system architecture.

The separate stages in the design process later in this section. However, you should not assume from this that design is a simple, well structured process. In reality, you develop a design by proposing solutions and refining these solutions as information becomes available. You inevitably have to backtrack and retry when problems arise. Sometimes you explore options in detail to see if they work; at other times you ignore details until late in the process. These process activities develop an example of an object-oriented design. The example that uses to illustrate object-oriented design is part of a system for creating weather maps using automatically collected meteorological data. The detailed requirements for such a weather mapping system would take up many pages. However, overall system architecture can be developed from a relatively brief system description:

A weather mapping system is required to generate weather maps on a regular basis using data collected from remote, unattended weather stations and other data sources such as weather observers, balloons and satellites. Weather stations transmit their data to the area computer in response to a request from that machine.

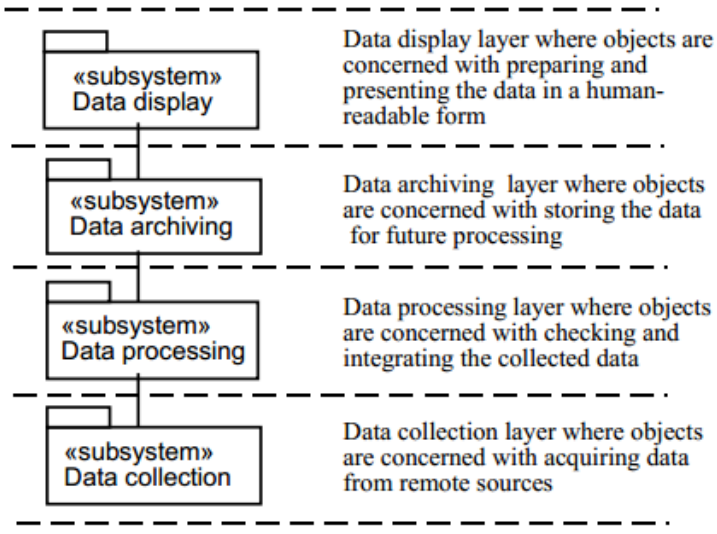


Figure 9.24: Layered architecture for weather mapping system.

The area computer system validates the collected data and integrates the data from different sources. The integrated data is archived and, using data from this archive and a digitized map database, a set of local weather maps is created. Maps may be printed for distribution on a special-purpose map printer or may be displayed in a number of different formats.

This description shows that part of the overall system is concerned with collecting data, part with integrating the data from different sources, part with archiving that data and part with creating

weather maps. Figure 9.24 illustrates a possible system architecture that can be derived from this description. This is a layered architecture that reflects the different stages of processing in the system namely data collection, data integration, data archiving and map generation. A layered architecture is appropriate in this case because each stage only relies on the processing of the previous stage for its operation. Figure 9.24 shown the different layers and have included the layer name in a UML package symbol that has been denoted as a subsystem. A UML package represents a collection of objects and other packages. It is here to show that each layer includes a number of other components. In Figure 9.25 it has expanded on this abstract architectural model by showing that the components of the subsystems. Again, these are very abstract and they have been derived from the information in the description of the system. The design example is explained by focusing on the weather station subsystem that is part of the data collection layer.

9.11.1 System Context and Models of Use

The first stage in any software design process is to develop an understanding of the relationships between the software that is being designed and its external environment. Developing this understanding helps you decide how to provide the required system functionality and how to structure the system so that it can communicate effectively with its environment.

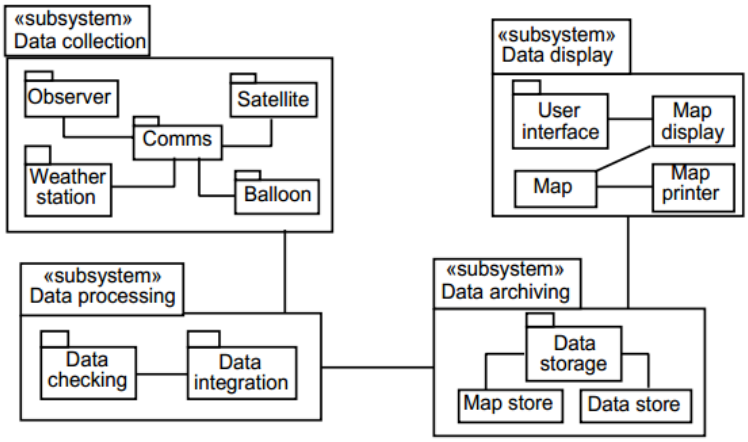


Figure 9.25: Subsystems in the weather mapping system.

The system context and the model of system use represent two complementary models of the relationships between a system and its environment:

1. The system context is a static model that describes the other systems in that environment.
2. The model of the system use is a dynamic model that describes how the system actually interacts with its environment.

The context model of a system may be represented using associations where, essentially, a simple block diagram of the overall system architecture is produced. This can be expanded by representing a subsystem model using UML packages as shown in Figure 9.25. This illustrates that the context of the weather station system is within a subsystem concerned with data collection. It also shows other subsystems that make up the weather mapping system.

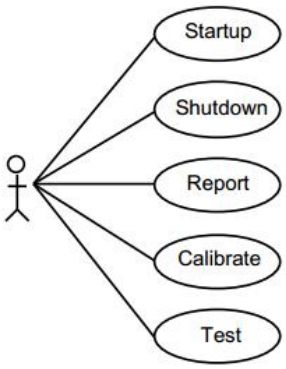


Figure 9.26: Use cases for the weather station.

System	Weather station
Use-case	Report
Actors	Weather data collection system, Weather station
Data	The weather station sends a summary of the weather data that has been collected from the instruments in the collection period to the weather data collection system. The data sent are the maximum minimum and average ground and air temperatures, the maximum, minimum and average air pressures, the maximum, minimum and average wind speeds, the total rainfall and the wind direction as sampled at 5 minute intervals.
Stimulus	The weather data collection system establishes a modem link with the weather station and requests transmission of the data.
Response	The summarised data is sent to the weather data collection system
Comments	Weather stations are usually asked to report once per hour but this frequency may differ from one station to the other and may be modified in future.

Figure 9.27: Report use case description.

When you model the interactions of a system with its environment you should use an abstract approach that does not include too much detail of these interactions. The approach that is proposed in the UML is to develop a use case model where each use case represents an interaction with the system. In use case models, each possible interaction is named in an ellipse and the external entity involved in the interaction is represented by a stick figure. In the case of the weather station system, this external entity is not a human but the data processing system for the weather data.

A use case model for the weather station is shown in Figure 9.26. This shows that weather station interacts with external entities for start-up and shutdown, for reporting the weather data that has been collected and for instrument testing and calibration.

Each of these use cases can be described using a simple natural language description. This helps designers identify objects in the system and gives them an understanding of what the system is intended to do. It uses a stylised form of this description that clearly identifies what information is exchanged, how the interaction is initiated etc. This is shown in Figure 9.27 where it has described the Report use case from Figure 9.26.

The use case description helps to identify objects and operations in the system. From the description of the Report use case, it is obvious that objects representing the instruments that collect weather data will be required as will an object representing the summary of the weather data. Operations to request weather data and to send weather data are required.

9.11.2 Architectural Design

Once the interactions between the software system that is being designed and the system's environment have been defined, you can then use this information as a basis for designing the system architecture. Of course, you need to combine this with your general knowledge of principles of architectural design and with more detailed domain knowledge

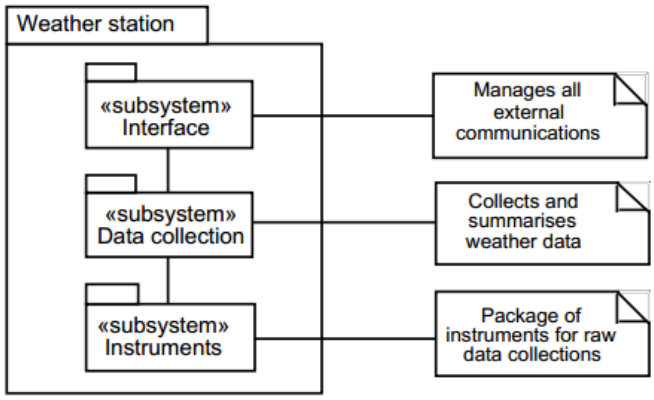


Figure 9.28: The weather station architecture.

The automated weather station is a relatively simple system and its architecture can again be represented as a layered model. It has illustrated this in Figure 9.28 as three UML packages within the more general Weather station package. Notice how it has used UML annotations (text in boxes with a folded corner) to provide additional information here.

The three layers in the weather station software are:

1. The interface layer which is concerned with all communications with other parts of the system and with providing the external interfaces of the system.
2. The data collection layer which is concerned with managing the collection of data from the instruments and with summarising the weather data before transmission to the mapping system.
3. The instruments layer which is an encapsulation of all of the instruments that are used to collect raw data about the weather conditions.

In general, you should try and decompose a system so that architectures are as simple as possible. A good rule of thumb is that there should not be more than seven fundamental entities included in an architectural model. Each of these entities can be described separately but, of course, you may choose to reveal the structure of the entities as it done in Figure 9.25.

9.11.3 Object Identification

By this stage in the design process, you will already have formulated some ideas about the essential objects in the system that you are designing. In the weather station system, it is clear that the instruments are objects and you need at least one object at each of the architectural levels. This reflects a general principle that objects tend to emerge during the design process. However, you usually also have to look for and document other objects that may be relevant.

Although it has headed in this section object identification, in practice this process is actually concerned with identifying object classes. The design is described in terms of these classes. Inevitably, you have to refine the object classes that you initially identify and revisit this stage of the process as you develop a deeper understanding of the design.

There have been various proposals made about how to identify objects classes:

1. Use a grammatical analysis of a natural language description of a system. Objects and attributes are nouns, operations or services are verbs. This approach has been embodied in the HOOD method for object oriented design that has been widely used in the European aerospace industry.
2. Use tangible entities (things) in the application domain such as aircraft, roles such as manager, events such as request, interactions such as meetings, locations such as offices, organizational units such as companies, etc. Support this by identifying storage structures (abstract data structures) in the solution domain that might be required to support these objects.
3. Use a behavioural approach where the designer first understands the overall behaviour of the system. The various behaviours are assigned to different parts of the system and an understanding is derived of who initiates and participates in these behaviours. Participants who play significant roles are recognised as objects.
4. Use a scenario-based analysis where various scenarios of system use are identified and analysed in turn. As each scenario is analysed, the team responsible for the analysis must identify the required objects, attributes and operations. A method of analysis called CRC cards where analysts and designers take on the role of objects is effective in supporting this scenario based approach.

These approaches help you get started with object identification. In practice, many different sources of knowledge have to be used to discover objects and object classes. Objects and operations that are initially identified from the informal system description can be a starting point for the design. Further information from application domain knowledge or scenario analysis may then be used to refine and extend the initial objects. This information may be collected from requirements documents, from discussions with users and from an analysis of existing systems.

There are five object classes shown in Figure 9.29. The Ground thermometer, Anemometer and Barometer represent application domain objects and the WeatherStation and WeatherData objects have been identified from the system description and the scenario (use case) description.
These objects are related to the different levels in the system architecture.

1. The WeatherStation object class provides the basic interface of the weather station with its environment. Its operations therefore reflect the interactions shown in Figure 9.26. In this case, we use a single object class to encapsulate all of these interactions but, in other designs, it may be more appropriate to use several classes to provide the system interface.

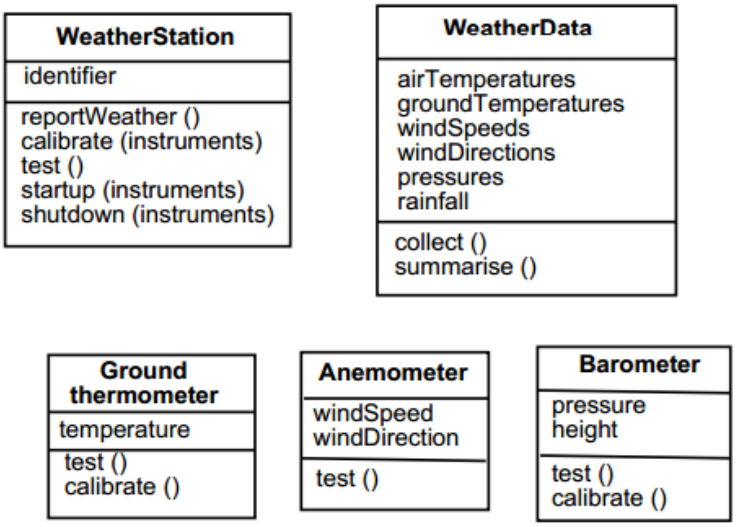


Figure 9.29: Examples of object classes in the weather station system.

2. The WeatherData object class encapsulates the summarised data from the different instruments in the weather station. It is associated operations are concerned with collecting and summarising the data that is required.
3. The GroundThermometer, Anemometer and Barometer object classes are directly related to instruments in the system. They reflect tangible hardware entities in the system and the operations are concerned with controlling that hardware.

9.12 Processing Controls and Data

An important function in computer auditing is the review and analysis of information systems security. One element of this is physical access control; another is software-based data processing controls that are intended to prevent data errors from occurring, to identify any errors that have occurred, and to assist with recovery from errors. Validation checks may be built into the database structure, or coded as part of a software application, to enforce business rules such as credit limits or minimum and maximum charges. Network and database access may be selectively restricted, and network monitoring software keeps track of which users logon to the system, what files they access, and what changes they make. Audit trails provide a transaction history showing where each entry originated and who was responsible, and serve an important function not only in identifying the source of problems but also in determining where staff training is needed. Sequential numbering of transactions and of audit reports ensures that every business activity is recorded. Where financial totals are transferred from subsidiary journals to the general ledger, the corresponding general ledger audit report should cross reference the audit report that reflects the individual transactions, and control totals in both ledgers must be reconciled. Exception reports should be produced in the case of transactions or totals exceeding previously specified limits.

Further data processing security for complex transactions can provide by database management systems. A single transaction may result in updates to a number of different files, for example a sale will affect both the accounts receivable and inventory files. In this case, all the required updates must be made successfully before the transaction is committed to the database; if an error occurs at any stage after the start of processing, then all the updates will be rolled back (i.e. undone), to prevent the possibility of inconsistent entries. Additional management controls should be implemented to reduce data processing risk. Employees should be thoroughly trained in the software and procedures that they are expected to use. Regular backups should be made and stored in secure off-site locations. Separation of duties among employees reduces the risk of deliberate fraud, and awareness of ethical standards should be a part of company policy.

9.13 Verification and Validation

The aim of software testing is to measure the quality of software in terms of number of defects found in the software, the number of tests run and the system covered by the tests. These tests are carried out for both the functional and non functional attributes of the software. When bugs or defects are found with the help of testing, the bug is logged and the developer's team fixes the bug. Once the bug is fixed and testing is carried out again to ensure that the bug was indeed fixed and no new defects have been introduced in the software. With the entire cycle the quality of the software increases. Verification and validation testing are two important tests, which are carried out on software, before it has been handed over to the customer. This makes sure, that the software testing life cycle starts early. The aim of both verification and validation is to ensure that the software product is made

according to the requirements of the client and does indeed fulfill the intended purpose. So that the software product is tested thoroughly without any bias, often the job of validation testing may also be given to third party validation testing services. Therefore, validation testing is an important part of software quality assurance procedures and standards.

9.13.1 Software Validation Testing

While verification is a quality control process, quality assurance process carried out before the software is ready for release is known as validation testing. The validation testing goals is to validate and be confident about the software product or system, that it fulfills the requirements given by the customer. The acceptance of the software from the end customer is also a part of validation testing.
Validation testing answers the question, "Are you building the right software system". Another question, which the entire process of validation testing in software engineering answers is, "Is the deliverable fit for purpose". In other words, does the software system provide the right solution to the problem? Therefore, often the testing activities are introduced early in the software development life cycle. The two major areas, when validation testing should take place are in the early stages of software development and towards the end, when the product is ready for release. In other words, it is acceptance testing which is a part of validation testing.

9.13.2 Validation Testing Types

If the testers are involved in the software product right from the very beginning, then validation testing in software testing starts right after a component of the system has been developed.
The different types of software validation testing are:

Component Testing

Component testing is also known as unit testing. The aim of the tests carried out in this testing type is to search for defects in the software component. At the same time, it also verifies the functioning of the different software components, like modules, objects, classes, etc., which can be tested separately.

Integration Testing

This is an important part of the software validation model, where the interaction between the different interfaces of the components is tested. Along with the interaction between the different parts of the system, the interaction of the system with the computer operating system, file system, hardware and any other software system it might interact with is also tested.

System Testing

System testing, also known as functional and system testing is carried out when the entire software system is ready. The concern of this testing is to check the behavior of the whole system as defined by the scope of the project. The main concern of system testing is to verify the system against the specified requirements. While carrying out the tester is not concerned with the internals of the system, but checks if the system behaves as per expectations.

Acceptance Testing

Here the tester especially has to literally think like the client and test the software with respect to user needs, requirements and business processes and determine, whether the software can be handed over to the client. At this stage, often a client representative is also a part of the testing team, so that the client has confidence in the system.

There are different types of acceptance testing:

- Operational acceptance testing
- Compliance acceptance testing
- Alpha testing
- Beta testing

Often when validation testing interview questions are asked, they revolve around the different types of validation testing. The difference between verification and validation is also a common software validation testing question.

Self Assessment Questions

7. The prototyping model cannot be used when the technical solutions are unclear to the development team.

(a) True (b) False

8. The JAD provides a working environment in which to accelerate methodology activities and deliverables.

(a) True (b) False

9.14 Audit Trail and Documentation Control

An important function of system controls is providing for an audit trail. An audit trail is a routine designed to allow the analyst, user, or auditor to verify a process or an area in the new system. In a manual system, the audit includes journals, ledgers, and other documents that the auditor use traces transactions through the system. In a computerized system, re content and format frequently make it difficult to trace a transaction completely.

Some reasons are the following:

1. Records stored on a magnetic medium (tape, disk, etc.) can be read only by a computer and with the use of a computer program.
2. Data processing activities are difficult to observe, since they take place within the computer system.
3. The record sequence is difficult to check without the assistance of a computer system.
4. Direct data entry eliminates the physical documentation for an audit program.

One way to overcome these constraints is to keep a file on all transactions as they occur. For example, transactions can be recorded on tape, which can be an input to an audit program. The program pulls selected transactions and makes them available for tracing their' status. The systems analyst must be familiar with basic auditing or work closely with an auditor to ensure an effective audit trail during the design phase.

The proper audit of a system also requires documentation. Documentation is the basis for the review of internal controls by internal or independent auditors. It also provides a reference for system maintenance. Preparing documentation occupies much of the analyst's time. When the implementation deadline is tight, documentation is often the first item to be ignored.

Documentation may be internal (in program documentation) or external hard-copy documentation. It must be complete and consistent for all systems prepared according to standards. So a plan to approve a new design should include documentation standards before programming and conversion.

The primary purpose of auditing is to check that controls built into the design of candidate systems ensure its integrity.

Caution
Audit considerations must be incorporated at an early stage in the system development so that changes, otherwise it can be timing consuming and confusing.

9.15 Summary

- Preliminary system study is the first stage of system development life cycle.
- Systems analysis is a process of collecting factual data, understand the processes involved, identifying problems and recommending feasible suggestions for improving the system functioning.
- Transform analysis is a set of design steps that map DFDs with transform characteristics into a design structure chart.
- Structured analysis and design technique is a data flow-oriented design approach.
- The SADT methodology provides a precise and concise representation scheme and a set of techniques to graphically define complex system requirements.
- The Jackson system development method is a data structure-oriented design approach.
- Functional decomposition is good in a procedural programming environment.
- The OOD methodology is a recent development as such it is still dynamic and evolving.
- Cohesion is a measure of the relative functional strength of a module.
- The basic concept of information engineering is that information systems should he engineered like other products.

9.16 Keywords

Coupling: A coupling is a device used to connect two shafts together at their ends for the purpose of transmitting power.
Data Flow Diagrams (DFDs): A data flow diagram is a graphical representation of the flow of data through an information system, modelling its process aspects.
Jackson System Development (JSD): Jackson system development is a method of system development that covers the software life cycle either directly by providing a framework into which more specialized techniques can fit.
Object-oriented Design (OOD): Object-oriented design is the process of planning a system of interacting objects for the purpose of solving a software problem. It is one approach to software design.
Structured Analysis and Design Technique (SADT): Structured analysis and design technique is a software engineering methodology for describing systems as a hierarchy of functions.
Structured Design (SD): Structured design is the art of designing the components of a system and the interrelationship between those components in the best possible way.

9.17 Review Questions

1. What is design process?
2. Describe the phases of design in system analysis.
3. What are software design methodologies?

4. What is method Jackson system development?
5. What is functional decomposition?
6. Explain the different types of coupling those two modules might exhibit.
7. Explain the different types of cohesion that a module might exhibit.
8. What is coupling and cohesion in reference to software design? How are these concepts useful in arriving at a good design of a system?
9. Discuss the prototyping model and joint application development.
10. What is rapid application development model? Explain with diagram.

Answers for Self Assessment Questions

1 (b)	2 (c)	3 (b)	4 (b)
5 (d)	6 (c)	7 (b)	8 (a)

9.18 Further Studies

1. Global SAP Systems - Design and Architecture by Sap PressAmerica, 2003

10

Input, Output and Form Design

STRUCTURE

10.0 Objectives

10.1 Introduction

10.2 Input Design

10.3 Output Design

10.4 Graphics

10.5 Desktop Publishing

10.6 Form Design

10.7 Layout Considerations

10.8 Automated Form Design

10.9 Forms Control

10.10 Summary

10.11 Keywords

10.12 Review Questions

10.13 Further Studies

10.0 Objectives

After studying this chapter, you will be able to:

- Discuss the input design of system
- Discuss the output design of system
- Explain graphics design
- Understand about desktop publishing
- Discuss the form design
- Explain layout considerations of form
- Discus how to design an automated form
- Explain form controls

10.1 Introduction

In this chapter we define systems design as the process of developing specifications for a candidate system that meet the criteria established in systems analysis. A major step in design is the preparation of input and the design of output reports in a form acceptable to the user. This chapter reviews input and output design and the basics of forms design. As we shall see, these steps are necessary for successful implementation.

10.2 Input Design

Inaccurate input data are the most common cause of errors in data processing. Errors entered by data entry operators can be controlled by input design. Input design is the process of converting user-originated input to a computer-based format. In the system design phase, the expanded data flow diagram identifies logical data flows, data stores, sources and destinations. A systems flowchart specifies master as (data base), transaction files, and computer programs. Input data are collected and organized into groups of similar data. Once identified, appropriate input media are selected for processing.

10.2.1 Input Data

The goal of designing input data is to make data entry as easy, logical, and free from errors as possible. In entering data, operators need to know the following:

- The allocated space for each field.
- Field sequence, which must match that in the source document.
- The format in which data fields are entered; for example, filling out the data is required through the edited format mm/dd/yy.

When we approach input data design. We design the source documents that capture the data and then select the media user to enter them into the computer.

10.2.2 Source Documents

Source data are captured initially on original paper or a source document. For example, a check written against an account is a source document. When it reaches the bank, it is encoded with special magnetic ink character recognition (MICR) so that it can be processed by a reader that is part of the information system of the bank. Therefore, source documents initiate a processing cycle as soon as they are entered into the system.

Source documents may be entered into the system from punch cards, from diskettes, or even directly through the keyboard. A source document may or may not be retained in the candidate system. Thus, each source document may be evaluated in terms of

- Its continued use in the candidate system
- The extent of modification for the candidate system and
- Replacement by an alternative source document should be logical and easy to understand.

10.3 Output Design

Computer output is the most important and direct sources of information to the user. Efficient, intelligible output design should improve the system's relationships with the user and help in decision making. A major form of output is a hard copy from the printer. Print outs should be designed around the output requirements of the user. The output devices to consider depend on factors such as compatibility of the device with the system response time requirements, expected

print quality, and number of the copies needed. The following media devices are available for providing computer and output:

1. MICR Readers
2. Line, Matrix and Daisy Wheel Printers
3. Computer Output Micro Film (COM)
4. CRT Screen Display
5. Graphic Plotters
6. Audio Response

10.4 Graphics

Get a better understanding of the basics of graphic design by studying the elements and principles of graphic design that govern effective design and page layout. Graphic design is the process and art of combining text and graphics and communicating an effective message in the design of logos, graphics, brochures, newsletters, posters, signs, and any other type of visual communication. Designers achieve their goals by utilizing the elements and principles of graphic design.

Graphic design is almost everywhere. Crammed into our homes, all over our cities and dotted around the countryside, its images, letters, colours and shapes are consciously put together to perform all sorts of functions.

In short, graphic design is visual communication. It employs lots of different techniques and modes, but is very seldom purely decorative: graphic design has a job to do and graphic designers are in the employ of their clients. The graphic designer may be briefed to create a piece of work which catches a customer's eye in a busy supermarket, or they may be required to herald the formation of a new business. Their client may want their work to impart cultural knowledge at a museum or help foreign tourists find their way to the bus station. Or graphic designers could be employed for something as run of the mill as creating a new look for the company stationery. Using an array of visual elements – including type, colour, shape, photography, illustration, painting, and digital imagery and so on – graphic designers work with their clients to deliver the required message in the most effective way.

10.4.1 The origins of graphic design

While it can be hard to find an exact definition of graphic design it is equally hard to pinpoint its origins: some say the invention of the mechanical printing press in the 15th century, while others point to the work of commercial artists in the 19th century or illuminated manuscripts of the middle ages. And although most graphic design is undertaken as a commercial service, visual art is sometimes included in its definition.

The discipline, already wide and varied, is also changing and expanding. Traditionally concerned with print, graphic design now includes other media, notably design for digital screens, as well as being a vital component in other specialist disciplines such as branding, packaging and information design.

But to start with, here is a small selection of designs to offer a feel for the many and varied areas in which graphic designers work. They span a wide range of industries, media, styles and objectives. In this set we have album artwork, the editorial design for a newspaper, a magazine layout, a corporate logo, road signage, a visual identity for an awards scheme, a transport map, packaging for bread, brochures for an arts centre, a poster and title sequences for a film and a mobile phone application. All draw on the skills of graphic designers and all meet commercial or functional, as well as aesthetic, needs.

10.5 Desktop Publishing

Desktop publishing is a term coined after the development of a specific type of software. It is about using that software to combine and rearrange text and images and creating digital files.

Before the invention of desktop publishing software the tasks involved in desktop publishing were done manually, by a variety of people and involved both graphic design and prepress tasks which sometimes leads to confusion about what desktop publishing is and how it is done.

Desktop publishing is the use of the computer and software to create visual displays of ideas and information. Desktop publishing documents may be for desktop or commercial printing or electronic distribution including PDF, slide shows, email newsletters, and the Web.

Old/Traditional Definition: Desktop publishing is the use of the computer and specialized software to create documents for desktop or commercial printing. Desktop publishing refers to the process of using the computer to produce documents such as newsletters, brochures, books, and other publications that were once created manually using a variety of non-computer techniques along with large complex phototypesetting machines. Today desktop publishing software does it all - almost. But before PageMaker and other desktop publishing software there were e-scales, paste-up, and other non-desktop computer ways of putting together a design for printing.

Properly speaking, desktop publishing is the technical assembly of digital files in the proper format for printing. In practical use, much of the "graphic design" process is also accomplished using desktop publishing and graphics software and is sometimes included in the definition of desktop publishing. Comparison between desktop publishing and graphic design:

What is Desktop Publishing: It is the process of using the computer and specific types of software to combine text and graphics to produce documents such as newsletters, brochures, books, etc.

What is Graphic Design: It is the process and art of combining text and graphics and communicating an effective message in the design of logos, graphics, brochures, newsletters, posters, signs, and any other type of visual communication. Desktop publishing software is a tool for graphic designers and non-designers to create visual communications.

10.5.1 Use of Desktop Publishing

Just about anyone can do desktop publishing with the right software? However, there are specific groups who tend to make the most use of the techniques of desktop publishing.

Freelance and in-house graphic designers, small business owners, secretaries, teachers, students, and individual consumers do desktop publishing. When desktop publishing software first debuted, desktop publishing was almost the exclusive realm of graphic designers. However, with the advent of more desktop publishing software and easier-to-use, consumer-oriented software desktop publishing became accessible to a wider range of people, including non-designers and others without graphic design experience.

Here are more detailed explanations of who does desktop publishing for profit (business people):

- A graphic designer or other professional (often with some formal training or background) who uses desktop publishing software (most often the high-end programs such as Adobe PageMaker or QuarkXPress) to design documents for clients.
- A (often) small or home-based business person (often self-trained or with a background other than design or printing) who uses their computer and desktop publishing software to design documents for clients.

- A (often) small or home-based business person with a word processing, resume, or office support business that uses their word processing software (usually) to design documents for their word processing clients.
- A person who uses their computer and desktop publishing software to design documents for their own business (not for clients). Will usually use low-end to midrange desktop publishing software and print documents directly to their desktop printers.
- A secretary, administrative assistant, or other office support person who uses word processing or desktop publishing software to design documents for an employer — in addition to their normal, non-design duties.
- Additionally, desktop publishing software is frequently used by consumers for personal use for various desktop printed projects such as greeting cards, family newsletters, and other creative printing projects.

10.5.2 Where is Desktop Publishing Used?

Desktop publishing software can be found in both graphic design firms and other types of businesses of all sizes, homes, schools, quick copy centres, service bureaus, and print shops.

Just about anywhere there is a need to produce printed communications from billboards to baby announcements to business cards; you can find desktop publishing software and people using it.

The type of desktop publishing software in use varies. Homes and small businesses may utilize consumer-level desktop publishing software packages designed for non-designers while freelance graphic designers, graphic design firms, corporations, and print shops utilize high-end applications designed for professional and high-volume use. Despite the differences in the software, types of documents created, and where the software is used, all these types of uses fall under the umbrella of desktop publishing.

Did You Know?

Desktop publishing began in 1983 with a program developed by James Bessen at a community newspaper in Philadelphia.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Make a list of software through which we create desktop applications.

……

……

……

10.6 Form Design

10.6.1 What is the Form?

People read from forms, write on forms and spend billions of hours handling forms and filing forms. The data the forms carry come from people and the information output of the system goes to people. So the form is a tool with a message; it is the physical carrier of data of information. It also can constitute authority for action.

For example:

Purchase Order says buy, a customer's order says and pay check says pay to the order.

10.6.2 Classification of Forms

A printed form is generally classified by what it does in the system. There are three primary classifications:

- Action
- Memory
- Report

Form action form requests the user to do something get action.

Example:

Purchase Orders and shop orders

A memory form is a record of history data that remains in a file, is used for reference and serves as control on details.

Example:

Inventory Records, Purchase Records and Bond Record activities

It provides data on a project or a job.

Example:

Profit and statements and sales analysis reports

10.6.3 Requirements of Forms Design

Form design follows analysing forms, evaluating present documents and creating new or improved forms. Bear in mind that detailed analysis occurs only after the problem definition stage and the beginning of designing the candidate system. Since the purpose of a form is to communicate effectively through forms design, there are several major requirements:

Identification and Wording

The form title must clearly identify its purpose. Columns and rows should be labelled to avoid confusion. The form should also be identified by firm's name or code number to make it easy to reorder.

Maximum Readability and Use

The forms must be easy to use and fill out. It should be legible, intelligible and uncomplicated. Ample writing space must be provided for inserting data. This means analysing for adequate space and balancing the overall forms layout, administration, and use.

Physical Factors

The form's composition, colour, layout and paper stock should lend them to easy reading. Pages should be numbered when multi page reports are being generated for the user.

Order of Data Items

The data requested should reflect a logical sequence. Related data should be in adjacent positions. Data copied from source documents should be in the same sequence on both forms. Much of this design takes place in the forms analysis phase.

Ease of data entry

It used for data entry, the form should have field positions indicated under each column of data and should have some indication of where decimal points are (use broken vertical lines).

Size and Arrangement

The form must be easily stored and filed. It should provide for signatures. Important items must be in a prominent location on the form.

Use of Instructions

The instructions that accompany a form should clearly show how it is used and handled.

Efficiency Consideration

The form must be cost effective. This means eliminating unnecessary data and facilitating reading lines across the form. The illustrate, if a poorly designed form causes 10 supervisors to waste 30 seconds each, then 5 minutes are lost because of the form. If the firm uses 10,000 of these forms per year, then 833 hours of lost time could have been saved by a better forms design.

Type of Report

Forms design should also consider whether the content is executive summary, intermediate, managerial information, or supporting data. The user requirements for each type often determine the final form design.

Types of Forms

Forms are classified into several categories:

- Flat Forms,
- Unit Set/Snap out Forms
- Continuous Strip/Fanfold Forms,
- NCR Paper
- Pre-printed Forms.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Prepare a list of some software to created automated form.

……

……

……

Did You Know?

While Graphic Design as a discipline has a relatively recent history, with the term "graphic design" first coined by William Addison Dwiggins in 1922,

10.7 Layout Considerations

Forms design is the outgrowth of forms analysis. Only after you have established the need for the form and determined the procedures affecting its use can you realistically decide on the right design. The design comes about as you integrate the needs of the people completing the information requested, the people processing the data, and the people involved in mailing, filing, and any other disposition of the form.

10.7.1 Principles of Design Form Layout

Good principles of design will result in a form that is easier to read and understand, to fill in, and to file.

There are five parts to almost every form:

1) Heading - includes a title and a form number.
2) Instructions - explain how to fill in the form and what to do with the completed form.
3) Introduction - explains the purpose of the form.
4) Body - consists of the sequenced items requesting the desired information.

5) Closing - provides space for approval signatures.



The application of the following design rules for simple forms will usually result in an efficient, easy to use product:

- Study the purpose and use of the form and design it with the user in mind.
- Keep the design simple. Use a minimum of type fonts and sizes; eliminate unnecessary information and lines.
- Include a form number and name on each form.
- Use standard sizes of paper (or screens, if automated) where practical.
- Use standard terminology in wording instructions.
- Arrange items in a logical sequence.
- Arrange items in the sequence in which the information will be extracted during processing.
- Preprint constant data (such as agency name) so as to keep variable (fill-in) data to a minimum.
- Allow sufficient spacing for the method of fill-in (manual, typewriter, and computer).

10.7.2 Identification

When using a form for the first time, a person reads the title first, in order to gain an idea of the purpose of the form. Some kind of identification is needed to make the purpose and function understandable to the user. In addition to the title, identification will include:

- Agency name.
- Form number, date of edition.
- Any suppression notices (“This form replaces RMD 101”).
- Any internal control symbols.

Different users will pay attention to different parts of the identification. A member of the public is interested in the agency name and form title; a stock clerk is interested in the form number, edition date, and suppression notice; a file clerk is primarily interested in the form number. Place the

identification information on the form to make it accessible to all users, no matter what their emphasis.

10.7.3 Title and Subtitle

Where the title is placed can be determined by how the form is used. Top left is often used when the upper right corner is reserved for filing data. Or the title can be cantered across the top to increase its visibility in filing equipment or to eliminate a break in typing sequence. A subtitle is helpful, especially if the form is used by the public, to explain or qualify the main title. If there is more than one type of a category of form, for example, "Daily Contact Report," each form should be distinguished by a subtitle under the main title, such as "Requests from Field Offices" or "Shipments."

10.7.4 Organizational Name

The agency name should appear on all forms. If the form is to be filled in by the public or by personnel from other agencies, definitely include the agency name to avoid confusion and aid routing.

10.7.5 Form Number

The best place for the form number is either the lower right or lower left margin of the page, for the following reasons:

- Prevents tearing or obliterating the number if the form is stapled at either upper corner.
- Permits the number to be readily seen if the forms are bound at the top.
- Serves as an aid in stocking forms, especially when forms are stocked in small quantities in supply cabinets.

When a form consists of several pages (or screens), put the form number on each page. Similarly, if a form is printed on the back as well as the front, have the form number appear on the back side as well.

10.7.6 Edition Date

Edition dates are valuable in determining that the current form is being used, in disposing of obsolete stock, in showing how long it has been since a form has been changed, and for reference when writing procedures. Place the edition date adjacent to the form number.

10.7.7 Page Identification

When a form consists of two or more pages which are folded, stapled, or bound, page numbers help to:

- Aid the printer in assembling and collating.
- Aid instructions in the use of the form, i.e., "Turn to page 2 if changing claim status."
- Identify the form, especially if pages are separated for processing.

The page number is usually placed in the upper right corner. If the number of continuation pages is unknown, number the pages with a "Page …….. of ……." The page number and total number of pages can be filled in by the person completing the form.

10.7.8 Suppression Notice

It is a good forms management practice to have some way of notifying users and those in charge of supplies when a form has been revised, or when an existing form is being replaced by a new form. The suppression notice does this, and it is usually placed near the form number. Some ways of wording the suppression notice include:

1) When revising an existing form, use:

a. Previous editions are obsolete.
b. Previous editions may be used until supply is exhausted.
c. Existing stocks of (form number and edition date) will be used.
d. Existing stocks of (form number and edition date) will not be used.

2) When replacing an existing form with a different number, use:
a. Replaces (form number and edition date) which is obsolete.
b. Replaces (form number and edition date) which may be used until supply is exhausted.

10.7.9 Instructions

Place brief instructions at the top of the form below or near the title to tell the user how many copies are required, who should submit the form, and where, when, and to whom copies are to be sent.

If detailed instructions are included elsewhere, such as on the reverse, direct the user to that area ("See reverse for instructions.") Short instructions that relate to a specific section should be placed at the top of that section.

Longer, more involved instructions are placed:

- On the front of the form if there is enough room for both the instructions and the fill-in data.
- On the back, if there is not sufficient space on the front.
- On a separate sheet or in a booklet.
- In an administrative directive or agency operations manual.

10.7.10 Routing

You can significantly simplify the handling of forms by incorporating effective routing and mailing design techniques. These techniques can also reduce the chance of errors and speed the delivery of the mail.

When possible, make the form self-routing, eliminating the need for a routing slip or transmittal letter. This can be done in several ways, such as:

- A "to/from" line or lines which the user fills in.
- A routing name or address pre-printed, for constant routing.
- A pre-printed multiple arrangements of departments, with instructions to route in the order listed.

10.7.11 Arrangement

The arrangement of the items on the form, other than the identification data, should facilitate the entering and retrieving of information. There should be a discernible sequence of the fields or items, so the user does not have to move from one area of the form to another and back again to either complete or interpret the form. Three basic arrangement factors are involved:

1) Grouping data.
2) Establishing item sequence.
3) Aligning data.

Grouping Data - If different people are to be entering data on the same form, have the item arrangement match the sequence of the processing steps so that the last person to enter data on the form is entering it at the bottom or on the last page of the form. This eliminates the need for backtracking or searching the form for the correct entry area. If the form is used as a source document to collect data on different types of materials, group the related items together.

It is helpful to identify the groupings either by numbering or lettering. You can identify subgroups the same way you would in an outline, with letters following numbers in the progression of importance.

Establishing Item Sequence: Grouped related items, put them in a sequence that will allow the user to move from one to the next without having to look back up the page. This can aid the person filling in the form as well as the person transcribing or reviewing the information. Numbering the items can inform the user that there is a prescribed sequence to follow.

Aligning Data: Arrange the items to follow people's visual habits, from left to right and from top to bottom. By arranging the form in this manner, you can relieve the user of wasted motion. If the information is to be entered using a typewriter, punch machine, or computer, help the user by aligning items vertically to reduce the number of tabular or marginal stops.

10.7.12 Paper Size

Sizes of paper stock are standardized. The most common sizes are letter (8 1/2 × 11), legal (8 1/2 × 14), and ledger (17 × 22). Reduced paper cost is not the only reason for choosing standard sizes. Greater cost savings are to be found in using equipment and supplies which have also been standardized. Each time a nonstandard size form is used, it forces the use of nonstandard equipment, and costs rise.

State agencies are required by Texas Government Code, §2051.021, to reduce the use of legal size paper for any application, including forms. Careful consideration should be given any variation from the established paper sizes.

10.7.13 Margins

Some space should be allowed around the text area of a form for utility as well as appearance. Some printers and reproduction shops require margins as working space for the sprocket holes that permit machines to grip the paper during printing, or for trimming the paper when several copies of a form are printed on larger sheets. Allow at least 1/3 inch at the top, 1/2 inch at the bottom, and 3/10 inch at the sides. If using card stock, allow 1/8 inch on all sides.

Sometimes it is necessary for the image on the form to extend to the edge of the paper, for example, if the form consists of several pages put together in an overlapping configuration to indicate comparative or cumulative figures. Printing the image to the edge requires printing on a sheet of paper larger than the finished form size, then trimming the printed page to the finished size. This process is called bleeding, which means to run off the edge of the trimmed printed sheet. If the form is designed for offset printing, a good practice is to draw lines beyond the image size. When trimmed, the lines will bleed off the edge of the paper, leaving a clean edge. Because of the extra handling and trimming, bleeding can be expensive, so use it only if you need it for making a form more effective.

10.7.14 Spacing

Space requirements will be determined by the amount of fill-in data needed as well as the amount of printed material such as captions, headings, and instructions. The writing method (hand, typewriter, computer or other office machine) determines the amount of space you should allow for fill-in data, and the number of characters per inch of typeface used determines the amount of space needed for printed matter.

Horizontal spacing is based on the number of characters written per inch and is determined by the writing method. Vertical spacing is based on the number of writing lines that can be written per inch.

Many forms are typewritten, some are handwritten, and a small percentage combines the two methods. Since most forms created on computer for computer entry are designed to be compatible with the machine spacing, incorrect horizontal or vertical spacing is rarely a problem.

Typewritten Spacing

Horizontal Spacing: There are 12 characters of elite type and 10 characters of pica type to the horizontal inch on standard typewriters. Accordingly, when counting horizontal spaces, allow 1/12 inch for elite and 1/10 inch for pica type; 1/10 inch accommodates either elite or pica type and allows maximum entry space. Whenever possible, add a minimum of one extra space to the required number of characters to prevent crowding.

Vertical Spacing: There are six vertical lines per inch on the standard typewriter, elite or pica. Accordingly, 1/6 inch, or a multiple of 1/6 inch, should be allowed for each line of typing. By measuring spacing this way, you require the user to adjust the form in the typewriter for the first line of typing only, after which no further adjustments are needed.

Handwritten Spacing

Horizontal Spacing: Provide 1/10 to 1/6 inch per character to avoid crowding or excessive abbreviation.

Vertical Spacing: Provide 1/4 inch to 1/3 inch per line. When using a box design, allow 1/3 inch. Otherwise, 1/4 inch is usually plenty of space for handwritten entries.

If a form is filled in by both typewriter and handwritten methods, determine the horizontal space by hand fill-in requirements and the vertical space by typewriter requirements. The 1/3 inch vertical spacing will accommodate either method of entry.

Caution

Lack of information of each section of a form can generate a problem while filling the form.

Self Assessment Questions

1. ..... data are captured initially on original paper document. For example, a check written against an account is a source document.

(a) Source (b) Destination (c) Both (a) and (b) (d) None of these.

2. Punch cards are either.....or 96 columns wide.

(a) 79 (b) 80 (c) 81 (d) 82

3. Mark-sensing readers automatically convert pencil marks in predetermined locations on a card to punched holes on the .....card.

(a) same (b) different (c) Both (a) and (b) (d) All of these

4. Optical character recognition (OCR) readers are similar to .....readers.

(a) MCIR (b) MIRC (c) MICR (d) MRIC

5. A source document collects.....an input action, and provides a record of tilt original transaction.

(a) input data (b) triggers

(c) authorizes an input action (d) All of these

10.8 Automated Form Design

These are basically online forms such as reservation forms, online forms online shopping forms etc. The automated or electronic forms are created through some software such as Microsoft Word, Adobe Designer, etc.

10.8.1 Creating an Automated Form in a Word 2007

This example describes how to create a simple form in a Word document that automatically prompts a user to fill in information.

How to Create the Template

To create a template with automatic FILLIN fields, follow these steps:

1. On the File menu, click New.
2. In the new document task pane, click general templates under new from template. In the templates dialog box, select the template that you want to use. Under Create New, click template. Click OK.
3. Create the FILLIN fields. To do this, use either of the following methods. Method 1: Create a Field by Using the Menus
   - Position the insertion point where you want to insert the text field.
   - On the Insert menu, click Field.
   - In the Categories list, click Mail Merge.
   - In the Field Names list, click Fill-in.
   - In the Field properties box, type quotation marks around the message that you want to display.

For example, use the following syntax to display a message that prompts users to enter a first and last name:

FILLIN "Please enter your first and last name."

- Click OK.

A sample of the message appears. Click OK to return to your document.

NOTE: To view the field code that you just inserted, press ALT+F9.

- Repeat steps a through f for every place in the document where you want to insert a FILLIN field.

Method 2: Create a Field by Using Keystrokes

- Position the insertion point where you want to insert the field.
- Press CTRL+F9.

Field braces ({ }) appear in the document.

- Position the insertion point inside the field braces.
- Type the following

FILLIN "message"

Where message is the instruction that Word prompts the user with (about what to enter in this field).NOTE: If you press F9 while the insertion point is still on the field, you can see a sample of the message that will be displayed. You do not have to follow this step to create the FILLIN field.

4. On the File menu, click Save As.
5. Name the template appropriately.

How to Use the Template

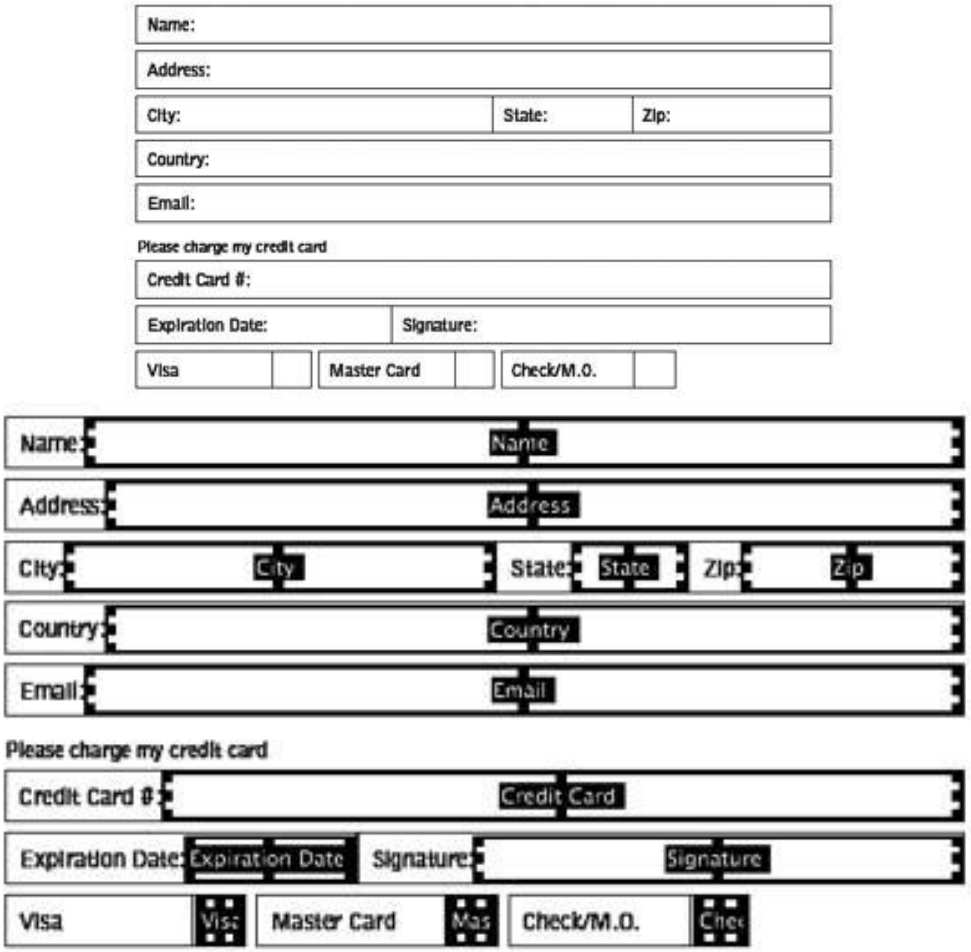
To create new documents based on the new template, follow these steps:

1. On the file menu, click New.
2. In the new document task pane, click general templates.
3. In the templates dialog box, select your template, and then click OK.

Word creates a new document, automatically searches for all FILLIN fields in the document, and then starts to prompt the user for input.

10.8.2 Automated Forms with Acrobat 8 and In Design

One of the hottest new features in Acrobat 8 Professional is its Run Form Field Recognition (RFFR) feature. Almost everyone needs to work with forms sometimes, and often we did like to create a "live" PDF form which can be filled out and submitted with a web browser or emailed. Forms have been part of Acrobat for several versions, but the downside of creating them has been the tedious labour of manually creating form fields, one by one.With Acrobat 8 Professional, if you have a relatively simple form composed of lines, checkboxes, and form field labels, RFFR uses some artificial intelligence to automate the process. To use the feature, you simply choose Forms > Run Form Field Recognition. RFFR can recognize many of the features of a form, and can often deduce from the placement of labels where form fields can be placed. Here is an example of such a form and the results of the default conversion:



The results are not perfect, but it is a great time-saver. Often, for a simple form, you only have to do minor tweaking to get a useful form.

10.9 Forms Control

Before a form is created, determine the need for its creation. Chart the entire life cycle of the form, from initial use, through all processes required, to final disposition. The person creating the form should be charged with these responsibilities:

- Prove the need for the form.
- Identify the purpose for which the form will be used.
- Specify how the form will be used.
- Determine the number of copies needed for the procedure.
- List the titles of personnel (not just the department) who will receive copies.
- Specify how the copies will be prepared.
- Specify how the copies will be distributed.
- Identify where the record copy and working copies will be retained.
- Determine how long the record copy and working copies will be retained.

Many forms originate in the offices of agency officials. The officials know how the forms are used and how their use affects other operations, although they may not be aware of changes that require revisions to the forms.

If a form is to be revised, that official usually must approve the final revisions. It is natural for this manager, supervisor, or management officer to feel that the daily responsibilities of the job are enough to handle without worrying about the analysis and improvement of forms. As a result, agency officials may be overlooking that unsuitable forms, or the need for more or fewer forms, may actually be causing operating problems in the agency. Supervisors should be aware of the time that they and their staff spend both in studying forms for possible changes and in handling problems caused by inefficient forms. Suggestions arising from these problems should be forwarded to the forms management staff (or the records management officer) for a detailed forms analysis.

Many forms travel from the originating office to other departments for preparation and use. In the majority of agencies, most forms are administrative, and the various departments have similar operations. This often results in forms from the different departments being similar or redundant. The analyst can assist with standardization of these forms or with the elimination of needless forms.

10.9.1 Control Files

The use of forms control files can be very effective in controlling the creation of unnecessary forms. Forms control files are also a basic part of the overall forms management program. The two most common types of forms control files are the numeric history file and the functional classification file.

Numeric History File

This control file provides a complete profile of each form from its creation to its current status. A number assigned to each form indicates that it has been approved for use, and the numbering system should be as simple as possible. The numeric file consists of copies of each form used by the organization, arranged in numerical order so that forms can be easily located. A numeric file folder for each form should eventually contain the following:

- A copy of the current edition of the form and any previous editions.
- Drafts showing only significant stages of development and pertinent correspondence.
- A copy of any directive authorizing use of the form.
- The original request for approval of the form and any requests for revisions.
- A list of all departments using the form and the rate of use.
- Final official agency approval for the printing, reproduction, and distribution of the form.
- Cross reference to the functional classification file.

The numeric history file must be consistently reviewed and updated. Remove files for discontinued and obsolete forms, annotate the folders accordingly, and place in a separate discontinued numeric file for the amount of time specified by your agency records retention schedule.

Functional Classification File

This control file brings together forms dealing with related subjects. Classify one copy of each form by its purpose and place it in the appropriate subject title folder. For example, in the "Hospitalization" folder will be the forms for hospitalization claims, hospitalization premiums, surgery forms, and so on. This classification can also group forms that serve similar areas in the agency, such as payroll, personnel, and purchasing. Each of the classifications can then be subdivided as you determine appropriate. The main purpose of the functional classification file is to accomplish the following:

- Avoid the creation of new forms that are very similar to existing forms.
- Revise existing forms to serve changing needs.
- Detect those forms that could be eliminated or consolidated.
- Single out those forms that should be standardized for agency-wide use.
- Identify forms that should be analyzed and redesigned.
- Generate studies of forms in their relation to their specific systems and procedures.

An important benefit of maintaining a functional control file is the detection and elimination of unauthorized forms (often called "bootleg forms"). These are forms that have been designed or reproduced outside of the established forms management program. By knowing which forms are approved for use within the agency, personnel can eliminate those forms which should not be used. The functional classification file can be difficult to develop, but it is an excellent type of control file to use in analyzing the agency's forms and their use.

Self Assessment Questions

6. Desktop publishing is the use of the computer and .....to create visual displays of ideas and information.

(a) hardware (b) software
(c) operating system (d) None of these.

7. A memory form is a record of history data that remains in a file, is used for reference and serves as control on details.

(a) True (b) False

10.10 Summary

- Input design is the process of converting user-originated input to a computer-based format. In the system design phase, the expanded data flow diagram identifies logical data flows, data stores, sources and destinations.
- Computer output is the most important and direct sources of information to the user. Efficient, intelligible output design should improve the system's relationships with the user and help in decision making
- Graphic design is the process and art of combining text and graphics and communicating an effective message in the design of logos, graphics, brochures, newsletters, posters, signs, and any other type of visual communication.
- Desktop publishing software allows the user to rearrange text and graphics on screen, change typefaces as easily as changing shoes, and resize graphics on the fly, before finally committing a design to paper

- Automated forms are the infrastructure use to submit the information or data. These are basically online forms such as reservation forms, online forms online shopping forms etc.

10.11 Keywords

Control File: It is provides a complete profile of each form from its creation to its current status.
Desktop Publishing: It is the process of using the computer and specific types of software to combine text and graphics to produce documents such as newsletters, brochures, books, etc.
Graphic Design: It is the process and art of combining text and graphics and communicating an effective message in the design of logos etc.
Systems Flowchart: It is referring to specifies master as (data base), transaction files, and computer programs.
Video display terminal: It is a computer terminal having a video display that uses a cathode-ray tube.

10.12 Review Questions

1. What is the input design?
2. Describe the output design?
3. Difference ate between input and form design.
4. Explain the input media and device.
5. What do you understand by online data entry?
6. Explain the graphics.
7. Describe the desktop publishing.
8. What do you understand by form design? With example.
9. What is the forms control?
10. Describe the layout considerations.

Answers for Self Assessment Questions

1. (a) 2 (b) 3 (a) 4 (c) 5 (d) 6 (b) 7 (a)

10.13 Further Studies

1. System Analysis and Design, Fifth Edition By: Alan Dennis; Barbara Haley Wixom; Roberta M. Roth
2. http://my.safaribooksonline.com/book/software-engineering-and-development/9781118057629/chapter-9-user-interface-design/navpoint-119

11

Hardware/Software Selection and the Computer Contract

STRUCTURE

11.0 Objectives

11.1 Introduction

11.2 Supplier and Types

11.3 Software Industry

11.4 Role of Consultant

11.5 Post Installation Review

11.6 Hardware and Software Selection

11.7 Ownership

11.8 Financial Consideration in Selection

11.9 Used Computer

11.10 Computer Contract

11.11 Art of Negotiation

11.12 Responsibilities and Remedies

11.13 Delivery and Acceptance of Hardware and Software

11.14 Warranties

11.15 Guarantee for Reliability

11.16 Summary

11.17 Keywords

11.18 Review Questions

11.19 Further Studies

11.0 Objectives

After studying this chapter, you will be able to:

- Explain the supplier and types
- Discuss the software industry
- Explain the role of consultant

- Understand the post installation review
- Discuss about the hardware and software selection
- Define ownership
- Understand the financial consideration in selection
- Explain about used computer
- Define computer contract

11.1 Introduction

In the computer software stream learn the basics of operating system structures, memory management, compilers, middleware, etc. Computers today are designed in conjunction with compiler technology and almost all make use of an operating system - this includes laptops, cell phones, and PDAs. Students will also study the basics of data structures, programming languages, databases, security, and software engineering.

Computer hardware stream will learn the basics of digital design at the gate and system/architectural level. Most people will spend their entire life no more than one meter away from some type of digital system (e.g. laptop, cell phone, PDA, iPod, GPS, auto, controllers, etc.) Digital hardware surrounds us all and affords many interesting careers. Students in this stream will study computer hardware, computer architecture, and digital systems design.

Hardware Selection Criteria

- Computer hardware must support the software available currently and that which will be purchased in the next year.
- Hardware must have compatibility and connectivity to existing networks or that which is planned for immediate purchase.
- Hardware must be upgradeable and expandable to meet the needs of the next several years.
- Hardware will be purchased with an appropriate length of warranty.
- Hardware maintenance shall be performed by a local vendor or in-house personnel.
- Whenever feasible, hardware standards will be maintained so that like brands and configurations are installed together to simplify installation and support.
- Attempts will be made to stay abreast of innovative mechanisms for extending the useful life of installed equipment.

Software Selection Criteria

- Software must be compatible with current and near future hardware.
- Software is standardized throughout the district whenever possible to improve purchasing power, simplify training, and facilitate support.
- Software must comply with current standards set by the technology committee or be approved by the Technology Director.
- Curricular software must support and enhance the local curriculum and school goals.

11.2 Supplier and Types

To get significant benefits from IT, you need to be able to trust your IT systems. This means having confidence in the company supplying, managing and maintaining them. Choosing the right IT supplier is therefore an essential part of selecting an effective IT solution.

11.2.1 The Different Types of IT Suppliers

Though most IT suppliers can provide a comprehensive range of hardware, software and services, there are some differences between them. For example:

- Resellers act as agents for hardware manufacturers, but can also offer software and a wide range of IT services and support
- System integrators select the appropriate hardware and software for your specific needs and deliver an integrated, working system
- Specialist suppliers have expertise in one specific area, such as customer-relationship management systems
- Consultants may just provide consultancy, with no hardware, software or maintenance services
- Retail, Mail order and Online suppliers may be appropriate if you are confident you do not need advice and simply need someone to supply the IT system.

Did You Know?

The software industry is the largest and most booming industry in the world. The total amount of revenues earned by software companies was highest.

11.3 Software Industry

Software industry encompasses all the activities and businesses involved with development, maintenance and distribution of computer software. Software industry started its operation. In this modern era of technology, software industry can be regarded as the most booming industry in the world. Software industry also covers the activities like software servicing, training and consultancy.

Software industry primarily concerned with the development of two types of software. One is proprietary software, which are owned by a single organization or individual and the other is open source software, which are written to use, distribute, recode and decode free of cost. The software industry business is principally based on proprietary software. Proprietary software needs to deal with the activities of licensing and security and these require lots of money to invest, whereas earning through open source software comes specifically from selling of services, or through training activities.

11.4 Role of Consultant

While there are negative stereotypes attached to the consulting profession, it is to be noted that consultants add value to their client's businesses. This is achieved through various avenues.

11.4.1 The Consultant's Contribution

Consulting assignments, whatever their nature, invariably begin with a problem! The consultant's client realizes that there is a need, that they lack the specific skills or staff to address in house.

That is where the business consultants job begins, and is an indication of what their contribution should be – skills, experience and knowledge.

The consultant's contribution therefore, can be defined as filling specific skills needs within an organization.

11.4.2 Why Are Consultants Necessary?

As mentioned, often, organizations lack full time employees with sufficient skills and experience to tackle complex problems. On the other hand, a situation arises where an objective, third party, unbiased opinion or input is required on a project. Or perhaps, an organization in the midst of a corporate reshuffle is in need of direction, in order to streamline processes, devise systems, or reallocate staff. Often, these types of situation would call for the use of a consultant who specializes in a specific field. Consultants bring a much needed objectivity, knowledge or innovative approach to organizations, augmenting and supplementing the teams within that organization.

11.4.3 Proven Results

Consultants, in almost every field, can produce documented results of how, when and why consulting has assisted their clients, or industry clients on the whole. This not only lends the consulting profession credibility, but also creates a knowledge base from which to work documents results and return on investment, and, overall, increases trust. Having shown proven results in a particular field, consultants are more likely to be engaged again to address similar problems.

11.4.4 How Does This Influence You, The Consultant?

At the end of the day, there will always be skeptics, who denounce the need for consultants.
By conducting your consulting business with ethics in mind, ensuring that you offer a valued contribution, and documenting results, you make it easier for potential clients to make the decision to hire you. Making sure that you bear in mind that your role, as the consultant, is to identify, analyze and solve your clients' problems, whatever they may be, and consistently doing so, will help to build your individual reputation.

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: How consultant helping in the selecting software/ Hardware? Show with example.

……

……

……

11.5 Post Installation Review

After a client has utilised a fully customised Sunrise application for a period of time, areas may come to light that require further customisation or training. The post-installation review (PIR) service provides Sunrise's clients with the opportunity to make such changes to the application, confident that they are undertaken in a controlled and consistent way. All changes are forwarded to our own client service desk (CSD) in order to provide qualified post- sales support.

11.5.1 Basic Concept

The PIR is normally scheduled to take place between one and four weeks after the system has been installed and customised. It is designed to ensure that the client's system is configured in accordance with their processes, which may have changed since the initial implementation.

If there are any minor enhancements that can be made to increase the performance of the system or improve the client's workflow, they should be completed during the PIR. Any enhancements that may require extra time can be scheduled and quoted for as a follow up.

Caution

The document produced during the workflow and process planning stage should be used as a guide against which the system will be tested.

Self Assessment Questions

1. The next stage is to research the potential solutions.....

(a). suppliers (b). hardware

(c). software (d). None of these.

2. The procedures for obtaining any further information from you should the suppliers have.....

(a). queries (b). system analysis

(c). design (d). None of these.

3. This process should usually include a demonstration of their proposed offering by each of the.....

(a). selling (b). buying (c). suppliers (d). None of these.

4. .....industry encompasses all the activities and businesses involved with development, maintenance and distribution of computer software

(a). Software (b). Hardware

(c). Both (a) and (b) (d). None of these.

11.6 Hardware and Software Selection

There is a bewildering array of IT hardware and services available to businesses today. But lines of business managers typically do not have the background, the time, or the inclination to educate themselves on all the features, pros and cons of alternative solutions to their business problems. In fact, too often, the way that new technology comes into an organization is like this:

- Somebody goes to a trade show
- A vendor gloms onto them and, of course, has the answer to all their problems
- The vendor pitches to an ad hoc procurement team, which vows to research alternatives and perhaps even issue a Request For Proposal (RFP)
- Due to the press of business, the process is short-circuited and the decision comes down to “Can we afford what the vendor is selling?” rather than “Is this the right solution of the many alternatives we have researched?”
- The purchase is made and never evaluated to see if it a) solved the problem and b) delivered true ROI

11.6.1 Hardware Selection Criteria

- Hardware must support current software as well as software planned for procurement over the next planning interval.
- Hardware must be compatible with existing or planned networks

- Hardware must be upgradeable and expandable to meet the needs of the next planning interval
- Hardware warranties must be of an appropriate length
- Hardware maintenance must be performed by.
- Whenever feasible, hardware standards will dictate procurement of like brands and configurations to simplify installation and support
- Routine assessments of installed infrastructure will feed an upgrade/replace decision process

Software selection is a critical aspect of system development. The search starts with the software, followed by the hardware. There are two ways of acquiring software: custom – made or "off – the – shelf" packages. Today's trend is toward purchasing packages, which represent roughly 10% of what it costs to develop the same in house. In addition to reduced cost, there are other advantages:
1. A good package can get the system running in a matter of days rather than the weeks or months required for "home-grown" packages.
2. MIS personnel are released for other projects.
3. Packages are generally reliable and perform according to stated documentation.
4. Minimum risks are usually associated with large – scale systems and programming efforts.
5. Delays in completing software projects in house often occur because programmers quit in midstream.
6. It is difficult to predict the cost of "home-grown" software.
7. The user has a change of seeing how well the package performs before purchasing it.

There are drawbacks, however, to software packages:
1. The package may not meet user requirements adequately.
2. Extensive modification of a package usually results in loss of the vendor's support.
3. The methodology for package evaluation and selection is often poorly defined. The result is a haphazard review based on a faulty process or questionable selection criteria.
4. For first – time software package users, the overall expectation from a package is often unclear and ill defined.
It can be seen, then, that the quality of a software package cannot be determined by price alone. A systematic review is crucial.

11.6.2 Criteria for Software Selection

Prior to selecting the software the project team must set up criteria for selection. Selection criteria fall into the categories described here.

Reliability

It is the probability that the software will execute for a specified time period without a failure, weighted by the cost to the user of each failure encountered. It relates to the ease of recovery and ability to give consistent results. Reliability is particularly important to the professional user. For example, a pharmacist relies on past files on patients when filling prescriptions. Information accuracy is crucial.

Functionality

It is a definition of the facilities, performance, and other factors that the user requires in the finished product. All such information comes from the user. The following are key questions to consider:
1. Do the input transactions, files, and reports contain the necessary data elements?
2. Are all the necessary computations and processing performed according to specifications?

Capacity

Capacity refers to the capability of the software package to handle the user's requirements for size of files, number of data elements, volume of transactions and reports and number of occurrences of data elements. All limitations should be checked.

Flexibility

It is a measure of the effort required to modify an operational program. One feature of flexibility is adaptability, which is a measure of the ease of extending the product.

Usability

This criterion refers to the effort required to operate, prepare the input, and interpret the output of a program. Additional points to be considered are portability and understand ability. Portability refers to the ability of the software to be used on different hardware and operating systems. Understand ability means that the purpose of the product is clear to the evaluator and that the package is clearly and simply written, is free of jargon, and contains sufficient references to readily available documents so that the reader can comprehend advance contents.

Security

It is a measure of the likelihood that a system's user can accidentally or intentionally access or destroy unauthorized data. A key question is how well can one control access of software or data file? Control provides system integrity.

Performance

It is a measure of the capacity of the software package to do what it is expected to do. This criterion focuses on throughput, or how effectively a package performs under peak loads. Each package should be evaluated for acceptance on the user's system.

The language in which a package is written and the operating system are additional performance considerations. If we plan to modify or extend a package, it is easier if it is written in a language that is commonly known to programmers. Likewise, if the package run only under a disk operating system and the installation is under a full operating system, then either the package will have to be upgraded to the larger operating system or the system downgraded to handle the package as is. In either case, the change could be costly and counterproductive.

Serviceability

This criterion focuses on documentation and vendor support. Complete documentation is critical for software enhancement. It includes a narrative description of the system, system logic and logic instructions. Vendor support assures the user adequate technical support for software installation, enhancements, and maintenance, the user should determine how much on – site technical assistance is provided by the vendor, especially during the first few weeks after the installation. The user expects on – site training and support as part of most commercial packages. It is vital to inquire about the amount of training provided. The user may require training at several levels clerical, operations, programming, and management.

Ownership

Who owns the software once it is "sold" to the user? Most of the standard license agreement forms essentially lease the software to the user for an indefinite time. The user does not "own" it, which means that the source code is inaccessible for modification, except by the vendor. Many users enter into an escrow arrangement whereby the vendor deposits code to the user if the vendor goes out of business or is unable to perform the services specified in the license.

In acquiring software, several questions should be asked:

1. What rights to the software is the user buying?
2. Can the user sell or modify the software?

3. If the vendor is modifying the package especially for the user, can the vendor sell it to other within the same industry the user is in?
4. What restrictions are there to copying the software or documentation?

Did You Know?
A "personal, portable information manipulator" was imagined by Alan Kay at Xerox PARC in 1968, and described in his 1972 paper as the "Dynabook".

11.7 Ownership

The primary objective of policies concerning ownership of intellectual property must be to preserve, protect and foster the open and creative expression and exchange of information, ideas and research results. This is not only the responsibility of a public educational institution; it is the basic premise on which a university must exist.
To encourage the production and distribution of creative works, our legal system has established property rights for inventions and writings through patents and copy-rights. Ownership of these properties is reserved to the creator for a limited time during which the creator may sell, lease or distribute the product of his/her efforts. The purpose of these limited rights is to establish an incentive to make the fruits of individual creativity available to society at large.
Although governmental agencies and most businesses assert ownership of the intellectual property created by their employees, the University of Wisconsin System has not typically done so. Such ownership provides the opportunity to withhold as well as disseminate. Ownership of intellectual property by the University would provide a general right and perhaps a responsibility to censor; this runs counter to the University mission to engage in open innovation and inquiry. Individual ownership is also more consistent with the self-directed nature of much university work and of Wisconsin traditions in particular.
The intellectual property in original works of authorship such as books, articles and similar works is protected by copyright, which is held to exist at the point the material is created. As with patents, ownership at the University is normally vested in the creator. For example, faculty have ordinarily owned rights to books created in the course of their scholarly activities, regardless of the funding mix supporting their work and the extent to which University resources have been used in their preparation. Copyright law has been amended recently to include computer software. One provision called "work for hire" states that when an employee is specifically directed to produce a software product as a condition of employment, ownership rights including copyright rest with the employer.

11.8 Financial Consideration in Selection

The acquisition of and payment for a computer system are usually handled through one of three common methods: rental, lease, or purchase. Determining which option is appropriate depends on the characteristics and plans of the organization at the time the acquisition is made. No one option is always better than the other. (Table 11.1 summarizes the features of each method of acquisition.)

11.8.1 Rental

Computer rental is for the short – term use of a system, generally form 1 to 12 months. Each month a payment is made for the use of the equipment. Both the user and supplier have the option of cancelling the rental with advance notice, usually 30 or 60 days ahead of the termination date.

Because the commitment is short-term, the renter has a great deal of flexibility. The decision to purchase a system can be delayed until financing is adequate, until a new generation of equipment is available, or until such time as the organization wishes, for whatever reason. Flexibility can be particularly important when an organization is experiencing planned rapid growth and will outgrow a specific system in a brief period, when important reorganizations of divisions and departments that will affect computing resources are in progress, or when the enterprise is in a period of dynamic change.

11.8.2 Lease

As lease is a commitment to use a system for a specific time, generally from three to seven years. Payments are predetermined and do not change throughout the course of the lease. Depending on the terms of the lease, payments are monthly, quarterly, semi-annual, or annual and include the cost of equipment service and maintenance. At the end of the lease period the lessor generally does not own the equipment. (If that is not the case, and the equipment becomes the property of the lessor, the Internal Revenue Service considers the agreement a conditional sale and the entire transaction must then be treated as a purchase.)

11.8.3 Purchase

The ownership of computers through outright purchase is the most common method of computer acquisition and is increasing in popularity as lease costs rise. Over time, the purchase option frequently costs the least, especially in light of the tax advantages that can some – times be gained. Under purchase, the organization takes title to the equipment. Of course, the money for the purchase must be taken from operating funds or borrowed. And, in a sense the organization is locked in to the system it purchases, since changing to a different computer system is more difficult; either the system must be sold or arrangements must be negotiated to trade it in on a different computer.

The organization must acquire its own maintenance services (for parts and labour), usually from the manufacturer, and pay the monthly charges, which fluctuate from year to year. In addition, if the equipment was financed, payment on the loan must be made periodically. The cash outflow still may be lower than with renting or leasing, depending on the terms arranged by the purchaser. In return for the outgoing cash, purchase offers specific tax advantages:

1. The monthly maintenance charges are deductible as a business expense.
2. Interest on any loan to finance the purchase is deductible as a business expense.
3. The cost of the equipment can be depreciated over time; this also lowers the taxable income and therefore the income taxes paid.
4. Local, state, and federal taxes paid on the purchase may be deductible from income taxes.

The purchase option indicates the use of depreciation to reduce taxes. In a sense then, depreciation deductions on income tax reduce the cost of the computer to the organization. Normally, this benefit is not possible under lease agreements and it is never feasible for short – term rentals. Of course, the tax benefits described apply only to firms that operate for profit. Non profit firms that do not pay income taxes thus do not receive tax benefits from computer purchase.

Table 11.1 Comparison of Computer Systems Financing Options

Method of acquisition	Advantages	Disadvantages
Rental	Short – term commitment. High level of flexibility. Does not require cash up front.	Most expensive option. Little control of equipment change. Not all vendors will rent.
Lease	Predetermined payments for fixed period. Does not require cash up front. Usually better service from vendor than under rental. Little risk of obsolescence. Less expensive than rental	More expensive than purchase. May have limitations on hours of equipment use.
Purchase	Least cost in long run. Distinct tax advantages if a profit – making firm. A business investment. Full control over equipment use.	Risk of obsolescence. Permanent commitment. Full responsibility for all problems. Greater early cash requirements than other options.

11.9 Used Computer

Computers are definitely not the cheapest products on the market. One option that many shoppers often forget to take into consideration is buying refurbished products. There are many benefits to buying refurbished computers. Below are some points to consider when deciding whether or not to purchase refurbished machines?

11.9.1 Benefits of used Computer

Aesthetics

Many computer manufacturers, such as HP, Dell and Apple, use a very stringent refurbishment process. They want consumers to continue to order from them, even refurbished products. So, they are not going to send computers with case scratches or any major physical defects. Most times, refurbished desktops and laptops are actually given a brand new casing prior to sale. If a company receives a computer for refurbishment that shows only light wear, it may choose to replace only certain parts with new ones. Expect refurbished computers to look 99% just like brand new.

Functionality

Most of the time, refurbished computers are just customer returns. In some cases, there could be nothing wrong with the computers at all. The customer just did not like the products, so they returned them within the warranty period. In cases like this, the computer manufacturers will sell these computers as refurbished. They cannot sell them as new because they have been opened at used slightly for a short period of time.

The other instance of refurbished computers is defective parts. This could range from anything to a RAM module to the motherboard. When the manufacturers refurbish these computers, they normally replace the faulty part with a brand new one. If the issue is with installation, they will properly reinstall the part.

Manufacturers also test for like-new functionality before selling. Not only do they test the replacement parts, but everything else as well. Expect refurbished computers to function just as well as new ones.

Warranty

Large computer companies will offer the exact same warranty on refurbished products as they will on new products. So as far as the warranty goes, you really are losing nothing by purchasing refurbished.

If a warranty is important to you, make sure that you buy refurbished from the manufacturer rather than some independent reseller that refurbishes computers. This will give you the full warranty that you want.

Price

Computer manufacturers cannot sell any product as new unless it is truly 100% new. This means that customer returned computers, extra computers and previous generation computers, must be sold under a different name. So, manufacturers sell all these types of products with a refurbished label.

They also know that consumers will not pay near as much for refurbished computer as they would brand new ones. In turn, refurbished computers are often offered at rock bottom prices--often times up to 20% off the new price.

Here is a good example of when to buy refurbished: Apple originally sold a Mac Book for 80000, new. The company also offered the same laptop for $1300 refurbished. A month later, Apple gets newer processors in and overhauls all of their notebooks. The original laptop is now offered at 45000 refurbished.

Accessories

One question you may have with buying refurbished is: Do you receive everything that comes with a new computer? Yes. Computer manufacturers will provide you with all original documentation, user manuals, power cords and other accessories. The one downfall is that your item will usually come in a plain brown packaging rather than the original box.

11.10 Computer Contract

The covers the full range of contracts for buying, selling, leasing and licensing hardware, software and services, with legal analysis, sample clauses, and complete contracts to help you handle contracts for computer users and vendors, marketing arrangements, software development, maintenance, licensing arrangements and telecommunications contracts.

Unique double-column format provides invaluable clause-by-clause examination of a wide range of documents, with background material, and the full text of the agreement or clause under discussion.

Primary source volume includes a glossary of computer terminology, pertinent statutes and regulatory materials, and selected cases.

Unreported and otherwise unavailable computer contract cases. To points out potential benefits, pitfalls, problems in using various agreements and clauses which gives you alternative language for custom - tailoring documents.

Includes proprietary and financial considerations, important federal and state tax aspects, protection of intellectual property and security rights.

11.11 Art of Negotiation

Whether it is time to negotiate for a mortgage, a software or even hardware, preparation is the key to victory. When you know what you want and how you intend to get it, you are much more likely to come out of the appointment smiling.

“Start out with an ideal and end up with a deal.” Karl Albrecht, co-founder of Aldi.

Let's go through the different phases of a deal and prep for each one. Skipping one of these will most likely hurt your chances so do not slack off on any of them.

11.11.1 before the Appointment

Know what you want. Sit down and work out exactly:

- What you need and what want. Secure the needs first and negotiate with the wants (aka the extra stuff you can do without)
- How much you can afford
- How far you are prepared to compromise on both (afford vs. spend)

If you are looking at a long-term deal, look beyond the immediate price and make sure you can afford the payments without stretching your budget to its breaking-point.

11.12 Responsibilities and Remedies

The following pint used for responsibility and remedies for Software/ hardware selection of computer:

- Get to know about the Roles, Tasks, Duties and Responsibilities of Computer Support Specialist.
- Provide in-depth support to computer programmers.
- Ensure proper configuration and functioning of desktop computers, servers and networks.
- Provide support in troubleshooting software and hardware problems.
- Identify and trouble network problems.
- Ensure correct analysis of problems.
- Provide technical support in the use of business applications.
- Provide technical support and guidance in third party products.
- Provide remedies, solutions and recommendations in a timely manner.
- Support and assist in installation and repairs of laptops, desktop workstations, networking and other related systems.
- Develop and implement preventive maintenance measures.
- Assist and support in writing security code to all the systems.

11.13 Delivery and Acceptance of Hardware and Software

11.13.1 Delivery

A "delivery model" refers to the approach taken to "deliver" enterprise software. It is usually used when referring to a software application. There are three primary delivery models for enterprise software:

Licensed model: Licensed model in which a customer purchases software and owns it.

Software as a Service (SaaS): Saas model in which the customer "rents" the software and the supplier runs and updates the software, providing the software as a service offering rather than as a product.One of the key tenets of SaaS applications is a concept called "multi-tenancy". In SaaS deployments–all customers simultaneously access the same single instance of the application in the remote data centre. This means the vendor can amortize a world-class infrastructure across multiple customers while reducing costs. What is more, well-designed SaaS applications allow for instant provisioning and upward/downward scalability, parameter-driven customization and configuration and regular upgrades (instead of the daunting customization-breaking upgrades required with on-premises or hosted software implementations). So which model is best for your organization.

Hosted model: Hosted model is called in which a customer purchases or leases the software, and a third party runs the software for that customer. The hosted model varies from the SaaS model in that hosted customers still own the software and it is the third-party company that provides the service. In

the SaaS model, the software itself is designed for use by many companies at once (called "multitenant") and the service is provided by the software vendor.

On-Premises Software Deployment: Before the widespread availability, affordability and adoption of networks particularly the Internet on premises software deployments were virtually the only choice for businesses Some examples include large-scale enterprise resource planning (ERP) and customer relationship management (CRM) systems as well as single-user programs such as QuickBooks or Microsoft Office. As its name implies, an on-premises implementation is software that's installed and operated on computer(s) located on the premises of the software licensee, rather than at a remote facility. This model largely defined and drove the first generation of business computing.
However, on-premises software is limited in its ability to support remote access to computing services. Customizations if allowed– can be difficult and expensive. Software vendors also make significant investments in legacy code that tend to work poorly in off-premises configurations.

11.13.2 Acceptance
In today's marketplace, removing all potential objections to accepting hardware solutions is critical. Designers are under increasing pressure to develop products faster and cheaper. Potential customers are increasingly demanding complete solutions to their problems. Hardware alone is rarely the answer.

In order to provide a more complete solution to customers, significant software support including software drivers, network protocol support, configuration, management and control applications, reference software, or development tools are needed to improve the ability of a customer to successfully move beyond just a device to a real solution. In many situations, software can be the differentiating factor in making a component choice.

Providing a complete solution addressing these requirements allows:
Rapid acceptance of solutions: By providing all the software components necessary for completely supporting a hardware device, the time to adopt the hardware is shortened, requiring less investment by potential customers to accept the solution and reducing time to market.
Entry to new markets: Software support for a variety of network and operating environments insures that no barriers to particular markets exist.
Positioning as a technology leader: The hardware provider can position itself as a technology leader by demonstrating commitment to providing complete support (including high quality software solutions for a variety of environments and addressing as many customer concerns as possible).
Unfortunately, providing software for complete solutions is often problematic. In most cases, the supporting software must be made available simultaneously with the hardware in order to create a fully saleable solution for potential customers. Even if sufficient resources are available to commit to software development, the expertise is difficult to acquire and often distracts from the core capabilities and mission of hardware providers.
Even in situations where the software can be developed or is readily available, customers require additional assistance and services to port, integrate, and otherwise adapt this software to their devices, environments, and markets. In today's difficult business environment, hardware providers can offer this software and associated support to only their very highest volume customers. While these high volume customers justify the additional effort due to their purchasing power, there is a significant gap in support for mid-tier or smaller customers.

Self Assessment Questions

5. The software industry is the largest and most booming.....in the world.

(a). system analysis (b). industry

(c). organization (d). None of these.

6. Software industry primarily concerned with the development of two types of software.

(a). True (b). False.

7. India's software exporting industry is one of the world's successful information technology industries.

(a). True (b). False.

11.14 Warranties

An implied warranty is one that arises from the nature of the transaction, and the inherent understanding by the buyer, rather than from the express representations of the seller.

11.14.1 Covered (and is free)

- All individual hardware parts are covered for a minimum of one year, including the cost of removing the faulty part at our base and reconnecting the new one. Parts need to be demonstrably faulty, either through a manufacturer's software test routine, or other repeatable test, which takes less than 30mins. Less frequent faults (eg occurring once or twice in day) would be classified as reliability issues.
- Where individual parts have a manufacturer's warranty greater than a year (eg SCSI disks), then the one year minimum is extended accordingly. Where the terms of the manufacturer's warranty exceeds our own (eg on-site swap-out of monitors), you will be entitled to this service provided you have complied with the manufacturer's requirements (eg registering your product with them).
- At our discretion and where applicable and mutually convenient, repair work may be carried out on site. The inclusion of on-site installation at the time of initial purchase does not constitute an entitlement to subsequent on-site support unless specifically itemised on the invoice.
- Faulty or damaged items should be notified within seven days of receipt of goods, and will be dealt with in accordance with the returns procedure laid out by the manufacturer. All goods returned must be in the manufacturers' original packaging complete with all ancillary items. The company reserves the right to refuse returns for items which have become obsolete or were part of a special order, regardless of the time the return is requested or the condition of the goods.

11.14.2 Not covered (or is chargeable)

- Any work or time spent on diagnostics which does not result in a fault being found within the warranty terms (above). We do not guarantee the reliability either of hardware or software, and diagnostic work on reliability issues is fully chargeable.
- Customers are responsible for the cost of inbound shipping for parts or systems intended for repair, and the outbound shipping cost should no fault be found. Customers should report the nature of the problem and we will advise on what parts to return. We will not be responsible for the return of un-requested parts.
- Damage caused in return transit, particularly through the use of inappropriate or inadequate packaging. It is the customer's responsibility to provide insurance, if they require it, for returns to base.

- Unless specifically covered by “advance replacement warranty”, parts will usually need to be returned to suppliers/manufacturers for repair/replacement. This process can take anything from days to weeks, and we can make no guarantees about such timescale.
- We do not accept responsibility for the contents of any re-writeable storage media (ie software and data on disk), any more than suppliers of other media like video tape or 35mm film. Whilst we can undertake “data recovery/transfer” work, it is not within the scope of warranty repairs.
- We provide all original disks for software sold, and customers are responsible for the software configuration of their own system. In particular, customers are responsible for registering their products and obtaining updates and fixes made available over the internet. The inclusion of software installation and set-up at the time of purchase does not constitute an entitlement to the restoration of such set-up following a hardware replacement.
- Software is supplied in accordance with the publisher's licence agreement, and we do not offer any warranties beyond the scope of that user licence, which the purchaser is deemed to have accepted.
- Any consequential losses however caused, through the use, or inability to use, any hardware, software or systems that we may have supplied or worked on. We do not guarantee continuity of use should any part of the hardware develop any kind of fault. Should you wish to bypass the normal returns/repair/replacement procedures, due to pressure of ongoing work and the interruption that would occur, you are at liberty to do so, but must take responsibility for the cost and consequences We will be happy to assist in whatever way we can and liase with whoever you choose to work on the system.

11.15 Guarantee for Reliability

As we have seen, there are ranges of ways in which failures can happen within a distributed system. In this section we will translate those into concrete examples of how failures could affect and applications deployed on it. In the section on Recommendations we shall cover ways in which you can configure to better tolerate these faults, or how you should approach your application development.

The failure of some of them may go unnoticed to some or all of your applications depending upon when the failure occurs. For example, if the Registry Service crashes after your consumer have successfully obtained all necessary EPR information for the services it needs in order to function, then it will have no adverse affect on your application. However, if it fails before this point, your application will not be able to make forward progress. Therefore, in any determination of reliability guarantees it is necessary to consider when failures occur as well as the types of those failures.

It is never possible to guarantee 100% reliability and fault tolerance. The laws of physics (namely thermodynamics and the always increasing nature of entropy) mean that hardware degrades and human error is inevitable. All we can ever do is offer a probabilistic approach: with a high degree of probability, a system will tolerate failures and ensure data consistency/make forward progress. Furthermore, proving fault-tolerance techniques such as transactions or replication comes at a price: performance. This trade-off between performance and fault-tolerance is best achieved with application knowledge: any attempts at opaquely imposing a specific approach will inevitably lead to poorer performance in situations where it is simply not necessary.

Exercise: Check Your Progress 2

Case Study-Computer Software History

There is a high likelihood that every individual from developed countries has to deal with some kind of software. Computer software is a rather broad term that is used to encompass the different types of software based on which a computer actually works. Yet, even though software engineering has become a large and profitable industry just few people known how it all started. Here one can find out more about the history of software and how it all came together.

It all started with Alan Turing who proposed a theory about software in 1935 in his essay called Computable numbers with an application to the decision problem. The term software was however not used in the written literature until 23 years later when John W Tukey used it in print. The term is commonly used to describe application software but in computer engineering this word encompasses all the information that is processed by the computer system, programs in general and data. It is believed that the history of software as we know it began in 1946 when the first software bug was developed. Software has become in time cheaper and faster as did hardware. At first some elements of the computer were considered to be software but then they have joined the ranks of hardware. Software has constantly increased in popularity as the importance of computers has increased. Moreover, individuals started to want more from computers and this caused a need to further progress and development of the software that was being produced. For a long time however, software was bundled with the hardware by original equipment manufacturers. This meant that a new computer would have not come with pre-installed software but that the software had to be installed by the specialists working for the original equipment manufacturers. Nowadays things are much simpler since people can perform software download from the internet whenever they need a new program.

Questions

1. Who was the founder of computer software?
2. When software industries began?

11.16 Summary

- Software is supplied in accordance with the publisher's license agreement, and we do not offer any warranties beyond the scope of that user license, which the purchaser is deemed to have accepted.
- IT suppliers range from small local outfits to global organizations. Even the largest suppliers can provide systems, services and consultancy to small businesses.
- Client/Server computing is a technique in which application is shared between a desktop "client" and one or more network attached "servers".
- The Post Installation Review (PIR) is also an ideal opportunity for the client to identify any additional requirements such as training or report writing.
- The primary objective of policies concerning ownership of intellectual property must be to preserve, protect and foster the open and creative expression and exchange of information, ideas and research results.

11.17 Keywords

Capacity: Capacity refers to the capability of the software package to handle the user's requirements for size of files, number of data elements, volume of transactions and reports and number of occurrences of data elements.

Delivery Model: A "delivery model" refers to the approach taken to "deliver" enterprise software. It is usually used when referring to a software application.

Lease: As lease is a commitment to use a system for a specific time, generally from three to seven years.

Proprietary: A proprietary design or technique is one that is owned by a company. It also implies that the company has not divulged specifications that would allow other companies to duplicate the product.

Reliability: It is the probability that the software will execute for a specified time period without a failure. It is particularly important to the professional user.

System integrators: System integrators select the appropriate hardware and software for your specific needs and deliver an integrated, working system.

Answers of Self Assessment Question

1. (a) 2. (a) 3. (c) 4. (a) 5. (b) 6. (a) 7. (a)

11.18 Review Questions

1. What are the types of supplier?
2. Discuss the software industry.
3. What is the post installation review?
4. What do you understand by hardware and software selection?
5. Discuss the criteria of hardware and software selection.
6. Define the ownership.
7. What is the financial consideration in selection?
8. Discuss the computer contract.
9. Explain the art of negotiation and guarantee for reliability.
10. Discuss the acceptance of hardware and software.

11.19 Further Studies

1. Systems Analysis and Design By Goyal Arunesh

12

System Security and Disaster Recovery Planning

STRUCTURE

12.0 Objectives

12.1 Introduction

12.2 System Security

12.3 System Security is an Important Concern

12.4 Threats to System Security

12.5 System Integrity

12.6 Risk Analysis

12.7 Control Measures

12.8 Recovery/Restart Requirements

12.9 The Recovery Aid

12.10 System Failures and Recovery

12.11 Disaster/Recovery Planning

12.12 Plans

12.13 Team

12.14 Planning Tasks

12.15 Ethics in System Development

12.16 Ethics Codes and Standards of Behaviour

12.17 Summary

12.18 Keywords

12.19 Review Questions

12.20 Further Studies

12.0 Objectives

After studying this chapter, you will able to:

- Understand system security
- Define that why system security is an important concern
- Discuss the threats to system security

- Explain personal computer and system integrity
- Define risk analysis

12.1 Introduction

Disaster recovery closely parallels computer security operations in several functional areas. Threat evaluation, risk assessment, mitigation, and service priorities are but only a few of the items that are on the event horizon. Traditional disaster recovery procedure looks at the varying aspects of planning and implementation from an administrative perspective, focusing primarily on physical infrastructure, backup and restoration procedure, staffing, logistical operations, and connectivity. Attention to computer security must be given at all levels of recovery to ensure the integrity of the system(s).

12.2 System Security

The meaning of the term computer security has evolved in recent years. Before the problem of data security became widely publicized in the media, most people's idea of computer security focused on the physical machine. Traditionally, computer facilities have been physically protected for three reasons:

- To prevent theft of or damage to the hardware
- To prevent theft of or damage to the information
- To prevent disruption of service

Strict procedures for access to the machine room are used by most organizations, and these procedures are often an organization's only obvious computer security measures. Today, however, with pervasive remote terminal access, communications, and networking, physical measures rarely provide meaningful protection for either the information or the service; only the hardware is secure. Nonetheless, most computer facilities continue to protect their physical machine far better than they do their data, even when the value of the data is several times greater than the value of the hardware.



Figure 12.1: System Security.

In a generic sense, security is "freedom from risk or danger." In the context of computer science, security is the prevention of, or protection against,

- access to information by unauthorized recipients, and
- intentional but unauthorized destruction or alteration of that information

12.3 System Security is an Important Concern

Computer security is frequently associated with three core areas, which can be conveniently summarized by the acronym "CIA":

- ***Confidentiality*** Ensuring that information is not accessed by unauthorized persons
- ***Integrity*** Ensuring that information is not altered by unauthorized persons in a way that is not detectable by authorized users
- ***Authentication*** Ensuring that users are the persons they claim to be a strong security protocol addresses all three of these areas. Take, for example, Netscape's SSL (Secure Sockets Layer) protocol. It has enabled an explosion in ecommerce which is really about trust (or more precisely, about the lack of trust). The SSL overcomes the lack of trust between transacting parties by ensuring confidentiality through encryption, integrity through checksums, and authentication via server certificates.

Computer security is not restricted to these three broad concepts. Additional ideas that are often considered part of the taxonomy of computer security include:

- ***Access control*** Ensuring that users access only those resources and services that they are entitled to access and that qualified users are not denied access to services that they legitimately expect to receive
- ***Nonrepudiation*** Ensuring that the originators of messages cannot deny that they in fact sent the messages
- ***Availability*** Ensuring that a system is operational and functional at a given moment, usually provided through redundancy; loss of availability is often referred to as "denial-of-service"
- ***Privacy*** Ensuring that individuals maintain the right to control what information is collected about them, how it is used, who has used it, who maintains it, and what purpose it is used for

Did You Know?

Fred Cohen formally defined the term "computer virus" in 1983 when he performed academic experiments on a Digital Equipment Corporation VAX system.

12.4 Threats to System Security

Computer systems are vulnerable to many threats that can inflict various types of damage resulting in significant losses. This damage can range from errors harming database integrity to fires destroying entire computer centres. Losses can stem, for example, from the actions of supposedly trusted employees defrauding a system, from outside hackers, or from careless data entry clerks. Precision in estimating computer security-related losses is not possible because many losses are never discovered, and others are "swept under the carpet" to avoid unfavourable publicity.

The effects of various threats vary considerably: some affect the confidentiality or integrity of data while others affect the availability of a system.

12.4.1 Errors and Omissions

Errors and omissions are an important threat to data and system integrity. These errors are caused not only by data entry clerks processing hundreds of transactions per day, but also by all types of users who create and edit data. Many programs, especially those designed by users for personal computers, lack quality control measures. However, even the most sophisticated programs cannot detect all types of input errors or omissions. A sound awareness and training program can help an organization reduce the number and severity of errors and omissions.

Users, data entry clerks, system operators, and programmers frequently make errors that contribute directly or indirectly to security problems. In some cases, the error is the threat, such as a data entry error or a programming error that crashes a system. In other cases, the errors create vulnerabilities. Errors can occur during all phases of the systems life cycle.

12.4.2 Fraud and Theft

Computer systems can be exploited for both fraud and theft both by "automating" traditional methods of fraud and by using new methods. For example, individuals may use a computer to skim small amounts of money from a large number of financial accounts, assuming that small discrepancies may not be investigated. Financial systems are not the only ones at risk. Systems that control access to any resource are targets (e.g., time and attendance systems, inventory systems, school grading systems, and long-distance telephone systems). Computer fraud and theft can be committed by insiders or outsiders. Insiders (i.e., authorized users of a system) are responsible for the majority of fraud.

Since insiders have both access to and familiarity with the victim computer system (including what resources it controls and its flaws), authorized system users are in a better position to commit crimes. Insiders can be both general users (such as clerks) and technical staff members. An organization's former employees, with their knowledge of an organization's operations, may also pose a threat, particularly if their access is not terminated promptly.

12.4.3 Employee Sabotage

Employees are most familiar with their employer's computers and applications, including knowing what actions might cause the most damage, mischief, or sabotage. The downsizing of organizations in both the public and private sectors has created a group of individuals with organizational knowledge, who may retain potential system access (e.g., if system accounts are not deleted in a timely manner). The number of incidents of employee sabotage is believed to be much smaller than the instances of theft, but the cost of such incidents can be quite high.

***Common examples of computer-related employee sabotage include*:**

- destroying hardware or facilities,
- planting logic bombs that destroys
- Programs or data,
- entering data incorrectly,
- "crashing" systems,
- deleting data,
- Holding data hostage, and
- Changing data.

Did You Know?

One 1992 study of a particular Internet site (i.e., one computer system) found that hackers attempted to break in at least once every other day.

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: Prepare the list how to secure computer to threads.

……

……

……

12.5 System Integrity

System integrity begins with selecting and deploying the right hardware and software components to authenticate a user's identity—and help prevent others from assuming it. In doing so, it needs to offer efficient administrative functions to restrict access to administrator-level functions, and give administrators processes and controls to manage changes to the system. There are many individual components to system integrity, such as vulnerability assessment, antivirus, and anti-malware solutions. However, the ultimate goal from an access control standpoint is to prevent the installation and execution of malicious code—while protecting valuable data—from the outset.

12.5.1 System Integrity Issues

Essentially, system integrity seeks to ensure the security of system hardware, software, and data. Microsoft invests significant resources in providing prevention and mitigation against malware at both the platform and application level. For Microsoft® Windows Vista™, the next generation of the Microsoft Windows® operating system, Microsoft has identified the following issues that have an impact on System Integrity.

12.5.2 Detecting Malware

Malware is one of the most insidious threats that enterprises encounter today. Malware is a malicious type of software designed to install itself on a computer, and because it spoofs the administrator login, it operates at the administrator level without the user knowing it is there. There are many types of malware, but it is always designed with intent of doing damage. The most dangerous malware are designed to find and transmit confidential data, credit card information, user names and passwords, and more. Many third-party applications require administrator access to function properly, and this is one reason that malware can be so pernicious. It takes advantage of administrator settings, which often bypass security features and give the application or user access to a number of critical, system-level functions.

To stop malware from being installed, enterprises must:

- Control administrator-level access privileges
- Screen incoming files for malware and prevent their download/installation
- Scan and remove malware from computers

12.5.3 Preventing Malware

Malware has reached epidemic proportions in many organizations, and even the best firewalls cannot always filter it out. Many computers are running rogue code without even knowing it, leaving entire organizations at the mercy of code that can disable systems, expose confidential user information, and more.

To prevent malware from running, enterprise computers need the ability to authenticate code and assess applications at the system level before it runs. The Microsoft response to unauthorized code running on client computers includes:

- **Code Integrity** to validate the integrity of each binary image with per-page hashes as each page is loaded into the system. Images that fail are not loaded.
- **Security Restriction Policy** to help prevent damage to systems by identifying and then controlling software running on an enterprise domain.
- **Microsoft Authenticode** to digitally sign and validate application code.
- **Internet Explorer 7** which helps protect a user's system from malware attacks while he or she is browsing the internet

12.5.4 Protecting the Operating System

Hundreds of thousands of computers are lost or stolen every year, posing an enormous liability to organizations—and not just in terms of lost user productivity and physical assets. Intellectual property and trade secrets are also at risk.

Many organizations are justifiably concerned about confidential information being accessed by unauthorized users. Hacker programs can bypass the Windows XP default data security mechanisms with an offline attack that exposes core system keys. Even more problematic, a hacker can browse and copy the contents of a hard disk drive by removing it from its computer and attaching it to another.

Did You Know?

The first computer viruses were developed in the early 1980s. The first viruses found in the wild were Apple II viruses such as Elk Cloner, which was reported in 1981.

12.6 Risk Analysis

There may be some terminology and definition differences related to risk analysis, risk assessment and business impact analysis. Although several definitions are possible and can overlap, for purposes of this section, please consider the following definitions:

- A risk analysis involves identifying the most probable threats to an organization and analyzing the related vulnerabilities of the organization to these threats.
- A risk assessment involves evaluating existing physical and environmental security and controls, and assessing their adequacy relative to the potential threats of the organization.
- A business impact analysis involves identifying the critical business functions within the organization and determining the impact of not performing the business function beyond the maximum acceptable outage. Types of criteria that can be used to evaluate the impact include: customer service, internal operations, legal/statutory and financial.

12.6.1 Risk Analysis Process

Regardless of the prevention techniques employed, possible threats that could arise inside or outside the organization need to be assessed. Although the exact nature of potential disasters or their resulting consequences are difficult to determine, it is beneficial to perform a comprehensive risk assessment of all threats that can realistically occur to the organization. Regardless of the type of threat, the goals of business recovery planning are to ensure the safety of customers, employees and other personnel during and following a disaster.

The relative probability of a disaster occurring should be determined. Items to consider in determining the probability of a specific disaster should include, but not be limited to: geographic location, topography of the area, proximity to major sources of power, bodies of water and airports, degree of accessibility to facilities within the organization, history of local utility companies in providing uninterrupted services, history of the area's susceptibility to natural threats, proximity to major highways which transport hazardous waste and combustible products.

Potential exposures may be classified as natural, technical, or human threats. Examples include:
Natural Threats: internal flooding, external flooding, internal fire, external fire, seismic activity, high winds, snow and ice storms, volcanic eruption, tornado, hurricane, epidemic, tidal wave, typhoon.
Technical Threats: power failure/fluctuation, heating, ventilation or air conditioning failure, malfunction or failure of CPU, failure of system software, failure of application software, telecommunications failure, gas leaks, communications failure, nuclear fallout.
Human Threats: robbery, bomb threats, embezzlement, extortion, burglary, vandalism, terrorism, civil disorder, chemical spill, sabotage, explosion, war, biological contamination, radiation contamination, hazardous waste, vehicle crash, airport proximity, work stoppage (Internal/External), computer crime.

Self Assessment Questions

1. .....is a property of individuals.
(a) Security (b) Privacy (c) Safety (d) None of these

2. ..... is a property assigned to computer hardware and software systems.
(a) Security (b) Privacy (c) Safety (d) None of these

3. Actions taken to ensure application security are sometimes called.....
(a) security measures (b) window measures
(c) countermeasures (d) None of these

4. The most basic .....countermeasure is an application firewall.
(a) software (b) hardware (c) system (d) None of these

5. Errors can occur .....all phases of the systems life cycle.
(a) after (b) before (c) between (d) during

6. Trojan Horse commonly use network services to propagate to other host systems.
(a) True (b) False

7. .....is one of the most insidious threats that enterprises encounter today.
(a) Malware (b) Virus (c) Worm (d) Trojan Horse

12.7 Control Measures

Security and controls refer to all the measures adopted within an organization to safeguard assets, ensure the accuracy and reliability of records, and encourage operational efficiency and adherence to

prescribed procedures. The system of internal controls also includes the measures adopted to safeguard the computer system.

The nature of internal controls is such that certain control procedures are necessary for a proper execution of other control procedures. This interdependence of control procedures may be significant because certain control objectives that appear to have been achieved may, in fact, not have been achieved because of weaknesses in other control procedures upon which they depend.

Concern over this interdependence of control procedures may be greater with a computerized system than with a manual system because computer operations often have a greater concentration of functions, and certain manual control procedures may depend on automated control procedures, even though that dependence is not readily apparent. Adequate computer internal controls are a vital aspect of an automated system.

Security is an increasing concern because computer systems are increasingly complex. Particular security concerns result from the proliferation of PCs, local area networking, and on-line systems that allow more access to the mainframe and departmental computers. Modern technology provides computer thieves with powerful new electronic safecracking tools.

12.7.1 Insurance Considerations

Adequate insurance coverage is a key consideration when developing a business recovery plan and performing a risk analysis. Having a disaster plan and testing it regularly may not, in itself, lower insurance rates in all circumstances.

However, a good plan can reduce risks and address many concerns of the underwriter, in addition to affecting the cost or availability of the insurance.

Most insurance agencies specializing in business interruption coverage can provide the organization with an estimate of anticipated business interruption costs. Many organizations that have experienced a disaster indicate that their costs were significantly higher than expected in sustaining temporary operations during recovery.

Most business interruption coverage's include lost revenues following a disaster. Extra expense coverage includes all additional expenses until normal operations can be resumed. However, coverage's differ in the definition of resumption of services. As a part of the risk analysis, these coverage's should be discussed in detail with the insurer to determine their adequacy.

To provide adequate proof of loss to an insurance company, the organization may need to contract with a public adjuster who may charge between three and 10% of recovered assets for the adjustment fee. Asset records become extremely important as the adjustment process takes place.

Types of insurance coverage's to be considered may include: computer hardware replacement, extra expense coverage, business interruption coverage, valuable paper and records coverage, errors and omissions coverage, fidelity coverage, media transportation coverage.

With estimates of the costs of these coverage's, management can make reasonable decisions on the type and amount of insurance to carry.

These estimates also allow management to determine to what extent the organization should self-insure against certain losses.

12.7.2 Records

Records can be classified in one of the three following categories: vital records, important records, and useful records.

Vital records are irreplaceable. Important records can be obtained or reproduced at considerable expense and only after considerable delay. Useful records would cause inconvenience if lost, but can be replaced without considerable expense.

Vital and important records should be duplicated and stored in an area protected from fire or its effects.
Protection of records also depends on the particular threat that is present. An important consideration is the speed of onset and the amount of time available to act. This could range from gathering papers hastily and exiting quickly to an orderly securing of documents in a vault. Identifying records and information is most critical for ensuring the continuity of operations.
A systematic approach to records management is also an important part of the risk analysis process and business recovery planning. Additional benefits include: reduced storage costs, expedited service, federal and state statutory compliance.
Records should not be retained only as proof of financial transactions, but also to verify compliance with legal and statutory requirements. In addition, businesses must satisfy retention requirements as an organization and employer. These records are used for independent examination and verification of sound business practices.
Federal and state requirements for records retention must be analyzed. Each organization should have its legal counsel approve its own retention schedule. As well as retaining records, the organization should be aware of the specific record salvage procedures to follow for different types of media after a disaster.

Caution
Records kept in the computer room should be minimized and should be stored in closed metal files or cabinets. Records stored outside the computer room should be in fire-resistant file cabinets with fire resistance of at least two hours.

12.8 Recovery/Restart Requirements

This section discusses the design aspects of database recovery and restart. Proper recovery and restart planning is an important part of the design of the system, particularly in a database environment. Although Adabas provides facilities to perform both restart and recovery, the functions must be considered separately.

12.8.1 Recovery

Recovery of database integrity has the highest priority; if a database transaction fails or must be cancelled, the effects of the transaction must be removed and the database must be restored to its exact condition before the transaction began.
The standard Adabas system provides transaction logic (called *ET logic*), extensive checkpoint/logging facilities, and transaction-reversing backout processing to ensure database integrity.
Restarting the database following a system failure means reconstructing the task sequence from a saved level before the failure, up to and including the step at which the failure occurred-including, if possible, successfully completing the interrupted operation and then continuing normal database operation. Adabas provides a recovery aid that reconstructs a recovery job stream to recover the database.

Recoverability is often an implied objective. Everyone assumes that whatever happens, the system can be systematically recovered and restarted. There are, however, specific facts to be determined about the level of recovery needed by the various users of the system. Recoverability is an area where the DBA needs to take the initiative and establish necessary facts. Initially, each potential user of the

system should be questioned concerning his recovery/restart requirements. The most important considerations are:

- how long the user can manage without the system;
- how long each phase can be delayed;
- what manual procedures, if any, the user has for checking input/output and how long these take;
- What special procedures, if any, need to be performed to ensure that data integrity has been maintained in a recovery/restart situation?

12.8.2 Planning and Incorporating Recoverability

Once the recovery/restart requirements have been established, the DBA can proceed to plan the measures necessary to meet these requirements. The methodology provided in this section may be used as a basic guideline.

1. A determination should be made as to the level and degree to which data is shared by the various users of the system.
2. The recovery parameters for the system should be established. This includes a predicted/actual breakdown rate, an average delay and items affected, and items subject to security and audit.
3. A determination should be made as to what, if any, auditing procedures are to be included in the system.
4. An outline containing recovery design points should be prepared. Information in this outline should include.

- Validation planning. Validation on data should be performed as close as possible to its point of input to the system. Intermediate updates to data sharing the record with the input will make recovery more difficult and costly;
- dumps (back-up copies) of the database or selected files;
- user and Adabas checkpoints;
- use of ET logic, exclusive file control, ET data;
- audit procedures.

5. Operations personnel should be consulted to determine if all resources required for recovery/restart can be made available if and when they are needed.
6. The final recovery design should be documented and reviewed with users, operations personnel, and any others involved with the system.

12.8.3 Matching Requirements and Facilities

Once the general recovery requirements have been designed, the next step is to select the relevant Adabas and non-Adabas facilities to be used to implement recovery/restart. The following sections describe the Adabas facilities related to recovery/restart.

12.8.4 Transaction Recovery

Almost all online update systems and many batches update programs process streams of input transactions which have the following characteristics:

- The transaction requires the program to retrieve and add, update, and/or delete only a few records. For example, an order entry program may retrieve the customer and product records for each order, add the order and order item data to the database, and perhaps update the quantity-on-order field of the product record.
- The program needs exclusive control of the records it uses from the start of the transaction to the end, but can release them for other users to update or delete once the transaction is complete.

- A transaction must never be left incomplete; that is, if it requires two records to be updated, either both or neither must be changed.

12.8.5 End Transaction (ET) Command

The use of the Adabas ET command

- ensures that all the adds, updates, and/or deletes performed by a completed transaction are applied to the database;
- ensures that all the effects of a transaction which is interrupted by a total or partial system failure are removed from the database;
- Allows the program to store up to 2000 bytes of user-defined restart data (ET data) in an Adabas system file. This data may be retrieved on restart with the Adabas OP or RE commands. The restart data can be examined by the program or TP terminal user to decide where to resume operation;
- Releases all records placed in hold status while processing the transaction.

12.9 The Recovery Aid

When a system failure disrupts database operation, the Recovery Aid can create a job stream that reconstructs the database to the point of failure.

The Recovery Aid combines the protection log (PLOG) and the archived database status from previous ADASAV operations with its own recovery log (RLOG) information to reconstruct the job sequence. The result is a reconstructed job statement string (recovery job stream) that is placed in a specially named output data set.

The two major parts of the Adabas Recovery Aid are the recovery log (RLOG) and the recovery aid utility ADARAI. The RLOG is formatted like other Adabas files, using ADAFRM, and then defined with the ADARAI utility.

The DBA must run the Recovery Aid utility, ADARAI, to

- define the RLOG and set up the Recovery Aid environment;
- display current RLOG information;
- create the recovery job stream.

This section covers the following topics:

- The Recovery Log (RLOG)
- Starting the Recovery Aid

The Recovery Log (RLOG)

The recovery log (RLOG) records the essential information that, when combined with the PLOG, is used by the ADARAI utility's RECOVER function to rebuild a job stream to recover and restore the database status up to the point of failure.

The RLOG information is grouped in generations, where each generation comprises the database activity between consecutive ADASAV SAVE, RESTORE (database) or RESTORE GCB operations.

The RLOG holds a minimum of four consecutive generations, up to a maximum value specified when the RLOG is activated; the maximum is 32. If RLOG space is not sufficient to hold the specified number of generations, the oldest generation is overwritten with the newest in wraparound fashion.

The RLOG file is formatted like other database components by running the ADAFRM utility (SIZE parameter), and then defined using the PREPARE function of the Recovery Aid ADARAI utility

(with the RLOGSIZE parameter). The space required for the RLOG file is approximately 10 cylinders of 3380 or equivalent device space.

The ADARAI PREPARE function must be performed just before the ADASAV SAVE run that begins the first generation to be logged. After ADARAI PREPARE is executed, all subsequent nucleus and utility jobs that update the database must specify the RLOG file. Of course, the RLOG file can be included in any or all job streams, if desired.
The RLOG file job statement should be similar to the following:
//DDRLOGR1 DD DISP=SHR,DSN=... .RLOGR1

12.9.1 Starting the Recovery Aid

The activity of the Recovery Aid and RLOG logging begins when the first ADASAV SAVE/RESTORE database or RESTORE GCB function is executed following ADARAI PREPARE.
All activity between the first and second ADASAV SAVE/RESTORE database or RESTORE GCB operations following the ADARAI PREPARE operation belongs to the first generation. When viewing generations with the ADARAI utility's LIST function, generations are numbered relatively in ascending order beginning with the oldest generation.

12.10 System Failures and Recovery

When running without K-safety (in other words, a K-safety value of zero) any node failure is fatal and will bring down the database (since there are no longer enough partitions to maintain operation). When running with K-safety on, if a node goes down, the remaining nodes of the database cluster log an error indicating that a node has failed.
By default, these error messages are logged to the console terminal. Since the loss of one or more nodes reduces the reliability of the cluster, you may want to increase the urgency of these messages. For example, you can configure a separate Log4J appender (such as the SMTP appender) to report node failure messages. To do this, you should configure the appender to handle messages of class HOST and severity level ERROR or greater.
When a node fails with K-safety enabled, the database continues to operate. But at the earliest possible convenience, you should repair (or replace) the failed node.
To replace a failed node to a running VoltDB cluster, you restart the VoltDB server process specifying the deployment file and adding the argument “rejoinhost” with the address of one of the remaining nodes of the cluster. For example, to rejoin a node to the VoltDB cluster where myclusternode5 is one of the current member nodes, you use the following command:

```
$ voltdb  rejoinhost myclusternode5 \
        deployment mydeployment.xml
```

Note that the node you specify may be any active cluster node; it *does not* have to be the node identified as the lead node on the command line when the cluster was originally started. Also, the deployment file you specify must be the currently active deployment settings for the running database cluster.
If security is enabled for the cluster, you must also specify a username and, optionally, a password on the command line. (If you specify a username but not a password, you will be prompted for the password.) The full syntax for specifying the node to reconnect to is as follows. You only need to specify the port number if the server was started using a different port number than the default.

username:password@nodename:port

For example, the following command attempts to rejoin the current system to the cluster that includes the node *voltserver2* using the username *operator*. VoltDB will prompt for the password.

```
$ voltdb  rejoinhost operator@voltserver2 \
        deployment mydeployment.xml
```

12.10.1 What Happens When a Node Rejoins the Cluster?

When you issue the rejoin command, the node first rejoins the cluster, then retrieves a copy of the application catalogue and the appropriate data for its partitions from other nodes in the cluster. Rejoining the cluster only takes seconds and once this is done and the catalogue is received, the node can accept and distribute stored procedure requests like any other member.

However, the new node will not actively participate in the work until a full working copy of its partition data is received. What's more, the update process for each partition operates as a single transaction and will block further transactions on the partition which is providing the data.

While the node is rejoining and being updated, the cluster continues to accept work. If the work queue gets filled (because the update is blocking further work), the client applications will experience back pressure. Under normal conditions, this means the calls to submit stored procedures with the callProcedure method (either synchronously or asynchronously) will wait until the back pressure clears before returning control to the call application.

The time this update process takes varies in length depending on the volume of data involved and network bandwidth. However, the process should not take more than a few minutes.

More importantly, the cluster is not fully K-safe until the restoration is complete. For example, if the cluster was established with a K-safety value of two and one node failed, until that node rejoins and is updated, the cluster is operating with a K-safety value of one. Once the node is up to date, the cluster becomes fully operational and the original K-safety is restored.

12.10.2 Where and When Recovery May Fail

It is possible to rejoin any node with the appropriate catalogue and deployment file to the cluster. It does not have to be the same physical machine that failed. This way, if a node fails for hardware reasons, it is possible to replace it in the cluster immediately with a new node, giving you time to diagnose and repair the faulty hardware without endangering the database itself.

It is also possible to rejoin multiple nodes simultaneously, if multiple nodes fail. That is, assuming the cluster is still viable after the failures. As long as there is at least one active copy of every partition, the cluster will continue to operate and be available for nodes to rejoin.

There are a few conditions in which the rejoin operation may fail. Those situations include the following:

- Insufficient K-safety

If the database is running without K-safety, or more nodes fail simultaneously than the cluster is capable of sustaining, the entire cluster will fail and must be restarted from scratch. (At a minimum, a VoltDB database running with K-safety can withstand at least as many simultaneous failures as the K-safety value. It may be able to withstand more node failures, depending upon the specific situation. But the K-safety value tells you the minimum number of node failures that the cluster can withstand.)

- Mismatched deployment file

If the deployment file that you specify when issuing the rejoin command does not match the current deployment configuration of the database, the cluster will refuse to let the node rejoin.

- More nodes attempt to rejoin than have failed

If one or more nodes fail, the cluster will accept rejoin requests from as many nodes as failed. For example, if one node fails, the first node requesting to rejoin with the appropriate catalogue and deployment file will be accepted. Once the cluster is back to the correct number of nodes, any further requests to rejoin will be rejected. (This is the same behaviour as if you tried to add more nodes than specified in the deployment file when initially starting the database.)

- The rejoining node does not specify a valid username and/or password

When rejoining a cluster with security enabled, you must specify a valid username and password when issuing the rejoin command. The username and password you specify must have sufficient privileges to execute system procedures. If not, the rejoin request will be rejected and an appropriate error message displayed.

12.11 Disaster/Recovery Planning

Because a goal of business recovery planning is to ensure the safety of personnel and assets during and following a disaster, a critical aspect of the risk analysis process is to identify the preparedness and preventive measures in place at any point in time. Once the potential areas of high exposure to the organization are identified, additional preventative measures can be considered for implementation.

Disaster prevention and preparedness begins at the top of an organization. The attitude of senior management toward security and prevention should permeate the entire organization. Therefore, management’s support of disaster planning can focus attention on good security and prevention techniques and better prepare the organization for the unwelcome and unwanted.

Disaster prevention techniques include two categories: procedural prevention and physical prevention. Procedural prevention relates to activities performed on a day-to-day, month-to-month, or annual basis, relating to security and recovery. Procedural prevention begins with assigning responsibility for overall security of the organization to an individual with adequate competence and authority to meet the challenges. The objective of procedural prevention is to define activities necessary to prevent various types of disasters and ensure that these activities are performed regularly. Physical prevention and preparedness for disaster begins when a site is constructed. It includes special requirements for building construction, as well as fire protection for various equipment components. Special considerations include: computer area, fire detection and extinguishing systems, record(s) protection, air conditioning, heating and ventilation, electrical supply and UPS systems, emergency procedures, vault storage area(s), archival systems.

12.12 Plans

A disaster recovery plan is a written plan describing the steps a company or individual would take to restore computer operations in the event of a disaster. Every company and each of its' department or division within an enterprise usually has it's own disaster recovery plan. A disaster recovery plan contains four major components: the emergency plan, the backup plan, the recovery plan, the backup plan, the recovery plan, and the test plan.

12.12.1 The Emergency Plan

An emergency plan specifies the steps to be taken immediately after a disaster strikes. The emergency plan usually is organized by type of disaster, such as fire, flood, or earthquake. Depending on the nature of disaster and the extent of the disaster, the procedures that are followed in an emergency will differ.

12.12.2 The Backup Plan

Once the procedures in the emergency plan have been executed, the backup plan is executed. The backup plan specifies how a company uses backup files and equipment to resume business processing. The backup plan should specify the location of an alternate computer facility in the event the company's normal location is destroyed or unusable.

When operations are so important that a company cannot afford to lose the operations to a disaster, the company often maintains a hot site, warm site or cold site:

Hot site—is a separate facility that mirrors the systems and operations of the critical site. The hot site always operates concurrently with the main site. This type of backup site is the most expensive to operate. Hot sites are popular with stock exchanges and other financial institutions that may need to evacuate due to potential bomb threats and must resume normal operations as soon as possible.

Warm site—A warm site is a location where the business can relocate to after the disaster that is already stocked with computer hardware similar to that of the original site, but does not contain backed up copies of data and information.

Cold site—is a site that mirrors some of the critical site hardware, but does become operational until the critical site becomes unavailable. It's the most inexpensive type of backup site for a business to operate. It does not include backed up copies of data and information from the original location of the business, nor does it include hardware already set up.

The location of the alternate site facility is important. It should be close enough to be convenient, yet not too close that a single disaster, such as an earthquake, could destroy both facilities. All sites should have high-speed Internet services.

12.12.3 The Recovery Plan

The recovery plan specifies the actions to be taken to restore full information processing operations. As with the emergency plan, the recovery plan differs for each type of disaster. To prepare for disaster recovery, a company should establish planning committees, with each one responsible for different forms of recovery.

12.12.4 The Test Plan

To provide assurance that the disaster plan is complete, it should be tested, several times. A disaster recovery test plan contains information for simulating various levels of disasters and recording an organization's ability to recover. Any needed recovery actions that are not specified in the plan should be added.

12.12.5 Security in the Enterprise and Small Business

Large organizations must deal with complex computer security issues. Web sites and hardware must be secure from threats both inside and outside the enterprise. Users should be allowed access only to the data and information for which they are authorized, which typically is limited to the amount necessary to do their job. Small business faces some of the same issues and should take the necessary steps to secure their systems.

12.12.6 Developing a Computer System Security Plan

A computer security plan summarizes in writing all of the safeguards that are in place to protect a company's information assets. A computer security plan should do the following:

- Identify all information assets of an organization, including hardware, software, documentation, procedures, people, data, facilities, and supplies.
- Identify all security risks that may cause a security breach which cause a loss in assets. Rank risks from most likely to occur to least likely to occur.
- Place an estimated value on each risk, including the value of lost business.
- For each risk, identify the safeguards that exist to detect, prevent, and recover from a loss.

Keep in mind that some degree of risk is unavoidable. The more secure a system is, the more difficult it is for everyone to use.

12.13 Team

General roles and responsibilities for teams involved in systems recovery are defined below. Specific tasks for each team related to recovery in the event of an incident are listed in section 8.0. Individuals currently filling these positions, along with their contact information, can be found in Appendix B.

12.13.1 Recovery Management Team

Team Leader – Director
Backup Team Leader – Assoc Director

Team Members:
Associate Director
Facilities Manager
Telecom Manager
Internet Services Manager
Data Services Manager

Responsibilities:
This team is responsible for the overall coordination of responses to all emergencies affecting information and telecommunication systems.

- Readiness Responsibilities
  - Support related training
  - Test and update System Recovery Plan.
  - Manage Assets and Services Database
  - Conduct an annual review of the Systems Recovery Plan along with a reassessment of risks and an update of the Risk Mitigation Plan.
  - Market IT Systems Recovery Plan to campus.
- Recovery Responsibilities
  - Assess severity of service interruption and declare disaster if warranted.
  - Initiate action by appropriate recovery team(s).
  - Manage communications amongst recovery teams.
  - Manage communications to Campus.
  - Co-ordinate resources and financial requirements needed to effect recovery.
  - Declare return to service under temporary operations.
  - Declare return to service under normal operations
  - Assess recovery process after return to normal operation.
  - Implements updates or improvements to SRP

12.13.2 Facilities Recovery Team

Team Leader – Telecom Manager
Backup Team Leader – Facilities Manager

Team Members:

Facilities Manager
Network Analyst
Telecom Electrician
Telecom Technician
Physical Plant Person(s)

Responsibilities

This team is responsible for responding to emergencies which physically impact the computer rooms, network and telephone equipment rooms, cabling and wiring infrastructure, ancillary equipment such as UPS and air-conditioning, servers and network devices. The focus of this recovery team is to provide the facilities and devices necessary to restore services that have been disrupted.

- Readiness Responsibilities
  - Review backup power and cooling capabilities.
  - Ensure related contracts are in place.
  - Ensure network and wiring diagrams are up-to-date and a printed copy is available.
  - Ensure information from Physical Plant is available as needed.
  - Conduct annual review of IT facilities with regard to power, cooling, security, fire detection and suppression and water detection.
  - Ensure that Asset and Services Management Database (ASMD) is up to date and a copy is available.
  - Document power management grid layout for data centre.
  - Insure an appropriate level of spares is in place.
- Recovery Responsibilities
  - Perform assessment and recovery tasks as outlined in section 8.4.
  - Proactively communicate to Recovery Management Team.

12.13.3 Server Recovery Team

Team Leader – Internet Services Manager
Backup Team Leader – Data Services Manager

Team Members:

Data Services Manager
Facilities Manager
System Support Specialist
System Support Specialist
System Support Specialist
System Support Specialist

Responsibilities

This team is responsible for response and resolution of all emergencies affecting the server hardware, operating system and applications as identified in the asset and service management database. Examples of these services include Email, Banner, WebCT.

- Readiness Responsibilities
  - Ensure operating system, data and application files are backed up.
  - Conduct periodic test to ensure backup and recovery procedures are current and tested.
  - Ensure currency of configuration parameters, procedures, tools, and processes for re-build of services.
  - Ensure that service contracts are up-to-date.
  - Train backup personnel in rebuild procedures.
  - Keep Assets and Services Management Database up to date as systems are added, changed or removed.

- Recovery Responsibilities
  - Perform assessment and recovery tasks as outlined in section 8.5.
  - Proactively communicate to Recovery Management Team.

12.13.4 Desktop Services Recovery Team

Team Leader – Assoc Director
Backup Team Leader – Facilities Manager

Team Members:

Lab Supervisor
Desktop Team Lead
Help Desk Team Lead

Responsibilities

This team is responsible for response and resolution of all emergencies affecting all desktops except where units have their own desktop support staff. Information Technology will work with any such units to assist them with their own system recovery plan.

- Readiness Responsibilities
  - Identify source of replacement desktops.
  - Keep inventory of assets up to date.
  - Ensure ghost images of default configurations are in place.
- Recovery Responsibilities
  - Perform assessment and recovery tasks as outlined in section 8.6.
  - Proactively communicate to Recovery Management Team.

12.13.5 Network and Telephone Systems Recovery Team

Team Leader – Telecom Manager
Backup Team Leader – Internet Services Manager

Team Members:

Network Analysis
Telecom Electrician
Telecom Technician

Responsibilities

This team is responsible for response and resolution of all emergencies affecting network and telephone equipment, and cabling. Networks include Corporate Network, Residence Network, Supernet, Netera, and LAnet, routers, switches, firewalls.

- Readiness Responsibilities
  - Ensure telephone switch data is backed up and available to recovery team.

- Keep inventory of network and telephone assets up to date.
- Document configuration data and rebuild instructions.
- Maintain and update network diagrams.
- Update service and asset management database related to network functions.
- Review network security (operating and physical).
- Ensure that service contracts are up-to-date.
- Ensure backup personnel are trained in recovery procedures.

- Recovery Responsibilities
  - Perform assessment and recovery tasks as outlined in section 8.7.
  - Proactively communicate to Recovery Management Team.

12.14 Planning Tasks

Analysis, design, and implementation descriptions of security controls are completed and recorded in the system security planning (SSP). The following tasks are required for SSP development:
1. Review the Security Requirements Traceability Matrix (SRTM) for completeness.
2. Review the most current RA document to identify any additions to the controls listed in the SRTM.
3. Review Component policies to identify any modifications to the controls listed in the SRTM.
4. Document any planned compensating controls and track compensating controls into the ST&E Plan so that the compensating controls are tested as satisfying requirements.
5. Review acquisition documents to identify any modifications to the controls listed in the SRTM.
6. Draft the SSP, using the SSP template in RMS.
7. Draft Rules of Behaviour.
8. Draft Interconnection Security Agreements (ISA).
9. If required, together with the DHS Privacy Office, complete a PIA. See the section on System Security Characterization Process.
10. Conduct SSP review and obtain approval by the System Owner, Component CISO, and DAA.

12.15 Ethics in System Development

Ethics is the study of value concepts such as ‘good,’ ‘bad,’ ‘right,’ ‘wrong,’ ‘ought’, applied to actions in relation to group norms and rules. Therefore, it deals with many issues fundamental to practical decision-making. Computer software systems lie at the heart of modern decision making, including data/information storage and manipulation, data availability, and ‘alternatives’ formulation and selection. In fact, the very use of computer systems can often frame the types of questions that can be asked as well as their possible answers.
This is particularly evident when we incorporate software systems into our knowledge management methods, as they then play an essential role in institutional memory. The ubiquity of software systems in all aspects of public and private institutions means that the environment that they create needs to be critically examined as they are developed and deployed.
Two major ethical questions must be addressed with regard to software systems. Firstly, can these systems represent the different codes of ethics of the groups affected by software-mediated decisions? Secondly, what ethical considerations should guide the design and development of the software itself?

12.16 Ethics Codes and Standards of Behaviour

"When considering the issue of ethical… behaviour in the work setting… a basis for ethical behaviour can be found in the context of business as a social institution. Second, a rationale for ethical behaviour can be obtained from guidelines implied in the notion of professionalism." To determine the ethical standards recognized by computer professionals, the existing ethics codes of four major computer professional associations were compared.
The themes are:
(1) Personal integrity / claim of competence,
(2) Personal responsibility for work,
(3) Responsibility to employer/client,
(4) Responsibility to profession,
(5) Confidentiality of information,
(6) Conflict of interest,
(7) Dignity /worth of people,
(8) Public safety, health, and welfare,
(9) Participation in professional societies, and
(10) Increase public knowledge about technology.

It is encouraging that in all of the ethics codes of the computer professional societies there is an emphasis on the relationship and interaction of the computer professional with other people, rather than with machines. This properly places the focus of ethical behaviour upon ethical or right dealings with people, rather than upon the technology. One reason that the four codes are not only similar to each other, but also very similar to codes of non-computer professionals is that they take a generic approach to ethics. With the exception of the concern raised about privacy and the confidentiality of data, the codes could have been written to cover most professions and do not fully reflect the unique ethical problems raised by computer technology.

12.17 Summary

- Security is "freedom from risk or danger." In the context of computer science, security is the prevention against unauthorized recipients.
- Computer security is frequently associated with three core areas i.e. Confidentiality, Integrity, and Authentication.
- Privacy is a property of individuals; confidentiality is a property of data; and security is a property assigned to computer hardware and software systems.
- Analyzing security by function can be a valuable part of the security planning process.
- The term malicious hackers refer to those who break into computers without authorization.

12.18 Keywords

Errors and omissions are an important threat to data and system integrity.
***Network spoofing*:** In network spoofing, a system presents itself to the network as though it were a different system.
***Packet replay*:** This refers to the recording and retransmission of message packets in the network.
Trojan horse: A program that performs a desired task, but that also includes unexpected functions.
Virus: A code segment that replicates by attaching copies of itself to existing executables.

12.19 Review Questions

1. Define the term security system.
2. Why system security is an important concern?
3. What are the various threats to system security?
4. What do you mean by system integrity?
5. How risk can be analysed in system security?
6. Where and when recovery may fail?
7. Why disaster/recovery planning is useful?
8. What are the various ethics in system development?
9. Briefly explain the term security vulnerabilities.
10. What are the preventions to protect our systems from Malware?

Answers for Self Assessment Questions

1 (b) 2 (a) 3 (c) 4 (a) 5 (d) 6 (b) 7 (a)

12.20 Further Studies

1. System Analysis, Design, and Development: Concepts, Principles, and Practices By Charles S. Wasson

13

Electronic Data Processing

STRUCTURE

13.0 Objectives

13.1 Introduction

13.2 Data versus Information

13.3 Characteristics of useful Information

13.4 Data Processing

13.5 Need of Data Processing

13.6 Types of Data Processing

13.7 Data Management

13.8 Data Organization

13.9 Database Management Systems

13.10 Data Warehousing

13.11 Future Trends in Data Warehousing

13.12 Data Verification and Data Validation

13.13 EDP Organization

13.14 Data centre

13.15 Evolution of Data Centre

13.16 Requirements for Modern Data

13.17 Summary

13.18 Keywords

13.19 Review Questions

13.20 Further Studies

13.0 Objectives

After studying this chapter, you will be able to:

- Discuss the data processing concept
- Differentiate between data and information
- Discuss the characteristics of useful information
- Define the data processing

- Explain the need and approaches of data processing
- Discuss the types of data processing
- Define the data management
- Explain the data organization
- Discuss the database management systems

13.1 Introduction

The Electronic Data Processing (EDP) division provides computerized services for all departments in the TRC. TRC departments have direct access to the data with their personal computers (PC). The EDP division is continuing to give data management support including data entry/verification to various studies undertaken in the centre. Also, this division generates reports and prepares pre-printed forms for field activity and supply data tabulations for monitoring the studies and publication of research work. Also, helps in preparation of employees' pay roll, income-tax sheets, loan schedules and central bills.

This division takes care to serve all the departments in their computing, data sharing and helps in accessing internet connection throughout. At present time the EDP division supports three server systems and four network printers and catering support for over 80 Pentium computers. The EDP division protects data through frequent backups. Apart from data processing and data management, this division is taking care of all the servers, PC's and printers by bringing under comprehensive maintenance contract service to avoid breakdown. Also, annual procurement for computer consumables is done by making indent through this division for user departments.

In this division, at present, six data entry/verification operators, six data processing assistants and one EDP in charge are working.

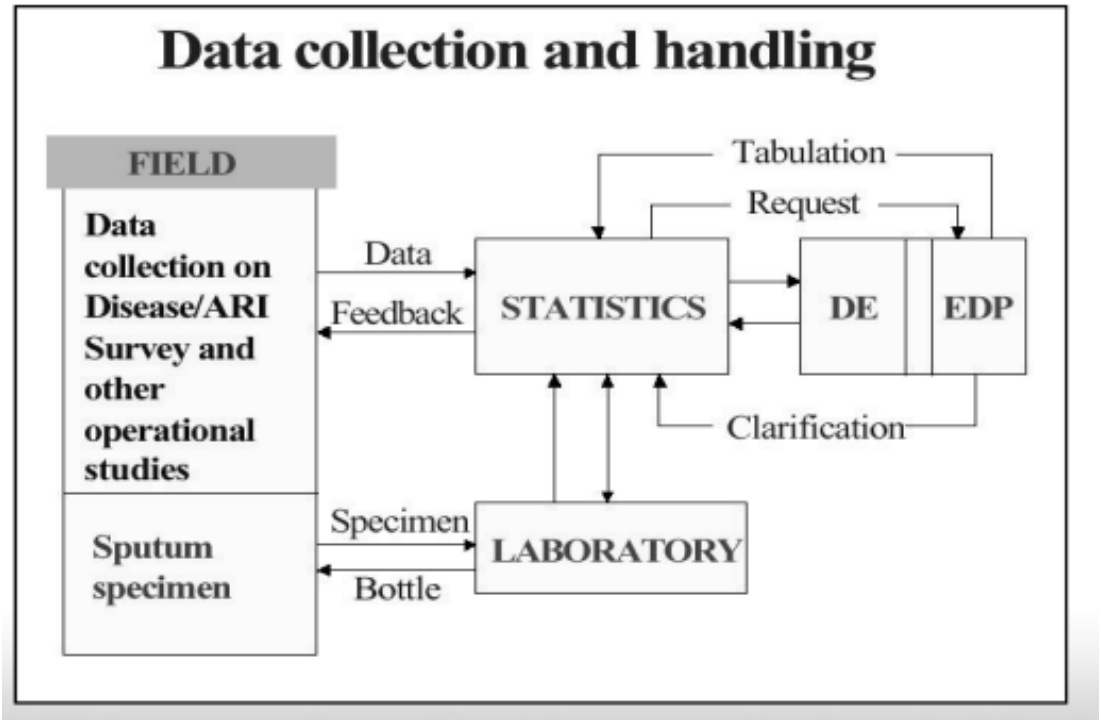


Figure 13.1: Data collection.

13.2 Data versus Information

The term data is used to describe raw facts (not yet processed) about something or someone. Data is raw facts from which the required information is derived.
Example: ROBCOR Company tracks all sales for its two divisions through invoices.

Each of the invoices contains raw facts such as these:
Invoice number = 300124, Invoice date = 12-Jan-2004, and Sales amount = 12500.98
Suppose that ROBCOR's two divisions have generated 1,380,456 and 1,453,907 invoices, respectively, between the first quarter of 1999 and the first quarter of 2004. These millions of invoices the ROBCOR had generated contain raw facts which do not have meaning unless and until they are processed into information.
Now, suppose that for the purpose of making conclusions and/or decisions, the ROBCOR's sales managers want information about sales productivity per employee for each of the two divisions. The generated sales productivity information will reveal meaning of the data exist in the sales invoices. An application program in an information system will generate the required information. Figure13.2, from the text book, shows a graph of the sales per employee for each of the ROBCOR's two divisions.

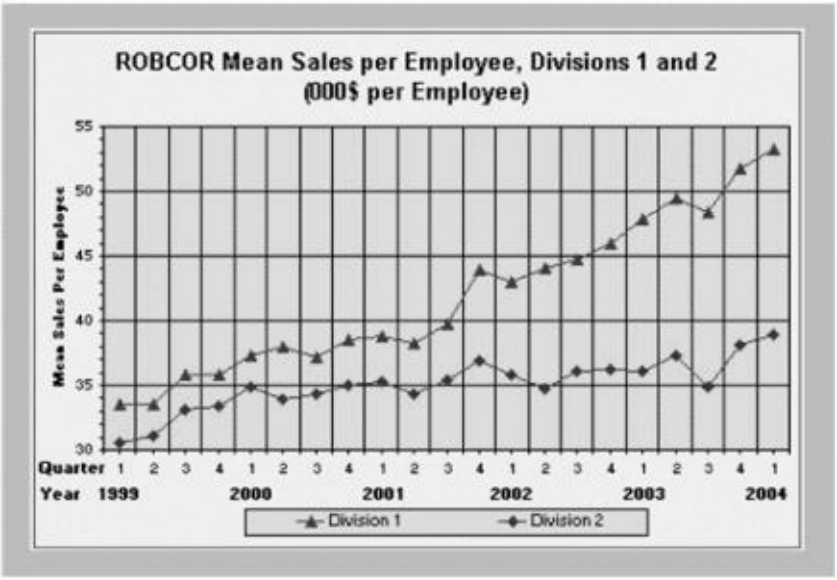


Figure 13.2: Sales per employee for each of the ROBCOR's two divisions.

When data is stored electronically in files, it can be used as input for an information system. An information system has programs to process (or transform) data to produce information as an output, see Figure 13.3. Information reveals meaning of data.
For example, student's data values such as ID, Name, Address, Major, and Phone number represent raw facts. Class roll is a list which shows students ID and Names of those students who are enrolled in particular class.

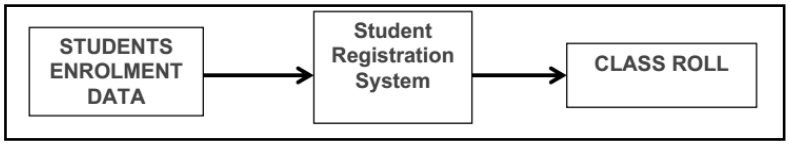


Figure 13.3: Data processed into information.

Let's summarize some key points:

- Data constitute the building blocks of information.
- Information is produced by processing data.
- Information is used to reveal the meaning of data.
- Good, relevant, and timely information is the key to good decision making.
- Good decision making is the key to organizational survival in a global environment.

Timely and useful information requires accurate data. To achieve accurate information, the data must be stored and generated properly. Also, the data must be stored in a format that is easy to access and process. And like any basic resource, the data environment must be managed carefully. Thus, Data management is a discipline that focuses on the proper generation, storage, and retrieval of data.

13.3 Characteristics of useful Information

There are five characteristics of useful information.

Relevance

The information that a person uses must be relevant to the current problem at hand. If the information is not relevant it would not help to solve the problem at all.

Complete

The information must be complete. If the information is not complete and only partial information is received, it may lead to wrong decisions being made as only half of an entirety of the information is known.

Accuracy

The information must be accurate as inaccurate information would lead to many different major problems as inaccurate information for example, may lead to people playing the stock market to lose allot of money.

Current

The information must be current as a fact of yesterday may not be a fact of today. For example, a criminal on the loose the day before would not be a criminal on the loose anymore if he gets caught today. As the stock market is ever-changing, current information is really a must.

Economical

The information used must be of economical use to companies that use them. Companies are built to make profit; if the information used is not economical it would result in companies making losses and get shut down. That is why economical information is important.

13.4 Data Processing

The data processing activities described above are common to all data processing systems from manual to electronic systems. These activities can be grouped in four functional categories, viz., data input, data processing, data output, and storage, constituting what is known as a data processing cycle.

13.4.1 Input

The term input refers to the activities required to record data and to make it available for processing. The input can also include the steps necessary to check, verify and validate data contents.

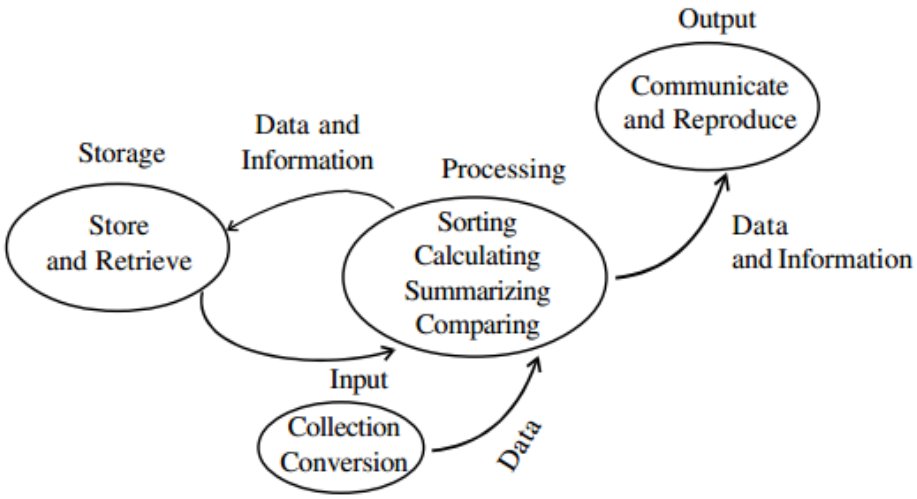


Figure 13.4: Data processing cycle.

13.4.2 Processing

The term processing denotes the actual data manipulation techniques such as classifying, sorting, calculating, summarizing, comparing, etc. that convert data into information.

13.4.3 Output

It is a communication function which transmits the information, generated after processing of data, to persons who need the information. Sometimes output also includes decoding activity which converts the electronically generated information into human-readable form.

13.4.4 Storage

It involves the filing of data and information for future use. The above mentioned four basic functions are performed in a logical sequence as shown in Figure 13.4 in all data processing systems.

Did You Know?

In 1961 census, a modest beginning was made with the adoption of "Unit record" machines. Prior to 1961 censuses, the data collection, data entry and processing were done manually.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Draw the flow diagram of data processing.

……

……

……

13.5 Need of Data Processing

Services of Data Processing is consist of a variety of giving out of data from one data arrangement to another, data digitizing and data capturing. Exchange from single to another data format is database for effective analysis and research. Converting raw data into a specified format of information is

termed as data processing. Outsource data processing improve the quality of data entry form, word processing, image processing and data conversion.

Data processing Outsourcing is helpful in a various division of BPO industry for instance services providers as well as BPO professionals. Data processing outsources services ranging from:

- Data conversion
- Data entry
- Survey processing
- Word processing
- Database management
- Script processing
- Image processing
- Forms processing

Data Conversion

Data conversion is the process of converting one form of a data into another, for example converting data from paper source to digital database or converting from one operating system to another.

Data Entry

Data entry, automated data capture, quality checks and proof reading. The skilled staffs deliver 100% accurate outputs. Enter data from hand-written, printed document, electronic data, scanned image or any other type of documents. Also, offer 24 x 7 support and assistance in data entry works.

Process of Data

Data processing involves digitizing, capturing and handing out of data including word processing, form processing, image processing, data entry etc. from different resources as well as changing them into a database for effective analyzes and research.

Image Processing

Image processing is analyzing as well as manipulating images to an exact format or quality otherwise reporting based on the study of images. Image processing as well involves exchange of one arrangement of an image into an additional format as per the most of the company.

Form Processing

Form processing and survey processing are an important part in major domains such as banking, insurance, health care, billing etc. Persons use optical character recognition, intelligent character recognition software for quick processing and intelligent mark recognition. In many cases these software could not be used to capture data from owing to unreadable handwriting, etc. In such situation, we undertake manual data entry to undertake form processing. As well, make sure that the manually procedure forms are verified for accuracy before final production.

Caution

Data base must be secure with protect security system; do not give the permission to access all data to everyone.

Self Assessment Questions

1.....is the second phase of ‘cleaning’ in which thorough quality-check is been done.

(a) Coding (b) Tabulation

(c) Data Entry (d) Validation

2.....is the act of handling or manipulating data in some fashion.

(a) Data management
(b) Data organization
(c) Data processing
(d) None of these

3. The term.....refers to the activities required to record data and to make it available for processing.
(a) input
(b) output
(c) process
(d) storage

13.6 Types of Data Processing

Data processing is the sequence of operations performed on data to convert it into useful information. These operations may be arithmetic, statistical or logical operations. We can arrange or classify data. We can send and receive data from one point to another.
There are three types of data processing.
(1) Manual data processing
(2) Mechanical data processing
(3) Electronic data processing

13.6.1 Manual Data Processing

In manual data processing data is processed manually. No machine or tool is used.



Figure 13.5: Manual data processing.

All the calculations on data are performed manually. This is a slow method and errors may occur. This is an old method. It was used before the invention of calculators. But data is still processed manually in many small shops.
Example: A book seller (a small book shop) records his daily transactions manually. He prepares bills by pen, paper and carbon paper (no doubt, brain is the main data processor in this case). At the end of day he will use carbon copies made on a particular date to know how many books he sold and how much income he got.

13.6.2 Mechanical Data Processing



Figure 13.6: Mechanical data processing.

In mechanical data processing data is processed with the help of devices or machines. These machines that help the people in data processing may be calculators and type writers etc. Obviously, this method is faster easier and more accurate than manual data processing.



Figure 13.7: Data processing system.

Example: Book seller can use a calculator to speed up his data processing system. There will be a less chance of errors in calculations. Bill calculations will be much faster with a calculator and easier too.

13.6.3 Electronic Data Processing

It is the latest and fastest method of data processing. Data is processed with the help of computer. First of all, data and set of instructions are given to the computer. Computer gives output results after processing the data according to instructions. This method of data processing is very fast and accurate. Electronic data processing is used in banks and business firms, Government offices, colleges, schools, universities, businesses and other institutes. Electronic data processing is being used in almost every field of life.

Example: Suppose there are 800 students in a college. There is a manual library system in this college. If we want to know which students have not returned books since one year? We will have to search registers for 800 students' record. But computer can do this job within seconds.



Figure 13.8: Electronic data processing.

In short, we can say that:
Electronic data processing:
(i) Is more fast
(ii) Is less time consuming
(iii) Can process large amounts of data easily
(iv) Can present information in more attractive way
(v) Will give 100% error free results, if input and instructions are accurate.

13.7 Data Management

Data management falls under the rubric of project management. Most researchers are unprepared for project management, since it tends to be underemphasized in training programs. An epidemiologic project is not unlike running a business project with one crucial difference; the project has a fixed life span. This difference will affect many aspects of its management. Some areas of management that are affected are hiring, firing, evaluation, organization, productivity, morale, communication, ethics, budget, and project termination. Although the production of a study proposal raises many management challenges, if the proposal is approved and funds allocated, the accomplishments of the project are dependent more upon its management than any other factor. A particular problem for investigators and staff, if they lack specific training or experience, is to fail to appreciate and prepare for the implications and exigencies of mass production.

13.8 Data Organization

The data processing cycle (also called Information Processing Cycle) and the components of a computer, we will now describe how data is organized before processing on a computer. Data can be arranged in a variety of ways, but a hierarchical approach to organization is generally recommended.

13.8.1 Data Item

A data item is the smallest unit of information stored in computer file. It is a single element used to represent a fact such as an employee's name, item price, etc. In a payroll application, the employee number 170 is a data item Pankaj the name is a data item.

13.8.2 Field

Data items are physically arranged as fields in a computer file. Their length may be fixed or variable. Since all individuals have 3 digit employee numbers, a 3-digit field is required to store the particular data. Hence, it is a fixed field. In contrast, since customer's name varies considerably from one customer to another, a variable amount of space must be available to store this element. This can be called as variable field.

13.8.3 Record

A record is a collection of related data items or fields. Each record normally corresponds to a specific unit of information. For example, various fields in the record, illustrated in Figure 13.9 are employee number, employee's name, basic salary and house rent allowance. This is the data used to produce the payroll register report. The first record contains all the data concerning the employee Pankaj. The second record contains all the data concerning the employee Rekha. Each subsequent record contains all the data for a given employee. It can be seen how each related item is grouped together to form a record.

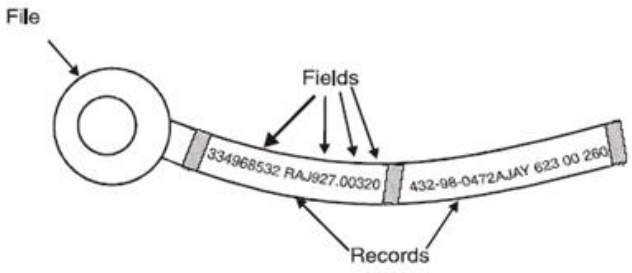


Figure 13.9: Data record.

13.8.4 File

The collection of records is called a file. A file contains all the related records for an application. Therefore, the payroll file contains all records required to produce the payroll register report. Files are stored on some medium, such as floppy disk, magnetic tape or magnetic disk.

13.8.5 Database

The collection of related files is called a database. A database contains all the related files for a particular application.

13.9 Database Management Systems

A DBMS is a complex set of software programs that controls the organization, storage, management, and retrieval of data in a database. DBMS are categorized according to their data structures or types, sometime DBMS is also known as Data base Manager. It is a set of prewritten programs that are used to store, update and retrieve a Database. A DBMS includes:

A modelling language to define the schema of each database hosted in the DBMS, according to the DBMS data model.

- The four most common types of organizations are the hierarchical, network, relational and object models. Inverted lists and other methods are also used. A given database management system may provide one or more of the four models. The optimal structure depends on the natural organization of the application's data, and on the application's requirements (which include transaction rate (speed), reliability, maintainability, scalability, and cost).
- The dominant model in use today is the ad hoc one embedded in SQL, despite the objections of purists who believe this model is a corruption of the relational model, since it violates several of its fundamental principles for the sake of practicality and performance. Many DBMSs also support the Open Database Connectivity API that supports a standard way for programmers to access the DBMS.

13.9.1 DBMS Benefits

- Improved strategic use of corporate data
- Reduced complexity of the organization’s information systems environment
- Reduced data redundancy and inconsistency
- Enhanced data integrity
- Application-data independence
- Improved security
- Reduced application development and maintenance costs
- Improved flexibility of information systems
- Increased access and availability of data and information
- Logical & Physical data independence
- Concurrent access anomalies.
- Facilitate atomicity problem.
- Provides central control on the system through DBA.

Did You Know?

As computers grew in speed and capability, a number of general-purpose database systems emerged; by the mid-1960s there were a number of such systems in commercial use.

13.10 Data Warehousing

Data warehousing concepts are used to design, create and manage a data warehouse that provides a centralized company database. Data warehouses were first developed in the late 1980 and early 1990 as a response to the need for business analysis that could not be effectively met by current operational database systems. To meet this need, the process of the recording, collecting, filtering and loading of data into a database was revised, streamlined and customized to support analysis and decision-making. This serves to differentiate these data repositories from the regular transactional systems that are central to operations.

13.10.1 Distinguishing Characteristics

Purpose is the distinguishing characteristic of these specialized data repositories, not form. This means that the form of the stored data or the type of the database used can vary widely. Data can either be normalized or de-normalized and the database itself can take on a number of forms from an object database to a hierarchical database, relational, flat file or multidimensional. The data itself can change a number of times and the database should be designed to accommodate this, but the most important basis for design is in the way it is set up to support decision-making for a specific action or entity.

13.10.2 Design and Implementation Challenges

The implementation challenge starts with the collection of disparate data from several sources, including but not limited to the transactional and operational databases. This database structure must be able to merge old or existing data with new data and transform it into a standard format compatible to the data warehouse platform. Integrating disparate data requires resolving conflicts in areas such as naming and grouping protocol, units of measuring and maybe even time zones.

13.10.3 Benefits and Rewards

Because an efficient database need to be both customized to meet a specific need and flexible enough to handle disparate and changing data, database design can be complicated. The rewards for successfully putting together and running a good database far outweighs the challenges though.
Some of these benefits include the following:

- Providing a standardized data format for different data sources and interest areas, which will not only streamline analysis and reporting but also make the data reusable for different departments, interest groups and levels
- Allowing for more user control over data paving the way for necessary purges and safer storage
- Faster data retrieval that does not impede or slow down operations
- Streamlines data processing for performance assessment, trend analysis and forecasting reports
- Strengthens and speeds up decision making processes for both core business operations and customer relationship management

In essence, data warehousing solutions are meant to enhance data collection and integration to enable accurate and timely reporting. Since good design translates to improved information handling and management, it supports quick, efficient and informed business analysis and decision-making, which are essential to staying competitive and profitable. With such clear benefits to data warehousing, companies should commit resources and develop a strong enterprise vision to ensuring that a workable data warehouse is put into place and maintained.

13.11 Future Trends in Data Warehousing

13.11.1 Top Ten Trends in Data Warehousing

Although data warehousing has greatly matured as a technology discipline over the past ten years, enterprises that undertake data warehousing initiatives continue to face fresh challenges that evolve with the changing business and technology environment. The data warehouse is being called on to support new initiatives, such as customer relationship management and supply chain management, and has also been directly impacted by the rise of e- business. Data warehousing vendors have developed new and more sophisticated technologies and have acquired and merged with other vendors. The number of home-grown and packaged software implementations throughout the average enterprise has grown rapidly, creating more data sources and information delivery options. With all of the activity surrounding data warehousing, it is hard to sort out which issues and trends are most pressing for enterprises. To that end, this section presents insights into the ten biggest data warehousing challenges facing organizations.

Data Warehouse Do-over

Data warehousing has matured as a technology discipline and most large enterprises have completed some form of data warehousing initiative, whether it is an enterprise-wide data warehouse or just one or two departmental data marts. These initiatives have achieved varying degrees of success, and many organizations are now in the process of reengineering or even totally rebuilding their data infrastructures. According to META Group, almost one- third of data warehousing efforts through 2001 will be do-over.

What problems and challenges have made these do-over necessary? There are some common pitfalls that many enterprise data warehousing initiatives have fallen into:

- Many organizations undertake data warehousing projects with a build-it-and-they-will-come attitude. Unfortunately, this philosophy has doomed many a data warehouse to failure. Data warehousing projects need to involve end users from the beginning to ensure buy-in when the data warehouse is complete. Some organizations also fail to create the killer apps that actually deliver the benefits of the data warehouse to end users.
- Another pitfall is not architecting the data warehouse for performance, scalability and reliability. Many enterprises do not take future needs into account when building their initial data warehouse and fail to anticipate the demands of warehouse operations. They are forced to rebuild their data warehouse from the ground up when data volumes and user demands overwhelm their original systems.
- Data quality issues are often ignored in initial data warehouse implementations. Enterprises do not feel the negative impact of poor data quality until after their data warehouse is already up and running. Many are now re-examining the quality of the data in their warehouses and are undertaking the painful process of resolving data quality problems.
- Some data warehouses are unsuccessful because their sponsors did not take the time to define success at the outset of the project. According to META Group, only 40% of enterprises measure ROI for their data warehousing initiatives. Without a clear definition of success, it is hard to determine whether the data warehouse is delivering real business benefits.
- Finally, many data warehousing projects simply fall into the late-and over budget trap. Enterprises fail to anticipate the scope of their data warehousing projects and do not implement proper project planning.

The good news behind past data warehousing "failures" is that enterprises have learned from their mistakes and are developing a set of best practices as they correct the problems. This means more successful implementations in the future as newcomers to data warehousing learn from those who have been there before.

Proliferation of Data Sources

The number of enterprise data sources is growing rapidly, with new types of sources emerging every year. The most exciting new source is, of course, enterprise e-business operations. Enterprises want to integrate click stream data from their Web sites with other internal data in order to get a complete picture of their customers and integrate internal processes. Other sources for valuable data include ERP programs, operational data stores, packaged and home-grown analytic applications and existing data marts. The process of integrating these sources into one data warehouse can be complicated and is made even more difficult when an enterprise merges with or acquires another enterprise.

Enterprises also look to a growing number of external sources to supplement their internal data. These might include prospect lists, demographic and psychographic data, and business profiles purchased from third-party providers. Enterprises might also want to use an external provider for help with address verification, where internal company sources are compared with a master list to ensure data accuracy. Additionally, some industries have their own specific sources of external data. For example, the retail industry uses data from store scanners, and the pharmaceutical industry uses prescription data that is aggregated by third- party vendors.

Outsourcing

Although enterprises have not yet begun to outsource their actual data warehouses, they are outsourcing other applications and, by extension, the data used and generated by those applications. The use of outsourcing is growing rapidly. Gartner, Inc. estimates that by 2003, 45% of large enterprises will host or rent some form of business application with an application service provider (ASP). ASPs offer fast application deployment and application expertise that an enterprise might not possess. While the benefits can be great, enterprises that use ASPs must manage the risks inherent in outsourcing data. First, enterprises should make sure that their ASP is taking adequate security measures to keep data separate and private from the data of the ASP's other customers. Second, the enterprise should ensure that the ASP has experience with moving large volumes of data so that migration of data to and from the ASP will go smoothly. Third, the ASP should have proven experience in backup and recovery for the database(s) being used. Finally, enterprises should ensure that the flow of data between the enterprise's internal systems and the ASP can be kept intact.

Hub versus Relational Databases

In an effort to control costs and improve performance, enterprises are increasingly implementing data hubs in their data warehouses instead of using relational databases. Keeping data in a relational database can be quite expensive, costing three to five times more than keeping data in a no relational repository. Additionally, queries on no relational data stores can outperform queries on relational databases. In hopes of achieving these benefits, enterprises are turning to compressed flat files to replace some of their RDBMSs. Despite the performance benefits and cost-effectiveness of these data hubs, they are limited by not having SQL and are not appropriate for all situations.

Active Data Warehouses

As enterprises face competitive pressure to increase the speed of decision making, the data warehouse must evolve to support real-time analysis and action. "Active" data warehouses are one way to meet

this need. In contrast to traditional data warehouses, active data warehouses are tied closely to operational systems are designed to hold very detailed and current data, and feature shortened batch windows. And unlike most operational data stores (ODS), active data warehouses hold integrated data and are open to user queries. All of the aforementioned characteristics make active data warehouses ideal for real-time analysis and decision-making as well as automated event triggering.

With an active data warehouse, an enterprise can respond to customer interactions and changing business conditions in real time. An active data warehouse enables a credit card company to detect and stop fraud as it happens, a transportation company to reroute its vehicles quickly and efficiently or an online retailer to communicate special offers based on a customer's Web surfing behaviour. The active data warehouse's greatest benefit lies in its ability to support tactical as well as strategic decisions.

Fusion with CRM

Customer relationship management (CRM) is one of the most popular business initiatives in enterprises today. CRM helps enterprises attract new customers and develop loyalty among existing customers with the end result of increasing sales and improving profitability. A data warehouse contains the information an enterprise needs to truly understand its customers and is, therefore, increasingly looked to as a prerequisite for a successful CRM initiative.

One of the most important requirements of CRM is the integration of sales, marketing and customer care – all of these customer-facing functions must share information and work together. In the past, enterprises seldom integrated these areas, but CRM initiatives are pushing them to do so in order to better understand and serve their customers. The data warehouse is essential in this integration process, as it collects data from all channels and customer touch points, and presents a unified view of the customer to sales, marketing and customer-care employees. Software packages are increasingly reflecting the need for integration of these functional areas as demonstrated by the trend towards merging customer-care and campaign-management software.

Growing Number of End Users

As vendors make data warehousing and business intelligence tools more accessible to the masses, the number of data warehousing end users is growing rapidly. Survey.com predicts that the number of data warehouse users wills more than quadruple by 2002, with an average of 2,718 individual users and 609 concurrent users per warehouse. In addition to coping with the growth in the number of end users, data warehousing teams will need to cater to different types of end users. In a recent study, Gartner found that the use of business intelligence tools is growing most rapidly among administration and operations personnel, followed closely by executive-level personnel. These findings demonstrate that business intelligence tools have become both easier to use and more strategic. Obviously, end users will have different needs depending on their position in the company – while the business analyst needs ad hoc querying capabilities, the CEO and COO may only want static reporting.

Enterprises can handle the growing number of end users through the use of several techniques including parallelism and scalability, optimized data partitioning, aggregates, cached result sets and single-mission data marts. These techniques allow a large number of employees to concurrently access the data warehouse without compromising performance. Accommodating the different needs of various end-user groups will require as much of an organizational solution as a technical one. Data warehousing teams should involve end users from the beginning in order to determine the types of data and applications necessary to meet their decision-making needs.

More Complex Queries

In addition to becoming more numerous, queries against the data warehouse will also become more complex. User expectations are growing in terms of the ability to get exactly the type of information needed, when it is needed. Simple data aggregation is no longer enough to satisfy users who want to be able to drill down on multiple dimensions. For example, it may not be enough to deliver a regional sales report every week. Users may want to look at the data by customized dimensions – perhaps by a certain customer characteristic, a specific sales location or the time of purchase.

Users are also demanding more sophisticated business intelligence tools. According to Gartner, data mining is the most rapidly growing business intelligence technology. Other sophisticated technologies are also becoming more popular. Vendors are developing software that can monitor data repositories and trigger reactions to events on a real-time basis. For example, if a telecom customer calls to cancel his call-waiting feature, real-time analytic software can detect this and trigger a special offer of a lower price in order to retain the customer. Vendors are also developing a new generation of data mining algorithms, featuring predictive power combined with explanatory components, robustness and self-learning features. These new algorithms automate data mining and make it more accessible to mainstream users by providing explanations with results, indicating when results are not reliable and automatically adapting to changes in underlying predictive models and/or data structures.

Enterprises can handle complex queries and the demands of advanced analytic technologies by implementing some of the same techniques used to handle the increasing number of users, including parallelism. These techniques ensure that complex queries will not compromise data warehouse performance. In trying to meet end-user demands, enterprises will also need to address data warehouse availability. In global organizations, users need 24 x 7 uptime in order to get the information they need. In enterprises with moderate data volumes, high availability is easily implemented with high redundancy levels. In enterprises with large data volumes, however, systems must be carefully engineered for robustness through the use of well-designed parallel frameworks.

Integrated Customer View

Obtaining a 360 degree view of the customer is rapidly becoming the single most popular rationale for large-scale data warehousing efforts. Enterprises want to get a complete picture of each customer across all channels and all lines of business. While this sounds like a simple concept, it can be very difficult to implement. Many enterprises have historically been organized around products, geographies or other business-related dimensions, and their IT systems reflect this. Moving to a customer-centric view requires a big change in the way they collect, store and disseminate information. Enterprises have to integrate the proliferating data sources previously mentioned and must be sure to handle data quality issues so that customers are represented accurately across all systems.

Exploding Data Volumes

One of the biggest technology issues facing enterprises today is the explosion in data volumes that is expected to occur over the next several years. According to Gartner, in 2004 enterprises will be managing 30 times more data than in 1999. And Survey.com found that the amount of usable data in the average data warehouse will increase 290% to more than 1.2 terabytes in 2002. E-business is one of the primary culprits in the data explosion, as click stream data is expected to quickly add terabytes to the data warehouse. As the number of other customer contact channels grows, they add even more data. Escalating end-user demands also play a part, as organizations collect more information and store it for longer periods.

The data explosion creates extreme scalability challenges for enterprises. A truly scalable data warehouse will allow an enterprise to accommodate increasing data volumes by simply adding more hardware. Scalable data warehouses typically rely on parallel technology frameworks. Fortunately, lower hardware costs are making parallel technology more accessible. Distributed memory parallel processor (DMPP) hardware is becoming less expensive, and alternatives to DMPP are also improving server clustering (of SMPs) is evolving as a substitute. Additionally, storage costs continue to decline every year, making it possible for enterprises to keep terabytes of detailed historical data.

13.12 Data Verification and Data Validation

13.12.1 Data Validation

Data validation deals with making sure the data is valid (clean, correct and useful). Data validation procedures use data validation rules (or check routines) to ensure the validity (mostly correctness and meaningfulness) of data. It also ensures the validity of input data to maintain the security of the system. These rules are automatically implemented through data dictionaries. Data validation can also be implemented through declaring data integrity rules or procedures enforcing business rules (especially in business applications). These business rules are usually captured during the initial business requirements analysis conducted by the business analysts. It is very important to implement business rules at the beginning of the process, because wrongly validated data usually have a negative impact on the execution of the business process.

The simplest form of validation is checking the input to make sure they are made up of characters from the "valid" set. For example, a validation process for telephone directory application should validate the input telephone numbers to make sure that they contain only numbers, plus/minus symbols and brackets (and nothing else). A little more advanced validation processes could also check the country code field to check if they are legitimate country codes.

13.12.2 Data Verification

Data verification is the process of checking a copy of data to make sure that it is exactly equal to the original copy of the data. Data verification is usually required when you have backed up your data. Most modern backup software has in-built verification functionality. Even, disc burning software allows you to perform verification at the end of the burning process. If the data on the burned disc is verified then you are fine. But if not, you have to throw away that disc and burn again. Data verification is a very important process as it makes you feel safe because you will be confident that you can actually use the backed up data in case the original data gets lost or corrupted. Verification software usually makes sure the copy is readable as well as the content is matched exactly with the original content. So, it takes more time than a simple backup, but it is well worth the trouble. But typically large enterprises perform the automatic backups at night, so the lengthening of time due to the verification process is not a serious problem.

13.13 EDP Organization

The history of computer cantered fraud shows that the persons responsible for frauds in many situations set up the system and control its use as programmer and operator.

The number of personnel and the organizational structure will of course determine the extent to which segregation of duties is possible. As a minimum, the function of programming should be separated from the functions controlling input to the computer programs, and the function of the

computer operator should be segregated from functions requiring de-tailed knowledge or custody of the computer programs. If one person is permitted to perform duties in several of these functions, internal control is weakened, and the opportunity exists for fraudulent data to be inserted in the system.

The rapid growth of electronic data processing (EDP) for business use is having a greater impact on public accounting than perhaps any other event in the history of the profession. No longer is the challenge of auditing EDP activities limited to a few large clients. With the advent of inexpensive minicomputer systems and PC Network, even the smallest audit clients are likely to use a computer for many accounting functions. Thus, auditors must be prepared to work in an ever-changing environment in which the client's accounting records are maintained on anything from a personal computer to a multimillion dollar mainframe system.

Did You Know?

The first commercial business computer was developed in the United Kingdom in 1951, by the J. Lyons and Co.

Self Assessment Questions

4.....is a communication function which transmits the information, generated after processing of data, to persons who need the information.

(a) Input (b) Output

(c) Processing (d) Storage

5..... is the process of converting one form of a data into another.

(a) Database management (b) Data entry

(c) Data processing (d) Data conversion

6. The.....capability is a major use of database management system.

(a) database maintenance (b) application development

(c) database interrogation (d) database development

13.14 Data centre

Data centres form the backbone of a wide variety of services offered via the Internet including Web-hosting, e-commerce, social networking, and a variety of more general services such as software as a service (SAAS), platform as a service (PAAS), and grid/cloud computing. Some examples of these generic service platforms are Microsoft's Azure platform, Google App engine, Amazon's EC2 platform and Sun's Grid Engine. Virtualization is the key to providing many of these services and is being increasingly used within data centres to achieve better server utilization and more flexible resource allocation. However, virtualization also makes many aspects of data centre management more challenging.

As the complexity, variety, and penetration of such services grows, data centres will continue to grow and proliferate. Several forces are shaping the data centre landscape and we expect future data centres to be lot more than simply bigger versions of those existing today. Data centres into distributed, virtualized, multi-layered infrastructures that pose a variety of difficult challenges.

In particular, we consider a layered model of virtualized data centres and discuss storage, networking, management, and power/thermal issues for such a model. Because of the vastness of the space, we

shall avoid detailed treatment of certain well researched issues. In particular, we do not delve into the intricacies of virtualization techniques, virtual machine migration and scheduling in virtualized environments.

13.15 Evolution of Data Centre

Traditional data centres have evolved as large computational facilities solely owned and operated by a single entity commercial or otherwise. However, the forces in play are resulting in data centres moving towards much more complex ownership scenarios. For example, just as virtualization allows consolidation and cost savings within a data centre, virtualization across data centres could allow a much higher level of aggregation. This notion leads to the possibility of ‘‘out-sourced” data centres that allows an organization to run a large data centre without having to own the physical infrastructure. Cloud computing, in fact, provides exactly such a capability except that in cloud computing the resources are generally obtained dynamically for short periods and underlying management of these resources is entirely hidden from the user. Subscribers of virtual data centres would typically want longer-term arrangements and much more control over the infrastructure given to them. There is a move afoot to provide Enterprise Cloud facilities whose goals are similar.
In the following we present a four layer conceptual model of future data centres shown in Figure 13.15 that subsumes a wide range of emergent data centre implementations. In this depiction, rectangles refer to software layers and ellipses refer to the resulting abstractions.

The bottom layer in this conceptual model is the Physical Infrastructure Layer (PIL) that manages the physical infrastructure (often known as ‘‘server farm”) installed in a given location. Because of the increasing cost of the power consumed, space occupied, and management personnel required, server farms are already being located closer to sources of cheap electricity, water, land, and manpower. These locations are by their nature geographically removed from areas of heavy service demand, and thus the developments in ultra high-speed networking over long distances are essential enablers of such remotely located server farms. In addition to the management of physical computing hardware, the PIL can allow for larger-scale consolidation by providing capabilities to carve out well-isolated sections of the server farm (or ‘‘server patches”) and assign them to different ‘‘customers.” In this case, the PIL will be responsible for management of boundaries around the server patch in terms of security, traffic firewalling, and reserving access bandwidth. For example, set up and management of virtual LANs will be done by PIL.

The next layer is the Virtual Infrastructure Layer (VIL) which exploits the virtualization capabilities available in individual servers, network and storage elements to support the notion of a virtual cluster, i.e., a set of virtual or real nodes along with QoS controlled paths to satisfy their communication needs. In many cases, the VIL will be internal to an organization who has leased an entire physical server patch to run its business. However, it is also conceivable that VIL services are actually under the control of infrastructure provider that effectively presents a virtual server patch abstraction to its customers. This is similar to cloud computing, except that the subscriber to a virtual server patch would expect explicit SLAs in terms of computational, storage and networking infrastructure allocated to it and would need enough visibility to provide its own next level management required for running multiple services or applications.

The third layer in our model is the Virtual Infrastructure Coordination Layer (VICL) whose purpose is to tie up virtual server patches across multiple physical server farms in order to create a

geographically distributed virtualized data centre (DVDC). This layer must define and manage virtual pipes between various virtual data centres. This layer would also be responsible for cross-geographic location application deployment, replication and migration whenever that makes sense. Depending on its capabilities, VICL could be exploited for other purposes as well, such as reducing energy costs by spreading load across time-zones and utility rates, providing disaster or large scale failure tolerance, and even enabling truly large-scale distributed computations.

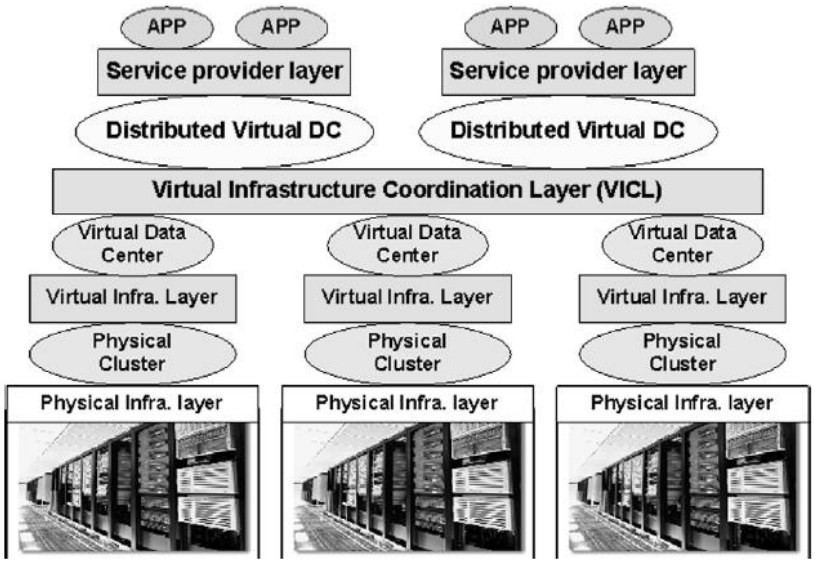


Figure 13.15: Logical organization of future data centres.

Finally, the Service Provider Layer (SPL) is responsible for managing and running applications on the DVDC constructed by the VICL. The SPL would require substantial visibility into the physical configuration, performance, latency, availability and other aspects of the DVDC so that it can manage the applications effectively. It is expected that SPL will be owned by the customer directly.

The model in Figure 13.15 subsumes everything from a non virtualized, single location data centre entirely owned by a single organization all the way up to a geographically distributed, distributed, fully virtualized data centre where each layer possibly has a separate owner. The latter extreme provides a number of advantages in terms of consolidation, agility, and flexibility, but it also poses a number of difficult challenges in terms of security, SLA definition and enforcement, efficiency and issues of layer separation. For this reason, real data centres are likely to be limited instances of this general model.
In subsequent sections, we shall address the needs of such DVDC's when relevant, although many of the issues apply to traditional data centres as well.

Exercise: Check Your Progress 2
Note: i) Use the space below for your answer.
Ex1: Evolution of Data Centers.

……

……

……

13.16 Requirements for Modern Data

Businesses and other organizations of all sizes rely heavily on their data centre in order to run their operations. In this section, we shall take a look at seven of the most crucial requirements of modern data centres.

1. Continuity

For many organizations, if a system was to become unavailable even for a short period of time, it could have devastating effects on their ability to function, with costs potentially running into the millions. As a result, one of the most important factors to consider in the modern data centre is continuity. A data centre must minimize or, ideally, eliminate the potential for downtime for example with emergency backup power generation should the data centre suffers a power outage.

2. Security

Another key concern is security. With so much critical and often classified information being stored on data centres, it is important that this information is protected from the threats of unauthorized eyes. Another security concern relates to the threat of an unexpected disaster such as fire or flood. Data centres should have backup options available should the system fail, and should remain secure at all times.

3. Individual or co-located?

Data centres can be split into two categories. The first category involves data centres that serve the needs of an individual company (a large data centre located on the premises of a large business, for example, and which has been customized to suit that business's purposes). The second involves data centre services where equipment and bandwidth can be rented by many different customers or businesses. This is known as a "co-located" data centre.

4. Low costs

With the data centre such a crucial part of so many businesses, it's obvious that the costs associated with installing, maintaining and upgrading a data centre are going to be significant. As a result, any steps that can be taken to lower the costs of data centres can result in huge savings for a company savings that have the potential to put them at a competitive advantage against other organizations.

5. Environmentally friendly

Data centres are one of the most energy-intensive parts of any organization. In fact, data centres alone make up about 2% of the world's annual electricity bill. Because data centres rely so heavily on energy, it means that their carbon footprint is quite high. With governments around the world taking steps to minimize carbon emissions, companies are now expected to find ways to lower the energy output of their data centres. This can have tremendous cost savings as well, both in terms of electricity bill spending and also through avoiding impending government carbon taxes.

6. Scalability

When a data centre is being installed, one of the most important considerations is its scalability. After all, as a company grows, data centres may reach capacity, at which point the data centre will need to be upgraded in order for an organization to continue its growth. Scalability must be planned for from the outset, otherwise companies may find themselves needing to replace their data centre altogether, which can be quite costly.

7. Automation

Data centres that can almost eliminate the need to be accessed by IT personnel (except under special circumstances) are known as "dark data centres", and these can had tremendous benefits for the business. Many of the maintenance processes associated with such data centres are automated, saving the cost of man hours and also saving electricity due to eliminating the need for lighting.

Self Assessment Questions

7. The.....process is accomplished by transaction processing programs and other end-user application packages within the support of the database management system.

(a) database maintenance (b) application development

(c) database interrogation (d) database development

8. The EDP programs and data files cannot be changed without the use of EDP equipment.

(a) True (b) False

9. The DBMS can maintain the integrity of the database by allowing more than one user to update the same record at the same time.

(a) True (b) False

10. Data centres are not one of the most energy-intensive parts of any organization.

(a) True (b) False

13.17 Summary

- Data processing is change of one format of a data into another format for better maintenance as well as effective analyzes and study.
- Data management is a discipline that focuses on the proper generation, storage, and retrieval of data.
- Data processing is a process of converting data into the information as well as it can convert information into a data.
- The data management system is the set of procedures and people through which information is processed.
- In mechanical data processing data is processed with the help of devices or machines.
- Data verification is the process of checking a copy of data to make sure that it is exactly equal to the original copy of the data.
- Data validation procedures use data validation rules (or check routines) to ensure the validity (mostly correctness and meaningfulness) of data.

13.18 Keywords

Customer Relationship Management (CRM): Customer relationship management is a widely implemented strategy for managing a company's interactions with customers, clients and sales prospects. It involves using technology to organize, automate, and synchronize business processes principally sales activities, but also those for marketing, customer service, and technical support.

Data Definition Languages (DDL): A DDL is a language used to define data structures within a database. It is typically considered to be a subset of SQL, the Structured Query Language, but can also refer to languages that define other types of data.

Data Processing: Data Processing System is a system which processes data which has been captured and encoded in a format recognizable by the data processing system or has been created and stored by another unit of an information processing system.

Database Maintenance: Database maintenance is an activity which is designed to keep a database running smoothly.

***Database Management System (DBMS)*:** A Database Management System is computer software designed for the purpose of managing databases based on a variety of data models.

Electronic Data Processing (EDP): Electronic data processing is use of computers in recording, classifying, manipulating, and summarizing data.

13.19 Review Questions

1. What is the electronic data processing?
2. Explain the data processing system.
3. Differentiate between data and information.
4. What are the future trends in data warehousing?
5. Describe the data processing cycle.
6. What are the need and approaches of data processing?
7. Discuss the types of data processing.
8. What is the data management system?
9. What are the database management systems?
10. Discuss the data verification and data validation.

Answers for Self Assessment Questions

1 (d)	2 (c)	3 (a)	4 (b)	5 (d)
6 (c)	7 (a)	8 (a)	9 (b)	10 (b)

13.20 Further Studies

1. http://books.google.co.in/books?id=KjG6UiiQxm8C&pg=PR13&dq=Electronic+Data+Processing+in+SAD&hl=en&sa=X&ei=lbnCUvSgOsaZiAfGioHQDg&ved=0CEEQ6AEwBA#v=onepage&q=Electronic%20Data%20Processing%20in%20SAD&f=false

OOP's Programming Using C++

(MCA/PGDCA - 108)



Jaipur National University
Directorate of Distance Education

Established by Government of Rajasthan
Approved by UGC under Sec 2(f) of UGC ACT 1956
(**Recognised by Joint Committee of UGC-AICTE-DEC, Govt. of India**)

1

Overview of C++

STRUCTURE

1.0 Objectives

1.1 Introduction

1.2 Software Crisis

1.3 Object-oriented Programming Paradigm

1.4 Basic Concepts of Object-oriented Programming

1.5 Advantages/Benefits of OOP

1.6 Application of OOP

1.7 Summary

1.8 Keywords

1.9 Review Questions

1.10 Further Studies

1.0 Objectives

After studying this chapter, you will be able to:

- Discuss the software crisis
- Explain the object-oriented programming paradigm
- Discuss the basic concepts of OOP
- Discuss the advantages/benefits of OOP
- Define the application of OOP

1.1 Introduction

A computer is a machine that receives instructions and produces a result after performing an appropriate assignment. Since it is a machine, it expects good and precise directives in order to do something. The end result depends on various factors ranging from the particular capabilities of the machine, the instructions it received, and the expected result.



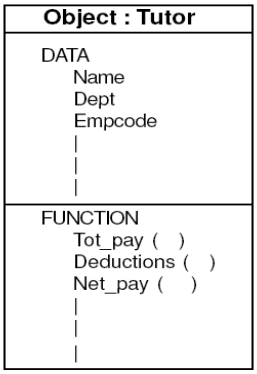
Computer programming is the art of writing instructions (programs) that ask the computer to do something and give a result. A computer receives instructions in many different forms.

Some of the operating systems on the market are: Microsoft Windows 3.X, Corel Linux, IBM OS\2, and Microsoft Windows 9X, Apple OS 10, Red Hat Linux, Microsoft Windows Millennium, BeOS, Caldera Linux, and Microsoft Windows 2000 etc. A particular OS (for example Microsoft Windows 98) depending on a particular processor (for example Intel Pentium) is sometimes referred to as a platform. Some of the computer languages running on Microsoft Windows operating systems are C++, Pascal, Basic, and their variants.

There are various computer languages, for different reasons, capable of doing different things. Fortunately, the computer can distinguish between different languages and perform accordingly. These instructions are given by the programmer who is using compilers, interpreters, etc, to write programs. Examples of those languages are Basic, C++, Pascal, etc.

C++ is an extension to C programming language. It was developed at AT&T Bell Laboratories in the early 1980s by Bjarne Stroustrup. It is a deviation from traditional procedural languages in the sense that it follows object-oriented programming (OOP) approach which is quite suitable for managing large and complex programs.

An object-oriented language combines the data to its function or code in such a way that access to data is allowed only through its function or code. Such combination of data and code is called an object. For example, an object called Tutor may contain data and function.



The data part contains the Name, Dept and Employee code. The function part consists of three functions: To_pay ( ), Deductions ( ) and Net_pay ( ).

1.2 Software Crisis

Developments in software technology continue to be dynamic. New tools and techniques are announced in quick succession. This has forced the software industry and software engineers to continuously look for new approaches to software design and development, which is becoming more

and more critical in view of the increasing complexity of software systems as well as the highly competitive nature of the industry.

The following issues needed to be resolved to overcome this crisis:

- How to represent real-life entities of problems in system-design?
- How to design systems with open interfaces?
- How to ensure reusability and extensibility of modules?
- How to develop modules that are tolerant to any change in future?
- How to improve software productivity and decrease software cost?
- How to manage time schedules?
- How to improve the quality of software?
- How to industrialize the software development process?

Many software products are either not finished, or not used, or else are delivered with major errors. Figure 1.1 shows the fate of the defence software projects undertaken in the 1970s, around 50% of the software products were never delivered, and one-third of those which were delivered were never used. It is interesting to note that only 2% were used as delivered, without being subjected to any changes. This illustrates that the software industry has a remarkably bad record in delivering products.

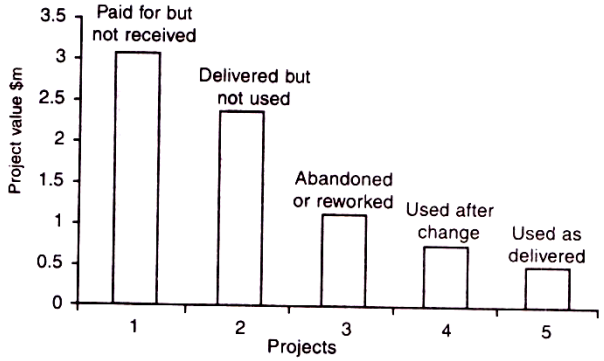


Figure 1.1: The state of defence projects.

Changes in user requirements have always been a major problem. Another study (Figure 1.2) shows that more than 50% of the systems required modifications due to changes in user requirements and data formats. It only illustrates that, in a changing world with a dynamic business environment, requests for change are unavoidable and therefore systems must be adaptable and tolerant to changes.

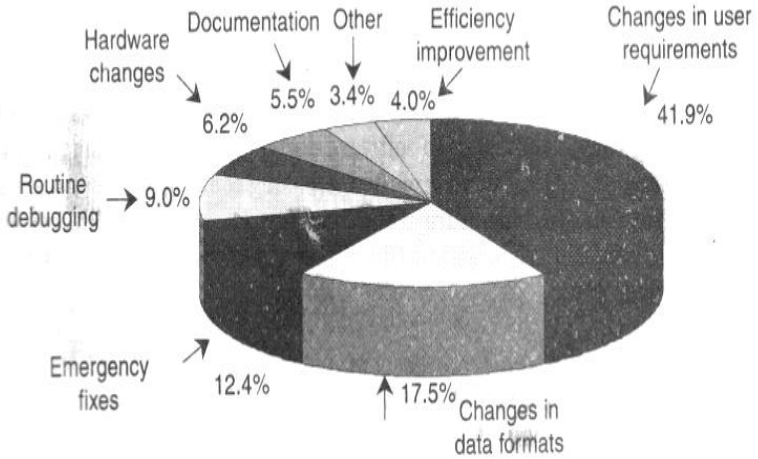


Figure 1.2: Breakdown of maintenance costs.

Problems occur when software products are either not finished, not used or are delivered with errors. Changes in user requirements have always been a major problem. The reports on software implementation suggest that software products should be evaluated carefully for their quality before they are delivered and implemented.
Some of the quality issues that must be considered for critical evaluation are:

- Correctness
- Maintainability
- Reusability
- Openness and interpretability
- Portability
- Security
- Integrity
- User friendliness

Selection and use of proper software tools would help resolving some of these issues.

Did You Know?
The term "software crisis" was coined by some attendees at the first NATO Software Engineering Conference in 1968 at Garmisch, Germany.

1.3 Object-oriented Programming Paradigm

The major motivating factor in the invention of object-oriented approach is to salvage some of the flaws encountered in the procedural approach. The OOP treats data as a critical element in the program development and does not allow it to flow freely around the system. It ties data more closely to the functions that operate on it and protects it from accidental modification from outside functions. The OOP allows us to decompose a problem into a number of entities called objects and then builds data and functions around these entities.
The organization of data and functions in object-oriented programs is shown Figure 1.3.

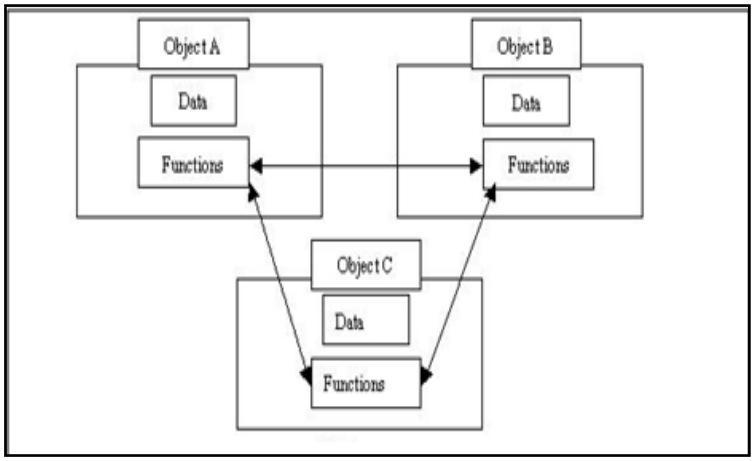


Figure 1.3: Organization of data and functions in OOP.

The data of an object can access the functions of other objects. However, functions of one object can access the functions of other objects.
Some of the striking features of object-oriented programming are:

- Emphasis is on data rather than procedure.
- Programs are divided into what are known as objects.

- Data structures are designed such that they characterize the objects.
- Functions that operate on the data of an object are tied together in the data structure.
- Data is hidden and cannot be accessed by external functions.
- Objects may communicate with each other through functions.
- New data and functions can be easily added whenever necessary.
- Follows bottom-up approach in program design.

Object-oriented programming is the most recent concept among programming paradigms is means different things to different people. It is therefore important to have a working definite object-oriented programming before we proceed further. Our definition of object-oriented programming is as follows "Object-oriented programming is an approach that provides a way of modelling programs by creating partitioned memory area for both data, and functions that can be used as templates for creating copies of such modules on demand."

That is, an object is considered to be a partitioned area of computer memory that stores data and set of operations that can access that data. Since the memory partitions are independent, the objects are used in a variety of different programs without modifications.

Object-oriented programming (OOP) have taken the best ideas of structured programming and combined them with several powerful new concepts that encourage the approach the task of programming in a new way. In general when programming in an object-oriented fashion you break down a problem into subgroup of related parts that take into account both code and data related to each group. Also, you organize this subgroup into a hierarchical for all intents and purpose, an object in a variable of area-defined type. It may seem strong at first to think of an object, which lines both code and data on variable. However, in object-oriented programming, this is precisely the case. When you define an object, you are implicitly creating a new data type.

Did You Know?

In the 1970s, Kay's Smalltalk work had influenced the Lisp community to incorporate object based techniques that were introduced to developers via the Lisp machine.

1.4 Basic Concepts of Object-oriented Programming

It is therefore necessary to understand some of the concepts used extensively in object-oriented programming.

These include:

1. Objects
2. Classes
3. Data abstraction
4. Inheritance
5. Dynamic binding
6. Data encapsulation
7. Polymorphism
8. Message passing

1.4.1 Objects

Objects are the basic run-time entities in an object-oriented system. They may represent a person, i.e. a bank account, a table of data or any item that the program must handle. They may also resent user-defined data such as vectors, time and lists. Programming problem is analyzed in terms of objects and

the nature of communication between them. Program objects should be chosen such that they match closely with the real-world objects. As pointed out earlier, objects take up space in the memory and have an associated address like a record in Pascal, or a structure in C.

When a program is executed, the objects interact by sending messages to one another. For example if "customer" and "account." are two objects in a program, then the customer object may send a message to the account object requesting for the bank balance. Each object contains data and code to manipulate the data. Objects can interact without having to know details of each other's data or code. It is sufficient to know the type of message accepted and the type of response returned by the objects. Although different authors represent them differently, Figure 1.4 shows two notations that are properly used in object-oriented analysis and design.

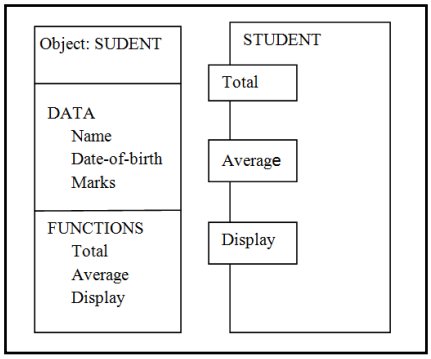


Figure 1.4: Two ways of representing an object.

1.4.2 Classes

We just mentioned that objects contain data and code to manipulate that data. The entire set of data and code of an object can be made a user-defined data type with the help of a class. In fact, objects are variables of type class. Once a class has been defined, we can create any number of objects belonging to that class. Each object is associated with the data of type class with which they created. A class is thus a collection of objects of similar type. For example, mango, apple and orange are members of the class fruit. Classes are user-defined data types and behave like the built-in types of a programming language. For example, the syntax used to create an object is no different than the syntax used to create an integer object in C.

If fruit has been defined as a class, then the statement fruit mango; will create an object mango belonging to the class fruit has been define as a class than the statement
Fruit mango; Will create an object mango belonging to the class fruit

Caution

Be careful not introduce local variables with the same names as the instance fields in the class. For example, the following constructor will not set the salary.

```
public Employee(String n, double s, . . .)
{
  String name = n; // ERROR
  double salary = s; // ERROR
  . . .
}
```

1.4.3 Data Abstraction

The wrapping up of data and functions into a single unit (called class) is known as encapsulation. Data encapsulation is the most striking feature of a class. The data is not accessible to the outside world and only those functions which are wrapped in the class can access it. These functions provide the interface between the objects data and the program. This insulation of the data from direct access by the program is called “data hiding” or “information hiding”.

Abstraction refers to the act of representing essential features without including the background details or explanations. Classes use the concept of abstraction and are defined as a list of abstract attributes such as size, weight and cost, and functions to operate on these attributes. They encapsulate al1 the essential properties of the objects that are to be created. The attribute are something called data member because they hold information. The function that operates on these data sometime called methods or member functions.Since the classes use the concept of data abstraction, they are known as Abstract Data Types (ADT).

1.4.4 Inheritance

Inheritance is the process by which objects of one class acquire the properties of objects of another class. It supports the concept of hierarchical classification. For example, the bird robin is a part the class flying bird which is again a part of the class bird. The principle behind this sort of division is that each derived class shares common characteristics with the class from which it is derived. As illustrated in Figure 1.5.

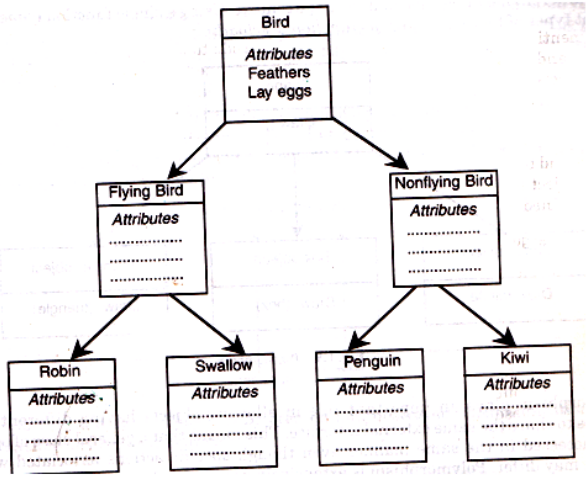


Figure 1.5: Property inheritance.

In OOP, the concept of inheritance provides the idea of reusability. This means that we can add additional features to an existing class without modifying it. This is possible by deriving a new class from the existing one. The new class will have the combined features of both the classes. The real appeal and power of the inheritance mechanism is that it allows the programmer to reuse a class that is almost, but not exactly, what he wants, and to tailor the class in such a way that it does not introduce any undesirable side effects into the rest of the classes.

Note that each sub-class defines only those features that are unique to it. Without the use of classification, each class would have to explicitly include all of its features.

1.4.5 Dynamic Binding

Binding refers to the linking of a procedure call to the code to be executed in response to the call. It is associated with polymorphism and inheritance. A function call associated with a polymorphic reference depends on the dynamic type of that reference.

Every object will have this procedure. Its algorithm is, however, unique to each object and so the draw procedure will be redefined in each class that defines the object. At run-time, the code matching the object under current reference will be called.

1.4.6 Data Encapsulation

Encapsulation in the mechanism that binds together code and data and that leaps both safe from outside interference. It also allows the creation of an object. More simply, an object in a logical entity that encapsulate both data and the code that manipulators that data.

Within an object, some of the code and/or data may be private to the objected and in accessible to anything outside the object. In this way and object provides a significant level of protection against some other unrelated part of the program accidentally modifying or incorrectly using the private parts of the object.

1.4.7 Polymorphism

Polymorphism is another important OOP concept. Polymorphism means the ability to take more than one form. For example, an operation may exhibit different behaviour in different instances. The behaviour depends upon the types of data, used in the operation. For example, consider the operation of addition. For two numbers, the operation will generate a sum. If the operands are strings, then the operation would produce a third string by concatenation.

The Figure 1.6 illustrates at a single function name can be used to handle different number and different types of arguments. This is something similar to a particular word having several different meanings depending on the context.

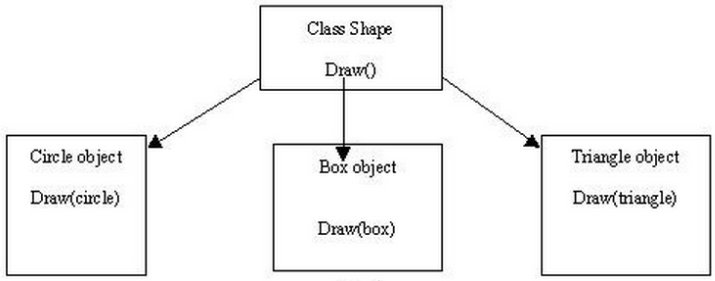


Figure 1.6: Property polymorphism.

Polymorphism plays an important role in allowing objects having different internal structures to share the same external interface. This means that a general class of operations may be accessed in the same manner even through specific actions associated with each operation may differ. Polymorphism is extensively used in implementing inheritance. Object-oriented programming languages support polymorphism, which in characterized by the phase “on interface multiple method”. In simple terms, polymorphism in an attribute that allows one interface to be used with a general class of actions. Polymorphism helps in reducing complexity by allowing the same interface to specify a general class of action. It is compiler’s job to select the “specify action” on it applies to each situation. The programmers do not need to make this selection manually operator, overloading, function, overloading

and overlooking example of polymorphism structure. Finally you translate these subgroups self-contained units called object.

In a multi-function program, many important data items are placed as global so that they may be accessed by all the functions. Each function-may have its own local data. Global data are more vulnerable to an inadvertent change by a function. In a large program it is very difficult to identify what data is used by which function. In case we need to revise an external data structure, we should also revise all functions that access the data. This provides an opportunity for bugs to creep in.
Another serious drawback with the procedural approach is that it does not model real world problems very well. This is because functions are action-oriented and do not really corresponding to the elements of the problem.
Some characteristics exhibited by procedure-oriented programming are:

- Emphasis is on doing things (algorithms).
- Large programs are divided into smaller programs known as functions.
- Most of the functions share global data.
- Data move openly around the system from function to function. Functions transforms data from one form to another

Employs top-down approach in program design.

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: Create a structure of polymorphism in real world.

……..……

……..……

……...

1.4.8 Message Communication
An object-oriented program consists of a set of objects that communicate with each other. The process of programming in an object-oriented language therefore involves the following basic steps:
1. Creating classes that define objects and their behaviour.
2. Creating objects from class definitions.
3. Establishing communication among objects.

Objects communicate with one another by sending and receiving information much the same way as people pass messages to one another. The concept of message passing makes it easier to talk about building systems that directly model or simulate their real-world counterparts.
A message for an object is a request for execution of a procedure, and therefore will invoke a function (procedure) in the receiving object that generates the desired result. Message passing involves specifying the name of the object, the name of the function (message) and the information to be sent.
Example
Objects have a life cycle. They can be created and destroyed. Communication with an object is feasible as long as it is alive.

Did You Know?
The terms “objects” and “oriented” in something like the modern sense of object-oriented programming seem to make their first appearance at MIT in the late 1950s and early 1960s.

Self Assessment Questions

1. ……..are divided into what are known as objects.
(a) Programs (b) Emphasis
(c) Data (d) Objects

2. ……..are designed such that they characterize the objects.
(a) Programs (b) Emphasis
(c) Data structures (d) Objects

3. ……..are the basic run-time entities in an object-oriented system.
(a) Inheritance (b) Data abstraction
(c) Data abstraction (d) Objects

4…… is the process by which objects of one class acquire the properties of objects of another class.
(a) Inheritance (b) Data abstraction
(c) Data abstraction (d) Objects

5. Major features that are not required for object-based programming are:
(a) Data encapsulation
(b) Data hiding and access mechanisms
(c) Software complexity can be easily managed
(d) Operator overloading

1.5 Advantages/Benefits of OOP

The OOP offers several benefits to both the program designer and the user. Object-orientation contribution to the solution of many problems associated with the development and quality of software products. The new technology promises greater programmer productivity, better quality of software and lesser maintenance cost.

The principal advantages are:

1. Through inheritance, we can eliminate redundant code and extend the use of existing classes.
2. We can build programs from the standard working modules that communicate with one another, rather than having to start writing the code from scratch. This leads to saving of development time and higher productivity.
3. The principle of data hiding helps the programmer to build secure programs that cannot be invaded by code in other parts of the program.
4. It is possible to have multiple instances of an object to co-exist without any interference.
5. It is possible to map objects in the problem domain to those objects in the program.
6. It is easy to partition the work in a project based on objects.
7. The data-cantered design approach enables us to capture more details of a model in implementable form.
8. Object-oriented systems can be easily upgraded from small to large systems.
9. Message passing techniques for communication between objects makes the interface descriptions with external systems much simpler.
10. Software complexity can be easily managed.

While it is possible to incorporate all these features in an object-oriented system, their importance their importance depends on the type of the project and the preference of the programmer. For instance, object libraries must be available for reuse. The technology is still developing and current products may be superseded quickly. Strict controls and protocols need to be developed if reuse is not to be compromised.
Developing software that is easy to use makes it hard to build. It is hoped that the object-oriented programming tools would help manage this problem.

1.5.1 Object-oriented Languages

Object-oriented programming is not the right of any particular language. Like structured programming, OOP concepts can be implemented using languages such as C and Pascal. However, programming becomes clumsy and may generate confusion when the programs grow large. A language that is specially designed to support the OOP concepts makes it easier to implement them.
The languages should support several of the OOP concepts to claim that they are object-oriented.
Depending upon the features they support, they can be classified into the following two categories:
1. Object-based programming languages
2. Object-oriented programming languages

Object-based programming is the style of programming that primarily supports encapsulation and object identity.
Major features that are required for object-based programming are:

- Data encapsulation
- Data hiding and access mechanisms
- Automatic initialization and clear-up of objects
- Operator overloading

Languages that support programming with objects are said to be object-based programming languages. They do not support inheritance and dynamic binding.
Object-oriented programming incorporates all of object-based programming features along with two additional features, namely, inheritance and dynamic binding.

1.6 Application of OOP

The Promising areas for application of OOPs include:

1. Real-time systems
2. Simulation and modelling
3. Object-oriented databases
4. Hypertext, hypermedia and expertext
5. AI and expert systems
6. Neural networks and parallel programming
7. Decision support and office automation systems
8. CIM/CAM/CAD systems

The object-oriented paradigm sprang from the language, has matured into design, and has recently moved into analysis. It is believed that the richness of OOP environment will enable the software industry to improve not only the quality of software systems but also its productivity. Object-oriented technology is certainly going to change the way the software engineers think, analyze, design and implement future systems.

Caution

Be careful before installing C++ software in your computer, you should install only supported C++ version in your running operating system.

Exercise: Check Your Progress 2

Case Study-Dedicated Developers and Programmers as ODC - Offshore Development Centre

Client is in to the development of specialized systems software in niche market. Since years of consistent development with high grade enterprise solutions for different industries are carried out by our perseverance therefore we are the right choice for the client.

Challenge

The client really had a concern over the security of the data regarding the maintenance of the quality on the development and deliverance on time where some of the major concerns like the cost and of course the management of the staff assigned for the project.

An array of projects that needs complex skills in C, C++ and good understanding of windows operating systems from 98 and till Vista or the 64 bit applications as a modest requirement.

While our initial discussions focused more on the solution needed for bringing forth their existing and functional solutions in to the future i.e. 64 bit operating system, the real aspiration was to benefit from the cost savings and eventually making the team in India to share some of their loads.

Solution

We provide them with the option to hire our resources on 'Time' and 'Resource' basis, under the contract we would depute full time developers and assist them with all kinds of technical support from our senior management. The other options include 'ODC Offshore Development Centre' and ODC with "Build Operate Transfer". The Client had opted for us to setup a dedicated "Offshore Development Centre" for them.

The project management happens from the Client's office and all the Project Managers operate from the Client divisions and most of them have never put their foot in India. It is the beginning of the process which would later on to be considered in parallel with Change Management.

The senior team provides approximately a man-week of free support to the team working in the office for the client while the developers took ideas and understanding of the project on a regular basis from the Project Managers in Client's country. Their main support is to come from Technical Head of Radix.

In order to ensure the smoothness in project coordination an experienced project coordinator has been appointed for the task. Though the technical communicator works only part time between the teams but the objective of having a proper and smoother communication is always unwavering.

Technical Supremacy

As the project execution is in turn then the management from the elite core of both Radix and Client and the developers themselves are better off cadre in C, C++ programming, combined experience, knowledge and efforts. Client is now supposed to leverage up on a new level of technical superiority in the market.

Risks

The pricing offered to client is very competitive and involved the salary and operational cost of Radix and the payments again has to be made on the month end. Project execution calls for collective skills

much in demand is that of developing skills, analytical skills and important among them as quality control aspects.
Single resource would be able to carry out all of the demands; however task bifurcation is imminent demand. Or else the efficient output of the resources could not be maintained.

Maintaining Quality and Efficient Output
In order to manage the demands while maintaining efficiency and quality the project "Work Breakdown Structure (WBS)" is created with clear set of 'Roles and Responsibilities' identified and escalation matrix drafted out.

Cost Benefits
Such procedures go on to help Client utilize same set of resources in a clearly productive manner with least of hassles that would have otherwise call for the additional resource hiring. Therefore client achieved significant cost savings even while outsourcing the project, and building the dedicated "Offshore Development Centre".
Build Operate and Transfer
Based on the success of such contracts this setup can eventually migrate to BOT and a company to mirror development operations in India.

Questions
1. Describe the role of developers and programmers in Offshore Development Centre.
2. Describe the Offshore Development Centre.

1.7 Summary

- Polymorphism means one name, multiple forms. It allows us to have more than one function with the same name in a program.
- Dynamic binding means that the code associated with a given procedure is not known until the time of the call at run-time.
- Message passing involves specifying the name of the object, the name of the function (message) and the information to be sent.
- Object-oriented technology offers several benefits over the conventional programming methods the most important one being the reusability.
- Applications of OOP technology has gained importance in almost all areas of computing including real-time business systems.

1.8 Keywords

Assembly Language: An assembly language is a low-level programming language for computers, microprocessors, microcontrollers, and other programmable devices in which each statement corresponds to a single machine language instruction.
Function Overloading: Function overloading is one of the most powerful features of C++ programming language. It forms the basis of polymorphism (compile-time polymorphism).
Machine language: Machine language is the programming language the computer understands; its native tongue. Machine language instructions are written with binary numbers, 0 and 1.
Object-oriented Programming: Object-oriented programming (OOP) is a programming paradigm using objects data structures consisting of data fields and methods together with their interactions to

design applications and computer programs.

Polymorphism: Polymorphism is a programming language feature that allows values of different data types to be handled using a uniform interface.

1.9 Review Questions

1. What are the major issues software industry facing today?
2. What is procedure-oriented programming? What are its main characteristics?
3. Discuss an approach to the development of procedure-oriented programs.
4. Describe how data are shared by functions in a procedure-oriented program.
5. What is object-oriented programming? How is it different from the procedure-oriented programming?
6. How are data and functions organized in an object-oriented program?
7. What are the unique advantages of an object-oriented programming paradigm?
8. What kinds of things can become objects in OOP?
9. Describe inheritance as applied to OOP.
10. What do you mean by dynamic binding? How is it useful in OOP?

Answers for Self Assessment Questions

1 (a) 2 (c) 3 (d) 4 (a) 5 (c)

1.10 Further Studies

1. The C++ Programming Language 4 Edition by Bjarne Stroustrup

2

C++ Environment

STRUCTURE

2.0 Objectives

2.1 Introduction

2.2 Program Development Environments

2.3 The Programming Language and C++ Standards

2.4 Introduction to Various C++ Compilers

2.5 The C++ Standard Library

2.6 Prototype of main () Function

2.7 Standard I/O Operator

2.8 Manipulators in C++

2.9 Comments in C++

2.10 Data Types

2.11 Summary

2.12 Keywords

2.13 Review Questions

2.14 Further Studies

2.0 Objectives

After studying this chapter, you will be able to:

- Discuss the C++ program development environment
- Explain the programming language and C++ standards
- Discuss about various C++ compilers
- Explain the C++ Standard Library
- Understand the prototype of main () function
- Explain about standard I/O operators

2.1 Introduction

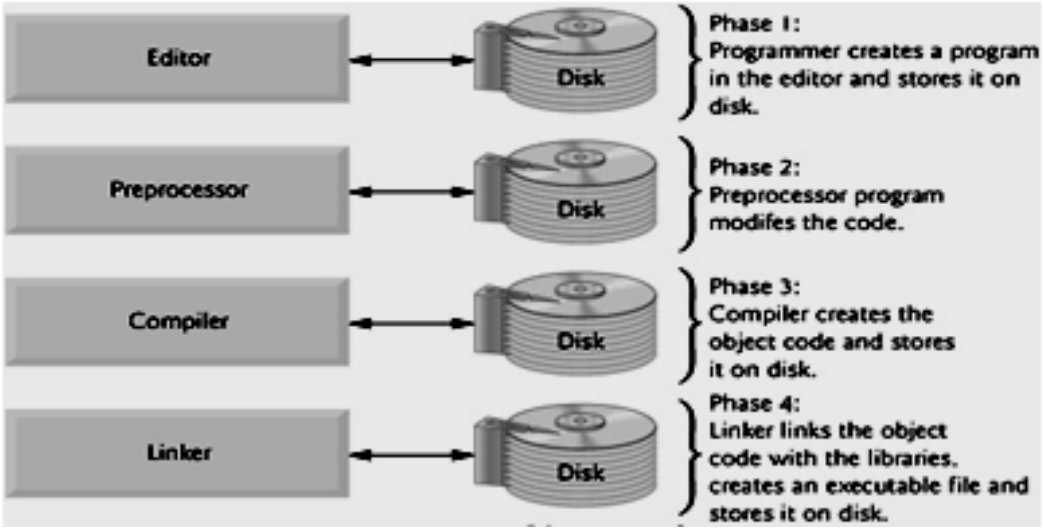
An integrated development environment (IDE) is a programming environment that has been packaged as an application program, typically consisting of a code editor, a compiler, a debugger, and a

graphical user interface (GUI) builder. The IDE may be a standalone application or may be included as part of one or more existing and compatible applications. The BASIC programming language, for example, can be used within Microsoft Office applications, which makes it possible to write a WordBasic program within the Microsoft Word application. IDEs provide a user-friendly framework for many modern programming languages, such as Visual Basic, Java, and PowerBuilder.
IDEs for developing HTML applications are among the most commonly used. For example, many people designing Web sites today use an IDE (such as HomeSite, DreamWeaver, or FrontPage) for Web site development that automates many of the tasks involved.
In this chapter we will discuss about the C++ programming development environment in which we develop different types of software's.

2.2 Program Development Environments

The term program is an organized list of instructions that, when executed, causes the computer to behave in a predetermined manner. Without programs, computers are useless.
A program is like a recipe. It contains a list of ingredients (called *variables*) and a list of directions (called *statements*) that tell the computer what to do with the variables. The variables can represent numeric data, text, or graphical images. There are many programming languages C, C++, Pascal, BASIC, FORTRAN, COBOL, and LISP is just a few. These are all high-level languages. One can also write programs in *low-level languages* called assembly languages, although this is more difficult. Low-level languages are closer to the language used by a computer, while high-level languages are closer to human languages.

Let us consider the steps in creating and executing a C++ application using a C++ development environment (illustrated in Figure 2.1). C++ systems generally consist of three parts: a program development environment, the language and the C++ Standard Library. C++ programs typically go through six phases: edit, pre-process, compile, link, load and execute. The following discussion explains a typical C++ program development environment.



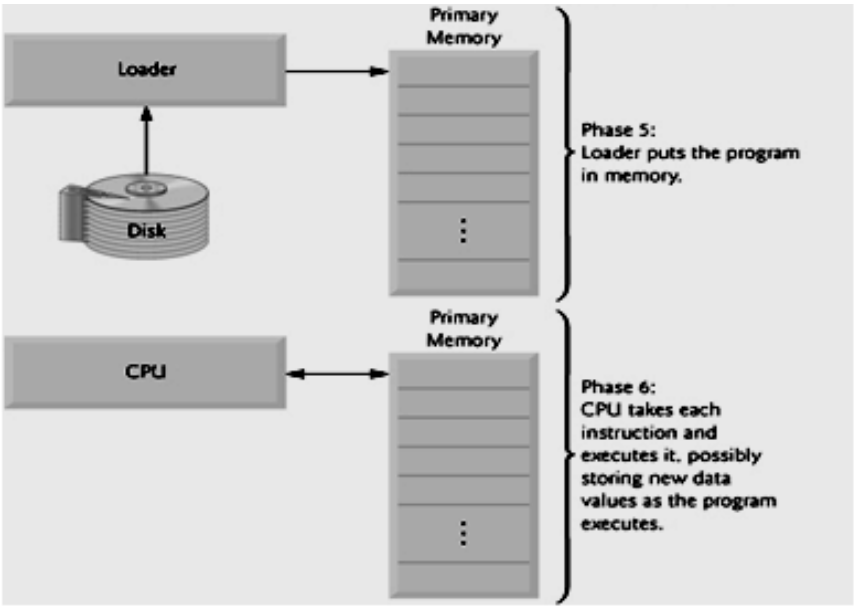


Figure 2.1: C++ environment.

Phase 1: Creating a Program

It consists of editing a file with an editor. You type a C++ program using the editor, make any necessary corrections and save the program on a secondary storage device, such as your hard drive. C++ source code filenames often end with the .cpp, .cxx, .cc or .C extensions (note that C is in uppercase) which indicate that a file contains C++ source code.

Two editors widely used on UNIX systems are vi and emacs. C++ software packages for Microsoft Windows such as Microsoft Visual C++ and cross-platform tools such as Eclipse have editors integrated into the programming environment.

Phases 2 and 3: Pre-processing and Compiling a C++ Program

In phase 2, you give the command to compile the program. In a C++ system, a pre-processor program executes automatically before the compiler's translation phase begins (so we call pre-processing phase 2 and compiling phase 3). The C++ pre-processor obeys commands called pre-processor directives, which indicate that certain manipulations are to be performed on the program before compilation. These manipulations usually include other text files to be compiled, and perform various text replacements. In phase 3, the compiler translates the C++ program into object code.

Phase 4: Linking

Phase 4 is called linking. C++ programs typically contain references to functions and data defined elsewhere, such as in the standard libraries or in the private libraries of groups of programmers working on a particular project. The object code produced by the C++ compiler typically contains "holes" due to these missing parts. A linker links the object code with the code for the missing functions to produce an executable image (with no missing pieces). If the program compiles and links correctly, an executable image is produced.

Phase 5: Loading

Before a program can be executed, it must first be placed in memory. This is done by the loader, which takes the executable image from disk and transfers it to memory. Additional components from shared libraries that support the program are also loaded.

Phase 6: Execution
Finally, the computer executes the program.

Problems That May Occur at Execution Time
Each of the preceding phases can fail because of various errors that we discuss throughout the chapter. This would cause the C++ program to display an error message. If this occurs, you would have to return to the edit phase, make the necessary corrections and proceed through the remaining phases again to determine that the corrections fix the problem(s).

Most programs in C++ input and/or output data. Certain C++ functions take their input from cin (the standard input stream; pronounced "see-in"), which is normally the keyboard, but cin can be redirected to another device. Data is often output to cout (the standard output stream; pronounced "see-out"), which is normally the computer screen, but cout can be redirected to another device. When we say that a program prints a result, we normally mean that the result is displayed on a screen. Data may be output to other devices, such as disks and hardcopy printers. There is also a standard error stream referred to as cerr. The cerr stream (normally connected to the screen) is used for displaying error messages. It is common for users to assign cout to a device other than the screen while keeping cerr assigned to the screen, so that normal outputs are separated from errors.

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: Draw the flow chart of C++ environment.

……..……

……..……

……...

Caution
The every program must be translated into a *machine language* that the computer can understand. This translation is performed by compilers, *interpreters*, and assemblers.

2.3 The Programming Language and C++ Standards

2.3.1 The Language
A programming language is an artificial language designed to communicate instructions to a machine, particularly a computer. Programming languages can be used to create programs that control the behaviour of a machine and/or to express algorithms precisely.
The earliest programming languages predate the invention of the computer, and were used to direct the behaviour of machines such as Jacquard looms and player pianos. Thousands of different programming languages have been created, mainly in the computer field, with many more being created every year. Most programming languages describe computation in an imperative style, i.e., as a sequence of commands, although some languages, such as those that support functional programming or logic programming, use alternative forms of description.
The description of a programming language is usually split into the two components of syntax (form) and semantics (meaning). Some languages are defined by a specification document (for example, the C, C++ programming language is specified by an ISO Standard), while other languages, such as Perl 5 and earlier, have a dominant implementation that is used as a reference.

A programming language is a notation for writing programs, which are specifications of a computation or algorithm. Some, but not all, authors restrict the term "programming language" to those languages that can express all possible algorithms.

2.3.2 C++ Standards

This International Standard specifies requirements for implementations of the C++ programming language. The first such requirement is that they implement the language, and so this International Standard also defines C++. Other requirements and relaxations of the first requirement appear at various places within this International Standard.

C++ is a general purpose programming language based on the C programming language as described in ISO/IEC 9899:1990 Programming languages - C. In addition to the facilities provided by C, C++ provides additional data types, classes, templates, exceptions, namespaces, inline functions, operator overloading, function name overloading, references, free store management operators, and additional library facilities.

The following standards contain provisions which, through reference in this text, constitute provisions of this International Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

- ISO/IEC 2382 (all parts), Information technology - Vocabulary
- ISO/IEC 9899:1990, Programming languages - C
- ISO/IEC 9899/Amd.1:1995, Programming languages - C, AMENDMENT 1: C Integrity
- ISO/IEC 10646-1:1993 Information technology - Universal Multiple-Octet Coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane

The library described in clause 7 of ISO/IEC 9899:1990 and clause 7 of ISO/IEC 9899/Amd.1:1995 is hereinafter called the Standard C Library.

For the purposes of this International Standard, the definitions given in ISO/IEC 2382 and the following definitions apply. Terms that are used only in a small portion of this International Standard are defined where they are used and italicized where they are defined.

2.4 Introduction to Various C++ Compilers

There are lot types of C++ compilers are used are discussed as:

2.4.1 C++Builder

C++Builder is a rapid application development (RAD) environment, developed by Borland and as of 2009 owned by Embarcadero Technologies, for writing programs in the C++ programming language. C++Builder combines the Visual Component Library and IDE written in Delphi with a C++ compiler. Traditionally, the release cycle was such that Delphi got major enhancements first, with C++Builder following, though recent versions have been released at the same time as their Delphi equivalents. Most components developed in Delphi can be used in C++Builder with no or little modification, although the reverse is not true.

C++Builder includes tools that allow true drag-and-drop visual development, making programming easier by incorporating a WYSIWYG GUI builder into its IDE.

C++Builder uses the same IDE as Delphi, and shares many core libraries. Notable shared Delphi (Pascal code) and C++ Builder routines include the FastMM4 memory manager, which was developed

as a community effort within the Fast Code project, the entire UI framework known as the VCL, which is written in Pascal, as well as base system routines, many of which have been optimized for both Delphi and C++Builder through the Fast Code project.
C++Builder projects can include Delphi code. The Delphi compiler emits C++ headers, allowing C++ code to link to Delphi classes and methods as though they were written in C++. The reverse (C++ code being used by Delphi) is not possible.

Version history
The following is a rough outline of product release information.

Year	Version
1997	1
1998	3
1999	4 (released as Inprise)
2000	5
2002	6
2003	X
2005	2006 (10)
2007	2007 (11)
Aug. 2008	2009 (12)
24 Aug. 2009	2010 (14)
30 Aug. 2010	XE (15)
31 Aug. 2011	XE2 (16)

2.4.2 x 86 Open64 Compiler Suites

The x 86 Open64 compiler systems is a high performance, production quality code generation tool designed for high performance parallel computing workloads. The x 86 Open64 environments provides the developer the essential choices when building and optimizing C, C++, and Fortran applications targeting 32-bit and 64-bit Linux platforms.
The x 86 Open64 compiler systems offers a high level of advanced optimizations, multi-threading, and processor support that includes global optimization, vectorization, interprocedural analysis, feedback directed optimizations, loop transformations, and code generation which extracts the optimal performance from each x86 processor core.

2.4.3 Turbo C++

Turbo C++ provides an environment called IDE (Integrated Development Environment). The editor is used to create the source file, compile it, link it and then execute it.
The first release of Turbo C++ was made available during the MS-DOS era on personal computers. Version 1.0, running on MS-DOS, was released in May 1990. An OS/2 version was produced as well. Version 1.01 was released on February 28, 1991[1], running on MS-DOS. The latter was able to generate both COM and EXE programs, and was shipped with Borland's Turbo Assembler compiler for Intel x86 processors. The initial version of the Turbo C++ compiler was based on a front end developed by TauMetric. This compiler supported the AT&T 2.0 release of C++.

Turbo C++ 3.0 was released in 1991 (shipping on November 20), and came in amidst expectations of the coming release of Turbo C++ for Microsoft Windows. Initially released as an MS-DOS compiler, 3.0 supported C++ templates, Borland's inline assembler, and generation of MS-DOS mode executables for both 8086 real-mode and 286-protected (as well as the Intel 80186.) 3.0's implemented AT&T C++ 2.1, the most recent at the time. The separate Turbo Assembler product was no longer included, but the inline-assembler could stand in as a reduced functionality version.
Soon after the release of Windows 3.0, Borland updated Turbo C++ to support Windows application development. The Turbo C++ 3.0 for Windows product was quickly followed by Turbo C++ 3.1 (and then Turbo C++ 4.5). It is possible that the jump from version 1.x to version 3.x was in part an attempt to link Turbo C++ release numbers with Microsoft Windows versions; however, it seems more likely that this jump was simply to synchronize Turbo C and Turbo C++, since Turbo C 2.0 (1989) and Turbo C++ 1.0 (1990) had come out roughly at the same time, and the next generation 3.0 was a merger of both the C and C++ compiler. Starting with version 3.0, Borland segmented their C++ compiler into two distinct product-lines: "Turbo C++" and "Borland C++". Turbo C++ was marketed toward the hobbyist and entry-level compiler market, while Borland C++ targeted the professional application development market. Borland C++ included additional tools, compiler code-optimization, and documentation to address the needs of commercial developers. Turbo C++ 3.0 could be upgraded with separate add-ons, such as Turbo Assembler and Turbo Vision 1.0. Version 4.0 was released in November 1993 and was notable (among other things) for its robust support of templates. In particular, Borland C++ 4 was instrumental in the development of the Standard Template Library, expression templates, and the first advanced applications of template multiprogramming. With the success of the Pascal-evolved product Delphi, Borland ceased work on their Borland C++ suite and concentrated on C++Builder for Windows. C++Builder shared Delphi's front-end application framework, but retained the Borland C++ back-end compiler. Active development on Borland C++/Turbo C++ was suspended until 2006.

2.5 The C++ Standard Library

The Standard Template Library provides a set of well structured generic C++ components that work together in a seamless way. Special care has been taken to ensure that all the template algorithms work not only on the data structures in the library, but also on built-in C++ data structures.
For example, all the algorithms work on regular pointers. The orthogonal design of the library allows programmers to use library data structures with their own algorithms, and to use library algorithms with their own data structures. The well specified semantic and complexity requirements guarantee that a user component will work with the library, and that it will work efficiently. This flexibility ensures the widespread utility of the library.
Another important consideration is efficiency. C++ is successful because it combines expressive power with efficiency. Much effort has been spent to verify that every template component in the library has a generic implementation that performs within a few percentage points of the efficiency of the corresponding hand coded routine.
The third consideration in the design has been to develop a library structure that, while being natural and easy to grasp, is based on a firm theoretical foundation.

2.5.1 Structure of the library

The library contains five main kinds of components:

- Algorithm: defines a computational procedure.

- Container: manages a set of memory locations.
- Iterator: provides a means for an algorithm to traverse through a container.
- Function object: encapsulates a function in an object for use by other components.
- Adaptor: adapts a component to provide a different interface.

2.5.2 Containers and Iterators

In particular, let us take a look at a simple algorithm and try to implement it using arrays and using linked lists. So, given a group of elements (say, integer values, to keep this example as simple as possible), we want to find the highest element.
A way of doing this would be to have a variable (say, HIGH) where we store the first element, and then, for each other element in the group, if its value is higher than HIGH, then we assign HIGH with the value of that particular element.
The pseudo-code for this algorithm could be as follows:

```
HIGH = first element
current_element = second element
while current_element is within the group of elements
  if current_element > HIGH, then HIGH = current_element
Advance to the next element end while
```

Notice that this pseudo-coded algorithm is valid for a group of elements, regardless how exactly those elements are stored (of course, provided that we are able to perform the required tests).
Let us try to implement it for both linked lists and arrays:

***Linked lists*:**

```
struct Element // This is an extremely simplified definition,
{              // but enough for this example.
  int value;
  struct Element * next;
};
int high = list->value; // list points to the first element
struct Element * current = list->next;
                    // refers (points) to second element
while (current != NULL) // test if within the group of elements
{
  if (current->value > high)
  {
    high = current->value;
  }
  current = current->next; // Advance to next element
}
```

Arrays:

```
int high = *array;
int * one_past_end = array + size;
int * current = array + 1;   // starts at second element
while (current != one_past_end) // test if within the group of elements
```

```
{
  if (*current > high)
  {
    high = *current;
  }
  current++;     // Advance to the next element
}
```

Surprise! Both fragments of code are almost identical. It is just the syntax that we use to manipulate and access the elements what changes. Notice that in both cases we have a pointer pointing to the current element. This pointer is compared to a particular value to test if we are within the group of values. Also, the pointer is dereferenced (in different concrete ways in both cases, but both are dereferencing operations) to obtain the particular value. This pointer allows us to advance to the next element (again, in different concrete ways, but still, in both cases we make the pointer point to the next element)

There is one important detail that makes the two examples conceptually identical: in this case, both data structures (array and linked list) are treated as a sequential group of elements; in both cases, the operations required are:
1. Point to a particular element
2. Access the element that is pointed
3. Point to the next element
4. Test if we are pointing within the group of elements

Notice that with these operations, we can implement any algorithm that requires sequential access to the elements of a group.

2.5.3 Operations other than Unary Predicates

The examples of find_if, count_if, remove_if have one common detail: they work with operations that represent a unary predicate (a condition on one element). We use them with function objects for which the operator() returns bool. Function objects may represent operations that are not necessarily a predicate. An obvious example is the algorithm transform. This algorithm receives four parameters: two iterators to specify the input sequence, one to specify the output sequence (client code is responsible of making sure there is enough room in the output sequence), and the operation. In this case, the operation represents a function that returns an output value given an input value.
Below is an example of using transform to obtain the lowercase equivalent of a string (yes, a string can be used with STL algorithms -- it is a “quasi-container”, in that it provides iterators, begin () and end () and other methods that make it compatible with STL containers):

```
class to_lower
{
public:
   char operator() (char c) const        // notice the return type
   {
      return tolower(c);
   }
};
```

```
string lower (const string & str)
{
   string lcase = str;
   transform (str.begin(), str.end(), lcase.begin(), to_lower());
   return lcase;
}
```

The transform line could have been:

```
   transform (lcase.begin(), lcase.end(), lcase.begin(), to_lower())
```

(remember that the output sequence can be the same input sequence, if we want in-place transformations)

2.5.4 Standard Library Function Objects

The STL provides a handful of ready-to-use function object classes, including predicates and arithmetic operations. These function objects are found in the <functional> library facility (i.e., we #include <functional> to use them).

The predicates include comparisons and logical operations, provided in the form of template classes, including the following: equal_to, not_equal_to, greater, less, greater_equal, less_equal.

These are binary predicates that can be used combined with algorithms that expect an operation. The implementation of these function objects is pretty straightforward. Except for one detail that is irrelevant for the purpose of this discussion, the implementation could be similar to this:

```
template <typename T>
class greater
{
public:
   bool operator() (const T & v1, const T & v2) const
   {
      return v1 > v2;
   }
};
```

For instance, we could use this function object greater to sort a sequence in descending order:

```
vector<int> values;
// ... add elements...
sort(values.begin(), values.end(), greater<int>());
```

The trick is that the third parameter is an operation that will be used instead of direct comparison, and that operation is supposed to emulate the “less-than” comparison. If we use “greater-than” instead, we are “lying” to the algorithm and always giving the opposite result- the outcome is that the sequence ends up sorted in the exact opposite order.

The function objects representing arithmetic operations include plus, minus, multiplies, and divides (and a couple others that will omit). These are binary operations that return the sum, difference, product, or division of the first argument and the second (in that order). You can imagine that their implementation is also straightforward.

We can use the multiplies function object to obtain the product of all the numbers in a sequence as shown below:

```
list<double> values;
// ... add elements ...
double product = accumulate (values.begin(), values.end(), 1.0,
                             multiplies<double>());
```

The trick here is that the user-provided operation is supposed to replace direct addition (e.g., we may want to accumulate the grades of all the students, or accumulate the lengths of a group of strings, etc.). We provide an operation that multiplies instead of adding.

2.6 Prototype of main () Function

In C and C++, the function prototype of the main function looks like one of the following:

```
int main(void)
int main(int argc, char **argv)
int main(int argc, char *argv[])

int main()
```

The parameters argc, argument count, and argv, argument vector, respectively give the number and value of the program's command-line arguments. The names of argc and argv may be any valid identifier in C, but it is common convention to use these names. In C++, the names are to be taken literally, and the "void" in the parameter list is to be omitted, if strict conformance is desired. Other platform-dependent formats are also allowed by the C and C++ standards, except that in C++ the return type must stay int; for example, Unix (though not POSIX.1) and Microsoft Windows have a third argument giving the program's environment, otherwise accessible through getenv in stdlib.h:

```
int main(int argc, char **argv, char **envp)
```

2.7 Standard I/O Operator

Using the C++ I/O stream library we will get the user's input from the keyboard and we will print messages onto the screen. The I/O stream library is part of the C++ standard library.

In C++, I/O is performed by using streams. A stream is a "stream of data" in which character sequences are "flow into" or "flow out off." A stream is an object with properties that are defined by a class. Global objects are predefined for the standard I/O channels.

2.7.1 Standard Output (cout)

The standard output of a program points at the screen. So with the cout operator and the "insertion" operator (<<) you can print a message onto the screen. Let us take a look at an example:

```
#include<iostream>
using namespace std;
int main()
{
        cout << "Hello World|";
        return 0;
}
```

Note: the double quotes around Hello World (because it is a constant string of characters.)

To print the content of a variable the double quotes are not used. Take a look at an example:

```
#include<iostream>
using namespace std;

int main()
{
        char Yes = 'y';
        cout << Yes;
        return 0;
}
```

Note: If you do not want to use the namespace std, you could write std::cout << Yes

The << operator can be used multiple times in a single statement. Take a look at an example:

```
#include<iostream>
using namespace std;

int main()
{
        cout << "Hello," << "this is a test" << "string.";
        return 0;
}
```

2.7.2 Standard input (cin)

The standard input device is the keyboard. With the cin and >> operators it is possible to read input from the keyboard.
Take a look at an example:

```
#include<iostream>
using namespace std;

int main()
{
        char MY_CHAR;
        cout << ":Press a character and press return:";
        cin >> MY_CHAR;
        cout << MY_CHAR;
        return 0;
}
```

Note: The input is processed by cin after the return key is pressed.
The cin operator will always return the variable type that you use with cin. So if you request an integer you will get an integer and so on. This can cause an error when the user of the program does

not return the type that you are expecting. (Example: you ask for an integer and you get a string of characters.)
Later on we will offer a solution to this problem.
The cin operator is also chainable. For example:

```
cin >> XX >> YY;
```

In this case the user must give two input values that are separated by any valid blank separator (tab, space or new-line).

Caution
The header file iostream must be included to make use of the input/output (cin/cout) operators.

2.8 Manipulators in C++

Manipulators are operators used with the insertion operator << to modify or manipulate the way data is displayed. It just changes certain characteristics of the input and output.
Manipulators are operators used in C++ for formatting output. The data is manipulated by the programmer's choice of display.
There are numerous manipulators available in C++. Some of the more commonly used manipulators are provided here below:

endl Manipulator
This manipulator has the same functionality as the 'n' newline character.

For example:
Sample Code

```
1. cout << "India" << endl;
2. cout << "Training";
```

produces the output:
India
Training

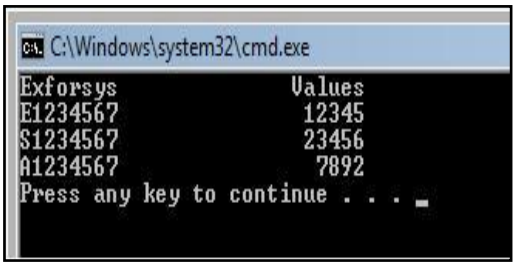
setw Manipulator
This manipulator sets the minimum field width on output. The syntax is:
setw(x)
Here setw causes the number or string that follows it to be printed within a field of x characters wide and x is the argument set in setw manipulator. The header file that must be included while using setw manipulator is .

Sample Code:

```
1. #include <iostream>
2. using namespace std;
3. #include <iomanip>
4. void main( )
5. {
6. int x1=12345,x2= 23456, x3=7892;
```

```
7.  cout << setw(8) << “India” << setw(20) << “Values” << endl
8.  << setw(8) << “E1234567” << setw(20)<< x1 << endl
9.  << setw(8) << “S1234567” << setw(20)<< x2 << endl
10. << setw(8) << “A1234567” << setw(20)<< x3 << endl;
11. }
```

The output of the above example is:



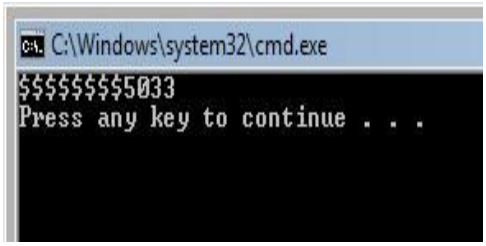
setfill Manipulator

This is used after setw manipulator. If a value does not entirely fill a field, then the character specified in the setfill argument of the manipulator is used for filling the fields.

Sample Code

```
1.  #include <iostream>
2.  using namespace std;
3.  #include <iomanip>
4.  void main()
5.  {
6.  cout << setw(10) << setfill(‘$’) << 50 << 33 << endl;
7.  }
```

The output of the above example is:



This is because the setw sets 10 for the width of the field and the number 50 has only 2 positions in it. So the remaining 8 positions are filled with $ symbol which is specified in the set fill argument.

Set precision Manipulator

The set precision Manipulator is used with floating point numbers. It is used to set the number of digits printed to the right of the decimal point. This may be used in two forms:

- fixed
- scientific

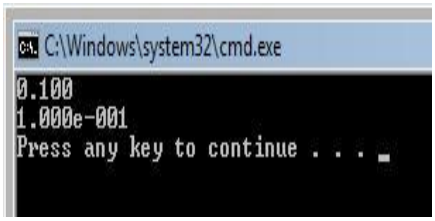
These two forms are used when the keywords fixed or scientific are appropriately used before the set precision manipulator. The keyword fixed before the setprecision manipulator prints the floating

point number in fixed notation. The keyword scientific, before the setprecision manipulator, prints the floating point number in scientific notation.

Sample Code

```
1. #include <iostream>
2. using namespace std;
3. #include <iomanip>
4. void main( )
5. {
6. float x = 0.1;
7. cout << fixed << set precision(3) << x << endl;
8. cout << scientific << x << endl;
9. }
```

The output of the above example is:



The first cout statement contains fixed notation and the set precision contains argument 3. This means that three digits after the decimal point and in fixed notation will output the first cout statement as 0.100. The second cout produces the output in scientific notation. The default value is used since no setprecision value is provided.

2.9 Comments in C++

Program comments are explanatory statements that you can include in the C++ code that you write and help anyone reading it is source code.

All programming languages allow for some form of comments.

C++ supports single line and multi-line comments. All characters available inside any comment are ignored by C++ compiler.

C++ comments start with /* and end with */. For example:

```
/* This is a comment */
/* C++ comments can also
 * span multiple lines
 */
```

A comment can also start with //, extending to the end of the line. For example:

```
#include <iostream>
using namespace std;
main()
{
   cout << “Hello World”; // prints Hello World
```

```
  return 0;
}
```
When the above code is compiled, it will ignore // prints Hello World and final executable will produce following result:

Hello World
Within a /* and */ comment, // characters have no special meaning. Within a // comment, /* and */ have no special meaning. Thus, you can “nest” one kind of comment within the other kind. For example:

```
/* Comment out printing of Hello World:
cout << “Hello World”; // prints Hello World
*/
```

2.10 Data Types

Data types are means to identify the type of data and associated operations of handling it. C++ provides a predefined set of data types for handling the data it uses. When variables are declared of a particular data type then the variable becomes the place where the data is stored and data types is the type of value (data) stored by that variable. Data can be of may types such as character, integer, real etc. since the data to be dealt with are of may types, a programming language must provide different data types.

2.10.1 Fundamental Data Types

As the name suggests these are the atomic or the fundamental data types in C++. Earlier there was five of these (int, char, float, double and void) but later two new data types namely bool and wchar_t have been added. Int stores integer data or whole numbers such as 10, -340 etc., char stores any of the ASCII characters, float is used to store numbers having fractional part such as 10.097, double stores the same data as float but with higher range and precision, bool can only store true and false values.

```
//Program to illustrate various fundamental data type
 #include<iostream>
 void main(void)
 {
 int age;
 float salary;
 char code;
 cout<< “Enter age, salary and code:”;
 cin>>age;
 cin>>salary;
 cin>>code;
 cout<<endl;//goto next line
 cout<< “DETAILS”<<endl;//short for cout<< “DETAILS”;cout<<endl;

 cout<< “Age:”<<age<<endl;
 cout<< “Salary:”<<salary<<endl; cout<<”Code:”<<code<<endl;
}
```

2.10.2 Derived Data Types

These are the data types which are derived from the fundamental data types.
It is further divide into two categories

i)Built-In and ii)User-defined, which are discussed below as separate topics.

Built-In Derived Data Type

Arrays: Arrays refer to a list of finite number of same data types. The data in the array can be accessed by an index number ranging from 0 to n(where n is the number of data element it can store). Ex- if arr[3] is an array of int(egers) then the different values in the array can be accessed as shown below. arr[0], arr[1],arr[2] when we declare an array such as the one sown above then by arr[3] we mean that we want three elements in the array and hence while accessing arr[2] is the last element.

```
//Program to illustrate arrays
#include<iostream>
void main(void)
{
int arr[3];//it will store 3 integer elements
cout<< "enter 3 numbers:";
cin>>arr[0]>>arr[1]>>arr[2];//this statement is same as using three cin's

cout<<endl;//goto next line
cout<<arr[0]<<arr[1]<<arr[2];

}
```

*Pointer***:** A pointer is a variable that holds the memory address of other variable. It is also of different data types, ex- char pointer can store address of only char variables, int pointer can store address of int variables and so on.

Reference: A reference in the simplest sense is an alias or alternate name for a previously defined variable.

```
//Program to illustrate References
#include<iostream>
void main(void)
{
int var;
int &refvar=var;//here a reference variable to var is declared remember var was previously declared

var=10;//var is given the value 10
cout<<var<<endl;
refvar=100;//reference variable of var is changed

cout<<var;//but var also gets changed

}
```

User-Defined Derived Data Types

1. ***Class:*** A class is a collection of variables and function under one reference name. it is the way of separating and storing similar data together. Member functions are often the means of accessing, modifying and operating the data members (i.e. variables). It is one of the most important features of C++ since OOP is usually implemented through the use of classes.
2. ***Structure*****:** In C++ structure and class same except for some very minor differences.
3. ***Union:*** A union is a memory location shared by two or more different variables, generally of different data types.
4. ***Enumerations*****:** It can be used to assign names to integer constants.

```
//Program to illustrate Enumerators
#include<iostream.h>
void main(void)
{
enum type{POOR,GOOD,EXCELLENT};//this is the syntax of enumerator

int var;
var=POOR;//this makes programs more understandable
cout<<var<<endl;
var=GOOD;
cout<<var<<endl;
var=EXCELLENT;
cout<<var;
}
```

2.10.3 Data Types Modifiers

- signed
- unsigned
- short
- long

int, char, float, double data types can be preceded with these modifiers to alter the meaning of the base type to fit various situations properly. Every data type has a limit of the larges and smallest value that it can store known as the range. An integer (usually 4 bytes long) can store any value ranging from -2147483648 to 2147483647. Data Type modifiers usually alter the upper and lower limit of a data type. Unsigned modifier makes a variable only to store positive values. For example - if a data type has a range from –a to a then unsigned variable of that type has a range from 0 to 2a. Preceding any data type with signed is optional because every data type is signed by default. Short integers are 2 bytes long and long integers are 8 bytes long.

Self Assessment Questions

1. .....it can be used to assign names to integer constants.

(a). Enumerations (b). Structure
(c). Unsigned (d). None of these.

2. A class is a collection of variables and function under one reference name it is the way of separating and storing similar data together.

(a). True (b). False.

3. .....are means to identify the type of data and associated operations of handling it. C++ provides a predefined set of data types for handling the data it uses.

(a). Variables (b). Data types

(c). Int (d). Float

4. The iostream library is part of the C++ standard library.

(a). True (b). False

5. The.....argc, argument count, and argv, argument vector, respectively give the number and value of the program's command-line arguments.

(a). Parameters (b). int

(c). long (d). None of these

6. The. Operations Template Library provides a number of useful, generic algorithms to perform the most commonly used operations on groups/sequences of elements.

(a).True (b). False

2.11 Summary

- The C++ is a general purpose programming language based on the C programming language as described in ISO/IEC 9899:1990 programming languages - C.
- The term computer language is sometimes used interchangeably with programming language.
- A complete specification for a programming language includes a description, possibly idealized, of a machine or processor for that language.
- The programming language is a notation for writing programs, which are specifications of a computation or algorithm.
- The earliest programming languages predate the invention of the computer, and were used to direct the behaviour of machines such as Jacquard looms and player pianos.

2.12 Keywords

Abstractions: Programming languages usually contain abstractions for defining and manipulating data structures or controlling the flow of execution.

C++: It is an object-oriented programming (OOP) language that is viewed by many as the best language for creating large-scale applications. C++ is a superset of the C language.

Structure: In C++ structure and class same except for some very minor differences.

Unsigned: The Variable type of int can hold negative and positive numbers but a signed int holds negative, zero or positive numbers.

Variable: It is a way of referring to a memory location used in a computer program. This memory location holds values- perhaps numbers or text or more complicated types of data like a payroll record.

2.13 Review Questions

1. What are the C++ program development environments?
2. Explain the programming language and C++ standards.

3. Discuss the various C++ compilers.
4. What is the C++ standard library? Explain with example.
5. Discuss the structure of the library and containers, iterators.
6. Explain the operations of unary predicates.
7. Discuss the function of standard library objects.
8. What is the prototype of main () function?
9. Explain the data types.
10. Differentiate between the fundamental and derived data types.

Answers for Self Assessment Questions

1. (a) 2. (a) 3. (b) 4. (a) 5. (a) 6. (b)

2.14 Further Studies

1. http://my.safaribooksonline.com/book/programming/cplusplus/9780136085669/introduction-to-computers-the-internet-and-the-world-wide-web/ch01lev1sec15
2. The C++ Programming Language 4 Edition by Bjarne Stroustrup

3

Creating and Compiling C++ Programs

STRUCTURE

3.0 Objectives

3.1 Introduction

3.2 Turbo C++ IDE

3.3 Creating, Compiling and Running a C++ Program Using IDE and Command Line

3.4 Elements of C++ Language

3.5 Structure of a C++ Program

3.6 C++ Tokens

3.7 Type Conversion in Expressions

3.8 Summary

3.9 Keywords

3.10 Review Questions

3.11 Further Studies

3.0 Objectives

After studying this chapter, you will be able to:

- Discuss the turbo C++ IDE
- Explain the creating, compiling and running a C++ program using IDE
- Define the elements of C++ language
- Explain the C++ tokens
- Explain the type conversion in expressions

3.1 Introduction

C++ is a third generation programming language. When computers were first invented, they were programmed with very simple, low-level commands. A programmer would design a program, and then translate the program into a specific set of codes, known as machine language. These codes would be fed into a computer with switches, punch-cards, or primitive keypads. These programs were cumbersome to write, and very hard to debug. (Debugging is the act of removing mistakes in a program.) Machine code is considered the first generation of programming languages. C++ is a

programming language substantially different from C. Many see C++ as “a better C than C,” or as C with some add-ons. C++ shares the same low level constructs as C, however, and assume some knowledge of C in this course. You might want to have a look at the C introduction course to get up to speed on that language. C++ is a programming language of many different dialects, similar to the way that each spoken language has many different dialects. In C++, dialects are not because the speakers live in the North or South. Instead, it is because there are many different compilers that support slightly different features. There are several common compilers: in particular, Borland C++, Microsoft C++, and GNU C++. There are also many front-end environments for the different compilers the most common is Dev-C++ around GNU’s g++ compiler. Some, such as g++, are free, while others are not.

3.2 Turbo C++ IDE

Turbo C++ was a C++ compiler and integrated development environment and computer language originally from Borland. Most recently it was distributed by Embarcadero Technologies, which acquired all of Borland’s compiler tools with the purchase of its CodeGear division in 2008. Turbo C++ 2006 was released on September 5, 2006 and was available in ‘Explorer’ and ‘Professional’ editions. In October 2009 Embarcadero Technologies discontinued support of its 2006 C++ editions. As such, the Explorer edition is no longer available for download and the Professional edition is no longer available for purchase from Embarcadero Technologies. Turbo C++ is succeeded by C++Builder.

Dev-C++, developed by Bloodshed Software, is a fully featured graphical IDE (Integrated Development Environment), which is able to create Windows or console-based C/C++ programs using the MinGW compiler system. MinGW (Minimalist GNU for Windows) uses GCC (the GNU g++ compiler collection), which is essentially the same compiler system that is in Cygwin (the unix environment program for Windows) and most versions of Linux. There are, however, differences between Cygwin and MinGW; link to Differences between Cygwin and MinGW for more information. C++Builder is a rapid application development (RAD) environment, developed by Borland and as of 2009 owned by Embarcadero Technologies, for writing programs in the C++ programming language. C++Builder combines the Visual Component Library and IDE.

C++ is currently one of the most popular third generation languages. It achieved popularity as a basic development platform for many operating systems. On a PC platform, there are various flavours available to support the creation of applications for virtually all OSes, eg. Turbo C++ for DOS, Borland C++ for Windows, Visual C++ for Win32, GNU C++ for DOS/Win/Unix This course shall concentrate on the principles of object-orientated programming and its uses and advantages in advanced API environments like that of Windows.

There are a myriad of reasons to study C++, included among which are:

- Since it is widely available, programmers that know C/C++ can migrate very easily to other platforms/operating systems
- Object-orientation is a principle component of many programming languages; thus all built-in code is already object-orientated
- Object-orientation makes it easier to understand the logic in programs since it is a better model of reality than classical programs
- Object-orientation lends itself to visual programming, which is fast becoming a trend in newer compilers
- Object-orientation allows for easy re-use, expansion and maintenance

Did You Know?

The original Turbo C++ product line was put on hold after 1994, and was revived in 2006 as an introductory-level IDE, essentially a stripped-down version of their flagship C++Builder.

3.3 Creating, Compiling and Running a C++ Program Using IDE and Command Line

3.3.1 Creating and Compile C++ Programs Using IDE

In order to run a program and see it doing wonderful things, you should first write the program. The program can be written in any text editor, such as vi and emacs in Unix environment and using command prompt in DOS. There are also several Integrated Development Environment (IDE) packages available which provide a complete programming environment for C++ in which you can write, compile, run, and debug your program.

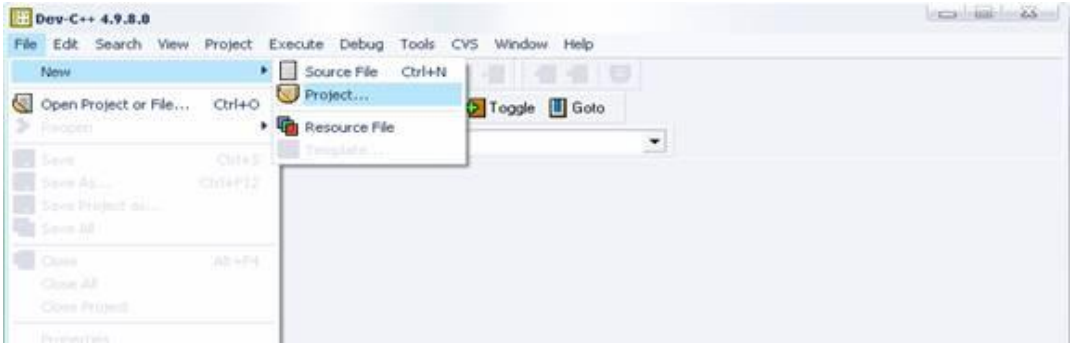
C++ programs are saved with extensions .C, .cc, .cpp, .cxx depending on the platform you are working upon.

Once you have saved a program, next stage is compiling it using a compiler which converts your C++ program into the object code which the computer can understand.

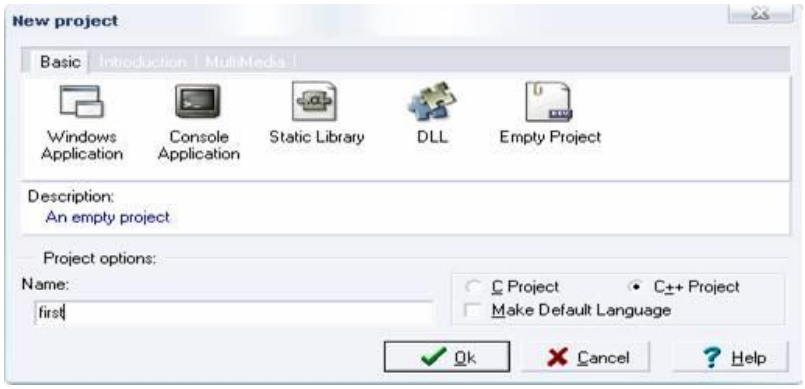
Compile using DevC++ compiler

If you work on Windows and use DevC++, then editing and compiling is as easy as click of a button. DevC++ is an IDE which allows you to edit, compile, run and debug your program in the IDE itself. If you want to install DevC++ then install it from here.

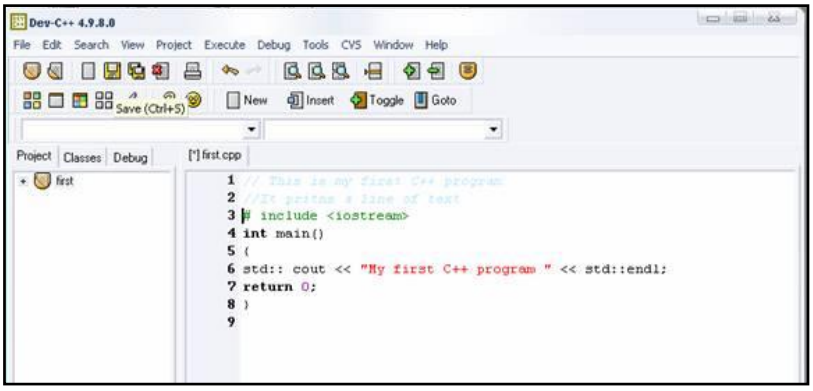
Once you have installed and configured the software, Write and save your program using DevC++ itself. Create a new program by clicking on New => Project.



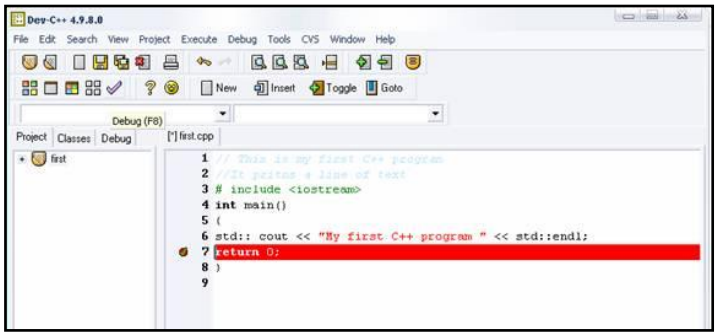
1. Choose Empty Project from New Project dialog and choose a name for the program, in our case first and click Ok.



Write the code of the program and save it.



1. Click on Compile button (third row, first button) to compile your source code. If there are any errors in your program, then a window at the bottom will specify the warnings.
2. After program is compiled, click on Run button (next to compile).
3. However, DevC++ has a problem. As soon as you run the program, output window opens momentarily and then it closes. So to come around this solution, set a breakpoint at the end of main function and then click on Debug instead of running it.



Output:

When you run the program, output window will show the string.



3.3.2 Creating and Compile C++ Programs Using Command Line

Creating the program

Create a file containing the complete program, such as the above example. You can use any ordinary editor with which you are familiar to create the file. One such editor is *textedit* available on most UNIX systems.

The filename must by convention end ".c" (full stop, lower case c), *e.g. myprog.c* or *progtest.c*. The contents must obey C syntax. For example, they might be as in the above example, starting with the

line /* Sample.... (or a blank line preceding it) and ending with the line} /* end of program */ (or a blank line following it).

Compilation

There are many C compilers around. The cc being the default Sun compiler. The GNU C compiler gcc is popular and available for many platforms. PC users may also be familiar with the Borland bcc compiler.

There are also equivalent C++ compilers which are usually denoted by CC (*note* upper case CC. For example Sun provides CC and GNU GCC. The GNU compiler is also denoted by g++

For the sake of compactness in the basic discussions of compiler operation we will simply refer to the cc compiler -- other compilers can simply be substituted in place of cc unless otherwise stated.

To compile your program simply invoke the command cc. The command must be followed by the name of the (C) program you wish to compile. A number of compiler options can be specified also. Thus, the basic compilation command is:

cc program.c

where *program.c* is the name of the file.

If there are obvious errors in your program (such as mistypings, misspelling one of the key words or omitting a semi-colon), the compiler will detect and report them.

There may, of course, still be logical errors that the compiler cannot detect. You may be telling the computer to do the wrong operations.

When the compiler has successfully digested your program, the compiled version, or executable, is left in a file called *a.out* or if the compiler option *-o* is used: the file listed after the *-o*.

It is more convenient to use a *-o* and filename in the compilation as in

cc -o program program.c

which puts the compiled program into the file program (or any file you name following the "-o" argument) instead of putting it in the file a.out .

3.4 Elements of C++ Language

When we consider a C++ program it can be defined as a collection of objects that communicate via invoking each other's methods.

Let us now briefly look into what do class, object, methods and instant variables mean.

- **Object -** Objects have states and behaviors. Example: A dog has states-color, name, breed as well as behaviors -wagging, barking, eating. An object is an instance of a class.
- **Class -** A class can be defined as a template/ blue print that describe the behaviors/states that object of its type support.
- **Methods -** A method is basically a behavior. A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed.
- **Instant Variables -** Each object has its unique set of instant variables. An object's state is created by the values assigned to these instant variables.

3.4.1 Semicolons and Blocks

In C++, the semicolon is a statement terminator. That is, each individual statement must be ended with a semicolon. It indicates the end of one logical entity.

For example, following are three different statements:

```
x = y;
y = y+1;
add(x, y);
```

A block is a set of logically connected statements that are surrounded by opening and closing braces. For example:

```
{
  cout << "Hello World"; // prints Hello World
  return 0;
}
```

C++ does not recognize the end of the line as a terminator. For this reason, it does not matter where on a line you put a statement.
For example:

```
x = y;
y = y+1;
add(x, y);
```

is the same as

```
x = y; y = y+1; add(x, y);
```

Caution
Missing semicolon after any statement cause the compile time error

3.4.2 Identifiers

A C++ identifier is a name used to identify a variable, function, class, module, or any other user-defined item. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores, and digits (0 to 9).
C++ does not allow punctuation characters such as @, $, and % within identifiers. C++ is a case sensitive programming language. Thus Manpower and manpower are two different identifiers in C++.
Here are some examples of acceptable identifiers:

```
mohd       zara    abc   move_name  a_123
myname50   _temp   j     a23b9      retVal
```

3.4.3 Trigraphs

A few characters have an alternative representation, called a trigraph sequence. A trigraph is a three-character sequence that represents a single character and the sequence always starts with two question marks.
Trigraphs are expanded anywhere they appear, including within string literals and character literals, in comments, and in preprocessor directives.

Following are most frequently used trigraph sequences:

Trigraph	Replacement
??=	#
??/	\
??’	^
??(	[
??)	]
??!	|
??<	{
??>	}
??-	~

All the compilers do not support trigraphs and they are not advised to be used because of their confusing nature.

3.4.4 Whitespace

A line containing only whitespace, possibly with a comment, is known as a blank line, and C++ compiler totally ignores it. Whitespace is the term used in C++ to describe blanks, tabs, newline characters and comments. Whitespace separates one part of a statement from another and enables the compiler to identify where one element in a statement, such as int, ends and the next element begins. Therefore, in the statement,

int age;

There must be at least one whitespace character (usually a space) between int and age for the compiler to be able to distinguish them. On the other hand, in the statement

fruit = apples + oranges; // Get the total fruit

No whitespace characters are necessary between fruit and =, or between = and apples, although you are free to include some if you wish for readability purpose.

Did You Know?

Whitespace is also an esoteric programming language developed by Edwin Brady and Chris Morris at the University of Durham (also developers of the Kaya programming language). It was released on 1 April 2003. Its name is a reference to whitespace characters.

Self Assessment Questions

1. The original Turbo C++ product line was put on hold after……, and was revived in 2006 as an introductory-level IDE, essentially a stripped-down version of their flagship C++Builder.

(a) 1990 (b) 1994

(c) 1995 (d) 1999

2. Turbo C++ is succeeded by ……

(a) Visual C++ (b) C++ programs
(c) Borland C++ (d) C++Builder

3. C++ programs are saved with extensions…… depending on the platform you are working upon.
(a) .cc (b) .c and .cxx
(c) .cpp (d) All of these

4. The GNU ……..gcc is popular and available for many platforms.
(a) C++ compiler (b) G++ compiler
(c) C compiler (d) DevC++ compiler

5. An object's state is created by the values assigned to these……
(a) instant variables (b) objects
(c) methods (d) class

3.5 Structure of a C++ Program

A computer program is a sequence of instructions that tell the computer what to do.

3.5.1 Statements and Expressions

The most common type of instruction in a program is the statement. A statement in C++ is the smallest independent unit in the language. In human language, it is analogous to a sentence. We write sentences in order to convey an idea. In C++, we write statements in order to convey to the compiler that we want to perform a task. Statements in C++ are terminated by a semicolon.

There are many different kinds of statements in C++.

The following are some of the most common types of simple statements:

```
1int x;
2x = 5;
3cout << x;
```

int x is a declaration statement. It tells the compiler that x is a variable. All variables in a program must be declared before they are used.

x = 5 is an assignment statement. It assigns a value (5) to a variable (x).

cout << x; is an output statement. It outputs the value of x (which we set to 5 in the previous statement) to the screen.

The compiler is also capable of resolving expressions. An expression is a mathematical entity that evaluates to a value. For example, in math, the expression 2+3 evaluates to the value 5. Expressions can involve values (such as 2), variables (such as x), operators (such as +) and functions (which return an output value based on some input value). They can be singular (such as 2, or x), or compound (such as 2+3, 2+x, x+y, or (2+x)*(y-3)).

For example, the statement x = 2 + 3; is a valid assignment statement. The expression 2+3 evaluates to the value of 5. This value of 5 is then assigned to x.

3.5.2 Functions

In C++, statements are typically grouped into units called functions. A function is a collection of statements that executes sequentially. Every C++ program must contain a special function called main (). When the C++ program is run, execution starts with the first statement inside of main ().

Functions are typically written to do a very specific job. For example, a function named Max () might contain statements that figures out which of two numbers is larger. A function named CalculateGrade () might calculate a student's grade.

3.5.3 Libraries

Libraries are groups of functions that have been "packaged up" for reuse in many different programs. The core C++ language is actually very small and minimalistic — however, C++ comes with a bunch of libraries, known as the C++ standard libraries, that provide programmers with lots of extra functionality. For example, the iostream library contains functions for doing input and output. During the link stage of the compilation process, the libraries from the C++ standard library are the runtime support libraries that are linked into the program.

Taking a look at a sample program, Now that you have a brief understanding of what statements, functions, and libraries are, let us look at a simple hello world program.

Consider our hello world program:

```
#include <iostream>
int main()
{
  using namespace std;
  cout << "Hello world!" << endl;
  return 0;
}
```

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Write a program to add two numbers.

……..……

……..……

……...

3.6 C++ Tokens

A token is a group of characters that logically belong together. A C++ program is written using these tokens, white spaces, and the syntax of the language. Most of the C++ tokens are basically similar to the C tokens with the exception of some additions and minor modifications. The programmer can write a program by using tokens.

C++ uses the following types of tokens.

- Keywords
- Identifiers
- Literals
- Punctuators
- Operators

3.6.1 Keywords

The keywords implement specific C++ language features. They are explicitly reserved identifiers and cannot be used as names for the program variables or other user-defined program elements. Table 3.1 gives the complete set of C++ keywords. Many of them are common to both C and C++. The ANSI C

keywords are shown in boldface. Additional keywords have been added to the ANSI C keywords in order to enhance its features and make it an object-oriented language. ANSI C++ standards committee has added some more keywords to make the language more versatile. There are some reserved words in C++ which have predefined meaning to complier called keywords. Some commonly used keywords are given as:

Table 3.1: List of keywords

asm	double	new	switch
auto	else	operator	template
break	enum	private	this
case	extern	protected	try
catch	float	public	typedef
char	for	register	union
class	friend	return	unsigned
const	goto	short	virtual
continue	if	signed	void
default	inline	sizeof	volatile
delete	int	static	while
do	long	struct	

3.6.2 Identifiers

Identifiers refer to the names of variables, functions, arrays, classes, etc. created by the programmer. They are the fundamental requirement of any language. Each language has its own rules for naming these identifiers.

The following rules are common to both C and C++:

- Only alphabetic characters, digits and underscores are permitted.
- The name cannot start with a digit.
- Uppercase and lowercase letters are distinct.
- A declared keyword cannot be used as a variable name.

Symbolic names can be used in C++ for various data items used by a programmer in his program. For example, if a programmer wants to store a value 50 in a memory location, he/she choose any symbolic name (say MARKS) and use it as given below:

MARKS = 50

The symbol '=' is an assignment operator. The significance of the above statement is that 'MARKS' is a symbolic name for a memory location where the value 50 is being stored.

A symbolic name is generally known as an identifier. The identifier is a sequence of characters taken from C++ character set.

The rules for the formation of an identifier are:

(i) An identifier can consist of alphabets, digits and/or underscores.

(ii) It must not start with a digit.

(iii) C++ is case sensitive, i.e., upper case and lower case letters are considered different form each other. It may be noted that TOTAL and total are two different identifier names.

(iv) It should not be a reserved word.

3.6.3 Literals

Literals (often referred to as constants) are data items that never change their value during the execution of the program.

The following types of literals are available in C++.
(i) integer-constants
(ii) character-constants
(iii) floating-constants
(iv) string-literals

(i) Integer constants

Integer constants are whole numbers without any fractional part. It may contain either + or – sign, but decimal point or commas does not appear in any integer constant. C++ allows three types of integer constants.

- Decimal (Base 10)
- Octal (Base 8)
- Hexadecimal (Base 16)

Decimal integer constants

It consists of sequence of digits and should not begin with 0 (zero). For example 124, - 179, + 108.

Octal integer constants

It consists of sequence of digits starting with 0 (zero). For example, 014, 012.

Hexadecimal integer constant

It consists of sequence of digits preceded by ox or OX. For example OXD, OXC. The suffix l or L and u or U attached to any constant forces it to be represented as a long and unsigned respectively.

(ii)Character constants

A character constant in C++ must contain one or more characters and must be enclosed in single quotation marks. For example 'A', '9', etc. C++ allows non graphic characters which cannot be typed directly from keyboard, e.g., backspace, tab, carriage return etc. These characters can be represented by using a escape sequence. An escape sequence represents a single character. Table 3.2 gives a listing of common escape sequences.

Table 3.2: Common escape sequences

Escape Sequence	Nongraphic Character
\a	Bell (beep)
\b	Backspace
\f	Formfeed
\n	Newline or line feed
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\?	Question mark
\ \	Backslash
\ '	Single quote
\ "	Double quote
\ xhh	Hexadecimal number (hh represents the number in hexadecimal)
\ 000	Octal number (00 represents the number in octal)
\0	Null

(iii) Floating constants

They are also called real constants. They are numbers having fractional parts. They may be written in fractional from or exponent form. A real constant in fractional form consists of signed or unsigned digits including a decimal point between digits.

For example 3.0, -17.0, -0.627 etc.

A real constant in exponent form has two parts: a mantissa and an exponent. The mantissa is either an integer or a real constant followed by letter E or e and the exponent which must be an integer. For example 2E03, 1.23E07.

(iv) String Literals

A sequence of character enclosed within double quotes is called a string literal. String literal is by default (automatically) added with a special character '\0' which denotes the end of the string. Therefore the size of the string is increased by one character. For example "COMPUTER" will be represented as "COMPUTER\0" in the memory and its size is 9 characters.

3.6.4 Punctuators

The following characters are used as punctuators in C++. (Table 3.3)

Table 3.3: List of punctuators

Brackets [ ]	opening and closing brackets indicate single and multidimensional array subscript.
Parentheses ( )	opening and closing brackets indicate functions calls, function parameters for grouping expressions etc.
Braces { }	opening and closing braces indicate the start and end of a compound statement.
Comma ,	it is used as a separator in a function argument list.
Semicolon ;	it is used as a statement terminator.
Colon :	it indicates a labelled statement or conditional operator symbol.
Asterisk *	it is used in pointer declaration or as multiplication operator.
Equal sign =	it is used as an assignment operator.
Pound sign #	it is used as pre-processor directive.

3.6.5 Operators

Operators are special symbols used for specific purposes. C++ provides six types of operators.

- Arithmetical operators
- Relational operators
- Logical operators
- Unary operators
- Assignment operators
- Conditional operators
- Comma operator

Arithmetical operators

An operator that performs an arithmetic (numeric) operation: +, –, *, / , or %. For these operations always two or more than two operands are required. Therefore these operators are called binary operators. Table 3.4 shows the arithmetic operators.

Table 3.4 Arithmetic operators

Operators	Meaning	Example	Result
+	Addition	8 + 5	13
-	Subtraction	8 - 5	3
1	Multiplication	8 1 5	40
/	Division	8 / 5	1
%	Modulus/Remainder	8 % 5	3

Relational Operators

The relational operators are used to test the relation between two values. All relational operators are binary operators and therefore require two operands. A relational expression returns zero when the relation is false and a non-zero when it is true.

Table 3.5 shows the relational operators.

Table 3.5: Relational operators

Relational operators	Meaning
<	Less than
< =	Less than or equal to
= =	Equal to
>	Greater than
> =	Greater than or equal to
! =	Not equal

Example:

int x = 2; Int l = 1;

int y = 3;

int z = 5;

The following statements are true

(i) l = = 1

(ii) x < y

(iii) z > y

(iv) y > = 1

(v) x ! = 1

(vi) l < = 1

Logical Operators

The logical operators are used to combine one or more relational expression. Table 3.6 shows the logical operators.

Table 3.6: Logical operators

Operators	Meaning
||	OR
&&	AND
!	NOT

The NOT operator is called the unary operator because it requires only one operand.
Example

int x = 5; int z = 9; int y = 7;

(x > y) & & (z > y)

The first expression (x > y) evaluates to false and second expression (z > y) evaluates to true. Therefore, the final expression is false.
In AND operation, if any one of the expression is false, the entire expression is false.
In OR operation, if any one of the expression is true, the entire expression is true.
In NOT operation, only one expression is required.
If the expression is true, the NOT operation of true is false and vice versa.

Unary Operators

C++ provides two unary operators for which only one variable is required.
Example

a = - 50; a = - b;

a = + 50; a = + b;

Here plus sign (+) and minus sign (-) are unary because they are not used between two variables.

Assignment Operator

The assignment operator '=' stores the value of the expression on the right hand side of the equal sign to the operand on the left hand side.
Example

```
int m = 5, n = 7;
int x, y, z;
x = y = z = 0;
```

In addition to standard assignment operator shown above, C++ also supports compound assignment operators.
C++ provides two special operators viz '++' and '– –' for incrementing and decrementing the value of a variable by 1. The increment/decrement operator can be used with any type of variable but it cannot be used with any constant.
With the prefix version of these operators, C++ performs the increment or decrement operation before using the value of the operand. For instance the following code.

```
int sum, ctr;
sum = 12;
ctr = 4;
```

sum = sum + (++ctr);

will produce the value of sum as 17 because ctr will be first incremented and then added to sum producing value 17.
Similarly, the following code

sum = 12;

ctr = 4;

sum = sum + (- - ctr);

will produce the value of sum as 15 because ctr will be first decremented and then added to sum producing value 15.

With the postfix version of these operators, C++ first uses the value of the operand in evaluating the expression before incrementing or decrementing the operand's value.

For example, the following code

```
sum = 12;
ctr = 4;
sum = sum + (ctr + +);
```

will produce the value of sum as 16 because ctr will be first used in the expression producing the value of sum as 16 and then increment the value of ctr by 1 (ctr becomes now 5)

Similary, the following code

```
sum = 12;
ctr = 4;
```

sum = sum + (ctr – –) will produce the value of sum as 16 because ctr will be first used with its value 4 producing value of sum as 16 and then decrement the value of ctr by 1 (ctr becomes 3).

Let us study the use of compound assignment operators in the Table 3.7:

Table 3.7: Compound assignment operators

Operator	Example	Equivalent to
+ =	A + = 2	A = A + 2
- =	A - = 2	A = A - 2
% =	A % = 2	A = A % 2
/ =	A / = 2	A = A / 2
¹ =	A ¹ = 2	A = A ¹ 2

Example:

int x = 2; / / first

x + = 5; / / second

In the second statement, the value of x is 7.

Conditional Operator

The conditional operator ?: is called ternary operator as it requires three operands. The format of the conditional operator is:

Conditional_expression ? expression1 : expression2;

If the value of conditional_expression is true then the expression1 is evaluated,

otherwise expression2 is evaluated.

Example:

```
int a = 5;
int b = 6;
big = (a > b) ? a : b;
```

The condition evaluates to false, therefore big gets the value from b and it becomes 6.

Comma Operator

The comma operator gives left to right evaluation of expressions. It enables to put more than one expression separated by comma on a single line.

Example

```
int i = 20, j = 25;
int sq = i * i, cube = j * j * j;
```

In the above statements, comma is used as a separator between two statements / expressions.

3.7 Type Conversion in Expressions

When constants and variables of different types are mixed in an expression, they are all converted to the same type. The compiler converts all operands up to the type of the largest operand, which is called type promotion. First, all char and short int values are automatically elevated to int. (This process is called integral promotion.) Once this step has been completed, all other conversions are done operation by operation, as described in the following type conversion algorithm:

IF an operand is a long double
THEN the second is converted to long double
ELSE IF an operand is a double
THEN the second is converted to double
ELSE IF an operand is a float
THEN the second is converted to float
ELSE IF an operand is an unsigned long
THEN the second is converted to unsigned long
ELSE IF an operand is long
THEN the second is converted to long
ELSE IF an operand is unsigned int
THEN the second is converted to unsigned int

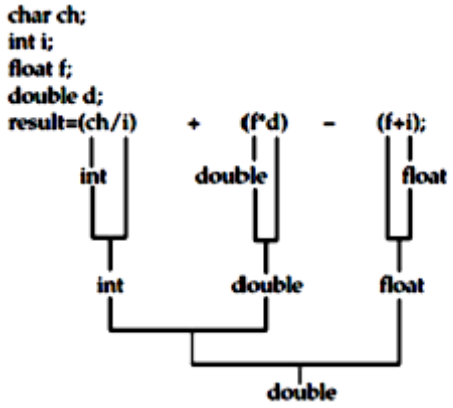


Figure 3.1: A type conversion.

There is one additional special case: If one operand is long and the other is unsigned int, and if the value of the unsigned int cannot be represented by a long, both operands are converted to unsigned long. Once these conversion rules have been applied, each pair of operands is of the same type and the result of each operation is the same as the type of both operands. For example, consider the type conversions that occur in Figure 3.1. First, the character ch is converted to an integer. Then the outcome of ch/i is converted to a double because f*d is double. The outcome of f+i is float, because f is a float. The final result is double.

The process in which one pre-defined type of expression is converted into another type is called conversion. There are two types of conversion in C++.

- Implicit conversion
- Explicit conversion

3.7.1 Implicit Conversion

Data type can be mixed in the expression.

Example:

```
double a;
int b = 5;
float c = 8.5;
a = b * c;
```

When two operands of different type are encountered in the same expression, the lower type variable is converted to the higher type variable. Figure 3.2 shows the order of data types.

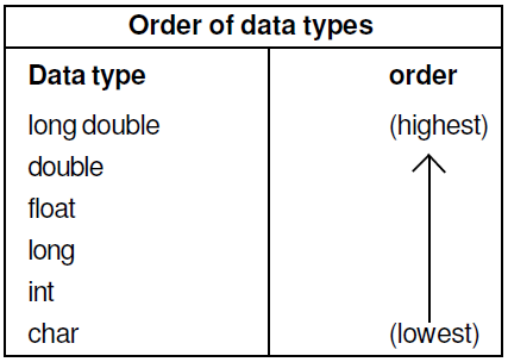


Figure 3.2: Order of data types.

The int value of b is converted to type float and stored in a temporary variable before being ultiplied by the float variable c. The result is then converted to double so that it can be assigned to the double variable a.

Self Assessment Questions

6. Integer constants are whole numbers without any fractional part.

(a) True (b) False

7. Decimal integer constants consist of sequence of digits and should begin with 0 (zero).

(a) True (b) False

8. In AND operation, only one expression is required.

(a) True (b) False

3.7.2 Explicit Conversion

It is also called type casting. It temporarily changes a variable data type from its declared data type to a new one. It may be noted here that type casting can only be done on the right hand side of the assignment statement.

C++ is a strong-typed language. Many conversions, especially those that imply a different interpretation of the value, require an explicit conversion.

We have already seen two notations for explicit type conversion: functional and c-like casting:

1 *short* a=2000;

```
2 int b;
3 b = (int) a;    // c-like cast notation
4 b = int (a);    // functional notation
```

The functionality of these explicit conversion operators is enough for most needs with fundamental data types.
For example, the following code is syntactically correct:

```
// class type-casting
#include <iostream>
using namespace std;
class CDummy {
   float i,j;
};
class CAddition {
        int x,y;
  public:
        CAddition (int a, int b) { x=a; y=b; }
        int result() { return x+y;}
};
int main () {
  CDummy d;
  CAddition * padd;
  padd = (CAddition*) &d;
  cout << padd->result();
  return 0;
}
```

The program declares a pointer to CAddition, but then it assigns to it a reference to an object of another incompatible type using explicit type-casting:

```
 padd = (CAddition*) &d;
```

Traditional explicit type-casting allows to convert any pointer into any other pointer type, independently of the types they point to. The subsequent call to member result will produce either a run-time error or a unexpected result.
In order to control these types of conversions between classes, we have four specific casting operators: dynamic_cast, reinterpret_cast, static_cast and const_cast. Their format is to follow the new type enclosed between angle-brackets (<>) and immediately after, the expression to be converted between parentheses.

```
dynamic_cast <new_type> (expression)
reinterpret_cast <new_type> (expression)
static_cast <new_type> (expression)
const_cast <new_type> (expression)
```

The traditional type-casting equivalents to these expressions would be:

```
(new_type) expression
new_type (expression)
```

Caution

The explicit conversion operators can be applied indiscriminately on classes and pointers to classes, which can lead to code that while being syntactically correct can cause runtime errors.

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Write a C++ program to convert a floating point number to integer number.

……..……

……..……

……...

Exercise: Check Your Progress 3

Case Study-String Indexing

The purpose of this appendix is to show at some length how a string indexing system is used to create an index display. The particular index displays chosen are the five-year cumulative indexes to The Canadian Journal of Information Science.

A major reason for choosing the CJIS cumulative indexes is the fact that the string indexing system used to produce them was NEPHIS. NEPHIS is far from being an ideal string indexing system, but it does have a number of features useful for purposes of illustration. First, the input strings are coded, and indexer control over index string syntax can readily be shown. Second, index string generation is neither too simple nor too complex: on the one hand, the procedure is sufficiently sophisticated to demonstrate some features of more sophisticated index string generators; on the other hand, it is sufficiently unsophisticated that it can be followed by the reader without extensive instruction. Third, NEPHIS software is readily available, notably versions for Commodore and IBM-PC-compatible microcomputers; and readers may care to follow the examples, and try alternatives of their own, with the aid of their own machines. Finally, of course, NEPHIS was designed by the author, who is thus uniquely familiar with its operation.

Other reasons for choosing the CJIS indexes as the case to be studied include the fact that the author was the indexer and index producer; the fact that the subject matter of the indexed items is likely to be somewhat familiar to many readers.

Preparation for indexing CJIS

No previous index to CJIS had been produced, and the index producer was not required to adhere to a set index display format.

The index producer found no existing controlled vocabulary suited to NEPHIS indexing of materials in the field of information science. Accordingly, vocabulary control was carried out parallel to the actual indexing. The vocabulary used was based partly on that of the indexed items and partly on the experience of the index producer. Rather than using NEPHIS' capacity for defining lead-only and alternate terms, it was decided to employ cross-references. Including lead-only and alternate terms in the input strings would have proved too clumsy in some cases.

Indexing CJIS

What follows will consider, item by item. The purpose is to demonstrate in some depth the kinds of decisions faced by the string indexer the NEPHIS input string or strings assigned, the resulting index strings, and discussion of problems and alternatives.

Questions:

1. What the CJIS stands for?

2. What is the string indexing?

3.8 Summary

- C++ is currently one of the most popular third generation languages.
- C++ is a case sensitive programming language.
- A function is a collection of statements that executes sequentially. Every C++ program must contain a special function called main ().
- A C++ program is written using these tokens, white spaces, and the syntax of the language.
- Whitespace is the term used in C++ to describe blanks, tabs, newline characters and comments.
- Libraries are groups of functions that have been "packaged up" for reuse in many different programs.

3.9 Keywords

Function: A function is a collection of statements that executes sequentially.
Identifiers: A C++ identifier is a name used to identify a variable, function, class, module, or any other user-defined item.
Libraries: Libraries are groups of functions that have been "packaged up" for reuse in many different programs.
Token: A token is a group of characters that logically belong together.
Trigraph: A trigraph is a three-character sequence that represents a single character and the sequence always starts with two question marks.
White space: Whitespace is the term used in C++ to describe blanks, tabs, newline characters and comments.

3.10 Review Questions

1. Explain the Turbo C++ IDE environment.
2. How do we create, compile and run a program through IDE and command line?
3. What are the C++ elements?
4. Differentiate between trigraph and identifiers.
5. Write the structure of a C++ program.
6. What are the C++ tokens?
7. Explain the literals and punctuators.
8. Explain the type conversion.
9. Write a C++ program to convert a data type implicitly.
10. Differentiate between explicit and implicit conversion.

Answers for Self Assessment Questions

1 (b)	2 (d)	3 (d)	4 (c)
5 (a)	6 (a)	7 (b)	8 (b)

3.11 Further Studies

1. http://en.wikibooks.org/wiki/C_Programming/Using_a_Compiler
2. The C++ Programming Language 4 Edition by Bjarne Stroustrup

4

Decision Making and Branching

STRUCTURE

4.0 Objectives

4.1 Introduction

4.2 Sequential Statement

4.3 Mathematical Functions

4.4 Branching Statements

4.5 Looping Statements

4.6 Nested Loops

4.7 Summary

4.8 Keywords

4.9 Review Questions

4.10 Further Studies

4.0 Objectives

After studying this chapter, you will be able to:

- Explain the sequential statements
- Discuss the mathematical functions
- Explain the branching and looping statements
- Discuss the nested loops

4.1 Introduction

C++ Language must be able to perform different sets of actions depending on the circumstances. A C++ program is a set of statements which are normally executed sequentially n the order in which they appear. This happens when no options or no repetitions of certain calculations are necessary.
However, in practice, we have a number of situations where we may have to change the order of execution of statements based on certain conditions; repeat a group of statements until certain specified conditions are met. This involves a kind of decision making to see whether a particular condition has occurred or not and then direct the computer to execute certain statements accordingly.
C++ has 4 decision making instructions, they are:
1. If-else statement

2. Switch statement
3. Conditional operator statement
4. Goto statement.

4.2 Sequential Statement

Sequential statements begin by stating a condition. This condition represents the Boolean logic binary choice, and begins the execution of the statement based on the value determined by the condition. For example, the condition may represent “true” or “false,” and whether the data or information is, indeed, true or false determines the nest sequential statement, leading the data or information through the rest of the process.

There are given following statement:

- If else statement
- Switch statement

4.2.1 If Statement

If statement is a powerful decision making statement and is used to control the flow of execution of statements. It is basically a two-way decision statement and is used in conjunction with an expression. It takes the following form:

if (test expression)

It allows the computer to evaluate the expression first and then, depending on whether the value of the expression (relation or condition) is “true” (non-zero) or “false” (zero), it transfers the control to a particular statement.

This point of program has two paths to follow, one for the true condition and the other for the false condition as shown in Figure 4.1.

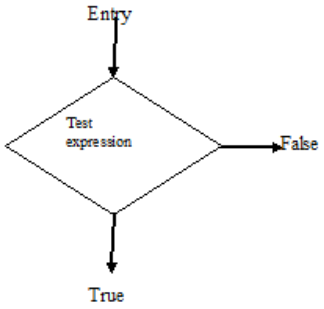


Figure 4.1: Two-way branching.

The if statement may be implemented in different forms depending on the complexity of conditions to be tested.

Simple if statement

If else statement

nested if....else statement

else if ladder.

Simple If Statement

The general form of a simple if statement is

```
if(test expression)
{
statement-block;
```

}
statement-x;

The “statement-block” may be a single statement or a group of statements. If the test expression is true, the statement-block will be executed; otherwise the statement-block will be skipped and the execution will jump to the statement-x. Remember, when the condition is true both the statement-block and the statement-x are executed in sequence. This is illustrated in Figure 4.2.

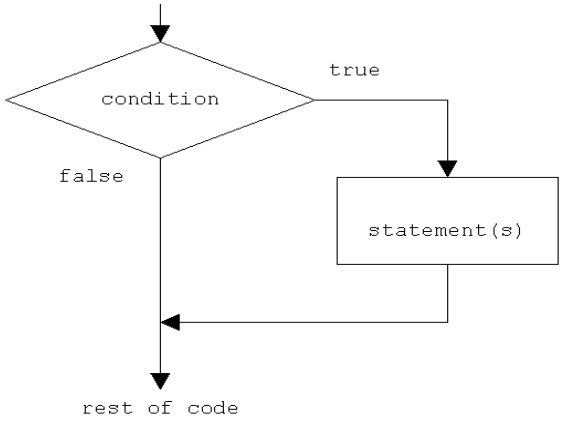


Figure 4.2: Flowchart of simple it controls.

If Else Statement

The if else statement is an extension of the simple if statement.

if (test expression)
{
True-block statement(s)
}
else
{
False-block statement(s)
}
statement-xs

If the test expression is true, then the true-block statement (S), immediately following the if statement is executed, otherwise, the false-block (S) is executed. In either case, either true-block or false-block will be executed, not both. This is illustrated in Figure.4.3 in both the cases, the control is transferred subsequently to statement-x.

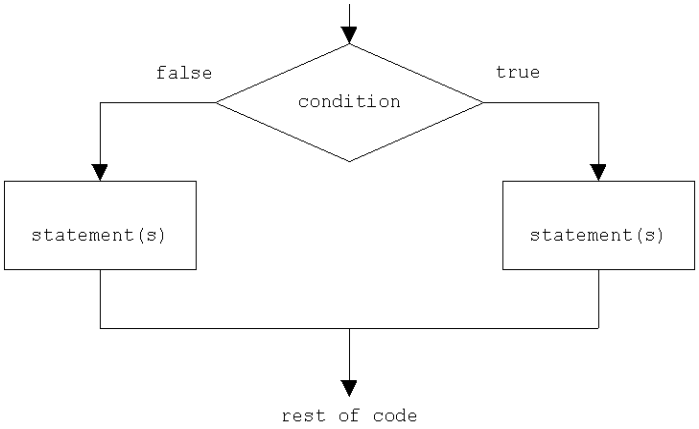
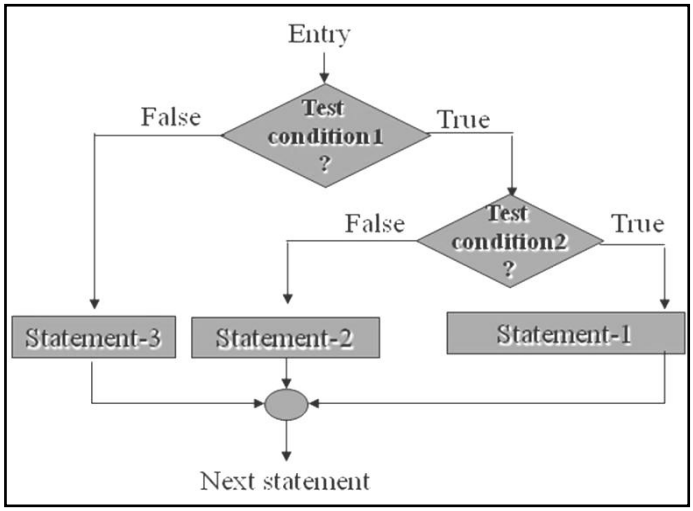


Figure 4.3: Flowchart of if else control.

Nesting of If Else Statements

When a series of decisions are involved, we may have to use more than one if else statement in nested form as follows:

Figure 4.4: Flowchart of nested lf else statements.



The logic of execution is illustrated in Figure 4.4. If the condition 1 is false, the statement 3 will be executed; otherwise it continues to perform the second test. If the condition 2 is true, the statement 1 will be evaluated; otherwise the statement 2 will be evaluated and then the control is transferred to the statement x.

Example:

```
#include <iostream>
#include <iomanip>
using namespace std;
int main ()
{
int first, second;
cout << “Enter two integers.” << endl;
cout << “First” << setw (3) << “: ”;
cin >> first;
cout << “Second ”<< setw (2) << “: ”;
cin >> second;
 if (first > second)
cout << “first is greater than second.” << endl;
return 0;
}
```

4.2.2 Switch Statement

We have seen that when one of the many alternatives is to be program using if statements to control the selection. However, the complexity of such a program increases dramatically when the number of alternatives increases the program becomes difficult to read and follow. At times, it may confuse even the person who designed it. Fortunately, C++ has a built-in multi way decision statement known as a “switch”. The switch statement tests the value of a given variable (or expression) against a list of case values and when a match is found. A block of statements associated with that case is executed. The general form of the switch statement is as shown below: The expression is an integer expression

or characters. Value l, Value 2 is constants or constant expressions (evaluable to an integral constant). Each of these values should be unique within a switch statement. The block l, block 2 are statement lists and may contain zero or more statements. There is no need to put braces around these blocks. Note that case labels end with a colon (:).
When the switch is executed, the value of the expression is successively compared against the values value 1, value 2,....If a case is found whose value matches with the value of the expression, then the block of statements that follows the case are executed.

```
switch (expression)
{
case value 1;
block 1
break;
case value 2
block 2
break;
default:
default-block
break;
}
statement x
```

The break statement at the end of each block signal the end of a particular case and causes an exit from the switch statement, transferring the control to the statement-x following the switch.
The default is an optional case. When present, it will be executed if the value of the expression does not match with any of the case values. If not present, no action takes place if all matches fail and the control goes to the statement-x.

Example:

```
#include <iostream>
using namespace std;
int main ()
{
char permit;
cout << “Are you sure you want to quit? (y/n) : ”;
cin >> permit;
switch (permit)
{
case “y” :
cout << “Hope to see you again!” << endl;
break;
case “n” :
cout << “Welcome back!” < < endl;
break;
default:
cout << “What? I do not get it!” << endl;
}
```

```
return 0;
}
```

Did You Know?
The release 2.0 of C++ came in 1989 and the updated second edition of the C++ programming language was released in 1991.

Caution
In the switch statement, the default case does not have to appear at the end. It can appear anywhere in the switch block.

4.3 Mathematical Functions

The standard function library contains several mathematical functions, which fall into the following categories:

- Trigonometric functions
- Hyperbolic functions
- Exponential and logarithmic functions
- Miscellaneous functions

All the math functions require the header <cmath>. (C++ programs must use the header file math.h.) In declaring the math functions, this header defines the macro called HUGE_VAL. The macros EDOM and ERANGE are also used by the math functions. These macros are defined in the header <cerrno> (or the file errno.h). If an argument to a math function is not in the domain for which it is defined, an implementation-defined value is returned, and the built-in global integer variable errno is set equal to EDOM. If a routine produces a result that is too large to be represented, an overflow occurs. This causes the routine to return HUGE_VAL, and errno is set to ERANGE, indicating a range error. If an underflow happens, the function returns zero and sets errno to ERANGE. All angles are in radians. Originally, the mathematical functions were specified as operating on values of type double, but Standard C++ added overloaded versions to explicitly accommodate values of type float and long double.
The operation of the functions is otherwise unchanged.

acos

```
#include <cmath>
float acos(float arg);
double acos (double arg);
long double acos(long double arg);
```

The acos() function returns the arc cosine of *arg*. The argument to acos() must be in the range -1 to 1; otherwise a domain error will occur.
Related functions are asin(), atan(), atan2(), sin(), cos(), tan(), sinh(), cosh(), and tanh().

asin

```
#include <cmath>
float asin(float arg);
double asin(double arg);
long double asin(long double arg);
```

The asin() function returns the arc sine of arg. The argument to asin() must be in the range –1 to 1; otherwise a domain error will occur. Related functions are acos(), atan(), atan2(), sin(), cos(), tan(), sinh(), cosh(), and tanh().

atan

#include <cmath>
float atan(float arg);
double atan(double arg);
long double atan(long double arg);
The atan() function returns the arc tangent of arg. Related functions are asin(), acos(), atan2(), tan(). cos(), sin(), sinh(). cosh(). and tanh().

atan2

#include <cmath>
float atan2(float y, float x);
double atan2(double y, double x);
long double atan2(long double y, long double x);
The atan2() function returns the arc tangent of y/x. It uses the signs of its arguments to compute the quadrant of the return value. Related functions are asin(). acos(). atan(). tan(), cos(). sin(), sinh(). cosh().
and tanh().

ceil

#include <cmath>
float ceil(float num);
double ceil(double num);
long double ceil(long double num);
The ceil() function returns the smallest integer (represented as a floating-point value) not less than num. For example, given 1.02, ceil() would return 2.0. Given –1.02, ceil() would return –1. Related functions are floor() and fmod().

cos

#include <cmath>
float cos(float arg);
double cos(double arg);
long double cos(long double arg);
The cos() function returns the cosine of arg. The value of arg must be in radians. Related functions are asin(), acos(), atan2(), atan(), tan(), sin(), sinh(), cos(), and tanh().

cosh

#include <cmath>
float cosh(float arg);
double cosh(double arg);
long double cosh(long double arg);
The cosh() function returns the hyperbolic cosine of arg. Related functions are asin(), acos(), atan2(), atan(), tan(), sin(), cosh(), and tanh().

exp
#include <cmath>
float exp(float arg);
double exp(double arg);
long double exp(long double arg);
The exp() function returns the natural logarithm base e raised to the arg power. A related function is log().

fabs
#include <cmath>
float fabs(float num);
double fabs(double num);
long double fabs(long double num);
The fabs() function returns the absolute value of num. A related function is abs().

floor
#include <cmath>
float floor(float num);
double floor(double num);
long double floor(long double num);
The floor() function returns the largest integer (represented as a floating-point value) not greater than num. For example, given 1.02, floor() would return 1.0. Given –1.02, floor() would return –2.0. Related functions are fceil() and fmod().

fmod
#include <cmath>
float fmod(float x, float y);
double fmod(double x, double y);
long double fmod(long double x, long double y);
The fmod() function returns the remainder of x/y. Related functions are ceil(), floor(), and fabs().

frexp
#include <cmath>
float frexp(float num, int * exp);
double frexp(double num, int * exp);
long double frexp(long double num, int * exp);
The frexp() function decomposes the number num into a mantissa in the range 0.5 to less than 1, and an integer exponent such that num = mantissa * 2exp. The mantissa is returned by the function, and the exponent is stored at the variable pointed to by exp. A related function is ldexp().

ldexp
#include <cmath>
float ldexp(float num, int exp);
double ldexp(double num, int exp);
long double ldexp(long double num, int exp);
The ldexp() returns the value of num * 2exp. If overflow occurs, HUGE_VAL is returned. Related functions are frexp() and modf().

log
#include <cmath>
float log(float num);
double log(double num);
long double log(long double num);
The log() function returns the natural logarithm for num. A domain error occurs if num is negative, and a range error occurs if the argument is zero. A related function is log10().

log10
#include <cmath>
float log10(float num);
double log10(double num);
long double log10(long double num);
The log10() function returns the base 10 logarithm for num. A domain error occurs if num is negative, and a range error occurs if the argument is zero. A related function is log().

modf
#include <cmath>
float modf(float num, float * i);
double modf(double num, double * i);
long double modf(long double num, long double * i);
The modf() function decomposes num into its integer and fractional parts. It returns the fractional portion and places the integer part in the variable pointed to by i. Related functions are frexp() and ldexp().

pow
#include <cmath>
float pow(float base, float exp);
float pow(float base, int exp);
double pow(double base, double exp);
double pow(double base, int exp);
long double pow(long double base, long double exp);
long double pow(long double base, int exp);
The pow() function returns base raised to the exp power ($base^{exp}$). A domain error may occur if base is zero and exp is less than or equal to zero. It will also happen if base is negative and exp is not an integer. An overflow produces a range error. Related functions are exp(), log(), and sqrt().

sin
#include <cmath>
float sin(float arg);
double sin(double arg);
long double sin(long double arg);
The sin() function returns the sine of arg. The value of arg must be in radians.
Related functions are asin(), acos(), atan2(), atan(), tan(), cos(), sinh(), cosh(), and tanh()

sinh
#include <cmath>

float sinh(float arg);
double sinh(double arg);
long double sinh(long double arg);
The sinh() function returns the hyperbolic sine of arg. Related functions are asin(), acos(), atan2(), atan(), tan(), cos(), tanh(), cosh(), and sin().

sqrt
#include <cmath>
float sqrt(float num);
double sqrt(double num);
long double sqrt(long double num);
The sqrt() function returns the square root of num. If it is called with a negative argument, a domain error will occur. Related functions are exp(), log(), and pow().

tan
#include <cmath>
float tan(float arg);
double tan(double arg);
long double tan(long double arg);
The tan() function returns the tangent of arg. The value of arg must be in radians. Related functions are acos(), asin(), atan(), atan2(), cos(), sin(), sinh(), cosh(), and tanh().

tanh
#include <cmath>
float tanh(float arg);
double tanh(double arg);
long double tanh(long double arg);

The tanh() function returns the hyperbolic tangent of arg. Related functions are acos(), asin(), atan(), atan2(), cos(), sin(), cosh(), sinh(), and tan().

Self Assessment Questions

1. Sequential statements begin by starting condition.
(a) True (b) False

2. The..... statement is a powerful decision making statement and is used to control the flow of execution of statements.
(a) if (b).break
(c) return (d) None of these.

3. The statement-block may be.....statements.
(a) single (b) group
(c) Both (a) and (b) (d) None of these

4.4 Branching Statements

The branching statements are also called decision making statement which is used to see whether a particular condition has occurred or not and then tell the compiler to execute certain statements accordingly.

- Break statement
- Continue statement
- Return statement

4.4.1 Break Statement

Technically, the break statements inside the switch statement are optional. They terminate the statement sequence associated with each constant. If the break statement is omitted, execution will continue on into the next case's statements until either a break or the end of the switch is reached. For example, the following function uses the "drop through" nature of the cases to simplify the code for a device-driver input handler:
The user's goal is to put in a target number for the large number that will stop the game.
break statement

```
// demonstrates break statement
#include <iostream.h>
int main()
{
unsigned short small;
unsigned long large;
unsigned long skip;
unsigned long target;
const unsigned short MAXSMALL=65535;
cout << "Enter a small number:";
cin >> small;
cout << "Enter a large number:";
cin >> large;
cout << "Enter a skip number:";
cin >> skip;
cout << "Enter a target number:";
cin >> target;
cout << "\n";
// set up 3 stop conditions for the loop
while (small < large && large > 0 && small < 65535)
{
small++;
if (small % skip == 0) // skip the decrement?
{
cout << "skipping on" << small << endl;
continue;
}
if (large == target)   // exact match for the target?
{
cout << "Target reached!";
break;
```

```
}
large-=2;
}                    // end of while loop
cout << "\nSmall:" << small << "Large:" << large << endl;
return 0;
}
```

Output

```
Enter a small number: 2
Enter a large number: 20
Enter a skip number: 4
Enter a target number: 6
skipping on 4
skipping on 8
Small: 10 Large: 8
```

4.4.2 Continue Statement

The continue statement forces the next iteration of the loop to take place, skipping any code following the continue statement in the loop body. In for loop, the continue statement causes the conditional test and then the re-initialization portion of the loop to be executed. In the while and do-while loops, program control passes to the conditional test. The following program illustrates the usage of the continue statement:

```
//This program finds the square of the numbers less than 100
#include<iostream.h>
void main()
{
int iNum;
char cReply = "y";
do
{
cout<< "Enter a number:";
cin>>iNum;
if(iNum > 100)
{
cout<< "The number is greater than 100, enter another"<<endl;
continue;
}
cout<< "The square of the number is:" <<iNum * iNum<<endl;
cout<< "Do you want to enter another(y/n)?";
cin>>cReply;
}
while(cReply != "n");
}
```

Caution

The continue statement is only valid inside a loop. If you write it outside the loop, your program will generate a compile error.

4.4.3 Return Statement

The expression clause, if present, is converted to the type specified in the function declaration, as if an initialization were being performed. Conversion from the type of the expression to the return type of the function can create temporary objects. For more information about how and when temporaries are created.

The value of the expression clause is returned to the calling function. If the expression is omitted, the return value of the function is undefined. Constructors and destructors, and functions of type void, cannot specify an expression in the return statement. Functions of all other types must specify an expression in the return statement.

When the flow of control exits the block enclosing the function definition, the result is the same as it would be if a return statement without an expression had been executed. This is invalid for functions that are declared as returning a value.

A function can have any number of return statements.

The following example uses an expression with a return statement to obtain the largest of two integers.

Examples of return statements. The following are examples of return statements:

```
return;           /* Returns no value           */
return result;    /* Returns the value of result */
return 1;         /* Returns the value 1        */
return (x * x);   /* Returns the value of x * x  */
```

The following function searches through an array of integers to determine if a match exists for the variable number. If a match exists, the function match returns the value of i. If a match does not exist, the function match returns the value -1 (negative one).

```
int match(int number, int array[ ], int n)
{
int i;
for (i = 0; i < n; i++)
if (number == array[i])
return (i);
return(-1);
}
```

A function can contain multiple return statements. For example:

```
void copy( int *a, int *b, int c)
{
  /* Copy array a into b, assuming both arrays are the same size */
if (!a || !b)       /* if either pointer is 0, return */
return;
if (a == b)         /* if both parameters refer */
return;          /*    to same array, return */
if (c == 0)         /* nothing to copy */
return;
for (int i = 0; i < c; ++i;) /* do the copying */
b[i] = a[1];
```

```
/* implicit return */
}
```

4.5 Looping Statements

In looping statement a sequence of statement executed repeatedly until some conditions for termination are satisfied. The condition for the loop is tested and if satisfied then it executes the body of the loop.
Generally there are three types of looping statements:

- While loop
- Do-while loop
- For loop

4.5.1 While Loop Statements

While loop: While loop is the simplest form of loop, and it is entry controlled loop statement.
Syntax:

```
while(test condition)
{
Body of loop
}
```

At the first, condition is evaluated and if it is true then the body of loop is executed and then again the condition is evaluated and if it is true then the body of loop executed again. This process goes again and again until the condition is not false. If the condition goes false then the control is exit out of loop.

Program to find reverse of a number

```
#include<iostream>
#include<conio.h>
void main()
{
int n,rev=0,rem,temp;
clrscr();
cout<< “Enter any number:”;
cin>>n;
temp=n;
while(temp>0)
{
rem=temp%10;
temp=temp/10;
rev=rev*10+rem;
}
cout<< “Reverse of”<<n << “is”<<rev;
getch();
}
```

4.5.2 Do-while Statements

In while loop the condition is evaluated first and then executes the body of loop and if the condition is not satisfied even at the very first attempt the body of the loop is not executed.
Syntax:

```
do
{
Body of loop
}
while(condition);
```

In the above syntax the body of the loop is must be executed even the condition is not satisfied because in this the condition is evaluated after the body, so it is provides exit-control loop.

4.5.3 for Loop Statements

The for loop is also entry-controlled loop.
Syntax:

```
for(initialization; condition; increment/decrement)
{
Body of loop
}
```

Initialization: in this part of the loop the variable is initialized such as i=0; and this is the control variable for the loop, which tells from where the loop is starting.
Condition: The condition is a relational expression and the value of the variable is evaluated using some condition for exampole i<=5;
Increment/decrement: After the execution of the statements which are in the body of the loop the control will again transfer to the loop and then the value of the variable is incremented. This value can be incremented by one or more according to our need. If we want to increment variable with one then we can use i++;

Example: Program to check whether a given number is prime or not.

```
#include<conio.h>
#include<iostream>
class prime_no
{
int n, i, f;
public:
void input();
void output();
};
void prime_no::input()
{
cout<< “Enter any number:”;
cin>>n;
}
```

```
void prime_no:output ()
{
f=0;
for(i=2;i<=(n/2);i++)
{
if(n%i==0)
f=1;
}
if(f==0)
cout<<n<< " is a prime number";
else
cout<<n<< "is not a prime number";
}
void main()
{
prime_no p1;
clrscr();
p1.input();
p1.output();
getch();
}
```

Caution
The do-while loop will be executed at least once, whereas, depending upon the expression, the while loop may not be executed at all.

Did You Know?
The vertical loop is not a recent roller coaster innovation. Its origins can be traced back to the 1850s when centrifugal railways were built in France and Great Britain.

4.6 Nested Loops

The syntax for a nested for loop statement in C++ is as follows:

```
for ( init; condition; increment )
{
for ( init; condition; increment)
{
statement(s);
}
statement(s); // you can put more statements
}
```

The syntax for a nested while loop statement in C++ is as follows:

```
while(condition)
{
while(condition)
{
```

```
    statement(s);
  }
  statement(s); // you can put more statements.
}
```

The syntax for a nested do. while loop statement in C++ is as follows:

```
do
{
  statement(s); // you can put more statements.
  do
  {
    statement(s);
  }while( condition );
}while( condition );
```

Example:

The following program uses a nested for loop to find the prime numbers from 2 to 100:

```
#include <iostream>
using namespace std;
 int main ()
{
int i, j;
for(i=2; i<100; i++) {
for(j=2; j <= (i/j); j++)
if(!(i%j)) break; // if factor found, not prime
if(j > (i/j)) cout << i << " is prime\n";
}
return 0;
}
```

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Write a program to demonstrate the use of switch statement in C++ by taking character input.

……..……

……..……

……...

Exercise: Check Your Progress 2

Ex1: Case Study-Decision Making and Branching in C++

……..……

……..……

……...

Applying De Morgan's Rule

While designing decision statements, we often come across a situation where the logical NOT operator is applied to a compound logical expression, like !(a&&b||!c). However, a positive logic is

always easy to read and comprehend than a negative logic. In such cases, we may apply what is known as De Morgan's rule to make the total expression positive. This rule is as follows:
"Remove the parentheses by applying the NOT operator to every logical expression component, while complementing the relational operators"

That is,
a becomes !a
!a becomes a
&& becomes ||
|| becomes &&

Examples:
!(a && b || !c) becomes !a || !b && c
!(a <= 0 || !condition) becomes a > 0 && condition

Dangling Else Problem

One of the classic problems encountered when we start using nested if else statements is the dangling else. This occurs when a matching else is not available for if. The answer to this problem is very simple. Always match an else to the most recent unmatched if in the current block. In some cases, it is possible that the false condition is not required. In such situations, else statement may be omitted.
"else is always paired with the most recent unpaired if"

Rules for Indentation

When using control structures, a statement often controls many other statements that follow it. In such situations it is good practice to use indentation to show that the indented statements are dependent on the preceding controlling statement. Some guidelines that could be followed while using indentation are listed below:

- Indent statements that are dependent on the previous statements; provide at least three spaces of indentation.
- Align vertically else clause with their matching if clause.
- Use braces on separate lines to identify a block of statements.
- Indent the statements in the block by at least three spaces to the right of the braces.
- Align the opening and closing braces.
- Use appropriate comments to signify the beginning and end of blocks.
- Indent the nested statements as per the above rules.
- Code only one clause or statement on each line.

Rules for Switch Statement

- The switch expression must be an integral type.
- Case labels must be constants or constant expressions.
- Case labels must be unique. No two labels can have the same value.
- Case labels must end with semicolon.
- The break statement transfers the control out of the switch statement.
- The break statement is optional. That is, two or more case labels may belong to the same statements.

- The default label is optional. If present, it will be executed when the expression does not find a matching case label.
- There can be at most one default label.
- The default may be placed anywhere but usually placed at the end.
- It is permitted to nest switch statements.

Question

1. What are rules for switch statement?
2. Discuss the De Morgan's rule for decision making.

Self Assessment Questions

4. Branching statement is contain.....

(a) break statement (b) Continue

(c) return statement (d) All of these

5. The condition for the loop is tested and if satisfied then it executes the body of the loop.

(a) True (b) False

6. Switch-Case Statement is a multi-way decision making statement

(a) True (b) False

4.7 Summary

- A sequential statement is used in the body of the process described in the C++.
- If statement is a powerful decision making statement and is used to control the flow of execution of statements.
- In while loop is the simple form of loop, and it is entry controlled loop statement.
- In for loop, the continue statement causes the conditional test and then the re-initialization portion of the loop to be executed.
- C++ has a built-in multi way decision statement known as a switch.

4.8 Keywords

Decision-making: The statements are needed to alter the sequence of the statements in the program depending upon certain circumstances.

***Do-while*:** The do-while loop is mostly used for writing menu driven programs.

Loops: It is used to repeat a block of code. Being able to have your program repeatedly execute a block of code is one of the most basic but useful tasks in programming.

Many Programs: It is require that a group of instructions be executed repeatedly, until some particular condition has been satisfied. This process is known as looping.

Switch-Case Statement: It is a multi-way decision making statement.

4.9 Review Questions

1. Define the sequential statements.
2. Explain the mathematical functions.

3. What are the branching statements?
4. Differentiate between nesting and if-else statements?
5. Explain the looping statements.
6. Define the nested loops.
7. Write a program uses a nested for loop to find the prime numbers from 2 to 100.
8. What are the decision making and branching?
9. Differentiate between sequential statements and control flow?
10. Define the do-while loop statement.

Answers for Self Assessment Questions

1 (a) 2 (a) 3 (c) 4 (d) 5 (a) 6 (a)

4.10 Further Studies

1. http://my.safaribooksonline.com/book/programming/cplusplus/0672322234/branching-statements-and-logical-operators/ch06
2. http://www.boloji.com/index.cfmmd=Content&sd=Articles&ArticleID=561
3. The C++ Programming Language 4 Edition by Bjarne Stroustrup

5

Arrays and Functions

STRUCTURE

5.0 Objectives

5.1 Introduction

5.2 Arrays

5.3 User Defined Functions

5.4 Return Values and Their Types

5.5 Function Calls

5.6 Passing Parameters to Functions

5.7 Recursion

5.8 Command Line Arguments

5.9 Storage Class Specifies

5.10 Summary

5.11 Keywords

5.12 Further Studies

5.0 Objectives

After studying this chapter, you will be able to:

- Explain the arrays
- Discuss the User Defined Functions
- Define the return values and their types
- Explain the function calls
- Discuss the passing parameters to functions

5.1 Introduction

An array is a data type used to represent a large number of values of the same type. An array might be used to represent all the salaries in a company or all the weights of participants in a fitness program. Each element in an array has a position, with the initial element having position zero. An array element's position can be used as an index or subscript to access that element. The elements of an array are randomly accessible through the use of subscripts.

A function is a group of statements that together perform a task. Every C++ program has at least one function which is main(), and all the most trivial programs can define additional functions.
It can divide up ones code into separate functions. How it divides up ones code among different functions is up to you, but logically the division usually is so each function performs a specific task.

5.2 Arrays

5.2.1 The Meaning of an Array

An array is a collection of elements of the same data type are reference by a common name. Each element of an array can be referred to by an array name and a subscript or index. Arrays can be one-dimensional or two-dimensional.
Arrays of all types are possible, including arrays of arrays. A typical array declaration allocates memory starting from a base address. An array name is, in effect, a pointer constant to this base address.

To illustrate some of these ideas, let us write a small program that fills an array, prints out values, and sums the elements of the array:

```
// Simple array processing
const int SIZE = 5;
int main()
{
int a[SIZE]; // get space for a[0],·····,a[4]
int sum = 0;
for (int i = 0; i < SIZE; ++i) {
a[i] = i * i;
cout << “a[” << i << “] = ” << a[i] << “ ”;
sum += a[i];
}
cout << “\nsum = ” << sum << endl;
}
```

The output of this program is

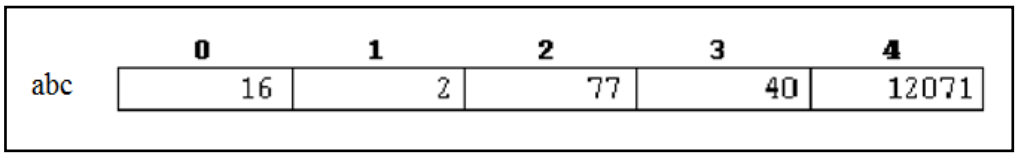
```
a[0] = 0 a[1] = 1 a[2] = 4 a[3] = 9 a[4] = 16
sum = 30
```

Initializing arrays

When declaring a regular array of local scope (within a function, for example), if we do not specify otherwise, its elements will not be initialized to any value by default, so their content will be undetermined until we store some value in them. The elements of global and static arrays, on the other hand, are automatically initialized with their default values, which for all fundamental types this means they are filled with zeros.
In both cases, local and global, when we declare an array, we have the possibility to assign initial values to each one of its elements by enclosing the values in braces { }. For example:

```
int abc [5] = { 16, 2, 77, 40, 12071 };
```

This declaration would have created an array like this:



The amount of values between braces { } must not be larger than the number of elements that we declare for the array between square brackets [ ]. For example, in the example of array abc we have declared that it has 5 elements and in the list of initial values within braces { } we have specified 5 values, one for each element.
When an initialization of values is provided for an array, C++ allows the possibility of leaving the square brackets empty [ ]. In this case, the compiler will assume a size for the array that matches the number of values included between braces { }:

int abc [] = { 16, 2, 77, 40, 12071 };

After this declaration, array abc would be 5 int long, since we have provided 5 initialization values.

5.2.2 Declaring One-Dimensional Arrays
The general form used to declare a single-dimensional array is:
type var-name[size];
Like other variables an array must be explicitly defined so that the compiler can allocate memory for the array. In the above declaration, the keyword type defines the data type of the array, which is the data type of each element in the array int, float, char and size defines the number of elements the array can hold.
An example of array declaration is:

char cArr[10];
The array subscript starts from zero. Therefore, cArr[2] would refer to the third element in the array cArr where 2 is the array subscript.

The following program illustrates the usage of an array:

```
//Program
//This program performs input-output operations on arrays
#include<iostream>
void main()
{
int iArr[10]; //Array of size 10 is defined
int i, j;
cout<< “Input the elements of the array:”<<endl;
for(i=0; i<10; i++)              //As last subscript is 9
{
cin>>iArr[i];                    //i is called the array subscript
}
cout<< “The elements stored in the array are:”<<endl;
for(j=0; j<10; j++)
cout<<iArr[j]<< “”;                    //Displays the elements of the array
}
```

Program illustrates the input and output operations using an array. Array iArr is declared to be of type int and size 10. A maximum of 10 numbers can be stored in the array and the individual elements are referenced using the array subscript.
The sample output of Program is:
Input the elements of the array:
1
2
7
9
13
16
3
8
100
21
The elements stored in the array are:
1 2 7 9 13 16 3 8 100 21

Figure 5.1 illustrates the storage of elements in the array.

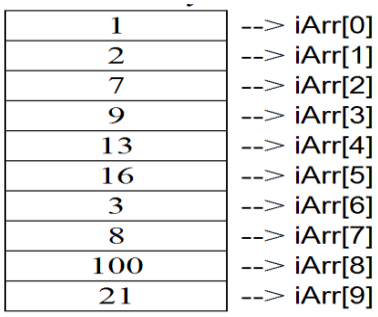
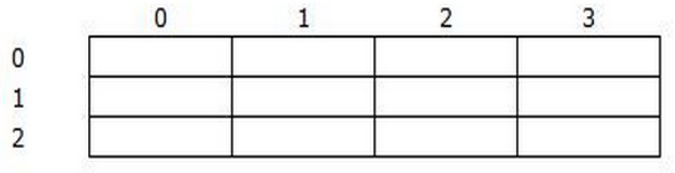


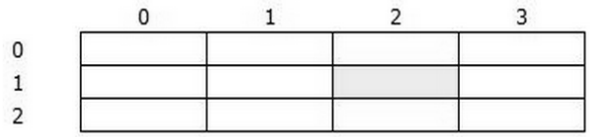
Figure 5.1: Array elements.

5.2.3 Multidimensional Arrays

A Multidimensional array is an array of arrays.
Suppose a programmer wants to represent the two-dimensional array India as an array with three rows and four columns all having integer elements. This would be represented in C++ as:
int India[3][4];
It is represented internally as:
India Data Type: int



How to access the elements in the Multidimensional Array
India Data Type: int



Based on the above two-dimensional arrays, it is possible to handle multidimensional arrays of any number of rows and columns in C++ programming language. This is all occupied in memory. Better utilization of memory must also be made.

Multidimensional Array Example:

Sample Code

```
#include <iostream>
using namespace std;
const int ROW=4;
const int COLUMN =3;
void main()
{
      int i, j;
      int India[ROW][COLUMN];
      for(i=0; i < ROW; i++)  //goes through the ROW elements
            for(j=0;j < COLUMN;j++)   //goes through the COLUMN elements
            {
                  cout << “Enter value of Row ” << i+1;
                  cout << “Column ” << j+1 << “:”;
                  cin>>India[i][j];
            }
            cout << “nnn”;
            cout << “ COLUMNn”;
            cout << “ 1 2 3”;
            for(i=0;i < ROW;i++)
            {
                  cout << “nROW ” << i+1;
                  for(j=0;j < COLUMN;j++)
                        cout << India[i][j];
            }
}
}}
```

Did You Know?

The first incarnation of what was later to be the APL (A Programming Language) programming language was published and formalized in A Programming Language, a book describing a notation invented in 1957 by Kenneth E.

5.3 User Defined Functions

The general form of a C++ function definition is as follows:

```
return_type function_name( parameter list )
{
  body of the function
}
```

5.3.1 Elements of User-Defined Functions

A C++ function definition consists of a function header and a function body. Here are all the parts of a function:

Return Type: A function may return a value. The return_type is the data type of the value the function returns. Some functions perform the desired operations without returning a value. In this case, the return_type is the keyword void.

Function Name: This is the actual name of the function. The function name and the parameter list together constitute the function signature.

Parameters: A parameter is like a placeholder. When a function is invoked, it passes a value to the parameter. This value is referred to as actual parameter or argument. The parameter list refers to the type, order, and number of the parameters of a function. Parameters are optional; that is, a function may contain no parameters.

Function Body: The function body contains a collection of statements that define what the function does.

Example:

Following is the source code for a function called max (). This function takes two parameters num1 and num2 and returns the maximum between the two:

```
// function returning the max between two numbers
 int max(int num1, int num2)
{
  // local variable declaration
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;
    return result;
}
```

C++ allows programmers to define their own functions. For example the following is a definition of a function which given the co-ordinates of a point (x, y) will return its distance from the origin.

```
float distance(float x, float y)
  // Returns the distance of (x, y) from origin
  {
   float dist;                          //local variable
   dist = sqrt(x * x + y * y);
   return dist;
  }
```

5.4 Return Values and Their Types

The purpose of a function identifies what the function is meant to do. When a function has carried its assignment, it provides a result. For example, if a function is supposed to calculate the area of a square, the result would be the area of a square. The result of a function used to get a student's first name would be a word representing a student's first name. The result of a function is called a return value. A function is also said to return a value.

There are two forms of expectations it will have from a function a specific value or a simple assignment. If it want the function to perform an assignment without giving it back a result, such a function is qualified as void and would be declared as

```
void FunctionName();
```

A function that does not return a value is declared and defined as void. Here is an example:

```
void Introduction()
{
cout << "This program is used to calculate the areas of some shapes.\n"
    << "The first shape will be a square and the second, a rectangle.\n"
    << "You will be requested to provide the dimensions and the program"
    << "will calculate the areas";
}
```

Any function could be a void type as long as it is not expecting it to return a specific value. A void function with a more specific assignment could be used to calculate and display the area of a square. Here is an example:

```
void SquareArea()
{
double Side;
cout << "\nEnter the side of the square:";
cin >> Side;
cout << "\nSquare characteristics:";
cout << "\nSide =" << Side;
cout << "\nArea =" << Side * Side;
}
```

When a function is of type void, it cannot be displayed on the same line with the cout extractor and it cannot be assigned to a variable (since it does not return a value). Therefore, a void function can only be called.

A return value, if not void, can be any of the data types we have studied so far. This means that a function can return a char, an int, a float, a double, a bool, or a string. Here are examples of declaring functions by defining their return values:

```
double FunctionName();
char FunctionName();
bool FunctionName();
```

```
string FunctionName();
```

If declare a function that is returning anything (a function that is not void), the compiler will need to know what value the function returns. The return value must be the same type declared. The value is set with the return keyword.

If a function is declared as a char, make sure it returns a character (only one character). Here is an example:

```
char Answer()
{
char a;
cout << "Do you consider yourself a reliable employee (y=Yes/n=No)?";
cin >> a;
return a;
}
```

A good function can also handle a complete assignment and only hand a valid value to other calling functions. Imagine it want to process member's applications at a sports club. You can define a function that would request the first and last names; other functions that need a member's full name would request it from such a function without worrying whether the name is complete. The following function is in charge of requesting both names. It returns a full name that any desired function can use:

```
string GetMemberName()
{
        string FName, LName, FullName;
        cout << "New Member Registration.\n";
        cout << "First Name:";
        cin >> FName;
        cout << "Last Name:";
        cin >> LName;
        FullName = FName + " " + LName;
        return FullName;
}
```

The return value can also be an expression. Here is an example:

```
double SquareArea(double Side)
{
        return (Side * Side);
}
```

A return value could also be a variable that represents the result. Here is example:

```
double SquareArea(double Side)
{
        double Area;
        Area = Side * Side;
        return Area;
}
```

If a function returns a value (other than void), a calling function can assign its result to a local variable like this:

```
Major = GetMajor();
```

Here is an example:

```
#include <iostream>
using namespace std;
int GetMajor()
{
	int Choice;
	cout << "\n1 - Business Administration";
	cout << "\n2 – History";
	cout << "\n3 – Geography";
	cout << "\n4 – Education";
	cout << "\n5 - Computer Sciences";
	cout << "\nYour Choice:";
	cin >> Choice;
	return Choice;
}

int main()
{
	int Major;
	cout << "Welcome to the student orientation program.";
	cout << "Select your desired major:";
	Major = GetMajor();
	cout << "You select" << Major; cout << "\n";
	return 0;
}
```

Did You Know?

In 1960, Kenneth E began work for IBM and, working with Adin Falkoff, created APL based on the notation he had developed.

5.5 Function Calls

One of the main reasons of using various functions in ones program is to isolate assignments; this allows it to divide the jobs among different entities so that if something is going wrong, it might easily know where the problem is. Functions trust each other, so much that one function does not have to know how the other function performs its assignment. One function simply needs to know what the other function does, and what that other function needs.

Once a function has been defined, other functions can use the result of its assignment. Imagine you define two functions A and B.

If Function A needs to use the result of Function B, function A has to use the name of function B. This means that Function A has to "call" Function B:

When calling one function from another function, provide neither the return value nor the body, simply type the name of the function and its list of arguments, if any. For example, to call a function named Welcome() from the main() function, simply type it, like this:

```
int main()
{
	Message(); // Calling the Message() function
	return 0;
}
```

The compiler treats the calling of a function depending on where the function is declared with regards to the caller. It can declare a function before calling it. Here is an example:

```
#include <iostream.h>
using namespace std;
void Message()
{
	cout << "This is C++ in its truest form.";
}
int main()
{
	Message(); // Calling the Message() function
	return 0;
}
```

5.5.1 Calling a Function before Defining

The example we saw above requires that you define a function before calling it. C++, like many languages, allows you to call a function before defining it. Unlike many languages, in C++, when calling a function, the compiler must be aware of the function. This means that, you must at least declare a function before calling it.

After calling the function, here is an example:

```
#include <iostream>
using namespace std;
int main()
{
	void Message();
	cout << "We will start with the student registration process.";
	Message(); // Calling the Message() function
	return 0;
}
void Message()
{
	cout << "Welcome to the Red Oak High School.";
}
```

To use any of the functions that ship with the compiler, first include the library in which the function is defined, then call the necessary function. Here is an example that calls the getchar() function:

```
#include <iostream>
#include <cstdio>
using namespace std;
int main()
{
        cout << "This is C++ in its truest form...\n\n";
        getchar();
        return 0;
}
```

Self Assessment Questions

1. The auto storage-class specified declares an automatic variable, a variable with a .....lifetime.

(a). global (b). local
(c). local and global (d). None of these

2. A declaration for an external variable can look just like a declaration for a variable that occurs .....a function or a block.

(a). inside (b). outside
(c). Both (a) and (b) (d). None of these

3. A command line argument is accomplished by passing information to main() method of the program.

(a) True (b) False

4. A function can call itself this is called.....

(a). Parameter (b). argument
(c). recursion (d). None of these

5.6 Passing Parameters to Functions

If a function is to use arguments, it must declare variables that accept the values of the arguments. These variables are called the formal parameters of the function.

They behave like other local variables inside the function and are created upon entry into the function and destroyed upon exit. As shown in the following function, the parameter declarations occur after the function name:

```
/* Return 1 if c is part of string s; 0 otherwise. */
int is_in(char *s,  char c)
{
while(*s)
if(*s==c) return 1;
else s++;
return 0;
}
```

The function is_in() has two parameters: s and c. This function return 1 if the character c is part of the string s; otherwise, it returns 0.
As with local variables, you may make assignments to a function's formal parameters or use them in an expression. Even though these variables perform the special task of receiving the value of the arguments passed to the function, you can use them as you do any other local variable.

5.6.1 Call by Value, and Call by Reference

In a computer language, there are two ways that arguments can be passed to a subroutine. The first is known as call by value. This method copies the value of an argument into the formal parameter of the subroutine. In this case, changes made to the parameter have no effect on the argument.
Call by reference is the second way of passing arguments to a subroutine. In this method, the address of an argument is copied into the parameter. Inside the subroutine, the address is used to access the actual argument used in the call. This means that changes made to the parameter affect the argument.
By default, C++ uses call by value to pass arguments. In general, this means that code within a function cannot alter the arguments used to call the function. Consider the following program:

```
#include <iostream>
int sqr(int x);
int main(void)
{
int t=10;
cout<<sqr(t) <<t;
return 0;
}
int sqr(int x)
{
x = x*x;
return(x);
}
```

In this example, the value of the argument to sqr(), 10, is copied into the parameter x. When the assignment x = x*x takes place, only the local variable x is modified. The variable t, used to call sqr(), still has the value 10. Hence, the output is 100 10.
Remember that it is a copy of the value of the argument that is passed into the function. What occurs inside the function has no effect on the variable used in the call.

Creating a Call by Reference

Even though C++ uses call by value for passing parameters, you can create a call by reference by passing a pointer to an argument, instead of the argument itself.
Since the address of the argument is passed to the function, code within the function can change the value of the argument outside the function.
Pointers are passed to functions just like any other value. Of course, you need to declare the parameters as pointer types. For example, the function swap(), which exchanges the values of the two integer variables pointed to by its arguments, shows how.

```
void swap(int *x, int *y)
{
int temp;
```

```
temp = *x;  /* save the value at address x */
*x = *y;    /* put y into x */
*y = temp;  /* put x into y */
}
```

swap() is able to exchange the values of the two variables pointed to by x and y because their addresses (not their values) are passed. Thus, within the function, the contents of the variables can be accessed using standard pointer operations, and the contents of the variables used to call the function are swapped. Remember that swap() (or any other function that uses pointer parameters) must be called with the addresses of the arguments. The following program shows the correct way to call swap():

```
void swap(int *x, int *y);
int main(void)
{
int i, j;
i = 10;
j = 20;
swap(&i, &j); /* pass the addresses of i and j */
return 0;
}
```

In this example, the variable i is assigned the value 10 and j is assigned the value 20. Then wap() is called with the addresses of i and j. (The unary operator & is used to produce the address of the variables.) Therefore, the addresses of i and j, not their values, are passed into the function swap( ).

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: Write a simple C++ program to familiar with 2-Dimensional Array as function arguments.

……..……

……..……

……...

5.7 Recursion

A function can call itself. This is called recursion, and recursion can be direct or indirect. It is direct when a function calls itself; it is indirect recursion when a function calls another function that then calls the first function.

Some problems are most easily solved by recursion, usually those in which you act on data and then act in the same way on the result. Both types of recursion, direct and indirect, come in two varieties: those that eventually end and produce an answer, and those that never end and produce a runtime failure. Programmers think that the latter is quite funny (when it happens to someone else).

The local variables in the second version are independent of the local variables in the first, and they cannot affect one another directly, any more than the local variables in main() can affect the local variables in any function it calls.

When implementing a recursive solution one usually has at least two cases:

- Base Case
- General Case

For a function/method to be called recursive, it usually has a call to itself within its code in the general case, with a smaller case being passed through.
A typical example to illustrate a recursive function is the factorial function (i.e. 5!)

```
//************************************
// Returns the factorial of a number
//************************************
int fact(int num)
{
   if(num <= 0)
      return 1;                    // Base Case
  else
      return num * fact(num-1);    // General Case - also note it calls itself
}
void CountDown(int nValue)
{
   using namespace std;
   cout << nValue << endl;
   CountDown(nValue-1);
}

int main(void)
{
   CountDown(10);
   return 0;
}
```

When CountDown(10) is called, the number 10 is printed, and CountDown(9) is called. CountDown(9) prints 9 and calls CountDown(8). CountDown(8) prints 8 and calls CountDown(7). The sequence of CountDown(n) calling CountDown(n-1) is continually repeated, effectively forming the recursive equivalent of an infinite loop.

Caution
It is important to note that when a function calls itself, a new copy of that function is run.

5.8 Command Line Arguments

A command line argument is a parameter which is passed to a program at the time or instant when it is invoked or executed from the command line .This is accomplished by passing information to main() method of the program. It is the information that directly follows the program's name on the command line when you run it This parameter may represent any information for example a filename the program should process. This eliminates the need for the program to request the user to enter the information during program execution. An application can accept any number of arguments from the command line. The space character usually separates the command line arguments.

These parameters are not passed to main() directly. Instead, every program's main() function is passed two parameters. The first is an integer count of the number of arguments on the command line. The program name itself is counted, so every program has at least one parameter.
The second parameter passed to main() is an array of pointers to character strings. Because an array name is a constant pointer to the first element of the array, you can declare this argument to be a pointer to a pointer to char, a pointer to an array of char, or an array of arrays of char.
Typically, the first argument is called argc (argument count), but you may call it anything you like. The second argument is often called argv (argument vector), but again this is just a convention.
It is common to test argc to ensure you have received the expected number of arguments, and to use argv to access the strings themselves.
Note that argv[0] is the name of the program, and argv[1] is the first parameter to the program, represented as a string. If your program takes two numbers as arguments, you will need to translate these numbers to strings. Listing illustrates how to use the command-line arguments.

5.9 Storage Class Specifies

The C++ storage-class specifies tell the compiler the duration and visibility of the object or function they declare, as well as where an object should be stored.
The following are storage class specifies:

- auto
- register
- static
- extern

5.9.1 Storage Class auto

Variables declared within function bodies are by default automatic, making automatic the most common of the four storage classes. An automatic variable is allocated within a block, and its lifetime is limited to the execution of that block. Once the block is exited, the value of such a variable is lost. If a compound statement contains variable declarations, these variables can be used within the scope of the enclosing compound statement. Recall that a compound statement with declarations is a block. Declarations of variables within blocks are implicitly of storage class automatic.

The keyword auto can be used to explicitly specify the storage class. An example is

```
auto int a, b, c;
auto float f = 7.78;
```

Because the storage class is automatic by default, the keyword auto is seldom used.
The auto storage-class specified declares an automatic variable, a variable with a local lifetime. It is the default storage-class specified for block-scoped variable declarations.
Few programmers use the auto keyword in declarations because all block-scoped objects not explicitly declared with another storage class are implicitly automatic.
Therefore, the following two declarations are equivalent:

```
// auto_keyword.cpp
int main()
{
  auto int i = 0;    // Explicitly declared as auto.
```

```
  int j = 0;    // Implicitly auto.
}
```

5.9.2 Storage Class Register

The storage class register tells the compiler that the associated variables should be stored in high-speed memory registers, provided it is physically and semantically possible to do so. Since resource limitations and semantic constraints sometimes make this impossible, the storage class register defaults to automatic whenever the compiler cannot allocate an appropriate physical register. Therefore, the register declaration can be used in place of an automatic declaration. When speed is of concern, the programmer may choose a few variables that are most frequently accessed and declare them to be of storage class register. Common candidates for such treatment include loop variables and function parameters.

Here is an example:

```
{
for (register i = 0; i < LIMIT; ++i) {
.....
}
}
```

Caution

The compiler does not accept user requests for register variables; instead, it makes its own register choices when global register-allocation optimization.

5.9.3 Storage Class static

Static declarations have two important and distinct uses. The more elementary use is to allow a local variable to retain its previous value when the block is re-entered. By contrast, ordinary automatic variables lose their value on block exit and must be reinitialized. The second, more subtle use is in connection with external declarations.

As an example of the value-retention use of static, we write a function that maintains a count of the number of times it is called:

```
int f()
{
static int called = 0;
++called;
.....
return called;
}
```

Caution

Assigning to a static local variable is not thread safe and is not recommended as a programming practice.

Task

Write the program for declare the static variable in C++.

5.9.4 Storage Class extern

One method of transmitting information across blocks and functions is to use external variables. When a variable is declared outside a function at the file level, storage is permanently assigned to it, and its storage class keyword is extern. A declaration for an external variable can look just like a declaration for a variable that occurs inside a function or a block. Such a variable is considered to be global to all functions declared after it. On block exit or function exit, the external variable remains in existence. Such variables cannot have automatic or register storage class.

```
// external.cpp
// defined in another translation unit
extern int DefinedElsewhere;
int main() {
  int DefinedHere;
  {
    // refers to DefinedHere in the enclosing scope
    extern int DefinedHere;
   }
}
```

Did You Know?

Call by sharing also known as “call by object” or “call by object-sharing” is an evaluation strategy first named by Barbara Liskov for the language CLU in 1974.

Exercise: Check Your Progress 2

Case Study-Support Vector Machine Training

Support Vector Machines (SVMs) are a widely used binary classification technique used to analyze data, or in other words, to label data as belonging to one of two categories. For example, SVMs can be used to determine whether email is important or unsolicited, whether an image contains a flower or not, or whether a DNA sequence codes for a particular trait, etc. Researchers in many fields, such as image, speech and gesture recognition, bioinformatics, and computer security have applied SVMs to enable programs to analyze data.

In order to use an SVM, it must first be trained, so that the classifier can “learn” the categories it distinguishes between. This training process is computationally intensive, the issues that arise when building an iterative solver for Support Vector Machine training, running on AMD GPUs, using OpenCL. The data structures we choose determine how efficiently we can use our memory resources.

Support Vector Machine Training

As previously mentioned, Support Vector Machines are a widely used classification technique. Support Vector Machines process data as points in some abstract, highly dimensional feature space. For example, it could describe objects by measuring their exterior dimensions (length, width, height), as well as their mass. Each object would then be described by four numbers, which we can think of as a four-dimensional vector in some feature space. Typical SVM applications use feature spaces with tens to hundreds of dimensions, or sometimes more.

Given a series of measurements of objects, along with their associated labels, for example: between objects made of wood, and those of stone. Since wood objects would tend to reside in a certain region of my four dimensional feature space, and stone objects would tend to reside in another region, we can draw a boundary in our feature space that separates the wood objects from the stone objects. Of course, there are many such boundaries we could draw, but we want to learn the most general

boundary we can, so that our classifier will be robust. Training an SVM is a specific approach for learning the boundary which separates our training data with maximal margin, meaning that the training data is as clearly classified as possible. Then, a program can classify new data by simply checking to see which side of the boundary the new data falls on.

SVMs can be trained using a Quadratic Programming optimization process called Sequential Minimal Optimization. In this algorithm, we take many tiny steps to update the boundary between our two object classes. This is done by finding the two training points which are explained most poorly by our current guess of what the boundary is, and then updating the boundary to fit to those points better.

The main computational load in this algorithm is choosing the two most poorly explained points out of our training set, which may be hundreds of thousands of points in total. This involves evaluating a distance function between all our training points and two of the training points. This distance function, called a kernel function, can take many forms, but one of the most useful is the so-called radial basis function: Our job, then, is to evaluate many copies of this function, for our many training points.

Initial Analysis

Let us look at a simple C++ version of the Radial Basis Function evaluations. We are going to evaluate all data points against two of the data points, one called high and the other called low.

We will store the results in two arrays, one called high_kernel and the other called low_kernel. We have nPoints training points, each of which is a sample from an nDim dimensional feature space. We will store the data points in an array called data, where each row represents one training point, and so the resulting matrix has nPoints rows and nDim columns. Gamma is the scalar parameter from our Radial Basis Function. Listing 1 shows some simple C++ code to accomplish this.

Questions

1. What is the support vector machine training?
2. What is the radial basis function?

5.10 Summary

- Array is a data structure which allows a collective name to be given to a group of elements which all have the same type. An individual element of an array is identified by its own unique index (or subscript).
- Command line argument is a parameter which is passed to a program at the time or instant when it is invoked or executed from the command line.
- Function calls itself. This is called recursion and recursion direct or indirect. It is direct when a function calls itself; it is indirect recursion when a function calls another function that then calls the first function.
- C++ provides a data structure, the array, which stores a fixed-size sequential collection of elements of the same type.
- A User-Defined Function is a function provided by the user of a program or environment, in a context where the usual assumption is that functions are built into the program or environment.

5.11 Keywords

Call by Reference: It is method the address of an argument is copied into the parameter.

Call by Value: This method copies the value of an argument into the formal parameter of the subroutine. Changes made to the parameter have no effect on the argument.

Extern: It variable is declared outside a function at the file level, storage is permanently assigned to it, and its storage class keyword is extern.
Function Body: It is contains a collection of statements that define what the function does.
Parameter List: It is refers to the type, order, and number of the parameters of a function.
void: Function that does not return a value is declared and defined as void.

5.12 Review Questions

1. What is the array and initializing arrays?
2. Define the one-dimensional arrays and multidimensional arrays.
3. What is the functions and also describe the type of the array?
4. Define the predefined functions and user defined functions.
5. Explain the elements of user-defined functions.
6. What is the function calls?
7. Explain the passing parameters to functions.
8. What is the call by value and call by reference?
9. What differentiate between external and static variable?
10. What are the difference between call by value and call by reference?

Answers for Self Assessment Questions

1. (b) 2. (a) 3. (a) 4. (c)

5.13 Further Studies

1- http://www.cplusplus.com/forum/beginner/58688/
2- http://my.safaribooksonline.com/book/programming/cplusplus/9788131703830/10dot-pointers-and-arrays/navpoint212
3- Author- The C++ Programming Language 4 Edition by Bjarne Stroustrup

6

Classes and Objects

STRUCTURE

6.0 Objectives

6.1 Introduction

6.2 Classes

6.3 Structures and Unions, Classes

6.4 Friend Function

6.5 Friend Classes

6.6 Inline Function

6.7 Scope Resolution Operator

6.8 Static Class Members

6.9 Static Data Members

6.10 Static Member Function

6.11 Passing Object to Functions

6.12 Returning Objects

6.13 Object Assignment

6.14 Summary

6.15 Keywords

6.16 Review Questions

6.17 Further Studies

6.0 Objectives

After studying this chapter, you will be able to:

- Discuss the classes
- Explain the structures and unions, classes
- Discuss the friend function and friend classes
- Explain the inline function
- Define the scope resolution operator
- Explain the static class and data members

- Define the static member function
- Explain the passing object to functions
- Discuss the returning objects

6.1 Introduction

In object-oriented programming languages like C++, the data and functions (procedures to manipulate the data) are bundled together as a self-contained unit called an object. A class is an extended concept similar to that of structure in C programming language; this class describes the data properties alone. In C++ programming language, class describes both the properties (data) and behaviours (functions) of objects. Classes are not objects, but they are used to instantiate objects.

6.2 Classes

A class is a way to bind the data and its associated functions together. It allows the data (and function) to be hidden, if necessary, from external use. When defining a class, we are creating a new abstract data type that can be treated like any other built-in data type. Generally, a class specification has two parts:

1. Class declaration
2. Class function definitions

The class declaration describes the type and scope of its members. The class function definition describes how the class functions are implemented.
The general form of a class declaration is:

```
class class_name
{
private:
        variable        declarations;
        function        declarations;
public;
        variable        declarations;
        function        declaration;
};
```

The class declaration is similar to a structure declaration. The keyword class specifies that what follows is an abstract data of type class_name. The body of a class is enclosed within braces and terminated by a semicolon. The class body contains the declaration of variables and functions. These functions and variables are collectively called class members. They are usually grouped under two sections, namely, private and public to denote which of the members are private and which of them are public. The keywords private and public are known as visibility labels. Note that these keywords are followed by a colon.
The class members that have been declared as private can be accessed only from within the class. On the other hand, public members can be accessed from outside the class also. The data hiding (using private declaration) is the key feature of object-oriented programming. The use of the keyword private is optional. By default, the members of a class are private. If both the labels are missing, then,

by default, all the members are private. Such a class is completely hidden from the outside world and does not serve any purpose.

The variables declared inside the class are known as data members and the functions are known as member functions. Only the member functions can have access to the private data members and private functions. However, the public members (both functions and data) can be accessed from outside the class. This is illustrated in Figure 6.1. The binding of data and the functions together into a single class-type variable is referred to as encapsulation.

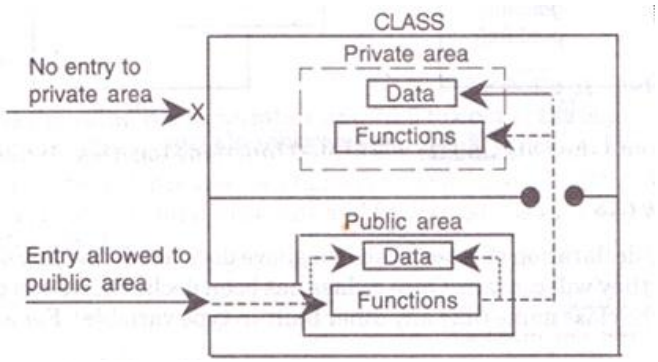


Figure 6.1: Data hiding in classes.

Simple Class Example

A typical class declaration would look like:

```
class item
{
        int number;     // variables declaration
        float cost;     // private by default
public:
        void getdata(int a, float b);     // functions declaration
        void putdata(void);               // using prototype
};
```

We usually give a class some meaningful name, such as item. This name now becomes a new type identifier that can be used to declare instances of that class type. The class item contains two data members and two function members. The data members are private by default while both the functions are public by declaration. The function getdata() can be used to assign values to the member variables number and cost, and putdata() for displaying their values. These functions provide the only access to the data members from outside the class. This means that the data cannot be accessed by any function that is not a member of the class item. Note that the functions are declared, not defined. Actual function definitions will appear later in the program. The data members are usually declared as private and the member functions as public.

Figure 6.2 shows two different notations used by the OOP analysts to represent a class.

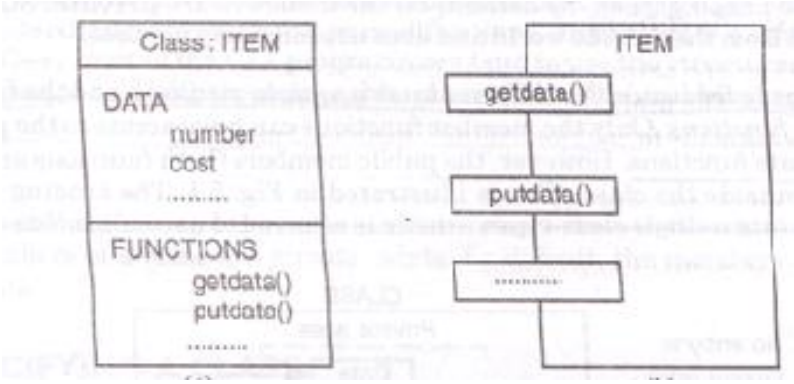


Figure 6.2: Representation of a class.

Did You Know?

The basis for OOP started in the early 1960s. A breakthrough involving instances and objects was achieved at MIT with the PDP-1 and the first programming language to use objects was Simula 67. It was designed for the purpose of creating simulations and was developed by Kristen Nygaard and Ole-Johan Dahl.

6.3 Structures and Unions, Classes

Data encapsulation is key to the functionality of many high level or later generation programming languages such as C++ and Java. It is simply a means of creating "containers" called objects, and holding multiple variables of varying data types called member variables inside of those containers.

C++ offers three methods of data-encapsulation; structure, union, and class. This is where C++ becomes object oriented.

1. Structure and Union
    - Common uses
    - Declaration and syntax
    - Accessing member variables
    - Accessing pointer member variables
2. Classes
    - Member functions
    - Levels of access: public vs. private vs. protected
    - Data hiding
    - Declaring and defining a class
    - Scope resolution operator
    - Conditional pre-processor directives
    - Constructors and destructors
3. The relationship between class, structure and union
    - Defaulting to public or private
    - Programming standards
4. Advanced methodology
    - Initialize lists
    - Overloading operators
    - Template classes

6.3.1 Structure and Union

A structure and union are fairly similar, so they will be treated as one entity. A struct is often useful to hold records of data, like a bank statement, or creating real-world objects, like dates and time.
The syntax is as follows.

```
struct {
float savings;
float checking;
float mortgage;
float interest;
float fees;
} BankStatement;
union {
short month;
short day;
int year;
} DATE;
typedef struct {
int hour;
int min;
int sec;
char AmPm;
} TIME;
```

A structure can also have a structure object as a member variable.

```
typedef struct {
bool deposit;
bool withdraw;
float Amount;
float newBalance;
DATE date;
TIME time;
} Transaction;
```

There are no limits to how many, or what data type a structure can hold. Remember though, a structure can become quite large sometimes because it is size in memory is that of the sum total of the memory size of all of it is member variables.

Now that we have all of these objects, what do we do with them? Accessing a member variable is very simple and uses dot-notation: object.member_variable

```
TIME newTime() {
TIME time;
cout << “Hour:”;
cin >> time.hour;
cout << “Minute:”;
cin >> time.min;
//... yadda yadda yadda
```

```
return time;
}
```

Now let us take a look at pointer objects. When any object, a structure, union, or class, is declared as a pointer, it no longer uses dot-notation, but pointer-notation: object->member_variable. The arrow or "pointer" is used to signify the difference between a normal object and a pointer.

```
DATE newDate() {
DATE date {07, 06, 2002};
/* A struct can be initialized in this manner.
month=07, day=10, year=2002
*/
char dummy;
cout << "Enter new date (mm/dd/yyyy):";
cin >> date.month >> dummy >> date.day >> dummy >> date.year;
return date;
}
```

As an alternative, the above code can request the date format to be (mm dd yyyy) and eliminate the need for the char dummy variable.

6.3.2 Classes

A class is where C++ becomes the most objects oriented that it can. Classes offer many abilities and methods of storing, handling, and retrieving data. A class is very similar to a structure in that it can have member variables, but it can also have member functions. These are functions within the class. Member functions are called the same way member variables are:

```
object.member_function (...)
class CAT {
int lives;
int getLives(); //returns the number of lives
};
```

In a class, as well as in a structure, there are levels of access; they are public, private, and protected. We would not worry about protected access, thought, since that pertains to inheritance, and is out of the scope of this tutorial.

Anything that is in the public section of the class can be referenced and changed by anything outside of the class like a normal variable. In the private section, opposite is true. Nothing but member functions can access, change or call member variables or functions.

```
class CAT {

public:
int getLives(); //returns the number of lives
private:
int lives;
};
```

6.3.3 Relationship between Class, Structure and Union

The separating factor between a structure and a union is that a structure can also have member functions just like a class. The difference between a structure and a class is that all member functions and variables in a structure are by default public, but in a class, they default to private as previously discussed.

It is often a good idea to use constructors to initialize the member variables of a structure. Other than that, though, it is against current standards, and usually looked down upon, to use functions in a structure, and is usually considered just being lazy.

Caution

A union member cannot be a class object that has a constructor, destructor, or overloaded copy assignment operator, nor can it be of reference type. A union member cannot be declared with the keyword static.

6.4 Friend Function

A C++ friend functions are special functions which can access the private members of a class. They are considered to be a loophole in the Object Oriented Programming concepts, but logical use of them can make them useful in certain cases. For instance, when it is not possible to implement some function, without making private members accessible in them. This situation arises mostly in case of operator overloading.

The following example shows a Point class and a friend function, Change Private. The friend function has access to the private data member of the Point object it receives as a parameter.

```
//friend_functions.cpp
//compile with: /EHsc
#include <iostream>
using namespace std;
class Point
{
   friend void ChangePrivate( Point &);
public:
   Point( void ) : m_i(0) {}
   void PrintPrivate( void ){cout << m_i << endl; }
private:
   int m_i;
};
void ChangePrivate ( Point &i ) { i.m_i++; }
int main()
{
  Point sPoint;
  sPoint.PrintPrivate();
  ChangePrivate(sPoint);
  sPoint.PrintPrivate();
}
```

Did You Know?
C++ was developed in bell labs by Bjarne Stroustrup in 1983-1985. Bjarne Stroustrup, after completing the doctoral degree at the Computing Laboratory of the Cambridge University, joined the Bell Laboratories.

6.5 Friend Classes

C++ provides the friend keyword to do just this. Inside a class, you can indicate that other classes (or simply functions) will have direct access to protect and private members of the class. When granting access to a class, you must specify that the access is granted for a class using the class keyword:

friend class aClass;

The friend declarations can go in either the public, private, or protected section of a class it does not matter where they appear. In particular, specifying a friend in the section marked protected does not prevent the friend from also accessing private fields.

Here is a more concrete example of declaring a friend:

```
class Node
{
   private:
   int data;
   int key;
   // ...
   friend class BinaryTree; // class BinaryTree can now access data directly
};
```

Now, Node does not need to provide any means of accessing the data stored in the tree. The BinaryTree class that will use the data is the only class that will ever need access to the data or key. (The BinaryTree class needs to use the key to order the tree, and it will be the gateway through which other classes can access data stored in any particular node.)
Now in the BinaryTree class, you can treat the key and data fields as though they were public:

```
class BinaryTree
{
   private:
   Node *root;
   int find(int key);
};
int BinaryTree::find(int key)
{
   // check root for NULL...
   if(root->key == key)
   {
      // no need to go through an access or function
      return root->data;
   }
   // perform rest of find
```

6.6 Inline Function

Inline functions are functions where the call is made to inline functions. The actual code then gets placed in the calling program.

Reason for the need of Inline Function:

Normally, a function call transfers the control from the calling program to the function and after the execution of the program returns the control back to the calling program after the function call. These concepts of function save program space and memory space and are used because the function is stored only in one place and is only executed when it is called. This execution may be time consuming since the registers and other processes must be saved before the function gets called.

The extra time needed and the process of saving is valid for larger functions. If the function is short, the programmer may wish to place the code of the function in the calling program in order for it to be executed. This type of function is best handled by the inline function. In this situation, the programmer may be wondering "why not write the short code repeatedly inside the program wherever needed instead of going for inline function?" Although this could accomplish the task, the problem lies in the loss of clarity of the program. If the programme repeats the same code many times, there will be a loss of clarity in the program. The alternative approach is to allow inline functions to achieve the same purpose, with the concept of functions.

The inline function takes the format as a normal function but when it is compiled it is compiled as inline code. The function is placed separately as inline function, thus adding readability to the source program. When the program is compiled, the code present in function body is replaced in the place of function call. General Format of inline Function:

The general format of inline function is as follows:

inline datatype function_name(arguments)

The keyword inline specified in the above example, designates the function as inline function. For example, if a programmer wishes to have a function named india with return value as integer and with no arguments as inline it is written as follows:

inline int india( )

Example:

The concept of inline functions:

```
#include <iostream>
using namespace std;
int india(int);
void main( )
{
      int x;
      cout << "\n Enter the Input Value:";
      cin>>x;
      cout << "\n The Output is:"<< india(x);
}
inline int india(int x1)
```

```
{
      return 5*x1;
}
```

The output of the above program is:
Enter the input value: 10
The output is: 50 press any key to continue.....

The output would be the same even when the inline function is written solely as a function. The concept, however, is different. When the program is compiled, the code present in the inline function india( ) is replaced in the place of function call in the calling program. The concept of inline function is used in this example because the function is a small line of code.
A programmer must make wise choices when to use inline functions. Inline functions will save time and are useful if the function is very small. If the function is large, use of inline functions must be avoided.

Self Assessment Questions

1. The……describes the type and scope of its members.
(a) class function (b) class function definitions
(c) class members (d) class declaration

2. The data members are …….by defaults while both the functions are public by declaration.
(a) public (b) private
(c) protected (d) None of these

3. A structure can also have a .....as a member variable.
(a) structure object (b) data structures
(c) structure declaration (d) structure class

4. The .....is placed only in the function declaration of the friend function and not in the function definition.
(a) friend class (b) friend function
(c) friend declarations (d) keyword friend

5. The inline function takes the format as a normal function but when it is compiled it is compiled as.....
(a) outline code (b) function code
(c) inline code (d) data member code

6. The concept of inline function is used in this example because the function is a small line of code.
(a) True (b) False

6.7 Scope Resolution Operator

Operator "::" is known as Scope Resolution Operator. C++ is a block structured language. Different program modules are written in various blocks. Same variable name can be used in different blocks.

Scope of a variable extends from the point of declaration to the end of the block. A variable declared inside a block is “local” variable. Blocks in C++ are often nested.
For example

```
….
….
{
        int x = 10;
        …..
        ……
        {
              int x = 20;
              …..
              ….
        }
        ….
}
```

The declaration of the inner block hides the declaration of same variable in outer block. This means, within the inner block, the variable x will refer to the data object declared there in. To access the global version of the variable, C++ provides scope resolution operator.
In the above example, x has a value of 20 but ::x has value 10. Similarly, this operator is used when a member function is defined outside the class.
For example

```
Class MyClass
{
int n1, n2;
public:
{
void func1();                  ---------Function Declaration
}
};

    public void MyClass::func1() ---Use of Scope Resolution Operator to write function definition
                                                outside class definition
{
            // Function Code
}
```

6.8 Static Class Members

The member variables declared in a class declaration are normally instance variables - a separate copy of them is created for each class object. Sometimes, however, it is convenient to have class variables that are associated with the class itself rather than with any class object. Every instance of a class accesses the same class variables, although each has its own private copy of any instance variables.
In C++, class variables are declared as static.

Thus in

```
class c{
int i;
int j;
static int m;
static int n;
public:
void zap();
static void clear();
};
void c::zap() {
i = 0; j = 0; m = 0; n = 0;
}
void c::clear() {
m = 0; n = 0;
}
```

There i and j are instance variables and m and n are class variables. Every object of class c will have its own private i and j, which can have different values for different objects; however, all objects will access the same m and n, which will, of course, have the same values for all objects.
Static variables are like non-inline member functions in that they are declared in a class declaration and defined in the corresponding source file. To define static variables m and n, the source file for class c must contain the following declarations (which are also definitions):

```
int c::m;
int c::n;
```

Caution
The static member variables must be defined outside the class.

6.9 Static Data Members

A data member inside the class can construct as static data member. There are few guidelines to be followed when declaring static data member.

- It is declared as a private data member.
- Only one copy of the static data member is created any numbers of objects are created from that class. Therefore the static data member will be same for all the objects created.
- The static member must be defined outside the class like function with class name.
- The static member can be initialized in the definition and can be used only within its class.

The following program will give clear difference between static data count and regular data number members. The static data is common to all objects but regular data is unique to its object.
class item.

```
{
int number;
static int count;
```

```
public:
void getdata()
{
++count;
number=count;
}
void putdata(void)
{
cout<< “Count is” <<count<< “\n”;
cout<< “Number is”<<number<< “\n”;
}
};
int item::count=0;
int main()
{
clrscr();
item x,y,z; //Three object created from class item
x.getdata();
y.getdata();
z.getdata();
x.putdata(); The count is 3 and number is 1
y.putdata(); The count is 3 and number is 2
z.putdata(); The count is 3 and number is 3
return 0;
}
```

6.10 Static Member Function

The static member function can be declared like static data member. It works like static member data except the static member data is used in static member function.
Rules of Static Member function:

- The static member function can read only static data member from its class.
- The static member function is called using class name instead of object name.

class_name::static_function_name ( ); It can also be called using object.static_fun( ).
The following program will illustrate how the static member function is used. class find

```
{
static int count;
int code;
public:
static void showcount(void)
{
cout<< “Count is” <<count<< “\n”;
}
void setcode(void)
{
```

```
code = ++count;
}
void setcount(void)
{
cout<< “Code is” <<code<< “\n”;
}
};
int find::count=0;
int main()
{
clrscr();
find x,y,z;
x.setcode(); //Code and count is 1.
y.setcode(); //Code and count is 2
find::showcount(); //Count is 2
z.setcode(); //Code and count is 3
find::showcount(); //Count is 3
x.setcount(); //Code is 1 for object x.
y.setcount(); //Code is 2 for object y.
z.setcount(); //Code is 3 for object z.
x.setcode(); //Code and count is 4.
y.setcode(); //Code and count is 5
z.setcode(); //Code and count is 6
find::showcount(); //Count is 6
//The value of code will increase from its previous value of that object.
x.setcount(); //Code is 4 for object x.
y.setcount(); /Code is 5 for object x.
z.setcount(); //Code is 6 for object x.
return 0;
}
```

Since the count is declared as static it has only one copy irrespective of number of objects created. When static count function called again and again it will increase count for all the objects created simultaneously therefore the count will be uniform for all object, but for the data member code it is unique for each object created because it is not a static data member. Therefore from the above example there will only one value for count data member irrespective of number of objects constructed since it is a static data member, where as for code data member the value will be different for each object.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Define a class to model a banking system. The function members should allow initializing the data members, a query to facilitate for account and a facility to deposit and withdraw from the account. WAP to implement the same.

……..……

……..……

……...

6.11 Passing Object to Functions

Like any other data type, an object may be used as a function argument.
This can be done in two ways:

- A copy of the entire object is passed to the function.
- Only the address of the object is transferred to the function.

The first method is called pass-by-value. Since a copy of the object is passed to the function any changes made to the object inside the function do not affect the object used to call the function. The second method is called pass-by-reference. When an address of the object is passed, the called function works directly on the actual object used in the call. This means that any changes made to the object inside the function will reflect in the actual object. The pass-by-reference method is more efficient since it requires passing only the address of the object and not the entire object.
Program illustrates the use of objects as function arguments. It performs the addition of time in the hour and minutes format.

```
#include <isotream>
using namespace std;
class time
{
            int hours;
           int minutes;

public:
           void gettime(int h, int m)
          { hour = h; minutes = m;}
          void puttime(void)
         {
      count<< hours << “hours and”;
      count<< minutes << “minute” << “\n”;
         }
     void sum (time, time);              //declaration with objects as arguments
};
     void time:: sum(time t1, time t2)//t1, t2 are objects
{
     minutes = t1.minutes + t2.minutes;
        hours = minutes/60;
        minutes = minutes%60;
       hours = hours + t1.hours + t2.hours;
}

int main()
{
```

```
        time T1, T2, T3;
        T1.gettime (2, 45);                  //get T1
        T2.gettime (3, 30);                  //get T2
        T3.sum(T1, T2);                      //T3 = T1 + T2
        count<< "T1 =";T1.puttime();        //display T1
        count << "T2="; T2.puttime();        //display T2
        count << "T3="; T3.puttime();        //display T3
        return 0;
}
```

The output of program would be:
T1 = 2 hours and 45 minutes
T2 = 3 hours and 30 minutes
T3 = 6 hours and 15 minutes

Since the member function sum ( ) is invoked by the object T3, with the objects T1 and T2 as arguments, it can directly access the hours and minutes variables of T3. But, the members of T1 and T2 can be accessed only by using the dot operator (like T1.hours and T1.minutes). Therefore, inside the function sum ( ), the variables hours and minutes refer to T3, T1.hours and T1.minutes refer to T1, and T2.hours and T2.minutes refer to T2. Figure 6.3 illustrates how the members are accessed inside the function sum ( ).

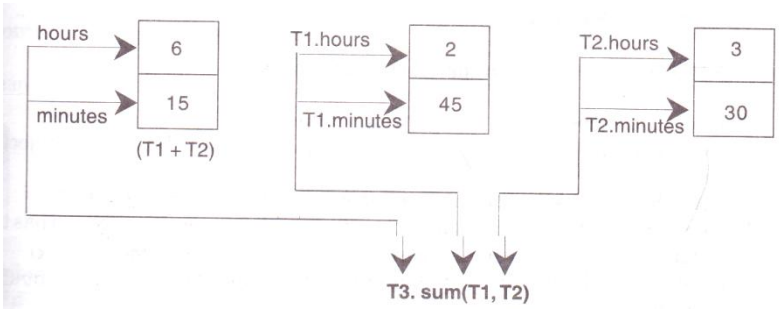


Figure 6.3: Accessing members of objects within a called function.

An object can also be passed as an argument to a non-member function. However, such functions can have access to the public member functions only through the objects passed as arguments to it. These functions cannot have access to the private data members.

6.12 Returning Objects

One can always return objects from class methods by reference, either constructor or non-constructor. A user can take the address of the reference, if necessary, to save the object. But there are no drawbacks to returning objects always as pointers. Consistency is preferable, and most API's (Application Programming Interface) return pointers.
If you return an object allocated on the heap (with a new), then be clear who has ownership of the object--your class, the recipient, or a third party.
Whether you are breaking encapsulation of member data in a way that will prevent modification later.

Never return a reference to a class member allocated on the stack in the header file. If class replaces the value, then the user may be left with an invalid reference, even though object still exists.
A method should modify an object constructed by the user by accepting it as a non-constructor reference. Returning the same object would be redundant and confusing.

Example:

```
class AnimalLister
{
  public:
  Animal* getNewAnimal()
  {
    Animal* animal1 = new Animal();
    return animal1;
  }
}
```

If 1 create an instance of Animal Lister and get Animal reference from it, then where 1 supposed to delete it?

```
int main() {
  AnimalLister al;
  Animal *a1, *a2;
  a1 = al.getNewAnimal();
  a2 = al.getNewAnimal();
}
```

The problem here is AnimalLister does not have a way to track the list of Animals Created, so how does 1 change the logic of such code to have a way to delete the objects created.

Exercise: Check Your Progress 2
Note: i) Use the space below for your answer.
Ex1: Create a class called time that has separate int member data for hours, minutes and seconds. Write functions for accepting time and displaying time.

……..……

……..……

……...

6.13 Object Assignment

Each Variable belonging to an object (including by inheritance) allows the object to also contain a single Assignment object for that variable. If no Assignment exists nor is inherited for a Variable belonging to a given object, the value of that Variable is undefined for the object. If an Assignment is inherited, a local assignment overrides it.

Example:

```
#include <iostream>
```

```
using namespace std;

class MyClass
{
  int a, b;
public:
  void setAB(int i, int j) { a = i, b = j; }
  void display() {
    cout << “\n a is” << a << “\n”;
    cout << “\n b is” << b << “\n”;
  }
};

int main()
{
  MyClass ob1, ob2;

  ob1.setAB(10, 20);
  ob2.setAB(0, 0);
  cout << “ob1 before assignment:”;
  ob1.display();
  cout << “ob2 before assignment:”;
  ob2.display();

  ob2 = ob1; //assign ob1 to ob2

  cout << “ob1 after assignment:”;
  ob1.display();
  cout << “ob2 after assignment:”;
  ob2.display();

  ob1.setAB(-1, -1); //change ob1

  cout << “ob1 after changing ob1:”;
  ob1.display();
  cout << “ob2 after changing ob1:”;
  ob2.display();

  return 0;
}
```

Output:

```
ob1 before assignment:
a is 10

b is 20
ob2 before assignment:
```

```
a is 0

b is 0
ob1 after assignment:
a is 10

b is 20
ob2 after assignment:
a is 10

b is 20
ob1 after changing ob1:
a is -1
b is -1
ob2 after changing ob1:
a is 10
b is 20
```

6.14 Summary

- A class is a combination of member functions (Methods) and member variables (fields)
- The class declaration is similar to a structure declaration.
- The data hiding is the key feature of object-oriented programming.
- Data-encapsulation is key to the functionality of many high level or later generation programming languages such as C++ and Java.
- Data hiding is the principle of using private member variables within a class.
- The keyword friend is placed only in the function declaration of the friend function and not in the function definition.
- A class is an extension to the structure data type. A class can have both variables and functions as members.

6.15 Keywords

Class: A class is very similar to a structure in that it can have member variables, but it can also have member functions.

Friend Class: A class can also be declared to be the friend of some other class. When we create a friend class then all the member functions of the friend class also become the friend of the other class.

Friend Function: A friend function is used for accessing the non-public members of a class. A class can allow non-member functions and other classes to access its own private data, by making them friends.

Inline Function: Inline function is the optimization technique used by the compilers. One can simply prepped inline keyword to function prototype to make a function inline.

Static Class Members: Static classes and class members are used to create data and functions that can be accessed without creating an instance of the class.

Static Data Members: Static data members are subject to class-member access rules, so private access to static data members is allowed only for class-member functions and friends.

6.16 Review Questions

1. What is return objects from class methods?
2. How to passed object to functions for any data type?
3. What is a class? How does it accomplish data hiding?
4. What is a Union?
5. How is a member function of a class defined?
6. How does a C++ structure differ from a C++ class?
7. What is a friend function? What are the merits and demerits of using friend functions?
8. What is the main reason for using structure?
9. When do we declare a member of a class static?
10. Describe the mechanism of accessing data members and member functions in the following cases:
    (a) Inside the main program.
    (b) Inside a member function of the same class.
    (c) Inside a member function of another class.

Answers for Self Assessment Questions

1 (d) 2 (b) 3 (a) 4 (d) 5 (c) 10 (a)

6.17 Further Studies

1. http://my.safaribooksonline.com/book/programming/cplusplus/9788131703830/6dot-classes-and-objects/navpoint108
2. The C++ Programming Language 4 Edition by Bjarne Stroustrup

7

Array, Pointers, References and the Dynamic Allocation Operators

STRUCTURE

7.0 Objectives

7.1 Introduction

7.2 Arrays of Objects

7.3 Pointers to Objects

7.4 Type Checking C++ Pointers

7.5 The this Pointer

7.6 Pointers to Derived Types

7.7 Pointers to Class Members

7.8 References

7.9 C++'s Dynamic Allocation Operators

7.10 Summary

7.11 Keywords

7.12 Review Questions

7.13 Further Studies

7.0 Objectives

After studying this chapter, you will be able to:

- Understand the Arrays of Objects
- Identifying and classifying the pointers to objects
- Discuss the type checking C++ pointers
- Explain the this pointer
- Discuss the pointers to derived types

7.1 Introduction

Pointers and arrays were examined as they relate to C++'s built-in types. Here, they are discussed relative to objects. This chapter also looks at a feature related to the pointer called a reference. The chapter concludes with an examination of C++'s dynamic allocation operators.

A pointer is a variable that is used to store a memory address. The address is the location of the variable in the memory. Pointers help in allocating memory dynamically. Pointers improve execution time and saves space. Pointer points to a particular data type.
The general form of declaring pointer is:-

```
type *variable_name;
```

type is the base type of the pointer and variable_name is the name of the variable of the pointer. For example,

```
int *x;
```

x is the variable name and it is the pointer of type integer.

7.2 Arrays of Objects

Arrays are data structures that store information in a set of adjacent memory addresses. In practice, this means is that you can store other variables and objects inside an array and can retrieve them from the array by referring to their position number in the array. Each variable or object in an array is called an element. An array of variables of type "class" is known as "Array of objects". The "identifier" used to refer the array of objects is a user defined data type.

It is possible to have arrays of objects. The syntax for declaring and using an object array is exactly the same as it is for any other type of array. For example, this program uses a three-element array of objects:

```
#include <iostream>
using namespace std;
class cl {
int i;
public:
void set_i(int j) { i=j; }
int get_i() { return i; }
};

int main()
{
cl ob[3];
int i;
for(i=0; i<3; i++) ob[i].set_i(i+1);
for(i=0; i<3; i++)
cout << ob[i].get_i() << "\n";
return 0;
}
```

7.2.1 Creating Initialized vs. Uninitialized Arrays

A special case situation occurs if you intend to create both initialized and uninitialized arrays of objects. Consider the following class.

```
class cl {
int i;
```

```
public:
cl(int j) { i=j; }
int get_i() { return i; }
};
```

Here, the constructor function defined by cl requires one parameter. This implies that any array declared of this type must be initialized. That is, it precludes this array declaration:

```
cl a[9]; // error, constructor requires initializes
```

The reason that this statement is not valid (as cl is currently defined) is that it implies that cl has a parameter less constructor because no initializes are specified. However, as it stands, cl does not have a parameter less constructor. Because there is no valid constructor that corresponds to this declaration, the compiler will report an error. To solve this problem, you need to overload the constructor function, adding one that takes no parameters. In this way, arrays that are initialized and those that are not are both allowed.

```
class cl {
int i;
public:
cl() { i=0; } // called for non-initialized arrays
cl(int j) { i=j; } // called for initialized arrays
int get_i() { return i; }
};
```

Given this class, both of the following statements are permissible:

```
cl a1[3] = {3, 5, 6}; // initialized
cl a2[34]; // uninitialized
```

7.3 Pointers to Objects

Just as you can have pointers to other types of variables, you can have pointers to objects. When accessing members of a class given a pointer to an object, use the arrow(–>) operator instead of the dot operator. The next program illustrates how to access an object given a pointer to it:

```
#include <iostream>
using namespace std;
class cl {
int i;
public:
cl(int j) { i=j; }
int get_i() { return i; }
};
int main()
{
cl ob(88), *p;
p = &ob; // get address of ob
cout << p->get_i(); // use -> to call get_i()
return 0;
}
```

When a pointer is incremented, it points to the next element of its type. For example, an integer pointer will point to the next integer. In general, all pointer arithmetic is relative to the base type of the pointer. (That is, it is relative to the type of data that the pointer is declared as pointing to.) The same is true of pointers to objects. For example, this program uses a pointer to access all three elements of array ob after being assigned ob's starting address:

```
#include <iostream>
using namespace std;
class cl {
int i;
public:
cl() { i=0; }
cl(int j) { i=j; }
int get_i() { return i; }
};
int main()
{
cl ob[3] = {1, 2, 3};
cl *p;
int i;
p = ob; // get start of array
for(i=0; i<3; i++) {
cout << p->get_i() << "\n";
p++; // point to next object
}
return 0;
}
```

7.4 Type Checking C++ Pointers

There is one important thing to understand about pointers. You may assign one pointer to another only if the two pointer types are compatible.
For example, given:

```
int *pi;
float *pf;
```

in C++, the following assignment is illegal:

```
pi = pf; // error -- type mismatch
```

Of course, you can override any type incompatibilities using a cast, but doing so bypasses C++'s type-checking mechanism.

Self Assessment Questions

1. A pointer to the base class can hold address of
(a) only base class object (b) only derived class object
(c) base class object as well as derived class object (d) None of the above

2. The address of a variable temp of type float is

(a) *temp
(b) &temp
(c) float& temp
(d) float temp&

3. If a variable is a pointer to a structure, then which of the following operator is used to access data members of the structure through the pointer variable?

(a). .
(b). &
(c). *
(d). ->

4. What is (void*)0?

(a). Representation of NULL pointer
(b). Representation of void pointer
(c). Error
(d). None of above

5. Pointers improve execution .....and save space.

(a). time
(b). location
(c). memory
(d). Variable

7.5 The this Pointer

When a member function is called, it is automatically passed an implicit argument that is a pointer to the invoking object (that is, the object on which the function is called). This pointer is called this. To understand this, first consider a program that creates a class called pwr that computes the result of a number raised to some power:

```
#include <iostream>
using namespace std;
class pwr {
double b;
int e;
double val;
public:
pwr(double base, int exp);
double get_pwr() { return val; }
};
pwr::pwr(double base, int exp)
{
b = base;
e = exp;
val = 1;
if(exp==0) return;
for( ; exp>0; exp--) val = val * b;
}
int main()
{
pwr x(4.0, 2), y(2.5, 1), z(5.7, 0);
cout << x.get_pwr() << " ";
cout << y.get_pwr() << "";
```

```
cout << z.get_pwr() << "\n";
return 0;
}
```

Within a member function, the members of a class can be accessed directly, without any object or class qualification. Thus, inside pwr(), the statement b = base; means that the copy of b associated with the invoking object will be assigned the value contained in base.
However, the same statement can also be written like this:
this->b = base;
The this pointer points to the object that invoked pwr(). Thus, this –>b refers to that object's copy of b. For example, if pwr() had been invoked by x (as in x(4.0, 2)), then this in the preceding statement would have been pointing to x. Writing the statement without using this is really just shorthand.
Here is the entire pwr() function written using the this pointer:

```
pwr::pwr(double base, int exp)
{
this->b = base;
this->e = exp;
this->val = 1;
if(exp==0) return;
for( ; exp>0; exp--)
this->val = this->val * this->b;
}
```

Actually, no C++ programmer would write pwr() as just shown because nothing is gained, and the standard form is easier. However, the pointer is very important when operators are overloaded and whenever a member function must utilize a pointer to the object that invoked it.
Remember that the this pointer is automatically passed to all member functions.

Therefore, get_pwr() could also be rewritten as shown here:
```
double get_pwr() { return this->val; }
```
In this case, if get_pwr() is invoked like this:
```
y.get_pwr();
```
then this will point to object y.
Two final points about this First, friend functions are not members of a class and, therefore, are not passed a this pointer. Second, static member functions do not have a this pointer.

Exercise: Check Your Progress 1
Note: i) Use the space below for your answer.
Ex1: Write a program to swap two number using pointers in C++.
……..……

……..……

……...

7.6 Pointers to Derived Types

A pointer of one type cannot point to an object of a different type. However, there is an important exception to this rule that relates only to derived classes. To begin, assume two classes called B and D. Further, assume that D is derived from the base class B. In this situation, a pointer of type B * may also point to an object of type D. More generally, a base class pointer can also be used as a pointer to an object of any class derived from that base.

Although a base class pointer can be used to point to a derived object, the opposite is not true. A pointer of type D * may not point to an object of type B. Further, although you can use a base pointer to point to a derived object, you can access only the members of the derived type that were imported from the base. That is, you would not be able to access any members added by the derived class. (You can cast a base pointer into a derived pointer and gain full access to the entire derived class, however.)

Here is a short program that illustrates the use of a base pointer to access derived objects.

```
#include <iostream>
using namespace std;
class base {
int i;
public:
void set_i(int num) { i=num; }
int get_i() { return i; }
};
class derived: public base {
int j;
public:
void set_j(int num) { j=num; }
int get_j() { return j; }
};
int main()
{
base *bp;
derived d;
bp = &d; // base pointer points to derived object
// access derived object using base pointer
bp->set_i(10);
cout << bp->get_i() << “”;
/* the following would not work. You cannot access element of
a derived class using a base class pointer.
bp->set_j(88); // error
cout << bp->get_j(); // error
*/
return 0;
}
```

As you can see, a base pointer is used to access an object of a derived class. Although you must be careful, it is possible to cast a base pointer into a pointer of the derived type to access a member of the derived class through the base pointer.
For example, this is valid C++ code:

```
// access now allowed because of cast
((derived *)bp)->set_j(88);
cout << ((derived *)bp)->get_j();
```

It is important to remember that pointer arithmetic is relative to the base type of the pointer. For this reason, when a base pointer is pointing to a derived object, incrementing the pointer does not cause it to point to the next object of the derived type. Instead, it will point to what it thinks is the next object of the base type. This, of course, usually spells trouble. For example, this program, while syntactically correct, contains this error.

```
// This program contains an error.
#include <iostream>
using namespace std;
class base {
int i;
public:
void set_i(int num) { i=num; }
int get_i() { return i; }
};
class derived: public base {
int j;
public:
void set_j(int num) {j=num;}
int get_j() {return j;}
};
int main()
{
base *bp;
derived d[2];
bp = d;
d[0].set_i(1);
d[1].set_i(2);
cout << bp->get_i() << “”;
bp++; // relative to base, not derived
cout << bp->get_i(); // garbage value displayed
return 0;
}
```

The use of base pointers to derived types is most useful when creating run-time polymorphism through the mechanism of virtual functions.

Caution

The array size must be declared at the time of declaration of array. Because arrays size is fixed and it must be specified at compile time.

Did You Know?
In 1957, the first of the major languages appeared in the form of FORTRAN.

7.7 Pointers to Class Members

C++ allows you to generate a special type of pointer that "points" generically to a member of a class, not to a specific instance of that member in an object. This sort of pointer is called a pointer to a class member or a pointer-to-member, for short. A pointer to a member is not the same as a normal C++ pointer. Instead, a pointer to a member provides only an offset into an object of the member's class at which that member can be found. Since member pointers are not true pointers, the . and -> cannot be applied to them. To access a member of a class given a pointer to it, you must use the special pointer-to-member operators .* and –>*. Their job is to allow you to access a member of a class given a pointer to that member.
Here is an example:

```
#include <iostream>
using namespace std;
class cl {
public:
cl(int i) { val=i; }
int val;
int double_val() { return val+val; }
};
int main()
{
int cl::*data; // data member pointer
int (cl::*func)(); // function member pointer
cl ob1(1), ob2(2); // create objects
data = &cl::val; // get offset of val
func = &cl::double_val; // get offset of double_val()
cout << "Here are values: ";
cout << ob1.*data << "" << ob2.*data << "\n";
cout << "Here they are doubled: ";
cout << (ob1.*func)() << "";
cout << (ob2.*func)() << "\n";
return 0;
}
```

In main(), this program creates two member pointers: data and func.
Note carefully the syntax of each declaration. When declaring pointers to members, you must specify the class and use the scope resolution operator. The program also creates objects of cl called ob1 and ob2. As the program illustrates, member pointers may point to either functions or data. Next, the program obtains the addresses of val and double_val(). As stated earlier, these "addresses" are really just offsets into an object of type cl, at which point val and double_val() will be found. Next, to display the values of each object's val, each is accessed through data. Finally, the program uses func to call the double_val() function. The extra parentheses are necessary in order to correctly associate the .* operator.

When you are accessing a member of an object by using an object or a reference, you must use the .* operator. However, if you are using a pointer to the object, you need to use the –>* operator, as illustrated in this version of the preceding program:

```
#include <iostream>
using namespace std;
class cl {
public:
cl(int i) { val=i; }
int val;
int double_val() { return val+val; }
};
int main()
{
int cl::*data; // data member pointer
int (cl::*func)(); // function member pointer
cl ob1(1), ob2(2); // create objects
cl *p1, *p2;
p1 = &ob1; // access objects through a pointer
p2 = &ob2;
data = &cl::val; // get offset of val
func = &cl::double_val; // get offset of double_val()
cout << “Here are values: ”;
cout << p1->*data << “” << p2->*data << “\n”;
cout << “Here they are doubled: ”;
cout << (p1->*func)() << “”;
cout << (p2->*func)() << “\n”;
return 0;
}
```

In this version, p1 and p2 are pointers to objects of type cl. Therefore, the –>* operator is used to access val and double_val(). Remember, pointers to members are different from pointers to specific instances of elements of an object. Consider this fragment (assume that cl is declared as shown in the preceding programs):

```
int cl::*d;
int *p;
cl o;
p = &o.val // this is address of a specific val
d = &cl::val // this is offset of generic val
```

Here, p is a pointer to an integer inside a specific object. However, d is simply an offset that indicates where val will be found in any object of type cl. In general, pointer-to-member operators are applied in special-case situations. They are not typically used in day-to-day programming.

Exercise: Check Your Progress 2
Note: i) Use the space below for your answer.

Ex1: Write a program in C++of bubble sort using pointer?

……..……

……..……

……...

7.8 References

C++ contains a feature that is related to the pointer called a reference. A reference is essentially an implicit pointer. There are three ways that a reference can be used: as a function parameter, as a function return value, or as a stand-alone reference. Each is examined here.

7.8.1 Reference Parameters

Probably the most important use for a reference is to allow you to create functions that automatically use call-by-reference parameter passing. Arguments can be passed to functions in one of two ways: using call-by-value or call-by-reference. When using call-by-value, a copy of the argument is passed to the function. Call-by-reference passes the address of the argument to the function. By default, C++ uses call-by-value, but it provides two ways to achieve call-by-reference parameter passing. First, you can explicitly pass a pointer to the argument. Second, you can use a reference parameter. For most circumstances the best way is to use a reference parameter.

To fully understand what a reference parameter is and why it is valuable, we will begin by reviewing how a call-by-reference can be generated using a pointer parameter. The following program manually creates a call-by-reference parameter using a pointer in the function called neg(), which reverses the sign of the integer variable pointed to by its argument.

```
// Manually create a call-by-reference using a pointer.
#include <iostream>
using namespace std;
void neg(int *i);
int main()
{
int x;
x = 10;
cout << x << " negated is ";
neg(&x);
cout << x << "\n";
return 0;
}
void neg(int *i)
{
*i = -*i;
}
```

7.8.2 Passing References to Objects

It was explained that when an object is passed as an argument to a function, a copy of that object is made. When the function terminates, the copy's destructor is called. If for some reason you do not want the destructor function to be called, simply pass the object by reference. When you pass by

reference, no copy of the object is made. This means that no object used as a parameter is destroyed when the function terminates, and the parameter's destructor is not called.
For example, try this program:

```
#include <iostream>
using namespace std;
class cl {
int id;
public:
int i;
cl(int i);
~cl();
void neg(cl &o) { o.i = -o.i; } // no temporary created
};
cl::cl(int num)
{
cout << "Constructing " << num << "\n";
id = num;
}
cl::~cl()
{
cout << "Destructing " << id << "\n";
}
int main()
{
cl o(1);
o.i = 10;
o.neg(o);
cout << o.i << "\n";
return 0;
}
```

Here is the output of this program:

```
Constructing 1
-10
Destructing 1
```

Did You Know?
In 1990, *The Annotated C++ Reference Manual* was published.

7.9 C++'s Dynamic Allocation Operators

C++ provides two dynamic allocation operators: new and delete. These operators are used to allocate and free memory at run time. Dynamic allocation is an important part of almost all real-world programs. As explained in Part One, C++ also supports dynamic memory allocation functions, called malloc() and free(). These are included for the sake of compatibility. However, for C++ code, you should use the new and delete operators because they have several advantages. The new operator

allocates memory and returns a pointer to the start of it. The delete operator frees memory previously allocated using new.
The general forms of new and delete are shown here:
p_var = new type;
delete p_var;
Here, p_var is a pointer variable that receives a pointer to memory that is large enough to hold an item of type type.
Since the heap is finite, it can become exhausted. If there is insufficient available memory to fill an allocation request, then new will fail and a bad_alloc exception will be generated. This exception is defined in the header <new>. Your program should handle this exception and take appropriate action if a failure occurs. If this exception is not handled by your program, then your program will be terminated.
Here is a program that allocates memory to hold an integer:

```
#include <iostream>
#include <new>
using namespace std;
int main()
{
int *p;
try {
p = new int; // allocate space for an int
} catch (bad_alloc xa) {
cout << “Allocation Failure\n”;
return 1;
}
*p = 100;
cout << “At ” << p << “”;
cout << “is the value ” << *p << “\n”;
delete p;
return 0;
}
```

7.9.1 Initializing Allocated Memory

You can initialize allocated memory to some known value by putting an initializer after the type name in the new statement.
Here is the general form of new when an initialization is included:
p_var = new var_type (initializer);

Of course, the type of the initializer must be compatible with the type of data for which memory is being allocated.
This program gives the allocated integer an initial value of 87:

```
#include <iostream>
#include <new>
using namespace std;
int main()
{
```

```
int *p;
try {
p = new int (87); // initialize to 87
} catch (bad_alloc xa) {
cout << “Allocation Failure\n”;
return 1;
}
cout << “At ” << p << “”;
cout << “is the value ” << *p << “\n”;
delete p;
return 0;
}
```

7.9.2 Allocating Arrays

You can allocate arrays using new by using this general form:

```
p_var = new array_type [size];
```

Here, size specifies the number of elements in the array.
To free an array, use this form of delete:

```
delete [ ] p_var;
```

Here, the [ ] informs delete that an array is being released.
For example, the next program allocates a 10-element integer array.

```
#include <iostream>
#include <new>
using namespace std;
int main()
{
int *p, i;
try {
p = new int [10]; // allocate 10 integer array
} catch (bad_alloc xa) {
cout << “Allocation Failure\n”;
return 1;
}
for(i=0; i<10; i++ )
p[i] = i;
for(i=0; i<10; i++)
cout << p[i] << “”;
delete [] p; // release the array
return 0;
}
```

Exercise: Check Your Progress 3

Case Study-C++ In Scientific Application

Galileo provided us the basis of all scientific computation when he wrote, “We must measure what is measurable, and make measurable what is not so.” However, mensuration in other contexts has a long history. For example, Leviticus 19:35 enjoins us, “Ye shall do no unrighteousness in judgment, in measures of length, of weight, or of quantity.”

Clearly, there has been significant progress in accuracy. Consider, as a very simple example, the value of pi. In I Kings 7:23 we read, “And he made a molten sea, ten cubits from brim to brim …, and a line of thirty cubits did compass it round about.” The ratio of these data yields $\pi = 3$, an error of less than 5%.

It is noteworthy that widespread use of standardized units came only much later. Among the significant provisions of the Magna Carta, we find the declarations, “Throughout the kingdom there shall be standard measures of wine, ale, and corn.… Weights [also] are to be standardized similarly.”

Accepted First Principles

A beginning student in any scientific discipline is taught, almost at once, the critical importance of calculations involving physical quantities. Often termed “dimensional analysis” or “quantity algebra,” the appropriate techniques are invariably stressed to

students early on. For example, Halliday & Resnick emphasize the importance of such discipline in such terms as:

- “In carrying out any calculation, always … attach the proper units to the final result, for the result is meaningless without this…”
- “You should check the dimensions of all the equations you use.”
- “One way to spot an erroneous equation is to check the dimensions of all its terms….”

If these are indicative of the accepted and expected methodology to be used in hand calculation of physical quantities, consider the formulation of analogous statements bearing on computer-based calculations.

For example, we might have, as a parallel to the last of the above (differences italicized):

- “One way to spot an erroneous program is to check the data types of all its objects.”

This clearly suggests that it is likely to be useful for computer-based calculations to carry out both units checking and type checking.

Type Checking Augments Traditional Units Checking

Type checking is far from a new concept. Aristotle, for example, wrote, “There is no transfer into another kind, like the transfer from length to area and from area to solid.” Indeed, there is considerable evidence that type checking is still needed.

As a representative case in point, a correspondent (who has requested his identity be withheld) writes, “A famous bug in this shop is 2*pi+r*w as the area of a strip of a cylinder, which escaped notice for years….” This incident provides compelling evidence of the subtle nature of computer-based calculation. It also strongly hints at the value of computer-based type checking, for such checking would have discovered that the above expression attempts to sum incommensurate quantities and is thus inconsistent with a purported area calculation.

Siunits Programming

Within Siunits, physical quantities (often known as dimensions) are implemented as types. Of particular interest, and consistent with SI, Siunits provides the seven base quantities (Length, Mass, Time, Current, Temperature, amount of substance, and Luminous Intensity). It is from these seven that all remaining quantities are derived. Indeed, 22 derived quantities are given special names by SI; these include, for example, Energy, Force, Entropy, etc. These and many, many more derived quantities are provided by Siunits.

Units behave as right-hand constants. Of special note are the 7 base units (meter, kilogram, second, Ampere, Kelvin, mole, candela) and the 22 composite units (joule, watt, hertz, katal, …) specified by

SI, as well as the various multiples and submultiples (mega-, nano-, etc.). All these are found in Siunits, too.

Questions

1. What is the type checking augments traditional units checking?
2. What is the Siunits programming?

7.10 Summary

- Pointer is a variable that is used to store a memory address. The address is the location of the variable in the memory. Pointers help in allocating memory dynamically.
- Reference is a relation between objects in which one object designates, or acts as a means by which to connect to or link to, another object. The first object in this relation is said to refer to the second object.
- The pointer accessible only within the nonstatic member functions of a class, struct, or union type. It points to the object for which the member function is called. Static member functions do not have a pointer.
- Dynamic allocation is one of the some ways of using memory provided by the C++ standard. To accomplish this in C++ the malloc function is used and the new keyword is used for C++.
- Memory allocated and deallocated dynamically by using the new and the delete operators.

7.11 Keywords

Array of Objects: An array of variables of type "class" is known as "Array of objects". The "identifier" used to refer the array of objects is a user defined data type.

malloc(): Allocates a block of size bytes of memory, returning a pointer to the beginning of the block. The content of the newly allocated block of memory is not initialized, remaining with indeterminate values.

new Operator: Pointers provide the necessary support for C++'s powerful dynamic memory allocation system. Dynamic allocation is the means by which a program obtains memory while it is running.

Pointer: It is a variable which contains the address in memory of another variable. We have a pointer to any variable type.

this Pointer: Every class member function has a hidden parameter the this pointer. this points to the individual object.

7.12 Review Questions

1. Which is the parameter that is added to every non-static member function when it is called?
2. What is pointer? Explain with examples.
3. What will be the output of the following program?

```
#include<iostream.h>
void main()
{
 float x=5,y=2;
int result;
result=x % y;
cout<<result;
```

```
}
```
4. What will be output of following program?
```
#include< iostream >
int main(){
   int a = 320;
   char *ptr;
   ptr =( char *)&a;
cout <<*ptr;
   return 0;
}
```
5. What will be output of following program?
```
#include< iostream >
int main(){
   int i = 3;
   int *j;
   int **k;
   j=&i;
   k=&j;
cout << k<<*k<<**k;
   return 0;
}
```
6. What will be output of following program?
```
#include< iostream >
int main(){
   void (*p)();
   int (*q)();
   int (*r)();
   p = clrscr;
   q = getch;
   r = puts;
  (*p)();
  (*r)("cquestionbank.blogspot.com");
  (*q)();
  return 0;
}
```
7. Define the array of objects and pointer to object.
8. What are the type checking C++ pointers?
9. What are the dynamic allocation operators?
10. Explain the pointer to derived types.

Answers for Self Assessment Questions

1. (c) 2. (b) 3. (b) 4. (a) 5. (a)

7.13 Further Studies

1. http://msdn.microsoft.com/en-us/library/hh184281(v=vs.88).aspx
2. The C++ Programming Language 4 Edition by Bjarne Stroustrup

8

Constructors and Destructor

STRUCTURE

8.0 Objectives

8.1 Introduction

8.2 Constructors

8.3 Default Constructor

8.4 Parameterized Constructors

8.5 Copy Constructors

8.6 Multiple Constructors in a Class

8.7 Constructors with Default Arguments

8.8 Special Characteristics of Constructor Functions

8.9 Destructors

8.10 Summary

8.11 Keywords

8.12 Review Questions

8.13 Further Studies

8.0 Objectives

After studying this chapter, you will be able to:

- Explain the constructors
- Discuss the default constructor
- Define the parameterized constructors
- Understand the concept of copy constructors

8.1 Introduction

It is very common for some part of an object to require initialization before it can be used. For example, think back to the stack class developed earlier. Before the stack could be used, to had to be set to zero. This was performed by using the function init(). Because the requirement for initialization

is so common, C++ allows objects to initialize themselves when they are created. This automatic initialization is performed through the use of a constructor function.
Because classes have complicated internal structures, including data and functions, object initialization and cleanup for classes is much more complicated than it is for simple data structures. Constructors and destructors are special member functions of classes that are used to construct and destroy class objects. Construction may involve memory allocation and initialization for objects. Destruction may involve cleanup and deallocation of memory for objects.

8.2 Constructors

A Class is defined as constructor by declaring a constructor in the form of function inside the class. In other word a function in the name of class is a constructor with few exceptions from the regular function. Constructor is a special member function. The constructor must have identical name as the class. Constructor is the extension of class with additional features and certain limitations. The class cannot be defined without parameters or arguments that are class can be defined in the form of function and thus class can be declared with argument and without argument. The class without argument is called default constructor with argument is known as parameterized constructors. The construction is used to initialize the class; as seen earlier a class cannot be initialized; this limitation is over come by having a constructor inside the class. The constructor can be overload.

Rules of Constructor:

- The constructors must have the same name as the class.
- The constructors will take the form of function prototype.
- The constructors are used to initialize the class and its objects.
- The constructors are invoked automatically as soon as the objects are created.
- The constructors do not return type even void like function.
- The constructors cannot be inherited.
- The constructors can have a default argument.
- The constructors may or may not have argument.
- The constructors can also be defined as inline function.
- The constructors cannot have its class as argument.
- The constructors can have its class as argument but only through reference. This is known as copy constructor.
- The constructor can be constant, thus constructor arguments never be changed.
- Constructor can be empty or without statement inside it. It is a do nothing constructor it must be created when overloading operator is defined in the class.
- Constructor can also have other than initialization statement

A constructor is defined as follows which will interpret the rule of constructor.

Constructor without argument or default constructor:

```
class integer
{
int m, n;
public:
integer(void);
```

```
};
integer::integer(void)
{
m=0; n=0;
}
```

Constructor with argument or parameterized constructor:

```
class integer
{
int m, n;
public:
integer(int x, int y);
};
integer::integer(int x, int y)
{m=x; n=y;
}
```

Calling a constructor:

1. Explicit calling. 2. Implicit calling.

Explicit calling, it is declared by combining the declaration and passing the argument to the constructor.

```
integer int1=integer(10,20);
```

The above statement can re written in the following form.

```
integer int1;
int1(10,20);
```

The above method is implicit calling it can also be made shorter in the following way.

```
integer int1(10,20);
```

Did You Know?

Around 1990, Ed Burton created the precursor to sodaconstructor - a BASIC program entitled "Constructor" which contained some of soda constructor's modern functionality, with both springs and masses, but not muscles, available for construction.

8.3 Default Constructor

A default constructor is a constructor that either has no parameters, or if it has parameters, all the parameters have default values.

If no user-defined constructor exists for a class A and one is needed, the compiler implicitly declares a default parameter less constructor A::A(). This constructor is an inline public member of its class. The compiler will implicitly define A::A() when the compiler uses this constructor to create an object of type A. The constructor will have no constructor initializer and a null body.

The compiler first implicitly defines the implicitly declared constructors of the base classes and nonstatic data members of a class A before defining the implicitly declared constructor of A. No default constructor is created for a class that has any constant or reference type members.

A constructor of a class A is trivial if all the following are true:

- It is implicitly defined

- A has no virtual functions and no virtual base classes
- All the direct base classes of A have trivial constructors
- The classes of all the nonstatic data members of A have trivial constructors

If any of the above is false, then the constructor is nontrivial.
A union member cannot be of a class type that has a nontrivial constructor.
Like all functions, a constructor can have default arguments. They are used to initialize member objects. If default values are supplied, the trailing arguments can be omitted in the expression list of the constructor.
Note that if a constructor has any arguments that do not have default values, it is not a default constructor.

Caution
The constructors must be declared in the public section of the class.

8.4 Parameterized Constructors

C++ permits us to achieve this objects but passing argument to the constructor function when the object are created. The constructors that can take arguments are called parameterized constructors.
When the object is created, we must supply arguments to the constructor function.
This can be done in two ways:
- By calling the function explicitly
- By calling the function implicitly

The first call is implemented as follows :
student S1 = student (1, 70) ;

The second call is implemented as follows :
student S1 (1, 70) ;
The second method is used very often as it is shorter.

Example:-
```
class abc
{
int m,n;
public:
abc(int x,int y); //paramererise constructor
.....
.....
};
abc::abc(int x,int y)
{
m=x; n=y;
}
```

If only the derived's constructor takes parameters then also it. But what if both the base and derived class contains parameterized constructors.

```
// this code contains ERRORS

// base class
class base
{
  int a;

public:
  base(int n)
  {
    //...
  }
};

// derived class
class derived:public base
{
  int b;

public:
  derived(int m)
  {
    //...
  }
};

//main
void main()
{
  base b(10); //ok

  derived d(10); // ERROR!!
    // base's constructor
    // will also be called and
    // it needs parameters too!!
}
```

We need to introduce expanded from of constructor declaration (of the derived class).

```
  derived-constructor (arg): base1 (arg),
                             base2 (arg),
                             ...
                             baseN (arg)
  {
    ...
    ...
  }
```

Here base1, base2 etc. are constructor functions of base classes, which the derived class inherits.

Let us have a look at an example program to understand this declaration:

```
// this code is ok

// base class
class base
{
  int a;

public:
  base(int n){a=n;}
};

// derived class
class derived:public base
{
  int b;

public:
  // since this class is derived
  // from only one class 'base'
  // therefore only one constructor
  // is listed
  derived(int m1,int m2):base(m2)
  {b=m1;}
};

// main
void main()
{
  base b(10); //ok

  derived d(10,100); //ok
}
```

One thing to note here is, the argument which is to be passed to the base's constructor should be declared and accepted from the derived class constructor, as in the line

derived(int m1,int m2):base(m2)
{b=m1;}

Caution

Constructors cannot return values. Specifying a constructor with a return type is an error, as is taking the address of a constructor.

8.5 Copy Constructors

A copy constructor is a special constructor in the C++ programming language used to create a new object as a copy of an existing object. This constructor takes a single argument: a reference to the object to be copied. Normally the compiler automatically creates a copy constructor for each class (known as an implicit copy constructor) but for special cases the programmer creates the copy constructor, known as an explicit copy constructor. In such cases, the compiler does not create one.
The copy constructor is a constructor which creates an object by initializing it with an object of the same class, which has been created previously.
The copy constructor is used to:
Initialize one object from another of the same type.

- Copy an object to pass it as an argument to a function.
- Copy an object to return it from a function.

If a copy constructor is not defined in a class, the compiler itself defines one. If the class has pointer variables and has some dynamic memory allocations, then it is a must to have a copy constructor.
The most common form of copy constructor is shown here:

```
classname (const classname &obj) {
  // body of constructor
}
```

Here, obj is a reference to an object that is being used to initialize another object.

```
#include <iostream>
using namespace std;
class Line
{
  public:
    int getLength( void );
    Line( int len );            // simple constructor
    Line( const Line &obj);  // copy constructor
    ~Line();                    // destructor

  private:
    int *ptr;
};
// Member functions definitions including constructor
Line::Line(int len)
{
   cout << “Normal constructor allocating ptr” << endl;
   // allocate memory for the pointer;
   ptr = new int;
   *ptr = len;
}
Line::Line(const Line &obj)
{
   cout << “Copy constructor allocating ptr. ” << endl;
```

```
    ptr = new int;
   *ptr = *obj.ptr; // copy the value
}

Line::~Line(void)
{
    cout << "Freeing memory! " << endl;
    delete ptr;
}
int Line::getLength( void )
{
    return *ptr;
}

void display(Line obj)
{
   cout << "Length of line : " << obj.getLength() <<endl;
}

// Main function for the program
int main( )
{
   Line line(10);

   display(line);

   return 0;
}
```

When the above code is compiled and executed, it produces following result:

```
Normal constructor allocating ptr
Copy constructor allocating ptr.
Length of line : 10
Freeing memory!
Freeing memory!
```

Self Assessment Questions

1. How many constructors can a class have?

(a) 0 (b) 1
(c) 2 (d) any number

2. What functions should all classes have?

(a) Constructor (b) Destructor
(c). Constructor and destructor (d). None of these

3. The copy constructor

(a) Creates an object from any object

(b). Creates an object from an object of the same class
(c). Creates an object that is a pointer to the copied object
(d). All the previous answers are incorrect

8.6 Multiple Constructors in a Class

We have used two kinds of constructors. They are:

```
integer();        // No arguments
integer(int, int);        // Two arguments
```

In the first case, the constructor itself supplies the data values and no values are passed by calling program. In the second case, the function call passes the appropriate values from main(). C++ permits us to use both these constructors in the same class. For example, we define a class as follows:

```
class integer
{
int m, n;
public:
integer(){m=0; n=0;}// constructor 1
integer(int a, int b)
{m = a; n = b;}//constructor 2
integer(integer & i)
{m = i.m; n= i.n;}      ;/ constructor 3
};
```

This declares three constructors for an integer object. The first constructor receives no arguments, the second receives two integer arguments and the third receives one integer object as an argument. For example, the declaration

```
integer I1;
```

Would automatically invoke the first constructor and set both m and n of 11 to zero. The statement

```
integer 12(20,40);
```

Would call the second constructor which will initialize the data members m and n of 12 to 20 and respectively. Finally, the statement

```
integer I3(I2);
```

Would invoke the third constructor which copies the values of 12 into 13. In other words, it sets the value of every data element of 13 to the value of the corresponding data element of 12. As mentioned earlier, such a constructor is called the copy constructor.

Example:

Program shows the use of overloaded constructors.

```
#include <iostream>
using namespace std;
class complex
{
Float x y;
public:
complex(){ }// constructor no arg
complex(float a) {x = y = a;}// constructor-one arg
complex(float real, float imag)// constructor-two orgs {x = real; y = imag;)
{x = real; y = imag;}
```

```
friend complex sum(complex, complex);
friend void show(complex);
};
complex sum(complex cl, complex c2)          // friend
{
Complex c3;
C3.x = c1.x + c2.x;
C3.y = c1.y + c2.y;
Return(c3);
}
void show(complex c) //friend
{
cout << c .x << " + j" << c .y  << "\n";
}
int main()
{
Complex A(2.7, 3.5); //define  & initialize
Complex B(1.6); //define & initialize
Complex C; // define
C = sum(A, B); //sum() is a friend
Cout << "A = "; show(A); //sum() is a also friend
Cout << "B = "; show(B);
Cout << "C = "; show(C);
// Another way to give initial values (second method)
complex P,Q,R; // define P, Q and R
P = complex(2.5, 3.9); //initialize P
Q = complex(1.6, 2.5); //initialize Q
R = sum(P,Q);
Cout << "\n";
Cout << "p = "; show(P);
Cout << "Q = "; show(Q);
Cout << "R = "; show(R);
Return 0;
}
```

The output of Program would be:

```
A = 2.7          + j3.5
B = 1.6 + j1.6
C = 4.3 + j5.1
P = 2.5 + j3.9
Q = 1.6 + j2.5
R = 4.1 + j6.4
```

There are three constructors in the class complex. The first constructor, which takes no arguments, is used to create objects which are not initialized; the second, which takes one argument, is used to create objects and initialize them; and the third, which takes two arguments, is also used to create objects and initialize them to specific values. Note that the second method of initializing values looks better.

Let us look at the first constructor again.
complex(){ }
It contains the empty body and does not do anything. We just stated that this is used to create objects without any initial values. Remember: we have defined objects in the examples without using such a constructor.

8.7 Constructors with Default Arguments

The constructor can be declared with default argument.
For example :
student (int rn, int total = 0) ;
Here the default value of total is zero.
Then the statement
student S1 (2) ;
assigns the value 2 to rn and 0 to total.
However, the statement
student S2 (3, 75) ;
assigns 3 to rn and 75 to total. In this case actual parameter takes the priority over default parameter. All default values should be on the right side.
Consider the following statement
A (int = 0) ;
It has only one argument. It can be called in two ways.
A B ;
A B (5) ;
In the first statement, no parameter is supplied. In the second statement, one parameter is supplied. When no parameter is supplied, it becomes a default constructor. When both the forms are used in a class (default constructor and constructor with one default argument), it causes ambiguity for a statement such as
A B
(Whether to call A ( ) or A (int = 0)

8.8 Special Characteristics of Constructor Functions

A constructor is an especially member function whose task is to initialize the objects of its class. It is special because its name is same as the class name. The constructor is invoked whenever an object of its associated class is created. It is called constructor because it construct the value data members of the class
The constructor functions have some special characteristics.

- They should be declared in the public section.
- They are invoked automatically when the objects are created.
- They do not have return types, not even void and therefore, they cannot return values.
- They cannot be inherited, though a derived class can call the base class constructor.
- Like other C++ functions, they can have default arguments.
- Constructors cannot be virtual.
- We cannot refer to their addresses.
- An object with a constructor (or destructor) cannot be used as a member of a union.

- They make implicit calls to the operations new and delete when memory allocation is required. Remember, when a constructor is declared for a class initialization of the class objects become mandatory.

8.9 Destructors

As opposed to a constructor, a destructor is called when a program has finished using an instance of an object. A destructor does the cleaning behind the scenes. Like the default constructor, the compiler always create a default destructor if you don't create one. Like the default constructor, a destructor also has the same name as its object. This time, the name of the destructor starts with a tilde.

To create your own destructor, in the header file, type ~ followed by the name of the object. Here is an example:

```
#ifndef BricksH
#define BricksH
class TBrick
{
public:
    TBrick();
    TBrick(double L, double h, double t);
    TBrick(const TBrick &Brk);
   ~TBrick();
   double  getLength() const;
   void  setLength(const double l);
   double  getHeight() const;
   void  setHeight(const double h);
   double  getThickness() const;
   void  setThickness(const double t);
   double  CementVolume();
   void  ShowProperties();
private:
   double Length;
   double Height;
   double Thickness;
};

#endif
```

As done with a default constructor, you don't need to put anything in the implementation of a destructor. In fact, when a program terminates, the compiler can itself destroy all of the objects and variables that your program has used. The only true time you will be concerned with destroying objects is if the objects were created dynamically, which we will learn when studying pointers.

You can implement your destructor in the header file by just providing it with empty parentheses:

```
ifndef BricksH
#define BricksH
```

```
class TBrick
{
public:
   ...~TBrick() {}
   ...private:
   ..};
#endif
```

Did You Know?

The destructor has the same name as the class, but with a tilde (~) in front of it. If the object was created as an automatic variable, its destructor is automatically called when it goes out of scope.

Self Assessment Questions

4. To initialize the data members of a class, it is used

(a) Constructor
(b). Destructor
(c). Data members are automatically initialized
(d). All the previous answers are incorrect

5. The copy constructor is invoked

(a) When an object is passed as a parameter to a function by reference
(b). When a function returns an object by value
(c). When one object is assigned to another object
(d). All the previous answers are incorrect

6. A class can have

(a) All constructors that are needed
(b) Only the default constructor and the copy constructor
(c). A destructor for each constructor
(d). All the previous answers are incorrect

Exercise: Check Your Progress 1

Case Study-Aspectual Mixin Layers clarify the use of Feature C++

In particular, it gives insight in how to implement Aspectual Mixin Layers. We choose the stock information broker example, adopted from, in order to point to the benefits of Aspectual Mixin Layers compared to common FOP approaches.

In particular, we show how FeatureC++ overcomes the problems stock Information Broker. A stock information broker provides information about the stock market. The central abstraction is the StockInformationBroker (SIB) that allows to lookup for information of a set of stocks (see Figure. 1). A Client can pass a StockInfoRequest (SIR) to the SIB by calling the method collectInfo. The SIR contains the names of all requested stocks. Using the SIR, the SIB queries the DBBroker in order to retrieve the requested information. Then, the SIB returns a StockInfo (SI) object, which contains the stock quotes, to the client.

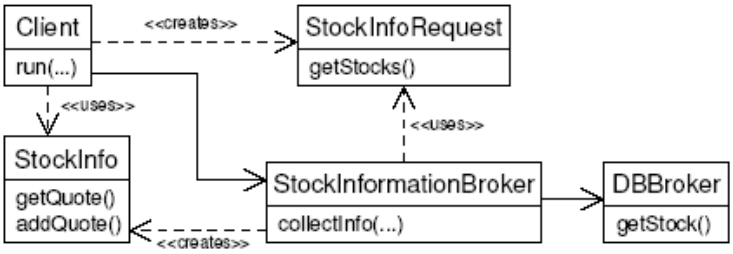


Figure 1: Stock information broker.

All classes are implemented as classes (constants) and are encapsulated by a Mixin Layer. In other words, this Mixin Layer implements a basic
stock information broker feature (BasicSIB).

```
1 class StockInformationBroker {
2 DBBroker m_db ;
3 public:
4 StockInfo & collectInfo ( StockInfoRequest & req ) {
5 string * stocks = req . getStocks ();
6 StockInfo * info = new StockInfo ();
7 for ( unsigned int i = 0; i < req . num (); i ++)
8 info -> addQuote ( stocks [i], m_db . get ( stocks [i ]));
9 return * info ; }
10 };
11
12 class Client {
13 StockInformationBroker & m_broker ;
14 public:
15 void run ( string * stocks , unsigned int num ) {
16 StockInfo & info = m_broker . collectInfo ( StockInfoRequest ( stocks , num ));
17 ... }
18 };
```

Pricing Feature as Mixin Layer: Now, we want to add a Pricing feature that charges the clients account depending on the received stock quotes. Depicts this feature implemented using common FOP concepts. Client is refined by an account management (Lines 16-23), SIR is refined by a price calculation (Lines 2-5), and SIB charges the clients account when passing information to the client (Lines 10-12). There are several problems to this approach:

(1) The Pricing features is expressed in terms of the structure of the BasicSIB feature. This problem is caused because FOP can only express hierarchy-conform refinements. It would be better to describe the pricing feature using abstractions as product and customer.

(2) The interface of collectInfo was extended. Therefore, the Client must override the method run in order to pass a reference of itself to the SIB. This is an inelegant workaround and increases the complexity.

(3) The charging procedure of the clients cannot be altered depending on the runtime control flow. Moreover, it is assigned to the SIB which is clearly not responsible for this function.

(4) An hypothetical accounting functionality that traces and logs the transfers (not depicted) suffers from excessive method shadowing because all affected methods, e.g. collectInfo, price, balance, etc., have to be shadowed. Pricing Feature as Aspectual Mixin Layer. Depicts the pricing feature

implemented by an Aspectual Mixin Layer. The key difference is the Charging aspect. It serves as an observer of calls to the method collectInfo. Every call to this method is intercepted and the client is charged depending on its request. This solves the problem of the extended interface because the client is charged by the aspect instead by the SIB. An alternative is to pass the client'

refines class StockInfoRequest

```
1 {
2 f loat basicPrice ();
3 f loat calculateTax ();
4 public:
5 f loat price ();
6 };
7
8 refines class StockInformationBroker {
9 public:
10 StockInfo & collectInfo ( Client &c, StockInfoRequest & req ) {
11 c. charge ( req );
12 return super:: collectInfo ( req ); }
13 };
14
15 refines class Client {
16 f loat m_balance ;
17 public:
18 f loat balance ();
19 void charge ( StockInfoRequest & req );
20 void run ( string * stocks , unsigned int num ) {
21 StockInfo & info = super:: m_broker . collectInfo (* this ,
22 StockInfoRequest ( stocks , num ));
23 ... }
24 };
```

reference to the extended collectInfo method (not depicted). In both cases, the Client does not need to override the run method. A further advantage is that the charging of client' accounts can be made dependent to the control flow (using the cflow pointcut). This makes it possible to implement the charging function variable. In this context, the method shadowing is prevented by using wildcard expressions in pointcuts, e.g., for capturing calls to all methods which are relevant for price transfer (accounting feature for tracing and logging transfers). Finally, our example shows that using Aspectual Mixin Layers we were able to refine only these classes that play the roles of product (SIR) and customer (Client).

Questions

1. What are the aspectual mixin layers clarify the use of Feature C++?
2. What is the pricing feature as mix in layer?

8.10 Summary

- Constructors are called automatically by the compiler when defining class objects. The destructors are called when a class object goes out of scope.

- A copy constructor is a special constructor in the C++ programming language for creating a new object as a copy of an existing object.
- Destructors are usually used to deallocate memory and do other cleanup for a class object and its class members when the object is destroyed.
- In computer programming languages the term "default constructor" refers to a constructor that is automatically generated in the absence of explicit constructors this automatically provided constructor is usually a nullary constructor.
- A destructor is called for a class object when that object passes out of scope or is explicitly deleted.

8.11 Keywords

Constructor: If however, any kind of constructor is declared by the programmer than a default is not supplied default values.

Default Arguments: C++ allows a function to assign default values to parameters. The default value is assigned when no argument corresponding to that parameter is specified in the call to that function.

Default Constructor: It is a constructor in C++ that has no parameters or where it has parameters they are all defaulted. If no constructor is supplied then the compiler will supply a default.

Destructor: It is used to destroy the objects that have been created by a constructor.

Parameterized Constructors: The constructors that take arguments are called parameterized constructors.

8.12 Review Questions

1. How did you pass arguments to the base class constructor?
2. Explain the constructor?
3. What are the special properties of a constructor function?
4. Define the default constructor.
5. What is a destructor?
6. What is parameterized constructor?
7. Explain the importance of destructors?
8. What is copy constructor?
9. Define a class Rectangle which has a length and a breadth. Define the constructors and the destructor and member functions to get the length and the breadth. Write a global function which creates an instance of the class Rectangle and computes the area using the member functions.
10. What are the special characteristics of constructor functions?

Answers for Self Assessment Questions

1. (d) 2. (c) 3. (b) 4. (a) 5. (b) 6. (a)

8.13 Further Studies

1. http://www.thegeekstuff.com/2013/04/cpp-constructors-destructors/
2. The C++ Programming Language 4 Edition by Bjarne Stroustrup

9

Function and Operator Overloading

STRUCTURE

9.0 Objectives

9.1 Introduction

9.2 Function Overloading

9.3 Overloading Constructor Function

9.4 Finding the Address of an Overloaded Function

9.5 Operator Overloading

9.6 Creating a Member Operator Function

9.7 Operator overloading using Friend Function

9.8 Overloading New and Delete Operator

9.9 Overloading Some Special Operators

9.10 Concatenating String

9.11 Overloading Comma (,) Operator

9.12 Overloading the I/O Operators

9.13 Summary

9.14 Keywords

9.15 Review Questions

9.16 Further Studies

9.0 Objectives

After studying this chapter, you will be able to:

- Describe the function overloading
- Explain the overloading constructor function
- Explain the finding address of overloaded function
- Describe the operator overloading

9.1 Introduction

Function overloading is one of the most powerful features of C++ programming language. It forms the basis of polymorphism (compile-time polymorphism). Most of the time you will be overloading the constructor function of a class.
Function overloading is a feature of C++ that allows us to create multiple functions with the same name, so long as they have different parameters.
C++ permits the use of two functions with the same name. However such functions essentially have different argument list. The difference can be in terms of number or type of arguments or both. This process of using two or more functions with the same name but differing in the signature is called function overloading. But overloading of functions with different return types is not allowed.

9.2 Function Overloading

Function overloading is the process of using the same name for two or more functions. The secret to overloading is that each redefinition of the function must use either different types of parameters or a different number of parameters. It is only through these differences that the compiler knows which function to call in any given situation. For example, this program overloads myfunc() by using different types of parameters.

```
#include <iostream>
using namespace std;
int myfunc(int i); // these differ in types of parameters
double myfunc(double i);
int main()
{
cout << myfunc(10) << " "; // calls myfunc(int i)
cout << myfunc(5.4); // calls myfunc(double i)
return 0;
}
double myfunc(double i)
{
return i;
}
int myfunc(int i)
{
return i;
}
```

The next program overloads myfunc() using a different number of parameters:

```
#include <iostream>
using namespace std;
int myfunc(int i); // these differ in number of parameters
int myfunc(int i, int j);
int main()
{
cout << myfunc(10) << " "; // calls myfunc(int i)
cout << myfunc(4, 5); // calls myfunc(int i, int j)
return 0;
```

```
}
int myfunc(int i)
{
return i;
}
int myfunc(int i, int j)
{
return i*j;
}
```

9.3 Overloading Constructor Function

Constructor functions can be overloaded; in fact, overloaded constructors are very common. There are three main reasons why you will want to overload a constructor function: to gain flexibility, to allow both initialized and uninitialized objects to be created, and to define copy constructors.

9.3.1 Overloading a Constructor to Gain Flexibility

Many times will create classes for which there are two or more possible ways to construct an object. In these cases, you will want to provide an overloaded constructor function for each way. This is a self-enforcing rule because if you attempt to create an object for which there is no matching constructor, a compile-time error results.

By providing a constructor for each way that a user of class may plausibly want to construct an object, you increase the flexibility of class. The user is free to choose the best way to construct an object given the specific circumstance. Consider this program that creates a class called date, which holds a calendar date. Notice that the constructor is overloaded two ways:

```
#include <iostream>
#include <cstdio>
using namespace std;
class date {
int day, month, year;
public:
date(char *d);
date(int m, int d, int y);
void show_date();
};
// Initialize using string.
date::date(char *d)
{
sscanf(d, "%d%*c%d%*c%d", &month, &day, &year);
}
// Initialize using integers.
date::date(int m, int d, int y)
{
day = d;
month = m;
year = y;
```

```
}
void date::show_date()
{
cout << month << "/" << day;
cout << "/" << year << "\n";
}
int main()
{
date ob1(12, 4, 2001), ob2("10/22/2001");
ob1.show_date();
ob2.show_date();
return 0;
}
```

In this program, you can initialize an object of type date, either by specifying the date using three integers to represent the month, day, and year, or by using a string that contains the date in this general form:
mm/dd/yyyy

9.3.2 Allowing both Initialized and Uninitialized Objects

Another common reason constructor functions are overloaded is to allow both initialized and uninitialized objects (or, more precisely, default initialized objects) to be created. This is especially important if you want to be able to create dynamic arrays of objects of some class, since it is not possible to initialize a dynamically allocated array. To allow uninitialized arrays of objects along with initialized objects, you must include a constructor that supports initialization and one that does not.

For example, the following program declares two arrays of type powers; one is initialized and the other is not. It also dynamically allocates an array.

```
#include <iostream>
#include <new>
using namespace std;
class powers {
int x;
public:
// overload constructor two ways
powers() { x = 0; } // no initializer
powers(int n) { x = n; } // initializer
int getx() { return x; }
void setx(int i) { x = i; }
};
int main()
{
powers ofTwo[] = {1, 2, 4, 8, 16}; // initialized
powers ofThree[5]; // uninitialized
powers *p;
int i;
// show powers of two
```

```
cout << “Powers of two: ”;
for(i=0; i<5; i++) {
cout << ofTwo[i].getx() << “ ”;
}
cout << “\n\n”;
// set powers of three
ofThree[0].setx(1);
ofThree[1].setx(3);
ofThree[2].setx(9);
ofThree[3].setx(27);
ofThree[4].setx(81);
// show powers of three
cout << “Powers of three: ”;
for(i=0; i<5; i++) {
cout << ofThree[i].getx() << “ ”;
}
cout << “\n\n”;
// dynamically allocate an array
try {
p = new powers[5]; // no initialization
} catch (bad_alloc xa) {
cout << “Allocation Failure\n”;
return 1;
}
// initialize dynamic array with powers of two
for(i=0; i<5; i++) {
p[i].setx(ofTwo[i].getx());
}
// show powers of two
cout << “Powers of two: ”;
for(i=0; i<5; i++) {
cout << p[i].getx() << “ ”;
}
cout << “\n\n”;
delete [] p;
return 0;
}
```

In this example, both constructors are necessary. The default constructor is used to construct the uninitialized ofThree array and the dynamically allocated array. The parameterized constructor is called to create the objects for the ofTwo array.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Write a program for copy constructor overloading.

……..……

……..……

……...

9.3.3 Copy Constructors

The copy constructor lets you create a new object from an existing one by initialization. A copy constructor of a class A is a non-template constructor in which the first parameter is of type A&, const A&, volatile A&, or const volatile A&, and the rest of its parameters (if there are any) have default values.

If you do not declare a copy constructor for a class A, the compiler will implicitly declare one for you, which will be an inline public member.

The following example demonstrates implicitly defined and user-defined copy constructors:

```
#include <iostream>
using namespace std;
struct A {
  int i;
  A() : i(10) { }
};
struct B {
  int j;
  B() : j(20) {
    cout << "Constructor B(), j = " << j << endl;
  }
  B(B& arg) : j(arg.j) {
    cout << "Copy constructor B(B&), j = " << j << endl;
  }
  B(const B&, int val = 30) : j(val) {
    cout << "Copy constructor B(const B&, int), j = " << j << endl;
  }
};
struct C {
  C() { }
  C(C&) { }
};
int main() {
  A a;
  A a1(a);
  B b;
  const B b_const;
  B b1(b);
  B b2(b_const);
  const C c_const;
//  C c1(c_const);
}
```

The following is the output of the above example:
Constructor B(), j = 20
Constructor B(), j = 20
Copy constructor B(B&), j = 20
Copy constructor B(const B&, int), j = 30
The statement A a1(a) creates a new object from a with an implicitly defined copy constructor. The statement B b1(b) creates a new object from b with the user-defined copy constructor B::B(B&). The statement B b2(b_const) creates a new object with the copy constructor B::B(const B&, int). The compiler would not allow the statement C c1(c_const) because a copy constructor that takes as its first parameter an object of type const C& has not been defined.

Exercise: Check Your Progress 2
Note: i) Use the space below for your answer.
Ex1: Write a program for function overloading using with different types of parameters.
……..……

……..……

……...

9.4 Finding the Address of an Overloaded Function

If you use an overloaded function name f without any arguments, that name can refer to a function, a pointer to a function, a pointer to member function, or a specialization of a function template. Because you did not provide any arguments, the compiler cannot perform overload resolution the same way it would for a function call or for the use of an operator. Instead, the compiler will try to choose the best viable function that matches the type of one of the following expressions, depending on where you have used f:
An object or reference you are initializing
The left side of an assignment
A parameter of a function or a user-defined operator
The return value of a function, operator, or conversion
An explicit type conversion
If the compiler chose a declaration of a nonmember function or a static member function when you used f, the compiler matched the declaration with an expression of type pointer-to-function or reference-to-function. If the compiler chose a declaration of a nonstatic member function, the compiler matched that declaration with an expression of type pointer-to-member function. The following example demonstrates this:

```
struct X {
  int f(int) { return 0; }
  static int f(char) { return 0; }
};
int main() {
  int (X::*a)(int) = &X::f;
//  int (*b)(int) = &X::f;
}
```

The compiler will not allow the initialization of the function pointer b. No nonmember function or static function of type int(int) has been declared.
If f is a template function, the compiler will perform template argument deduction to determine which template function to use. If successful, it will add that function to the list of viable functions. If there is more than one function in this set, including a non-template function, the compiler will eliminate all template functions from the set and choose the non-template function. If there are only template functions in this set, the compiler will choose the most specialized template function. The following example demonstrates this:

```
template<class T> int f(T) { return 0; }
template<> int f(int) { return 0; }
int f(int) { return 0; }
int main() {
  int (*a)(int) = f;
  a(1);
}
```

The function call a(1) calls int f(int).

9.5 Operator Overloading

In C++ the overloading principle applies not only to functions, but to operators too. That is, of operators can be extended to work not just with built-in types but also classes. A programmer can provide operator to a class by overloading the built-in operator to perform some specific computation when the operator is used on objects of that class. Is operator overloading really useful in real world implementations? It certainly can be, making it very easy to write code that feels natural.
An Example of Operator Overloading

```
Complex a(1.2,1.3);     //this class is used to represent complex numbers
Complex b(2.1,3);       //notice the construction taking to parameters for the real and imaginary part
Complex c = a+b;        //for this to work the addition operator must be overloaded
```

The addition without having overloaded operator + could look like this:

```
Complex c = a.Add(b);
```

This piece of code is not as readable. We are dealing with numbers, so doing addition should be natural.
In order to allow operations like Complex c = a+b, in above code we overload the “+” operator. The overloading syntax is quite simple, similar to function overloading, the keyword operator must be followed by the operator we want to overload:

```
class Complex

{
public:
      Complex(double re,double im)
            :real(re),imag(im)
            {};
      Complex operator+(const Complex& other);
      Complex operator=(const Complex& other);
private:
      double real;
```

```
    double imag;
};
Complex Complex::operator+(const Complex&  other)
{
   double result_real = real + other.real;
   double result_imaginary = imag + other.imag;
   return Complex( result_real, result_imaginary );
}
```

The assignment operator can be overloaded similarly. Notice that we did not have to call any accessor functions in order to get the real and imaginary parts from the parameter other since the overloaded operator is a member of the class and has full access to all private data. Alternatively, we could have defined the addition operator globally and called a member to do the actual work. In that case, we did also have to make the method a friend of the class, or use an accessor method to get at the private data:

```
friend Complex operator+(Complex);

Complex operator+(const Complex  &num1, const Complex &num2)
{
   double result_real = num1.real + num2.real;
   double result_imaginary = num1.imag + num2.imag;
   return Complex( result_real, result_imaginary );
}
```

Why would you do this? when the operator is a class member, the first object in the expression must be of that particular type:

```
Complex a( 1, 2 );
Complex a( 2, 2 );
Complex c = a.operator=( b );
```

When it is a global function, the implicit or user-defined conversion can allow the operator to act even if the first operand is not exactly of the same type:

```
Complex c = 2+b;          //if the integer 2 can be converted by the Complex class, this expression is
```

valid by the way, the number of operands to a function is fixed; that is, a binary operator takes two operands, a unary only one, and you cannot change it. The same is true for the precedence of operators too; for example the multiplication operator is called before addition. There are some operators that need the first operand to be assignable, such as: operator=, operator(), operator[] and operator->, so their use is restricted just as member functions(non-static), they cannot be overloaded globally. The operator=, operator& and operator, (sequencing) have already defined meanings by default for all objects, but their meanings can be changed by overloading or erased by making them private.

```
string prefix(“de”);
string word(“composed”);
string composed = prefix+word;
```

Caution
Operator overloading should only be utilized when the meaning of the overloaded operator's operation is unambiguous.

Did You Know?
C++ provides more than 35 operators, covering basic arithmetic, bit manipulation, indirection, comparisons, logical operations and others.

9.6 Creating a Member Operator Function

A member operator function takes this general form:
ret-type class-name::operator#(arg-list)
{
// operations
}
Often, operator functions return an object of the class they operate on, but ret-type can be any valid type. The # is a placeholder. When you create an operator function, substitute the operator for the #. For example, if you are overloading the/operator, use operator/. When you are overloading a unary operator, arg-list will be empty. When you are overloading binary operators, arg-list will contain one parameter. (The reasons for this seemingly unusual situation will be made clear in a moment.)

9.6.1 Creating Prefix and Postfix Forms of the Increment (++) and Decrement (– –) operators (Overloading Unary Operator)

In this program, only the prefix form of the increment operator was overloaded. However, Standard C++ allows you to explicitly create separate prefix and postfix versions of increment or decrement operators. To accomplish this, you must define two versions of the operator++() function. One is defined as shown in the foregoing program. The other is declared like this:
loc operator++(int x);
If the ++ precedes its operand, the operator++() function is called. If the ++ follows it is operand, the operator++(int x) is called and x has the value zero.
This example can be generalized. Here are the general forms for the prefix and postfix ++ and – – operator functions.

```
//Prefix increment
type operator++( ) {
//body of prefix operator
}
//Postfix increment
type operator++(int x) {
//body of postfix operator
}
//Prefix decrement
type operator– –( ) {
//body of prefix operator
}
//Postfix decrement
type operator– –(int x) {
```

```
//body of postfix operator
}
```

9.6.2 Overloading the Shorthand Operators (i.e. +=, == etc)

You can overload any of C++'s "shorthand" operators, such as +=, –=, and the like. For example, this function overloads += relative to loc:

```
loc loc::operator+=(loc op2)
{
longitude = op2.longitude + longitude;
latitude = op2.latitude + latitude;
return *this;
}
```

When overloading one of these operators, keep in mind that you are simply combining an assignment with another type of operation.

9.6.3 Operator Overloading Restriction (Rules)

There are some restrictions that apply to operator overloading. You cannot alter the precedence of an operator. You cannot change the number of operands that an operator takes. (You can choose to ignore an operand, however.) Except for the function call operator, operator functions cannot have default arguments. Finally, these operators cannot be overloaded:

. : : .* ?

When someone reading a program sees a statement like Ob1+Ob2, Then expects something resembling addition to be taking Place-not a disk access, for example. Therefore, before decoupling an overloaded operator from its normal meaning be sure that you have sufficient reason to do so. One good example where decoupling is successful is found in the way C++ overloads the << and >> operators for I/O. Although the I/O operations have no relationship to bit shifting, these operators provide a visual "clue" as to their meaning relative to both I/O and bit shifting, and this decoupling works. In general, however, it is best to stay within the context of the expected meaning of an operator when overloading it.

Except for the = operator, operator functions are inherited by any derived class. However, a derived class is free to overload any operator (including those overloaded y the base class) it chooses relative to itself.

Caution

Be aware when working with older C++ versions where the increment and decrement operators are concerned. In older versions of C++, it was not possible to specify separate prefix and postfix versions of an overloaded ++ or – –. The prefix form was used for both.

9.7 Operator Overloading using Friend Function

You can overload an operator for a class by using a non member function, which is usually a friend of the class. Since a friend function is not a member of the class, it does not have this pointer. Therefore, an overloaded friend operator function is passed the operands explicitly. This means that a friend function that overloads a binary operator has two parameters, and a friend function that overloads a unary operator has one parameter. When overloading a binary operator using a friend

function, the left operand is passed in the first parameter and the right operand is passed in the second parameter.
In this program, the operator+() function is made into a friend:

```
#include <iostream>
using namespace std;
class loc {
int longitude, latitude;
public:
loc() {} // needed to construct temporaries
loc(int lg, int lt) {
longitude = lg;
latitude = lt;
}
void show() {
cout << longitude << " ";
cout << latitude << "\n";
}
friend loc operator+(loc op1, loc op2); // now a friend
loc operator-(loc op2);
loc operator=(loc op2);
loc operator++();
};
// Now, + is overloaded using friend function.
loc operator+(loc op1, loc op2)
{
loc temp;
temp.longitude = op1.longitude + op2.longitude;
temp.latitude = op1.latitude + op2.latitude;
return temp;
}
// Overload - for loc.
loc loc::operator-(loc op2)
{
loc temp;
// notice order of operands
temp.longitude = longitude - op2.longitude;
temp.latitude = latitude - op2.latitude;
return temp;
}
// Overload assignment for loc.
loc loc::operator=(loc op2)
{
longitude = op2.longitude;
latitude = op2.latitude;
return *this; // i.e., return object that generated call
}
```

```
// Overload ++ for loc.
loc loc::operator++()
{
longitude++;
latitude++;
return *this;
}
int main()
{
loc ob1(10, 20), ob2( 5, 30);
ob1 = ob1 + ob2;
ob1.show();
return 0;
}
```

There are some restrictions that apply to friend operator functions. First, you may not overload the =, ( ), [ ], or –> operators by using a friend function. when overloading the increment or decrement operators, you will need to use a reference parameter when using a friend function.

Did You Know?

Ada supports overloading of operators from its inception, with the publication of the Ada 83 language standard.

Self Assessment Questions

1. C++ permits the use of two functions with the.....name.

(a) different (b) same

(c) Both (a) and (b) (d) None of these

2. The default constructor is used to construct the uninitialized of.....array and the dynamically allocated array.

(a) zero (b) one

(c). three (c) multi

3. The.....value is specified in a manner syntactically similar to a variable initialization.

(a) default (b) non default

(c) Both (a) and (b) (d) None of these

4. The overloading syntax is quite .....similar to function overloading.

(a) simple (b) difficult

(c) complex (d) None of these

5. All of the arithmetic operators are binary operators.

(a).True (b).False

9.8 Overloading New and Delete Operator

It is possible to overload new and delete operator. You might choose to do this if you want to use some special allocation method. For example, you may want allocation routines that automatically

begin using a disk file as virtual memory when the heap has been exhausted. Whatever the reason, it is a very simple matter to overload these operators.
The skeletons for the functions that overload new and delete are shown here:

```
//Allocate an object.
void *operator new(size_t size)
{
/*Perform allocation. Throw bad_alloc on failure.
Constructor called automatically. */
return pointer_to_memory;
}
//Delete an object.
void operator delete(void *p)
{
/*Free memory pointed to by p.
Destructor called automatically. */
}
```

The type size_t is a defined type capable of containing the largest single piece of memory that can be allocated. (size_t is essentially an unsigned integer.) The parameter size will contain the number of bytes needed to hold the object being allocated. This is the amount of memory that version of new must allocate. The overloaded new function must return a pointer to the memory that it allocates, or throw a bad_alloc exception if an allocation error occurs. Beyond these constraints, the overloaded new function can do anything else you require. When you allocate a
object using new the object's constructor is automatically called.
The delete function receives a pointer to the region of memory to be freed. It then releases the allocated memory back to the system. When an object is deleted, its destructor function is automatically called.
The new and delete operators may be overloaded globally so that all uses of these operators call custom versions. They may also be overloaded relative to one or more classes. Let us begin with an example of overloading new and delete relative to a class. For the sake of simplicity, no new allocation scheme will be used. Instead, the overloaded operators will simply invoke the standard library functions malloc() and free().
To overload the new and delete operators for a class, simply make the overloaded operator functions class members. For example, here the new and delete operators are overloaded for the loc class:

```
#include <iostream>
#include <cstdlib>
#include <new>
using namespace std;
class loc {
int longitude, latitude;
public:
loc() {}
loc(int lg, int lt) {
longitude = lg;
latitude = lt;
}
```

```
void show() {
cout << longitude << “ ”;
cout << latitude << “\n”;
}
void *operator new(size_t size);
void operator delete(void *p);
};
// new overloaded relative to loc.
void *loc::operator new(size_t size)
{
void *p;
cout << “In overloaded new.\n”;
p = malloc(size);
if(!p) {
bad_alloc ba;
throw ba;
}
return p;
}
// delete overloaded relative to loc.
void loc::operator delete(void *p)
{
cout << “In overloaded delete.\n”;
free(p);
}
int main()
{
loc *p1, *p2;
try {
p1 = new loc (10, 20);
} catch (bad_alloc xa) {
cout << “Allocation error for p1.\n”;
return 1;
}
try {
p2 = new loc (-10, -20);
} catch (bad_alloc xa) {
cout << “Allocation error for p2.\n”;
return 1;;
}
p1->show();
p2->show();
delete p1;
delete p2;
return 0;
}
```

Output from this program is shown here.

```
In overloaded new.
In overloaded new.
10 20
-10 -20
In overloaded delete.
In overloaded delete.
```

9.9 Overloading Some Special Operators

C++ defines array subscripting, function calling, and class member access as operations. The operators that perform these functions are the [ ], ( ), and –>, respectively. These rather exotic operators may be overloaded in C++, opening up some very interesting uses.

One important restriction applies to overloading these operators: They must be non static member functions. They cannot be friends.

9.9.1 Overloading [ ] (Subscripting) Operator

The [ ] is considered a binary operator when you are overloading it. Therefore, the general form of a member operator [ ] () function is as shown here:

```
type class-name::operator[](int i)
{
// . . .
}
```

Technically, the parameter does not have to be of type int, but an operator[ ]() function is typically used to provide array subscripting, and as such, an integer value is generally used.

Given an object called O, the expression

```
O[3]
```

translates into this call to the operator[ ]() function:

```
O.operator[](3)
```

That is, the value of the expression within the subscripting operators is passed to the operator[ ]() function in its explicit parameter. This pointer will point to O, the object that generated the call.

In the following program, atype declares an array of three integers. It is constructor function initializes each member of the array to the specified values. The overloaded operator[ ]() function returns the value of the array as indexed by the value of its parameter.

```
#include <iostream>
using namespace std;
class atype {
int a[3];
public:
atype(int i, int j, int k) {
a[0] = i;
a[1] = j;
a[2] = k;
}
int operator[](int i) { return a[i]; }
```

```
};
int main()
{
atype ob(1, 2, 3);
cout << ob[1]; // displays 2
return 0;
}
```

9.9.2 Overloading() (Function Call) Operator

When you overload the ( ) function call operator, Creating a new way to call a function. Rather, you are creating an operator function that can be passed an arbitrary number of parameters. Let us begin with an example. Given the overloaded operator function declaration

double operator()(int a, float f, char *s); and an object O of its class, then the statement O(10, 23.34, “hi”); translates into this call to the operator() function. O.operator()(10, 23.34, “hi”);

In general, when you overload the ( ) operator, you define the parameters that you want to pass to that function. When you use the ( ) operator in program, the arguments you specify are copied to those parameters. As always, the object that generates the call (O in this example) is pointed to by this pointer.

Here is an example of overloading ( ) for the loc class. It assigns the value of its two arguments to the longitude and latitude of the object to which it is applied.

```
#include <iostream>
using namespace std;
class loc {
int longitude, latitude;
public:
loc() {}
loc(int lg, int lt) {
longitude = lg;
latitude = lt;
}
void show() {
cout << longitude << “ ”;
cout << latitude << “\n”;
}
loc operator+(loc op2);
loc operator()(int i, int j);
};
// Overload ( ) for loc.
loc loc::operator()(int i, int j)
{
longitude = i;
latitude = j;
return *this;
}
// Overload + for loc.
loc loc::operator+(loc op2)
```

```
{
loc temp;
temp.longitude = op2.longitude + longitude;
temp.latitude = op2.latitude + latitude;
return temp;
}
int main()
{
loc ob1(10, 20), ob2(1, 1);
ob1.show();
ob1(7, 8); // can be executed by itself
ob1.show();
ob1 = ob2 + ob1(10, 10); // can be used in expressions
ob1.show();
return 0;
}
```

The output produced by the program is shown here.

```
10 20
7 8
11 11
```

When overloading (), you can use any type of parameters and return any type of value.

9.9.3 Overloading –>

The –> pointer operator, also called the class member access operator, is considered a unary operator when overloading. It is general usage is shown here:

```
object->element;
```

Here, object is the object that activates the call. The operator–>() function must return a pointer to an object of the class that operator–>() operates upon. The element must be some member accessible within the object.

The following program illustrates overloading the –> by showing the equivalence between ob.i and ob–>i when operator–>() returns the this pointer:

```
#include <iostream>
using namespace std;
class myclass {
public:
int i;
myclass *operator->() {return this;}
};
int main()
{
myclass ob;
ob->i = 10; // same as ob.i
cout << ob.i << " " << ob->i;
return 0;
}
```

An operator–>() function must be a member of the class upon which it works.

9.9.4 Overloading Binary Arithmetic Operators

Some of the most commonly used operators in C++ are the arithmetic operators - that is, the plus operator (+), minus operator (–), multiplication operator (*), and division operator (/). Note that all of the arithmetic operators are binary operators - meaning they take two operands - one on each side of the operator. All four of these operators are overloaded in the exact same way.

The following example shows how to overload operator plus (+) in order to add two “Cents” objects together:

```
class Cents
{
private:
   int m_nCents;
public:
   Cents(int nCents) { m_nCents = nCents; }
   // Add Cents + Cents
   friend Cents operator+(const Cents &c1, const Cents &c2);

   int GetCents() { return m_nCents; }
};
 // note: this function is not a member function!
Cents operator+(const Cents &c1, const Cents &c2)
{
   // use the Cents constructor and operator+(int, int)
   return Cents(c1.m_nCents + c2.m_nCents);
}
int main()
{
   Cents cCents1(6);
   Cents cCents2(8);
   Cents cCentsSum = cCents1 + cCents2;
   std::cout << “I have ” << cCentsSum .GetCents() << “ cents ” << std::endl;

   return 0;
}
```

This produces the result:

I have 14 cents.

9.10 Concatenating String

String concatenation consists of adding one string to another. To perform this operation, you can use the addition operator. Here is an example:

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
void main()
{
```

```
clrscr();
char str1[30],str2[30],str3[60];
int i,j;
cout<< “Enter first string: ”;
gets(str1);
cout<< “\nEnter second string: ”;
gets(str2);
for(i=0;str1[i]!='\0';++i)
str3[i]=str1[i];
for(j=0;str2[j]!='\0';++j)
str3[i+j]=str2[j];
str3[i+j]='\0';
cout<< “\nThe concatenate string is ”<<str3;
getch();
}
```

9.11 Overloading Comma (,) Operator

You can overload C++’s comma operator. The comma is a binary operator, and like all overloaded operators, you can make an overloaded comma perform any operation you want. However, if you want the overloaded comma to perform in a fashion similar to it is normal operation, and then version must discard the values of all operands except the rightmost. The rightmost value becomes the result of the comma operation. This is the way the comma works by default in C++.

Here is a program that illustrates the effect of overloading the comma operator.

```
#include <iostream>
using namespace std;
class loc {
int longitude, latitude;
public:
loc() {}
loc(int lg, int lt) {
longitude = lg;
latitude = lt;
}
void show() {
cout << longitude << “ ”;
cout << latitude << “\n”;
}
loc operator+(loc op2);
loc operator,(loc op2);
};
// overload comma for loc
loc loc::operator,(loc op2)
{
loc temp;
temp.longitude = op2.longitude;
```

```
temp.latitude = op2.latitude;
cout << op2.longitude << " " << op2.latitude << "\n";
return temp;
}
// Overload + for loc
loc loc::operator+(loc op2)
{
loc temp;
temp.longitude = op2.longitude + longitude;
temp.latitude = op2.latitude + latitude;
return temp;
}
int main()
{
loc ob1(10, 20), ob2( 5, 30), ob3(1, 1);
ob1.show();
ob2.show();
ob3.show();
cout << "\n";
ob1 = (ob1, ob2+ob2, ob3);
ob1.show(); // displays 1 1, the value of ob3
return 0;
}
```

This program displays the following output:

```
10 20
5 30
1 1
10 60
1 1
1 1
```

9.12 Overloading the I/O Operators

C++ is able to input and output the built-in data types using the stream extraction operator >> and the stream insertion operator <<. The stream insertion and stream extraction operators also can be overloaded to perform input and output for user-defined types like an object.

Here it is important to make operator overloading function a friend of the class because it would be called without creating an object.

Following example how extraction operator >> and insertion operator <<.

```
#include <iostream>
using namespace std;
class Distance
{
  private:
    int feet;             // 0 to infinite
    int inches;           // 0 to 12
```

```
  public:
    // required constructors
    Distance(){
      feet = 0;
      inches = 0;
    }
    Distance(int f, int i){
      feet = f;
      inches = i;
    }
    friend ostream &operator<<( ostream &output,
                                const Distance &D )
    {
      output << "F : " << D.feet << " I : " << D.inches;
      return output;
    }
    friend istream &operator>>( istream  &input, Distance &D )
    {
      input >> D.feet >> D.inches;
      return input;
    }
};
int main()
{
  Distance D1(11, 10), D2(5, 11), D3;
  cout << "Enter the value of object : " << endl;
  cin >> D3;
  cout << "First Distance : " << D1 << endl;
  cout << "Second Distance : " << D2 << endl;
  cout << "Third Distance :" << D3 << endl;
  return 0;
}
```

Output:

```
Enter the value of object :
70
10
First Distance : F : 11 I : 10
Second Distance :F : 5 I : 11
Third Distance :F : 70 I : 10
```

9.13 Summary

- Function overloading is a feature of C++ that allows us to create multiple functions with the same name.
- The default constructor is used to construct the uninitialized of three arrays and the dynamically allocated array. The parameterized constructor is called to create the objects of two arrays.
- Overloading the plus operator (+) is as simple as declaring a function named operator+, giving it

two parameters of the type of the operands.

- C++ is able to input and output the built-in data types using the stream extraction operator >> and the stream insertion operator <<.
- The –> pointer operator is called the class member access operator, and considered a unary operator when overloading.

9.14 Keywords

Comma Operator: It is a binary operator. It use to separate the arguments.
Friend Function: It is used for accessing the non-public members of a class
Function Overloading: It is the process of using the same name for two or more functions.
Parameterized Constructor: It is containing parameter.
String Concatenation: It is consists of adding one string to another.
String Objects: It is refer to special type of container, specifically designed to operate with sequences of characters.

9.15 Review Questions

1. What is the function overloading?
2. What is the operator overloading?
3. Explain an overloading constructor function.
4. What are the default function arguments?
5. What is the difference between function and operator overloading?
6. Describe the creating prefix and postfix forms of the increment (++) and decrement (– –) operators.
7. Explain a new and delete operator in overloading.
8. Which function is use to concatenating strings? Explain with example.
9. Describe the overloading of I/O operators.
10. What is the difference between subscripting and comma operator?

Answers for Self Assessment Questions

1. (b) 2 (c) 3 (a) 4 (a) 5 (a)

9.16 Further Studies

1. http://stackoverflow.com/questions/4421706/operator-overloading
2. The C++ Programming Language 4 Edition by Bjarne Stroustrup

10

Inheritance

STRUCTURE

10.0 Objectives

10.1 Introduction

10.2 Features/Advantages of Inheritance

10.3 Type of Inheritance

10.4 Based Classes and Derived Classes

10.5 Inherited Member Access

10.6 Inheriting Multiple Base Classes

10.7 Constructors, Destructors and Inheritance

10.8 Passing Parameters to Base Class Constructors

10.9 Granting Access

10.10 Virtual Base Classes

10.11 Summary

10.12 Keywords

10.13 Review Questions

10.14 Further Studies

10.0 Objectives

After studying this chapter, you will be able to:

- Describe the features or advantages of inheritance
- Explain the type of inheritance
- Describe the based classes and derived classes
- Explain the inheriting multiple base classes
- Describe the constructors, destructors and inheritance

10.1 Introduction

In the lesson on composition, you learned how to construct complex classes by combining simpler classes. Composition is perfect for building new objects that have a has-a relationship with their

subobjects. However, composition (and aggregation) is just one of the two major ways that C++ lets you construct complex classes. The second way is through inheritance.
Unlike composition, which involves creating new objects by combining and connecting other objects, inheritance involves creating new objects by directly acquiring the attributes and behaviors of other objects and then extending or specializing them. Like composition, inheritance is everywhere in real life. You inherited your parents genes, and acquired physical attributes from both of them.
Technological products (computers, cell phones, etc…) often inherit features from their predecessors. C++ inherited many features from C, the language upon which it is based, and C itself inherited many of its features from the programming languages that came before it.

10.2 Features/Advantages of Inheritance

Features of Inheritance

Inheritance is a prime feature of object oriented programming language. It is process by which new classes called derived classes(sub classes, extended classes, or child classes) are created from existing classes called base classes(super classes, or parent classes). The derived class inherits all the features (capabilities) of the base class and can add new features specific to the newly created derived class. The base class remains unchanged.
We can say that re-usability is concerned as to how we can use a system or its part in other systems.
Inheritance is a technique of organizing information in a hierarchical form. It is a relation between classes that allows for definition and implementation of one class based on the definition of existing classes.

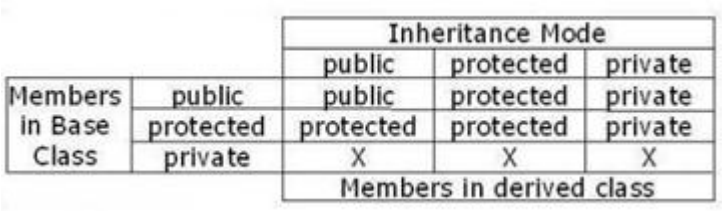
There are three basic advantages of inheritance:

1. Specialization - This is where a new class is derived from a base class and it does something more or slightly different form the base class.
2. Generalization - This is used when two or more classes are similar, and the similarties are factored together into a common base class.
3. Interface - This is used when a base class defines an interface, but doesn't implement some (or all) of it.

In all of these cases, there's one important thing to keep in mind. The derived class IS-A type of the base class. This means that anywhere that the base class works, the derived class can be used as well.

10.3 Type of Inheritance

Inheritance is the concept to inherit the properties of one class to another class. This has also known as class structure again. For example, classes A contains two-member function ads and subtracts and class B contain two another functions multiply and divide. We want to use all these function with one object then we need to use inheritance where class B inherits all the property of class, which is public, but class B cannot use the private properties of class A. There are following types of inheritance:



		Inheritance Mode		
		public	protected	private
Members in Base Class	public	public	protected	private
	protected	protected	protected	private
	private	X	X	X
		Members in derived class		

Single Class Inheritance

When class a gas inherited in class has known as base class and B class is known as derived class. Here only two classes have linked to each other. See the Figure 10.1.

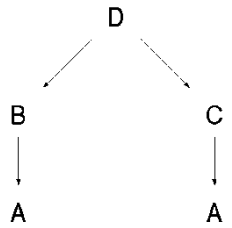


Figure 10.1: Single class inheritance.

Multilevel Inheritance

In this type of inheritance, there are number of level and it has used in that cases where we want to use all properties in number of levels to the requirement. For example, class A inherited in class B and class B has inherited in class C for class B so on. Where class A is base class C. In another way we can say B is derived class a base class for C and A indirect base class for C is indirect base class for C and C indirect derived class for class A (see the Figure 10.2):

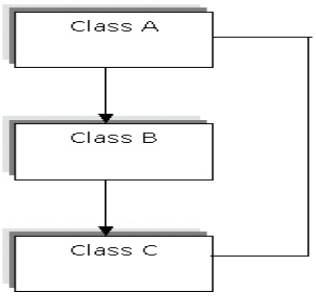


Figure 10.2: Multilevel inheritance.

Multiple Inheritances

In this type of inheritance, number of classes has inherited in a single class. Where two or more classes are, know as base class and one is derive class (see the Figure 10.3):

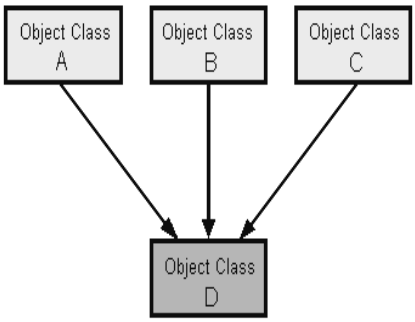


Figure 10.3: Multiple inheritances.

Hierarchical Inheritance

This type of inheritance helps us to create a baseless for number of classes and those numbers of classes can have further their branches of number of class (see the Figure 10.4).

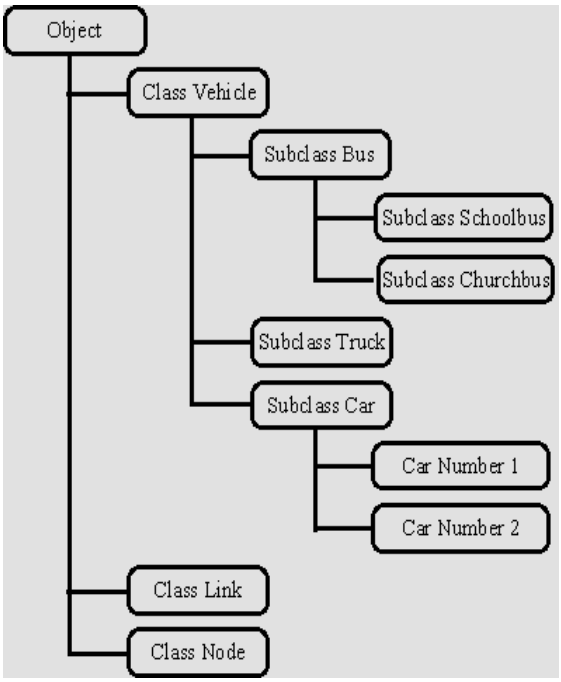


Figure 10.4: Hierarchical inheritance.

Hybrid Inheritance

In this type of inheritance, we can have mixture of number of inheritances but this can generate an error of using same name function from no of classes, which will bother the compiler to how to use the functions. Therefore, it will generate errors in the program. This has known as ambiguity or duplicity (see the Figure 10.5):

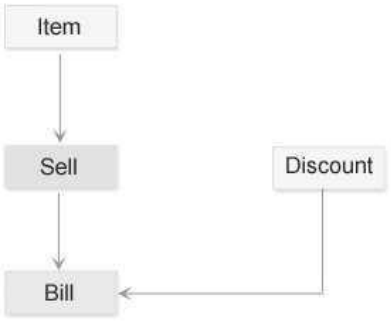


Figure 10.5: Hybrid inheritance.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Create a diagram of inheritance in real life with example.

……..……

……..……

……...

10.4 Based Classes and Derived Classes

Based Class

A class is a mechanism for creating user-defined data types. It is similar to the C language structure data type. In C, a structure is composed of a set of data members. In C++, a class type is like a C structure, except that a class is composed of a set of data members and a set of operations that can be performed on the class.

In C++, a class type can be declared with the keywords union, struct, or class. A union object can hold any one of a set of named members. Structure and class objects hold a complete set of members. Each class type represents a unique set of class members including data members, member functions, and other type names. The default access for members depends on the class key:

The members of a class declared with the keyword class are private by default. A class is inherited privately by default.
The members of a class declared with the keyword struct are public by default. A structure is inherited publicly by default.
The members of a union (declared with the keyword union) are public by default. A union cannot be used as a base class in derivation.
Once you create a class type, you can declare one or more objects of that class type. For example:

```
class X
{
   /* define class members here */
};
int main()
{
    X xobject1;      // create an object of class type X
    X xobject2;      // create another object of class type X
}
```

Derived Class

A derived class is a class that inherits the properties from its super class. For example, a Cat is a super class and Monx cat is a derived class which has all properties of a Cat and does not have a tail.
A concrete derived class is a derived class which implements the all functionality that are missed in the super class.
Explaination Derived class with an example using C++.
Inheritance is one of the important feature of OOP which allows us to make hierarchical classifications of classes. In this, we can create a general class which defines the most common features. Other more specific classes can inherit this class to define those features that are unique to them. In this case, the classes which inherit from other classes, is referred as derived class.

For example, a general class vehicle can be inherited by more specific classes car and bike. The classes car and bike are derived classes in this case.

```
class vehicle
{
```

```
    int fuel_cap;
    public:
        drive();
};

class car : public class vehicle
{
    public:
    roll_windows();
};

class bike : public class vehicle
{
    public:
    kick_start();
};
```

10.5 Inherited Member Access

The access rules affecting a protected non static base class member and how to declare a derived class using an access specifies:

- Protected members
- Access control of base class members

10.5.1 Protected Members

A protected non static base class member can be accessed by members and friends of any classes derived from that base class by using one of the following:

- A pointer to a directly or indirectly derived class
- A reference to a directly or indirectly derived class
- An object of a directly or indirectly derived class

If a class is derived privately from a base class, all protected base class members become private members of the derived class.

If reference a protected non static member x of a base class A in a friend or a member function of a derived class B, It must access x through a pointer to, reference to, or object of a class derived from A. However, if you are accessing x to create a pointer to member, you must qualify x with a nested name specifies that names the derived class B.

The following example demonstrates this:

```
class A {
public:
protected:
  int i;
};
class B : public A {
  friend void f(A*, B*);
  void g(A*);
};
void f(A* pa, B* pb) {
```

```
//  pa->i = 1;
  pb->i = 2;
//  int A::* point_i = &A::i;
  int A::* point_i2 = &B::i;
}
void B::g(A* pa) {
//  pa->i = 1;
  i = 2;
//  int A::* point_i = &A::i;
  int A::* point_i2 = &B::i;
}
void h(A* pa, B* pb) {
//  pa->i = 1;
//  pb->i = 2;
}

int main() { }
```

Class A contains one protected data member, an integer i. Because B derives from A, the members of B have access to the protected member of A. Function f() is a friend of class B:

- The compiler would not allow pa->i = 1 because pa is not a pointer to the derived class B.
- The compiler would not allow int A::* point_i = &A::i because i has not been qualified with the name of the derived class B.

Function g() is a member function of class B. The list of remarks about which statements the compiler would and would not allow apply for g() except for the following:

- The compiler allows i = 2 because it is equivalent to this->i = 2.

Function h() cannot access any of the protected members of A because h() is neither a friend or a member of a derived class of A.

10.5.2 Access Control of Base Class Members

When declare a derived class, an access specifies can precede each base class in the base list of the derived class. This does not alter the access attributes of the individual members of a base class as seen by the base class, but allows the derived class to restrict the access control of the members of a base class.

You can derive classes using any of the three access specifies:

- In a public base class, public and protected members of the base class remain public and protected members of the derived class.
- In a protected base class, public and protected members of the base class are protected members of the derived class.
- In a private base class, public and protected members of the base class become private members of the derived class.

In all cases, private members of the base class remain private. Private members of the base class cannot be used by the derived class unless friend declarations within the base class explicitly grant access to them.

In the following example, class d is derived publicly from class b. Class b is declared a public base class by this declaration.

```
class b { };
```

```
class d : public b // public derivation
{ };
```

We can use both a structure and a class as base classes in the base list of a derived class declaration:

- If the derived class is declared with the keyword class, the default access specifier in its base list specifiers is private.
- If the derived class is declared with the keyword struct, the default access specifier in its base list specifiers is public.

In the following example, private derivation is used by default because no access specifier is used in the base list and the derived class is declared with the keyword class:

```
struct B
{ };
class D : B // private derivation
{ };
```

Members and friends of a class can implicitly convert a pointer to an object of that class to a pointer to either:

- A direct private base class
- A protected base class (either direct or indirect)

Caution

The comma (,) character separates the base class names. And do not forget the public keyword-it should appear in front of every base class name. If the public keyword is omitted from one or more base class names, those base classes become private base classes.

Self Assessment Questions

1. .....is the concept to inherit the properties of one class to another class.

(a) Inheritance (b) Polymorphism
(c) Data hiding (d) None of these.

2. C++ inheritance is very similar to a parent-child relationship.

(a) True (b) False

3. .....are powered devices.

(a) Scanners (b).printers
(c) Both (a) and (b) (d) None of these.

4. Multiple inheritances seem like a simple extension of.....inheritance.

(a) multilevel (b) single
(c) Both (a) and (b) (d) None of these

5. Many object-oriented languages do not even support multiple inheritances like.....

(a) Smalltalk (b) Perl
(c) PHP (d) Both (a).and (c)

10.6 Inheriting Multiple Base Classes

As described in Multiple Inheritance, a class can be derived from more than one base class. In a multiple-inheritance model (where classes are derived from more than one base class), the base

classes are specified using the base-list grammar element. For example, the class declaration for Collection Of Book, derived from Collection and Book, can be specified:

```
// deriv_MultipleBaseClasses.cpp
// compile with: /LD
class Collection {
};
class Book {};
class CollectionOfBook : public Book, public Collection {
   // New members
};
```

The order in which base classes are specified is not significant except in certain cases where constructors and destructors are invoked. In these cases, the order in which base classes are specified affects the following:

The order in which initialization by constructor takes place. If your code relies on the Book portion of CollectionOfBook to be initialized before the Collection part, the order of specification is significant. Initialization takes place in the order the classes are specified in the base-list.
The order in which destructors are invoked to clean up. Again, if a particular "part" of the class must be present when the other part is being destroyed, the order is significant. Destructors are called in the reverse order of the classes specified in the base-list.

When specifying the base-list, you cannot specify the same class name more than once. However, it is possible for a class to be an indirect base to a derived class more than once.

Exercise: Check Your Progress 2
Note: i) Use the space below for your answer.
Ex1: Create a diagram of inheritance in real life with example.

……..……

……..……

……...

10.7 Constructors, Destructors and Inheritance

Constructors

A constructor is a member function with the same name as its class. For example:

```
class X {
public:
  X();      // constructor for class X
};
```

Constructors are used to create, and can initialize, objects of their class type.

You cannot declare a constructor as virtual or static, nor can you declare a constructor as const, volatile, or const volatile.

You do not specify a return type for a constructor. A return statement in the body of a constructor cannot have a return value.

Destructors

Destructors are used to control the behaviour of an object when it passes out of scope or is otherwise to be discarded. If the class is simple then a destructor may not be necessary: the default is simply to discard the store occupied by the data members. However, if pointers are involved, then you must explicitly delete objects pointed to so that memory leakage does not occur.

Inheritance

Inheritance is the process of creating new classes from the existing class or classes.

Using inheritance, one can create general class that defines traits common to a set of related items. This class can then be inherited (reused) by the other classes by using the properties of the existing ones with the addition of its own unique properties.

The old class is referred to as the base class and the new classes, which are inherited from the base class, are called derived classes.

Forms of Inheritance :

Single Inheritance - If a class is derived from a single base class, it is called as single inheritance.

Multiple Inheritance - If a class is derived from more than one base class, it is known as multiple inheritance.

Multilevel Inheritance - The classes can also be derived from the classes that are already derived. This type of inheritance is called multilevel inheritance.

Hierarchical Inheritance - If a number of classes are derived from a single base class, it is called as hierarchical inheritance.

10.8 Passing Parameters to Base Class Constructors

If I have a base class with a constructor that takes '...' parameters, and I want to derive a class from that and give it a similar constructor and have it pass the variable parameters to the base, is there a way to do that?

For example:

```
class Base {
public:
Base (const char *str, ...);
};

class Derived : public Base {
public:
Derived (const char *str, ...);
};
```

I want to have Derived's constructor call Base's constructor, passing all the optional parameters. Right now I am splitting things up into a separate initialization function that takes a va_list, but I'd really like to avoid having to separate it out:

```
class Base {
public:
Base (void);
Base (const char *str, ...); // <-- this would call Init()
```

```
protected:
void Init (va_list args);
};
class Derived : public Base {
public:
Derived (const char *str, ...) {
// call's Base Init with va_list...
va_list args;
va_start(args, str);
Init(args);
va_end(args);
}
};
```

10.9 Granting Access

There are following points inheritance for granting access

- When a base class is inherited as private, all public and protected members of that class become private members of the derived class.
- In certain circumstances, we want to restore one more inherited member to their original access specification. There are two ways to accomplish this:
  - Using namespaces via using statement
  - By employing an access declaration within the derived class using the scope resolution operator ::
- A member declared as private in a base class cannot be made public by a derived class by either of the tasks.
- Access declarations are supported in C++ to accommodate those situations in which most of an inherited class is intended to be made private, but a few members are to retain their public or protected status.
- While standard C++ still supports access declarations, they are deprecated
- The standard suggests achieving the same effect by applying the using keyword

Example:

```
#include <iostream>
using namespace std;

class base {
  int i; // private to base
public:
  int j, k;
  void seti(int x) { i = x; }
  int geti() { return i; }
};

// Inherit base as private.
class derived: private base {
public:
  base::j; // make j public again - but not k
  base::seti; // make seti() public
  base::geti; // make geti() public
```

```
  int a; // public
};
 int main()
{
  derived ob;

  ob.j = 20; // legal because j is made public in derived

  ob.a = 40; // legal because a is public in derived
  ob.seti(10);

  cout << ob.geti() << " " << ob.j << " " << ob.a;

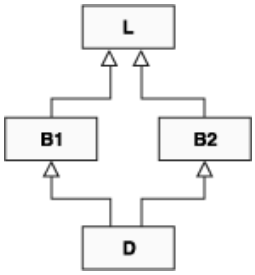
  return 0;
}
```

10.10 Virtual Base Classes

Suppose you have two derived classes B and C that have a common base class A, and you also have another class D that inherits from B and C. You can declare the base class A as virtual to ensure that B and C share the same subobject of A.

In the following example, an object of class D has two distinct subobjects of class L, one through class B1 and another through class B2. You can use the keyword virtual in front of the base class specifiers in the base lists of classes B1 and B2 to indicate that only one subobject of type L, shared by class B1 and class B2, exists.

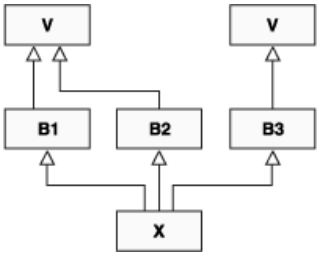
For example:




```
class L { /* ... */ }; // indirect base class
class B1 : virtual public L { /* ... */ };
class B2 : virtual public L { /* ... */ };
class D : public B1, public B2 { /* ... */ }; // valid
```

Using the keyword virtual in this example ensures that an object of class D inherits only one subobject of class L.

A derived class can have both virtual and non virtual base classes. For example:




```
class V { /* ... */ };
class B1 : virtual public V { /* ... */ };
class B2 : virtual public V { /* ... */ };
class B3 : public V { /* ... */ };
class X : public B1, public B2, public B3 { /* ... */
};
```

In the example, class X has two sub objects of class V, one that is shared by classes B1 and B2 and one through class B3.

Self Assessment Questions

6. Destructors show a.....

(a) *sign (b) &sign

(c) @sign (d) sign

7. Constructor and destructors are usually defined as..... members of their class and may never possess a return value.

(a) private (b) public

(c) protected (d) All of these

8. The constructors of any.....base classes are called first in the order of inheritance from the ultimate base class to the lowest virtual class in the inheritance hierarchy.

(a) virtual (b) static

(c) dynamic (d) All of these

9. The purpose of destructor is to release the memory when the compiler memory is reduced or insufficient to execute certain program.

(a) True (b) False

10. The derived class inherit some or all of the properties of the base class.

(a) True (b) False

Exercise: Check Your Progress 3

Case Study-Global Software Company

Challenge

For one leading software company, flexible workforce solutions are the best way to stay ahead of the technology curve in the fast-paced software industry. The firm must continually adjust the size and make-up of its workforce to match priority projects and initiatives. With custom products and U.S.

locations coast to coast, the global firm needed high-level recruiting expertise to find IT professionals with specialized skill sets.

Results

The global software company supplements its workforce with an average of 150 IT and business consultants drawn from Manpower Professional's broad talent pipeline.

Manpower Professional's on site management team generates streamlined, efficient recruiting and hiring in locations across the U.S. The onsite team conducts quarterly reviews aimed at continuous improvement in quality recruitment and client satisfaction. With a solid partnership in place, the firm is well positioned for new growth opportunities.

Manpower Professional

Manpower Professional's seasoned recruiters play a central role in flexible workforce strategy for many technology industry companies. We use a consultative approach and our knowledge of complex IT environments to deliver specialized IT talent at all levels for short- and long-term projects, project solutions and permanent hiring solutions.

Question

1. Describe the challenge Global Software Company.
2. What is the manpower professional in IT sector?

……..

..……

…..……

……..……

……..……

……...

Did You Know?

Virtual base classes allow class designers to specify that a base class will be shared among derived classes.

10.11 Summary

- Inheritance is the process of creating new classes, called derived classes, from existing or base classes.
- Derived class has access to most of the functions and variables of the base class.
- Constructor is a special member function. And it must have identical name as the class. Constructor is the extension of class with additional features and certain limitations.
- The purpose of destructor is to release the memory when the compiler memory is reduced or insufficient to execute certain program.
- The constructor and destructor of a base class are not inherited the assignment operator is not inherited the friend functions and friend classes of the base class are also not inherited.

10.12 Keywords

Derived Class: It is refers to class that inherits some or its entire member from another class.
Multiple Inheritances: It is refers to number of classes has inherited in a single class.
Private Class: It is accessible only to the member and friend function of the class.
Public Class: It is accessible from anywhere where the object is visible.
Reusability: It is refers to the ability for multiple programmers to use the same written and debugged existing class of data**.**

10.13 Review Questions

1. What does inheritance mean in C++?
2. What are the different forms of inheritance? Give an example.
3. What is virtual base class?
4. Describe the advantage of inheritance.
5. What is the difference between single level and multilevel inheritance?
6. Explain constructors, destructors and inheritance.
7. What is the difference between based and derived classes?
8. Explain a granting access.
9. Describe the inherited member access with example.
10. What are the protected members in heritance?

Answers for Self Assessment Questions

1 (a)	2 (a)	3 (c)	4 (b)	5 (d)
6 (d)	7 (b)	8 (a)	9 (a)	10 (a)

10.14 Further Studies

1. http://publib.boulder.ibm.com/infocenter/comphelp/v8v101/index.jsp?topic=%2Fcom.ibm.xlcpp8a.doc%2Flanguage%2Fref%2Finher.htm
2. The C++ Programming Language 4 Edition by Bjarne Stroustrup

11

Polymorphism

STRUCTURE

11.0 Objectives

11.1 Introduction

11.2 Concept and Type of Polymorphism

11.3 Virtual Functions and Polymorphism

11.4 Pure Virtual Functions

11.5 Early versus Late Binding

11.6 Summary

11.7 Keywords

11.8 Review Questions

11.9 Further Studies

11.0 Objectives

After studying this chapter, you will be able to:

- Discuss the concept and type of polymorphism
- Explain the virtual functions and polymorphism
- Discuss the pure virtual functions
- Explain the virtual functions
- Discuss the early versus late binding

11.1 Introduction

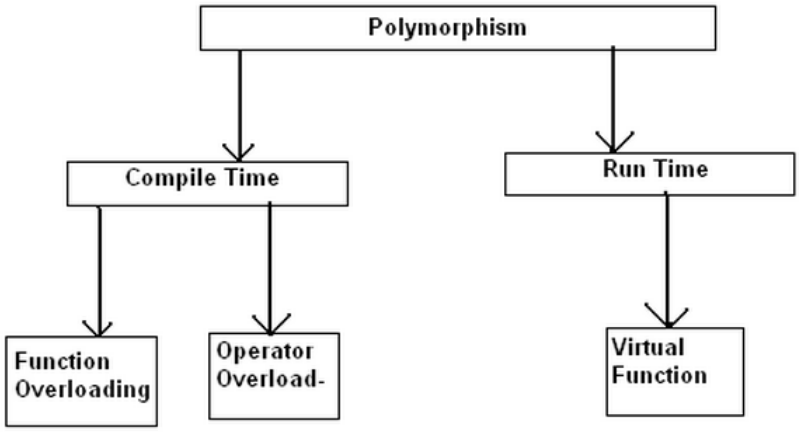
Polymorphism is the ability to use an operator or method in different ways. Polymorphism gives different meanings or functions to the operators or methods. Poly, refer too many, signifies the many uses of these operators and methods. A single method usage or an operator functioning in many ways can be called polymorphism. Polymorphism refers to codes, operations or objects that behave differently in different contexts.

Polymorphism is a powerful feature of the object oriented programming language C++. A single operator + behaves differently in different contexts such as integer, float or strings referring the concept of polymorphism. The concept of overloading is also a branch of polymorphism. When the

exiting operator or function operates on new data type it is overloaded. This feature of polymorphism leads to the concept of virtual methods.

11.2 Concept and Type of Polymorphism

In the word Polymorphism "Poly" means "Many". Polymorphism is one of the features of oops. It simply means one name many forms. The concept of polymorphism is implemented using overloaded function and operator. Polymorphism is a mechanism to use a single name with multiple forms.



There are two types of polymorphism

- Compile Time Polymorphism (Function Overloading, Operator Overloading)
- Run time Polymorphism (Virtual Function)

Compile Time Polymorphism

In this form of polymorphism the selection of the function invocation is done on compile time. This is also called early binding or static binding. This can be achieved by two ways:

- Function Overloading
- Operator Overloading

Function Overloading

Function overloading means multiple functions of same name with different arguments. C++ allows functions to be overloaded, that is the same function to have more than one definition.

Operator Overloading

Operators are similar to functions they take operands and return a value. For Example: the + operator can be used to add two integers, two real's or two addresses.

Runtime Polymorphism

If the member function is selected when the program is running then it called Runtime Polymorphism. This is also called late binding or dynamic binding. Runtime Polymorphism achieves the concept of Virtual Function.

Virtual Function

Virtual function is a member function of a class, whose functionality can be over-ridden in its derived classes. It can be used with virtual keyword. Virtual member functions are resolved during run-time.

This mechanism is known as dynamic binding. The non-virtual member functions are resolved at compile time. This mechanism is called static binding.
When people talk about polymorphism in C++ they usually mean the thing of using a derived class through the base class pointer or reference, which is called subtype polymorphism. But they often forget that there are all kinds of other polymorphisms in C++, such as parametric polymorphism, ad-hoc polymorphism and coercion polymorphism.
These polymorphisms also go by different names in C++,

- Subtype polymorphism is also known as runtime polymorphism.
- Parametric polymorphism is also known as compile-time polymorphism.
- Ad-hoc polymorphism is also known as overloading.
- Coercion is also known as (implicit or explicit) casting.

Here we illustrate all the polymorphisms through examples in C++ language and also give insight on why they have various other names.

Did You Know?
Ad-hoc polymorphism is a kind of polymorphism in which polymorphic functions can be applied to arguments of different types, because a polymorphic function can denote a number of distinct and potentially heterogeneous implementations depending on the type of argument(s) to which it is applied, was developed by Christopher Strachey in 1967.

11.2.1 Subtype Polymorphism (Runtime Polymorphism)

Subtype polymorphism is what everyone understands when they say "polymorphism" in C++. It is the ability to use derived classes through base class pointers and references.
Here is an example. Suppose you have various cats like these felines,



Since they are all of Felidae biological family, and they all should be able to meow, they can be represented as classes inheriting from Felid base class and overriding the meow pure virtual function,

```
// file cats.h
class Felid {
public:
virtual void meow() = 0;
};
class Cat : public Felid {
public:
void meow() { std::cout << "Meowing like a regular cat! meow!\n"; }
};
```

```
class Tiger : public Felid {
public:
 void meow() { std::cout << “Meowing like a tiger! MREOWWW!\n”; }
};
class Ocelot: public Felid {
public:
void meow() {std::cout << “Meowing like an ocelot! mews!\n”; }
};
```

Now the main program can use Cat, Tiger and Ocelot interchangeably through Felid (base class) pointer,

```
#include <iostream>
#include “cats.h”
void do_meowing(Felid *cat) {
 cat->meow();
}
int main() {
Cat cat;
Tiger tiger;
Ocelot ocelot;
do_meowing(&cat);
do_meowing(&tiger);
do_meowing(&ocelot);
}
```

Here the main program passes pointers to cat, tiger and ocelot to do_meowing function that expects a pointer to Felid. Since they are all Felids, the program calls the right meow function for each felid and the output is:
Meowing like a regular cat! meow!
Meowing like a tiger! MREOWWW!
Meowing like an ocelot! mews!

Subtype polymorphism is also called runtime polymorphism for a good reason. The resolution of polymorphic function calls happens at runtime through an indirection via the virtual table. Another way of explaining this is that compiler does not locate the address of the function to be called at compile-time, instead when the program is run; the function is called by dereferencing the right pointer in the virtual table.
In type theory it is also known as inclusion polymorphism.

Caution
When you want to perform the polymorphism than the signature of all the functions must be same as in base and in derived classes. Avoiding it can prevent to implement polymorphism.

11.2.2 Parametric Polymorphism (Compile-Time Polymorphism)

Parametric polymorphism provides a means to execute the same code for any type. In C++ parametric polymorphism is implemented via templates.

One of the simplest examples is a generic max function that finds maximum of two of its arguments,

```
#include <iostream>
#include <string>
template <class T>
T max(T a, T b) {
 return a > b ? a:b;
}
int main() {
 std::cout << ::max(9, 5) << std::endl;     // 9
 std::string foo("foo"), bar("bar");
 std::cout << ::max(foo, bar) << std::endl; // "foo"
}
```

Here the max function is polymorphic on type T. Note, however, that it does not work on pointer types because comparing pointers compares the memory locations and not the contents. To get it working for pointers you would have to specialize the template for pointer types and that would no longer be parametric polymorphism but would be ad-hoc polymorphism.

Since parametric polymorphism happens at compile time, it is also called compile-time polymorphism.

11.2.3 *Ad-hoc* Polymorphism (Overloading)

Ad-hoc polymorphism allows functions with the same name act differently for each type. For example, given two ints and the + operator, it adds them together. Given two std::strings it concatenates them together. This is called overloading.

Here is a concrete example that implements function add for ints and strings,

```
#include <iostream>
#include <string>
int add(int a, int b) {
 return a + b;
}
std::string add(const char *a, const char *b) {
std::string result(a);
result += b;
 return result;
}
int main() {
 std::cout << add(5, 9) << std::endl;
 std::cout << add("hello", "world") << std::endl;
}
```

Ad-hoc polymorphism also appears in C++ if you specialize templates. Returning to the previous example about max function, here is how you would write a max for two char *,

```
template <>
const char *max(const char *a, const char *b) {
 return strcmp(a, b) > 0 ? a : b;
}
```

Now you can call ::max("foo", "bar") to find maximum of strings "foo" and "bar".

11.2.4 Coercion Polymorphism (Casting)

Coercion happens when an object or a primitive is cast into another object type or primitive type. For example,

float b = 6; // int gets promoted (cast) to float implicitly
int a = 9.99 // float gets demoted to int implicitly

Explicit casting happens when you use C's type-casting expressions, such as (unsigned int *) or (int) or C++'s static_cast, const_cast, reinterpret_cast, or dynamic_cast.

Coercion also happens if the constructor of a class is not explicit, for example,

```
#include <iostream>
class A {
int foo;
public:
A(int ffoo) : foo(ffoo) {}
void giggidy() { std::cout << foo << std::endl; }
};
void moo(A a) {
 a.giggidy();
}
int main() {
 moo(55);     // prints 55
}
```

If you made the constructor of A explicit, that would no longer be possible. It is always a good idea to make your constructors explicit to avoid accidental conversions.

Also if a class defines conversion operator for type T, then it can be used anywhere where type T is expected.

For example,

```
class CrazyInt {
int v;
public:
CrazyInt(int i) : v(i) {}
operator int() const { return v; } // conversion from CrazyInt to int
};
```

The CrazyInt defines a conversion operator to type int. Now if we had a function, let us say, print_int that took int as an argument, we could also pass it an object of type CrazyInt,

```
#include <iostream>
void print_int(int a) {
 std::cout << a << std::endl;
}
int main() {
CrazyInt b = 55;
print_int(999);    // prints 999
print_int(b);      // prints 55
}
```

Subtype polymorphism that discussed earlier is actually also coercion polymorphism because the derived class gets converted into base class type.

Caution
Be aware while using runtime polymorphism only method should be overridden except data member.

Did You Know?
Parametric polymorphism was first introduced to programming languages in ML (Meta Language) in 1976.

Self Assessment Questions

1. Polymorphism is one of the features of……
(a) C++ (b) dot net
(c) OOPs (d) None of these.

2. …… is also known as overloading.
(a) Compile time polymorphism (b) Runtime polymorphism
(c) *Ad-hoc* polymorphism (d) Parametric polymorphism

3. *Ad-hoc* polymorphism also appears in …... if you specialize templates.
(a) C++ (b) dot net
(c) OOPs (d) None of these.

11.3 Virtual Functions and Polymorphism

11.3.1 Virtual Function

A virtual function is a member function that is declared within a base class and redefined by a derived class. To create a virtual function, precede the function's declaration in the base class with the keyword virtual. When a class containing a virtual function is inherited, the derived class redefines the virtual function to fit its own needs. In essence, virtual functions implement the "one interface, multiple methods" philosophy that underlies polymorphism. The virtual function within the base class defines the form of the interface to that function. Each redefinition of the virtual function by a derived class implements its operation as it relates specifically to the derived class. That is, the redefinition creates a specific method.

When accessed "normally," virtual functions behave just like any other type of class member function. However, what makes virtual functions important and capable of supporting run-time polymorphism is how they behave when accessed via a pointer. A base class pointer can be used to point to an object of any class derived from that base. When a base pointer points to a derived object that contains a virtual function, C++ determines which version of that function to call based upon the type of object pointed to by the pointer. And this determination is made at run time. Thus, when different objects are pointed to, different versions of the virtual function are executed. The same effect applies to base class references. To begin, examine this short example:

```
#include <iostream>
using namespace std;
class base {
public:
virtual void vfunc() {
cout << "This is base's vfunc().\n";
}
```

```
};
class derived1 : public base {
public:
void vfunc() {
cout << “This is derived1’s vfunc().\n”;
}
};
class derived2 : public base {
public:
void vfunc() {
cout << “This is derived2’s vfunc().\n”;
}
};
int main()
{
base *p, b;
derived1 d1;
derived2 d2;
// point to base
p = &b;
p->vfunc(); // access base’s vfunc()
// point to derived1
p = &d1;
p->vfunc(); // access derived1’s vfunc()
// point to derived2
p = &d2;
p->vfunc(); // access derived2’s vfunc()
return 0;
}
```

This program displays the following:

```
This is base’s vfunc().
This is derived1’s vfunc().
This is derived2’s vfunc().
```

As the program illustrates, inside base, the virtual function vfunc() is declared. Notice that the keyword virtual precedes the rest of the function declaration. When vfunc() is redefined by derived1 and derived2, the keyword virtual is not needed. (However, it is not an error to include it when redefining a virtual function inside a derived class; it is just not needed.)

In this program, base is inherited by both derived1 and derived2. Inside each class definition, vfunc() is redefined relative to that class. Inside main (), four variables are declared:

Name	Type
p	base class pointer
b	object of base
d1	object of derived1
d2	object of derived2

Next, p is assigned the address of b, and vfunc() is called via p. Since p is pointing to an object of type base, that version of vfunc() is executed. Next, p is set to the address of d1, and again vfunc() is

called by using p. This time p points to an object of type derived1. This causes derived1::vfunc() to be executed. Finally, p is assigned the address of d2, and p−>vfunc() causes the version of vfunc() redefined inside derived2 to be executed. The key point here is that the kind of object to which p points determines which version of vfunc() is executed. Further, this determination is made at run time, and this process forms the basis for run-time polymorphism.

Although you can call a virtual function in the "normal" manner by using an object's name and the dot operator, it is only when access is through a base class pointer (or reference) that run-time polymorphism is achieved. For example, assuming the preceding example, this is syntactically valid:

d2.vfunc(); // calls derived2's vfunc()

11.3.2 The Virtual Attribute Is Inherited

When a virtual function is inherited, its virtual nature is also inherited. This means that when a derived class that has inherited a virtual function is itself used as a base class for another derived class, the virtual function can still be overridden. Put differently, no matter how many times a virtual function is inherited, it remains virtual. For example, consider this program:

```
#include <iostream>
using namespace std;
class base {
public:
virtual void vfunc() {
cout << "This is base's vfunc().\n";
}
};
class derived1 : public base {
public:
void vfunc() {
cout << "This is derived1's vfunc().\n";
}
};
/* derived2 inherits virtual function vfunc() from derived1. */
class derived2 : public derived1 {
public:
// vfunc() is still virtual
void vfunc() {
cout << "This is derived2's vfunc().\n";
}
};
int main()
{
base *p, b;
derived1 d1;
derived2 d2;
// point to base
p = &b;
p->vfunc(); // access base's vfunc()
// point to derived1
p = &d1;
```

```
p->vfunc(); // access derived1's vfunc()
// point to derived2
p = &d2;
p->vfunc(); // access derived2's vfunc()
return 0;
}
```

As expected, the preceding program displays this output:

```
This is base's vfunc().
This is derived1's vfunc().
This is derived2's vfunc().
```

In this case, derived2 inherits derived1 rather than base, but vfunc() is still virtual.

11.3.3 Virtual Functions Are Hierarchical

As explained, when a function is declared as virtual by a base class, it may be overridden by a derived class. However, the function does not have to be overridden.

When a derived class fails to override a virtual function, then when an object of that derived class accesses that function, the function defined by the base class is used. For example, consider this program in which derived2 does not override vfunc():

```
#include <iostream>
using namespace std;
class base {
public:
virtual void vfunc() {
cout << "This is base's vfunc().\n";
}
};
class derived1 : public base {
public:
void vfunc() {
cout << "This is derived1's vfunc().\n";
}
};
class derived2 : public base {
public:
// vfunc() not overridden by derived2, base's is used
};
int main()
{
base *p, b;
derived1 d1;
derived2 d2;
// point to base
p = &b;
p->vfunc(); // access base's vfunc()
// point to derived1
p = &d1;
p->vfunc(); // access derived1's vfunc()
```

```
// point to derived2
p = &d2;
p->vfunc(); // use base's vfunc()
return 0;
}
```

The program produces this output:

```
This is base's vfunc().
This is derived1's vfunc().
This is base's vfunc().
```

Because derived2 does not override vfunc() , the function defined by base is used when vfunc() is referenced relative to objects of type derived2.

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Write a program to show the example of dynamic polymorphism and static polymorphism.

……..……

……..……

……...

11.4 Pure Virtual Functions

When a virtual function is not redefined by a derived class, the version defined in the base class will be used. However, in many situations there can be no meaningful definition of a virtual function within a base class. For example, a base class may not be able to define an object sufficiently to allow a base class virtual function to be created. Further, in some situations you will want to ensure that all derived classes override a virtual function. To handle these two cases, C++ supports the pure virtual function.

A pure virtual function is a virtual function that has no definition within the base class. To declare a pure virtual function, use this general form:

```
virtual type func-name(parameter-list) = 0;
```

When a virtual function is made pure, any derived class must provide its own definition. If the derived class fails to override the pure virtual function, a compile-time error will result.

The following program contains a simple example of a pure virtual function. The base class, number, contains an integer called val, the function setval() , and the pure virtual function show(). The derived classes hextype, dectype, and octtype inherit number and redefine show() so that it outputs the value of val in each respective number base (that is, hexadecimal, decimal, or octal).

```
#include <iostream>
using namespace std;
class number {
protected:
int val;
public:
void setval(int i) { val = i; }
// show() is a pure virtual function
```

```
virtual void show() = 0;
};
class hextype : public number {
public:
void show() {
cout << hex << val << "\n";
}
};
class dectype : public number {
public:
void show() {
cout << val << "\n";
}
};
class octtype : public number {
public:
void show() {
cout << oct << val << "\n";
}
};
int main()
{
dectype d;
hextype h;
octtype o;
d.setval(20);
d.show(); // displays 20 - decimal
h.setval(20);
h.show(); // displays 14 – hexadecimal
o.setval(20);
o.show(); // displays 24 - octal
return 0;
}
```

Exercise: Check Your Progress 2

Note: i) Use the space below for your answer.

Ex1: Write a C++ program to find the area of circle using virtual function.

……..……

……..……

……...

11.5 Early versus Late Binding

The virtual functions and run-time polymorphism, there are two terms that are need to be defined because they are used frequently in discussions of C++ and object-oriented programming: early binding and late binding.

Early binding refers to events that occur at compile time. In essence, early binding occurs when all information needed to call a function is known at compile time. (Put differently, early binding means that an object and a function call are bound during compilation.) Examples of early binding include normal function calls (including standard library functions), overloaded function calls, and overloaded operators. The main advantage to early binding is efficiency. Because all information necessary to call a function is determined at compile time, these types of function calls are very fast.
The opposite of early binding is late binding. As it relates to C++, late binding refers to function calls that are not resolved until run time. Virtual functions are used to achieve late binding. As you know, when access is via a base pointer or reference, the virtual function actually called is determined by the type of object pointed to by the pointer. Because in most cases this cannot be determined at compile time, the object and the function are not linked until run time. The main advantage to late binding is flexibility. Unlike early binding, late binding allows you to create programs that can respond to events occurring while the program executes without having to create a large amount of "contingency code." Keep in mind that because a function call is not resolved until run time, late binding can make for somewhat slower execution times.

Exercise: Check Your Progress 3

Case Study-A Polymorphic Genetic Algorithm

The ability to successfully apply polymorphic principles is one of the most valuable skills an object-oriented programmer can possess. Polymorphism drives adaptability and allows for complex functionality to be shared across even substantially different programming structures. It is also, fittingly, a very difficult skill to develop. It is really only by looking at many different examples of where polymorphism has been applied that we can begin to see the patterns that emerge.
Polymorphism is used all the way through the Java libraries - there are no shortages of areas that we could investigate for examples:

- The way an Array List stores its data.
- The way Swing interfaces work
- The way collections and arrays can be generically sorted and searched.

But more than this, we must look at otherwise exotic examples and see how polymorphism can be applied to provide a solid structure for expandable development. The best routines are applicable to as many different circumstances as possible. We are going to look at a fascinating tool of searching. The genetic algorithm is a powerful search technique often applied to very complex problems. The big benefit of a genetic algorithm (GA) is that we do not actually need to know what we are looking for. We just need to know when we have found it.

Brief Introduction to Genetic Algorithms

With a GA, we pose a problem. What that problem is does not matter. One common example is to maximize the results of a mathematical formula. We start off with a pool of solutions. Each of these solutions is a potential answer to the problem. To take a simplified example, if we had a GA that was to maximize the following formula:

$$A + B + (C * D)$$

Each solution would be a combination of values for A, B, C, and D. Usually we provide some limits to what these values are. For example, "A can be any number between one and one hundred. B will be between -100 and -200. C is a real number between zero and one inclusive, and D is a number between zero and one thousand).
A solution to this problem is a set of these four numbers. for example:

$$A = 20, B = -122, C = 0.50, D = 340$$

These values then get plugged into the formula:

$$20 + -122 + (0.50 * 340)$$

The results are then computed, and the result of this is classified as the fitness of a solution. In this case, the solution's fitness is 272. Traditionally, fitness will be a positive whole number. In cases where a particular solution calculates at a negative number, it will be given a fitness of 1

A genetic algorithm begins with an initial set of randomly generated solutions. Usually anything between 100 and 1000 solutions make a suitable starting set. Note that these are randomly generated - we have absolutely no way of knowing in advance how good any of these solutions are. The GA is an iterative process - each step of the iteration is called a generation. The driving force behind a GA is to "evolve" an answer to the problem. During each generation, the GA calculates how good each solution is, and then it picks a number of these to survive into the next generation - all of the others are discarded. It then "breeds" solutions to try and capitalize on their strengths (and minimize their weaknesses). It may also apply a "mutation" to a solution during the process. It will randomly change one value of the solution.

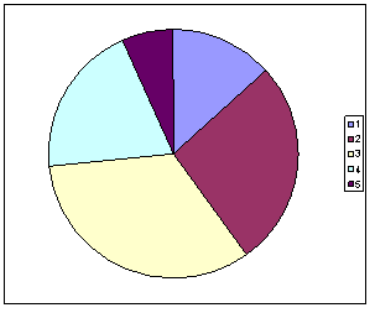
Each solution to a GA is tested against a fitness function. In the example, the function is our mathematical formula. During the evaluation phase, the fitness of each solution is calculated and stored.

During reproduction, we make use of a roulette wheel system to select a number of solutions for survival into the next generation. Consider if we had five solutions with the following fitness's:

Solution	Fitness
1	100
2	200
3	250
4	150
5	50

Starting Population

If we want to reproduce five solutions into the new generation, we sum up the fatnesses of all the current solutions (750) and then generate a random number that tells us which solution is to survive. The fitness of a particular solution relates to the proportional chance it has to reproduce into the next generation.



Fitness Graph

Once we have a new set of solutions, we then apply the cross-over principle. We pick two solutions at random, and then "breed" them. Let us imagine our solutions are represented by an array of four numbers to breed these solutions, we pick a random crossover point, and then we swap their arrays.

Questions

1. Describe the benefit of a genetic algorithm (GA).

2. What is the importance of genetic algorithm for polymorphism?

11.6 Summary

- Polymorphism is one of the features of oops. It simply means one name many forms.
- Virtual function is a member function of a class, whose functionality can be over-ridden in its derived classes.
- In C++ parametric polymorphism is implemented via templates.
- Operators are similar to functions they take operands and return a value.
- Virtual member functions are resolved during run-time. This mechanism is known as dynamic binding.

11.7 Keywords

Abstract Classes: An abstract class is a class that is designed to be specifically used as a base class. An abstract class contains at least one pure virtual function.

Compile Time Polymorphism: Compile time polymorphism is functions and operators overloading.

Function Overloading: Function overloading is one of the most powerful features of C++ programming language. It forms the basis of polymorphism (compile-time polymorphism).

Polymorphism: Polymorphism is a programming language feature that allows values of different data types to be handled using a uniform interface.

Runtime Time Polymorphism: Runtime time polymorphism is done using inheritance and virtual functions.

Virtual Function: A virtual function or virtual method is a function or method whose behaviour can be overridden within an inheriting class by a function with the same signature.

11.8 Review Questions

1. What is polymorphism?
2. Discuss the types of polymorphism.
3. What is compile-time polymorphism? Give the example.
4. What are the uses of virtual functions?
5. Write a program the base class reference is used to access a virtual function.
6. Define the pure virtual functions.
7. What is the importance of virtual functions?
8. Differentiate between early and late binding.
9. Give the example of runtime polymorphism.
10. What is ad-hoc polymorphism? Give the example.

Answers for Self Assessment Questions

1 (c) 2 (c) 3 (a)

11.9 Further Studies

1. http://stackoverflow.com/questions/5854581/polymorphism-in-c
2. The C++ Programming Language 4 Edition by Bjarne Stroustrup

12

C++ I/O System Basics

STRUCTURE

12.0 Objectives

12.1 Introduction

12.2 The C++ I/O System basics

12.3 Predefined Streams

12.4 Formatting Using the IOS Members

12.5 Clearing Format Flags

12.6 An Overloaded Form of setf( )

12.7 Examining the Formatting Flags

12.8 Using width ( ), precision ( ), and fill ( )

12.9 Using Manipulators to Format I/O

12.10 Creating Your Own Manipulator Functions

12.11 Summary

12.12 Keywords

12.13 Review Questions

12.14 Further Studies

12.0 Objectives

After studying this chapter, you will be able to:

- Discuss the formatted I/O
- Describe the using manipulators to format I/O
- Explain the overloading operators

12.1 Introduction

The C++ Programming Language", "Designing and implementing a general input/output facility for a programming language is notoriously difficult". He did an excellent job, and the C++ IOstreams library is part of the reason for C++'s success. IO streams provide an incredibly flexible yet simple way to design the input/output routines of any application.

IOstreams can be used for a wide variety of data manipulations thanks to the following features:

- A 'stream' is internally nothing but a series of characters. The characters may be either normal characters (char) or wide characters (wchar_t). Streams provide you with a universal character-based interface to any type of storage medium (for example, a file), without requiring you to know the details of how to write to the storage medium. Any object that can be written to one type of stream, can be written to all types of streams. In other words, as long as an object has a stream representation, any storage medium can accept objects with that stream representation.
- Streams work with built-in data types, and you can make user-defined types work with streams by overloading the insertion operator (<<) to put objects into streams, and the extraction operator (>>) to read objects from streams.
- The stream library.s unified approach makes it very friendly to use. Using a consistent interface for outputting to the screen and sending files over a network makes life easier. The programs below will show you what is possible.

12.2 The C++ I/O System basics

The C++ I/O system allows you to format I/O operations. For example, you can set a field width, specify a number base, or determine how many digits after the decimal point will be displayed. There are two related but conceptually different ways that you can format data. First, you can directly access members of the ios class.

Specifically, you can set various format status flags defined inside the ios class or call various ios member functions. Second, you can use special functions called manipulators that can be included as part of an I/O expression. We will begin the formatted I/O by using the ios member functions and flags.

12.3 Predefined Streams

When a C++ program begins execution, four built-in streams are automatically opened. They are:

Stream	Meaning	Default Device
cin	Standard input	Keyboard
cout	Standard output	Screen
cerr	Standard error output	Screen
clog	Buffered version of cerr	Screen

Streams cin, cout, and cerr correspond to C’s stdin, stdout, and stderr.

By default, the standard streams are used to communicate with the console. However, in environments that support I/O redirection (such as DOS, Unix, OS/2, and Windows), the standard streams can be redirected to other devices or files. For the sake of simplicity, the examples in assume that no I/O redirection has occurred. Standard C++ also defines these four additional streams: win, wout, werr, and wlog. These are wide character versions of the standard streams. Wide characters are of type wchar_t and are generally 16-bit quantities. Wide characters are used to hold the large character sets associated with some human languages.

Caution

If you use a string constant (or variable) to store the file name, you must add a special conversion when connecting the stream to avoid the error

12.4 Formatting Using the IOS Members

Each stream has associated with it a set of format flags that control the way information is formatted. The ios class declares a bitmask enumeration called fmtflags in which the following values are defined. (Technically, these values are defined within ios_base, which, as explained earlier, is a base class for ios.)

adjustfield	basefield	boolalpha	dec
fixed	floatfield	hex	internal
left	oct	right	scientific
showbase	showpoint	showpos	skipws
unitbuf	uppercase		

These values are used to set or clear the format flags. If you are using an older compiler, it may not define the fmtflags enumeration type. In this case, the format flags will be encoded into a long integer.
When the skipws flag is set, leading white-space characters (spaces, tabs, and newlines) are discarded when performing input on a stream. When skipws is cleared, white-space characters are not discarded.

When the left flag is set, output is left justified. When right is set, output is right justified. When the internal flag is set, a numeric value is padded to fill a field by inserting spaces between any sign or base character. If none of these flags are set, output is right justified by default. By default, numeric values are output in decimal. However, it is possible to change the number base. Setting the oct flag causes output to be displayed in octal. Setting the hex flag causes output to be displayed in hexadecimal. To return output to decimal, set the dec flag.

Setting showbase causes the base of numeric values to be shown. For example, if the conversion base is hexadecimal, the value 1F will be displayed as 0x1F.
By default, when scientific notation is displayed, the e is in lowercase. Also, when a hexadecimal value is displayed, the x is in lowercase. When uppercase is set, these characters are displayed in uppercase.

12.5 Clearing Format Flags

The complement of setf() is unsetf(). This member function of ios is used to clear one or more format flags. Its general form is

void unsetf(fmtflags flags);

The flags specified by flags are cleared. (All other flags are unaffected.)
The following program illustrates unsetf(). It first sets both the uppercase and scientific flags. It then outputs 100.12 in scientific notation. In this case, the “E” used in the scientific notation is in uppercase. Next, it clears the uppercase flag and again outputs 100.12 in scientific notation, using a lowercase “e.”

```
#include <iostream>
```

```
using namespace std;
int main()
{
cout.setf(ios::uppercase | ios::scientific);
cout << 100.12; // displays 1.0012E+02
cout.unsetf(ios::uppercase); // clear uppercase
cout << " \n" << 100.12; // displays 1.0012e+02
return 0;
}
```

12.6 An Overloaded Form of setf( )

There is an overloaded form of setf() that takes this general form:

```
fmtflags setf(fmtflags flags1, fmtflags flags2);
```

In this version, only the flags specified by flags2 are affected. They are first cleared and then set according to the flags specified by flags1. Note that even if flags1 contains other flags, only those specified by flags2 will be affected. For example,

```
#include <iostream>
using namespace std;
int main( )
{
cout.setf(ios::showpoint | ios::showpos, ios::showpoint);
cout << 100.0; // displays 100.0, not + 100.0
return 0;
}
```

Here, the basefield flags (i.,e., dec, oct, and hex) are first cleared and then the hex flag is set.
Remember, only the flags specified in flags2 can be affected by flags specified by flags1. For example, in this program, the first attempt to set the showpos flag fails.

```
//This program will not work.
#include <iostream>
using namespace std;
int main()
{
cout.setf(ios::showpos, ios::hex); // error, showpos not set
cout << 100 << "\n"; // displays 100, not +100
cout.setf(ios::showpos, ios::showpos); // this is correct
cout << 100; // now displays +100
return 0;
}
```

Keep in mind that most of the time you will want to use unsetf() to clear flags and the single parameter version of setf() (described earlier) to set flags. The setf(fmtflags, fmtflags) version of

setf() is most often used in specialized situations, such as setting the number base. Another good use may involve a situation in which you are using a flag template that specifies the state of all format flags but wish to alter only one or two. In this case, you could specify the template in flags1 and use flags2 to specify which of those flags will be affected

Self Assessment Questions

1. A ……defines the form of a class without fully specifying the data upon which it will operate.
(a) stream classes (b) template class (c) I/O classes (d) derived classes

2. Wide characters are of type wchar_t and are generally ……..quantities.
(a) 8-bit (b) 16-bit (c) 32-bit (d) None of these

3. ……..are used to hold the large character sets associated with some human languages.
(a) 8-bit (b) 16-bit
(c) Wide characters (d) White-space characters

4. The complement of setf() is……
(a) setprecision () (b) main() (c) flags() (d) unsetf()

5. The ……. function has a second form that allows you to set all format flags associated with a stream.
(a) setprecision () (b) main() (c) flags() (d) unsetf()

12.7 Examining the Formatting Flags

There will be times when you only want to know the current format settings but not alter any. To accomplish this goal, ios includes the member function flags(), which simply returns the current setting of each format flag. Its prototype is shown here:

fmtflags flags( );
The following program uses flags() to display the setting of the format flags relative to cout. Pay special attention to the showflags() function. You might find it useful in programs you write.

```
#include <iostream>
using namespace std;
void showflags() ;
int main()
{
// show default condition of format flags
showflags();
cout.setf(ios::right | ios::showpoint | ios::fixed);
showflags();
return 0;
}

// This function displays the status of the format flags.
void showflags()
```

```
{
ios::fmtflags f;
long i;
f = (long) cout.flags(); // get flag settings
// check each flag
for(i=0x4000; i; i = i >> 1)
if(i & f) cout << "1";
else cout << "0";
cout << " \n";
}
```

The output from the program is shown here:

```
0 0 0 0 0 1 0 0 0 0 0 0 0 0 1
0 1 0 0 0 1 0 1 0 0 1 0 0 0 1
```

12.8 Using width ( ), precision ( ), and fill ( )

In addition to the formatting flags, there are three member functions defined by ios that set these format parameters: the field width, the precision, and the fill character. The functions that do these things are width (), precision (), and fill (), respectively.

Each is examined in turn.

By default, when a value is output, it occupies only as much space as the number of characters it takes to display it. However, you can specify a minimum field width by using the width() function. Its prototype is shown here:

```
streamsize width(streamsize w);
```

Here, it becomes the field width, and the field width is returned. In some implementations, the field width must be set before each output. If it is not, the default field width is used. The streamsize type is defined as some form of integer by the compiler.

After you set a minimum field width, when a value uses less than the specified width, the field will be padded with the current fill character (space, by default) to reach the field width. If the size of the value exceeds the minimum field width, the field will be overrun. No values are truncated.

When outputting floating-point values, you can determine the number of digits to be displayed after the decimal point by using the precision() function. Its prototype is shown here:

```
streamsize precision(streamsize p);
```

Here, the precision is set to p, and the old value is returned. The default precision is 6.

In some implementations, the precision must be set before each floating-point output. If it is not, then the default precision will be used.

By default, when a field needs to be filled, it is filled with spaces. You can specify the fill character by using the fill() function. Its prototype is

```
char fill(char ch);
```

After a call to fill(), ch becomes the new fill character, and the old one is returned.

Here is a program that illustrates these functions:

```
#include <iostream>
using namespace std;
int main()
{
cout.precision(4) ;
cout.width(10);
cout << 10.12345 << "\n"; // displays 10.12
cout.fill("*");
cout.width(10);
cout << 10.12345 << "\n"; // displays *****10.12
// field width applies to strings, too
cout.width(10);
cout << "Hi!" << "\n"; // displays *******Hi!
cout.width(10);
cout.setf(ios::left); // left justify
cout << 10.12345; // displays 10.12*****
return 0;
}
```

This program's output is shown here:

```
10.12
*****10.12
*******Hi!
10.12*****
```

There are overloaded forms of width**()**, precision(), and fill() that obtain but do not change the current setting. These forms are shown here:

```
char fill( );
streamsize width( );
streamsize precision( );
```

Caution

Conditional and arithmetic expressions must be contained in parentheses.

12.9 Using Manipulators to Format I/O

The second way you can alter the format parameters of a stream is through the use of special functions called manipulators that can be included in an I/O expression. The standard manipulators are shown in Table 12.1. As you can see by examining the table, many of the I/O manipulators parallel member functions of the ios class. Many of the manipulators were added recently to C++ and will not be supported by older compilers.

Table 12.1: The C++ manipulators

Manipulator	Purpose	Input/Output
boolalpha	Turns on **boolapha** flag.	Input/Output
dec	Turns on **dec** flag.	Input/Output
endl	Output a newline character and flush the stream.	Output
ends	Output a null.	Output
fixed	Turns on **fixed** flag.	Output
flush	Flush a stream.	Output
hex	Turns on **hex** flag.	Input/Output
internal	Turns on **internal** flag.	Output
left	Turns on **left** flag.	Output
nobooalpha	Turns off **boolalpha** flag.	Input/Output
noshowbase	Turns off **showbase** flag.	Output
noshowpoint	Turns off **showpoint** flag.	Output
noshowpos	Turns off **showpos** flag.	Output

Manipulator	Purpose	Input/Output
noskipws	Turns off skipws flag.	Input
nounitbuf	Turns off unitbuf flag.	Output
nouppercase	Turns off uppercase flag.	Output
oct	Turns on oct flag.	Input/Output
resetiosflags (fmtflags *f*)	Turn off the flags specified in *f*.	Input/Output
right	Turns on right flag.	Output
scientific	Turns on scientific flag.	Output
setbase(int *base*)	Set the number base to *base*.	Input/Output
setfill(int *ch*)	Set the fill character to *ch*.	Output
setiosflags(fmtflags *f*)	Turn on the flags specified in *f*.	Input/output
setprecision (int *p*)	Set the number of digits of precision.	Output
setw(int *w*)	Set the field width to *w*.	Output
showbase	Turns on showbase flag.	Output
showpoint	Turns on showpoint flag.	Output
showpos	Turns on showpos flag.	Output
skipws	Turns on skipws flag.	Input
unitbuf	Turns on unitbuf flag.	Output
uppercase	Turns on uppercase flag.	Output
ws	Skip leading white space.	Input

To access manipulators that take parameters (such as setw()), you must include <iomanip> in your program.

Here is an example that uses some manipulators:

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
cout << hex << 100 << endl;
cout << setfill('?') << setw(10) << 2343.0;
return 0;
}
```

This displays:

```
64
??????2343
```

Exercise: Check Your Progress 1

Note: i) Use the space below for your answer.

Ex1: Write a C++ program to display a message on two new lines using the endl manipulator.

……..……

……..……

……...

12.10 Creating Your Own Manipulator Functions

In addition to overloading the insertion and extraction operators, you can further customize C++'s I/O system by creating your own manipulator functions. Custom manipulators are important for two main reasons. First, you can consolidate a sequence of several separate I/O operations into one manipulator. For example, it is not uncommon to have situations in which the same sequence of I/O operations occurs frequently within a program. In these cases you can use a custom manipulator to perform these actions, thus simplifying your source code and preventing accidental errors. A custom manipulator can also be important when you need to perform I/O operations on a nonstandard device. For example, you might use a manipulator to send control codes to a special type of printer or to an optical recognition system.

Custom manipulators are a feature of C++ that supports OOP, but also can benefit programs that are not object oriented. Custom manipulators can help make any I/O intensive program clearer and more efficient.

All parameterless manipulator output functions have this skeleton:

```
ostream &manip-name(ostream &stream)
{
// your code here
return stream;
}
```

Here, manip-name is the name of the manipulator. Notice that a reference to a stream of type ostream is returned. This is necessary if a manipulator is used as part of a larger I/O expression. It is important to note that even though the manipulator has as its single argument a reference to the stream upon which it is operating, no argument is used when the manipulator is inserted in an output operation.
As a simple first example, the following program creates a manipulator called sethex(), which turns on the showbase flag and sets output to hexadecimal.

```
#include <iostream>
#include <iomanip>
using namespace std;
// A simple output manipulator.
ostream &sethex(ostream &stream)
{
stream.setf(ios::showbase);
stream.setf(ios::hex, ios::basefield);
return stream;
}
int main()
{
cout << 256 << " " << sethex << 256;
return 0;
}
```

Exercise: Check Your Progress 2

Case Study-A Gentle Introduction to C++ IO Streams

One of the great strengths of C++ is its I/O system, IO Streams. As Bjarne Stroustrup says in his book "The C++ Programming Language", "Designing and implementing a general input/output facility for a programming language is notoriously difficult". He did an excellent job, and the C++ IO streams library is part of the reason for C++'s success. IO streams provide an incredibly flexible yet simple way to design the input/output routines of any application.
IO streams can be used for a wide variety of data manipulations thanks to the following features:

- A "stream" is internally nothing but a series of characters. The characters may be either normal characters (char) or wide characters (wchar_t). Streams provide you with a universal character-based interface to any type of storage medium (for example, a file), without requiring you to know the details of how to write to the storage medium. Any object that can be written to one type of stream can be written to all types of streams. In other words, as long as an object has a stream representation, any storage medium can accept objects with that stream representation.
- Streams work with built-in data types, and you can make user-defined types work with streams by overloading the insertion operator (<<) to put objects into streams, and the extraction operator (>>) to read objects from streams.
- The stream library's unified approach makes it very friendly to use. Using a consistent interface for outputting to the screen and sending files over a network makes life easier. The programs below will show you what is possible.

The IO stream class hierarchy is quite complicated, so rather than introduce you to the full hierarchy at this point, the concepts of the design and show you examples of streams in action. Once you are

familiar with elements of the design and how to apply those concepts to design a robust I/O system for your software, an understanding of what belongs where in the hierarchy will come naturally.
The most important of the basic stream operations are:

1. First, the stream is initialized with the appropriate type (like a std::string for a stringstream and the filename for an fstream) of values and suitable modes (like ios::in for input and ios::out for output and many more depending on the type of the stream).
2. After that, you can specify where the I/O should occur, through the get and put pointers. Depending on how you open the stream, the location may already be set appropriately (for example, if you open a file with ios::app, your get pointer set at the end of the stream, allowing appends).

The member functions associated with setting the get and put pointers are:

- seekg()and seekp() for dragging. the get and put pointer, respectively, to the position specified. Both seek methods take an argument (of type streampos) providing a position in the file relative to the beginning of the file (using ios::beg), the end of the file (using ios::end), or the current position (using ios::cur). You may also provide just a specific location, such as io::beg, for the beginning of the file.
- tellg() and tellp() provide the current location of the get and put pointers, respectively

The following one-liners should clear up most questions:

```
seekg(0); seekg(0,ios::beg); //sets the get pointer to the beginning.
seekg(5,ios::beg);           //sets the get pointer to 5 chars forward of the beginning.
tellp(); tellg()             //returns the current value of the put/get pointer
seekp(-10,ios::end);         //sets the put pointer to 10 chars before the end
seekp(1,ios::cur);           //proceeds to next char
```

Be careful when seeking the put pointer into the middle of in the stream. If you put anything in the stream, it will directly into the stream at the put location. In other words, if you need to insert data in the middle of a stream, you have to manually move the data that would be overwritten. As a side note, if you are finding yourself doing that too often, then you may want to use a string representation of your data, which can simplify this kind of random access operation.

3. Once you are at the right location in the stream, input and output is done through the << and >> operators. If you want to input an object to the stream, use the << operator; for output, use >>. The class for your object must, of course, have provided overloads for these methods. Here is a short example:

```
//Inserts var into string (like objects are displayed by
// putting them to cout)
output_stream<<var;
//Gets the value from the stream's characters positioned
// after the get pointer and puts it into var.
input_stream>>var;
```

If var is an object (either a built in class or a user defined type), the exact process of the input or output is dependent on the overloaded >> or << operator respectively.

Questions

1. What do input and output really mean?
2. How do streams work?

12.11 Summary

- A stream is a logical device that either produces or consumes information.
- The setf(fmtflags, fmtflags) version of setf() is most often used in specialized situations, such as setting the number base.
- The << output operator is referred to as the insertion operator because it inserts characters into a stream.
- Encapsulation is an essential component of object-oriented programming.
- An output manipulator is particularly useful for sending special codes to a device.
- The standard streams are used to communicate with the console.

12.12 Keywords

Field Width: describes the width of the next element to be output. This value can be obtained/modified by calling the member function width or parameterized manipulator setw.

Format Flags: a set of internal indicators describing how certain input/output operations shall be interpreted or generated. The state of these indicators can be obtained or modified by calling the members flags, setf and unsetf, or by using manipulators.

I/O Stream Library: The IO stream library is an object-oriented library that provides input and output functionality using streams.

Manipulators: Manipulators are functions specifically designed to be used in conjunction with the insertion (<<) and extraction (>>) operators on stream objects.

Setw( ): The Setw ( ) manipulator is used to set the width of the word to be displayed on screen.

12.13 Review Questions

1. What are stream classes? Give the example of stream classes.
2. Discuss the format flags.
3. What is importance of ios members?
4. Create a program using showflags() function.
5. Different between old-style I/O library and new-style I/O library.
6. Write a program using width ( ), precision ( ), and fill ( ).
7. Give the example that uses some manipulators.
8. Describe the inserter functions.
9. What is manipulator?
10. What is a manipulator function?

Answers for Self Assessment Questions

1 (b) 2 (b) 3 (c) 4 (d) 5 (c)

12.14 Further Studies

1. http://en.wikibooks.org/wiki/C++_Programming/Chapter_Learning_the_Basics
2. The C++ Programming Language 4 Edition by Bjarne Stroustrup