	Post-Graduate Diploma in **Paper : PGDCA-I** **Computer Applications** **Introduction to Information Technology**
Department of Distance Education **Punjabi University, Patiala** 	**Section-B** **LESSON NO. :** 2.1 : Computer Program 2.2 : Computer Languages 2.3 : Computer Software 2.4 : Operating System 2.5 : Data Communication 2.6 : Computer Network 2.7 : Database Fundamentals 2.8 : Database Management System

Lesson No. 2.1 **Author : Arun Sharma**

COMPUTER PROGRAM

Chapter Outline:

2.1.0 Objectives

- Learn the different phases of development of a program
- Different tools to develop the program solving logic like Algorithm, Flow Charts and Pseudocode
- Different approaches to test and debug a program
- Types of Documentation of a Program
- Characteristics of a Good Program

Computers have emerged as the most useful device in recent times. They perform a variety of tasks. However, being machines, computers cannot perform on their own. They need to be instructed to perform even something so simple as addition of two numbers. Computers work on a set of instructions called *computer program,* which clearly specify the ways to carry out a specific task.

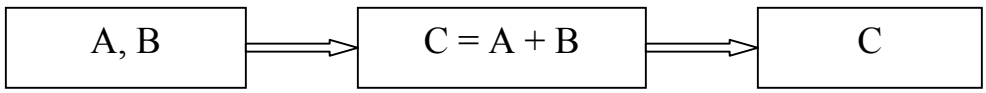
A computer specialist who is responsible for designing, writing and modifying computer programs, allowing end-users to interface directly with computer operating systems and hardware is known as a *computer programmer.* Broadly, programmers can be classified into two categories, namely, *system programmers* and *application programmers.* System programmers are mainly concerned with those programs, which provide interface and functionality to the hardware components. An application programmer writes programs to fulfill a specific task such as inventory control and payroll system.

Note: The collection of instructions forms a program and a group of programs forms software.

2.1.1 Developing a Program

As told earlier, a program consists of a series of instructions that a computer processes to perform the required operation. In addition, it also includes some fixed data required to perform the instructions and the process of defining those instructions and data. Thus, in order to design a program, a programmer must determine three basic rudiments:

- The instructions to be performed.
- The order in which those instructions are to be performed.
- The data required to perform those instructions.



Input Processing Output

Figure 2.1.1 Program Performing a Task

Suppose we want to Calculate the sum of two numbers, A and B and store the sum in C, here A and B are the inputs, addition is the process and C is the output of the program.

Programming involves many stages such as task analysis, planning the logic, program flowcharting, coding, program testing, implementation, documentation and maintenance.

2.1.1.1 Program Development Cycle

Before beginning with the process of coding (writing a program), the programmer has to determine the problem that needs to be solved. There are different approaches to problem solving. Most require breaking the problem into a series of smaller steps, independent of the programming language. Development cycle of a program includes the following phases:

1. **Analyse/Define the Problem:** In this phase, the problem is analysed precisely and completely. Based on his understanding, the developer knows about the scope within which the problem needs to be developed.
2. **Task Analysis:** After analysing the problem, the developer needs to develop various solutions to solve the given problem. From these solutions, the optimum solution (by experimenting with all the solutions) is chosen, which can solve the problem comfortably and economically.
3. **Using Programming Tools:** After selecting the appropriate solution, algorithm is developed to depict the basic logic of the selected solution. An algorithm depicts the solution in logical steps (sequence of instructions). Further, algorithm is represented by flowcharts and pseudocodes.
4. **Testing the Algorithm for Accuracy:** Before converting the algorithms into actual code, it should be checked for accuracy. The main purpose of checking algorithm is to identify major logical errors at an early stage, because logical errors are often difficult to detect and correct at later stages.
5. **Coding the Solution:** After meeting all the design considerations, the actual coding of the program takes place in the chosen programming language. Depending upon application domain and available resources, a program can be written by using computer languages
6. **Test and Debug the Program:** It is not unusual for the initial program code to contain errors. The errors may be in the form of logical errors ***(semantic errors)*** or incorrect use of programming language (grammatical errors also known as ***syntax errors***). *A program compiler and programmer designed test data machine tests the code for syntax errors*. The results obtained are compared with results calculated manually from this test data.
7. **Documentation:** Once the program is free from all the errors, it is the

duty of the program developers to ensure that the program is supported by suitable documentation. These documents should be supplied to the program users. Documenting a program enables the user to operate the program correctly. It also enables other persons to understand the program clearly.

8. **Implementation:** After performing all the aforesaid steps, the program is installed on the end user's machine. The user is also provided with all the essential documents so that he can understand how the program works. The implementation can be viewed as the final testing because only after using the program, the user can point out the weaknesses, if any, to the developers.
9. **Maintenance and Enhancement:** After the program is implemented (loading and executing the program on user's computer), it should be properly maintained taking care of the changing requirements of its users and system. The program should be regularly enhanced by adding additional capabilities. This phase is also concerned with detecting and fixing these errors, which were missed in testing phase.

2.1.2 Algorithm

Algorithms are one of the most basic tools that are used to develop the problem solving logic. An *algorithm* is defined as a finite sequence of explicit instructions that, when provided with a set of input values, produces an output and then terminates. To be an algorithm, the steps must be unambiguous and after a finite number of steps the solution of the problem is achieved. However, algorithms can have steps that repeat (iterate) or require decisions (logic and comparison) until the task is completed.

Different algorithms may accomplish the same task, with a different set of instructions, in more or less the same time, space and efforts. For example, two different recipes for preparing tea, one 'add the sugar' while 'boiling the water' and the other 'after boiling the water' produce the same result. However, performing an algorithm correctly does not guarantee a solution, if the algorithm is flawed or not appropriate to the context. For example, preparing the tea algorithm will fail if there are no tealeaves present; even if all the motions of preparing the tea are performed as if the tea leaves were there. Note that algorithms are *not* computer programs, as they cannot be executed by a computer.

An algorithm must have the following properties:

1. There must be no ambiguity in any instruction.

2. There should not be any uncertainty about which instruction is to be executed next.
3. The description of the algorithm must be finite. An algorithm cannot be open-ended.
4. The execution of the algorithm should conclude after a finite number of steps.
5. The algorithm must be general enough to deal with any contingency.

Consciously or subconsciously, we use algorithms in our daily life. For example, to determine the largest number out of three numbers A, B, and C, the following algorithm may be used.

Step 1: Start
Step 2: Read 3 numbers say: A, B, C
Step 3: Find the larger number between A and B and store it in MAX_AB and store it in MAX
Step 4: Find the larger number between MAX_AB and C
Step 5: Display MAX
Step 6: Stop

The above-mentioned algorithm terminates after six steps. This explains the feature of finiteness. Every action of the algorithm is precisely defined; hence, there is no scope of ambiguity.

The need of algorithms can be understood as it provides a logical structure to plan the solution. Once the solution is properly designed, the only job left is to code that logic into the respective programming language. For developing an effective algorithm, flowcharts and pseudocodes are used by programmers. They are further expressed in programming language to develop computer programs.

2.1.3 Flowchart

A *flowchart* is a pictorial representation of an algorithm in which the steps are drawn in the form of different shapes of boxes and the logical flow is indicated by interconnecting arrows. The boxes represent operations and the arrows represent the sequence in which the operations are implemented. The primary purpose of the flowchart is to help the programmer in understanding the logic of the program. Therefore, it is always not necessary to include all the required steps in detail. Flowcharts outline the general procedure. Since they provide an alternative, visual way of representing the information flow in a program, program developers often find them very

valuable.

Flowcharts can be compared with the blueprint of a building. Just as an architect draws a blueprint before starting construction of a building, a programmer draws a flowchart prior to writing a computer program. As in the case of the drawing of a blueprint, the flowchart is drawn according to defined rules I and using standard flowchart symbols prescribed by the American National Standard Institute (ANSI),

2.1.3.1 Importance of Flowcharts

A flowchart helps to clarify how things are currently working and how they could be improved. It also assists in finding the key elements of a process, while drawing clear lines between where one process ends and the next one starts. Developing a flowchart encourages communication among the participants and establishes a common understanding between them about the process. Flowcharts also help in revealing redundant or misplaced steps. In addition, flowcharts are used to identify appropriate team members by identifying who provides inputs or to whom the resources are to be allocated. It also helps in establishing important areas for monitoring or data collection and to identify areas for improvement or increase in efficiency. The reasons for using flowcharts as a problem solving tool are given below.

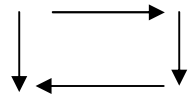
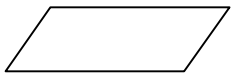
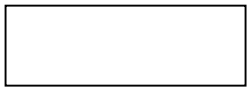
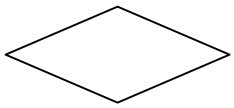
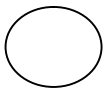
1. **Makes Logic Clear:** The main advantage of using a flowchart to plan a task is that it provides a pictorial representation of the task, which makes the logic easier to follow. The symbols are connected in such a way that they show the movement (flow) of information through the system visibly.
2. **Communication:** Being a graphical representation of a problem solving logic, flowcharts are a better way of communicating the logic of a system to all concerned. That is, the diagrammatical representation of logic is easier to communicate to all the interested parties as compared to actual program code as the users may not be aware of all the programming techniques and jargons.
3. **Effective Analysis:** With the help of a flowchart, a problem can be analysed in an effective way. This is because the analysing duties of the programmers can be delegated to other persons, who may or may not know the programming techniques as they have a broad idea about the logic.
4. **Useful in Coding:** The flowcharts act as a guide or blueprint during the analysis and program development phase. Once the flowcharts are ready,

the programmers can plan the coding process effectively. As a result, error free programs are developed in high-level languages and that too at a faster rate.

5. **Proper Testing and Debugging:** By nature, a flowchart helps in detecting the errors in a program, as the developers know exactly what the logic should do. Developers can test various data for a process so that the program can handle every contingency.
6. **Appropriate Documentation:** Flowcharts serve as a good program documentation tool. Since normally the programs are developed for novice users, they can take the help of the program documentation to know what the program actually does and how to use the program.

2.1.3.2 Flowchart Symbols

A flowchart uses special shapes to represent different types of actions or steps in a process. Some standard symbols, which are frequently required for flowcharts, are shown in Table 2.1.1. Note that each symbol has a different shape denoting a different type of operation.

	Flow Lines
	Terminal
	Input / Output
	Processing
	Decision
	Connector

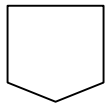
	Off-page Connector

Table 2.1.1 Flowchart Symbols

2.1.3.3 Guidelines for Preparing Flowcharts

The following guidelines should be used for creating a flowchart:

1. The flowchart should be clear, neat and easy to follow.
2. The flowchart must have a logical start and finish.
3. In drawing a proper flowchart, all necessary requirements should be listed in logical order.
4. The direction of the flow of a procedure should always be from left to right or top to bottom.
5. Only one flow line should come out from a process symbol.
6. Only one flow line should enter a decision symbol. However, two or three flow lines (one for each possible answer) may leave the decision symbol.
7. Only one flow line is used with a terminal symbol.
8. Within standard symbols, write briefly. If necessary, use the annotation symbol to describe data or process more clearly.
9. In case of complex flowcharts, connector symbols are used to reduce the number of flow lines.
10. Intersection of flow lines should be avoided to make it a more effective and better way of representing communication.
11. It is useful to test the validity of the flowchart by passing through it with normal/unusual test data.

2.1.3.4 Limitations of Flowcharts

1. **Complex:** The major disadvantage in using flowcharts is that when a program is very large, the flowcharts may continue for many pages, making them hard to follow. Flowcharts tend to get large very quickly and it is difficult to follow the represented process. It is also very labourious to draw a flowchart for a large program.
2. **Costly:** Drawing simple flowcharts may be a cost effective process. However, if the flowchart is to be drawn for a huge program, the time and cost factor of program development gets out of

proportion, making it a costly affair.

3. **Difficult to Modify:** Due to its symbolic nature, any changes or modification to a flowchart usually requires redrawing the entire logic again and redrawing a complex flowchart is not a child's play. It is not easy to draw thousands of flow lines and symbols along with proper spacing, especially for a large complex program.
4. **No Update:** Usually programs are updated regularly. However, the corresponding update of flowcharts may not take place, especially in case of large programs. As a result, the logic used in the flowchart may not match with the actual program's logic. This inconsistency in flowchart update defeats the main purpose of the flowcharts, that is, to give the users the basic idea about the program's logic.

2.1.4 Pseudocode (P-Code)

Pseudocode (pronounced *Soo-Doh-Kohd)* is made up of two words: *Pseudo* and *Code.* Pseudo means imitation and Code refers to instructions, written in a programming language. As the name suggests, pseudocode is not a real programming code, but it models and may even look like programming code. It is a generic way of describing an algorithm without using any specific programming language-related notations. Simply put, pseudocode is an outline of a program, written in a form that can be easily converted into real programming statements. Pseudocode uses plain English statements rather than symbols, to represent the processes of a computer program. Pseudocode is also known as *PDLC* (*pr*ogram Design Language), as it emphasises more on the design aspect of a computer program or stru*ctured english,* because usually pseudocode instructions are written in normal English but in a structured way.

Note: *Pseudocode cannot be compiled nor executed and there are no real formatting or syntax rules. It is simply one step, an important one, in producing the final code.*

2.1.4.1 Why Pseudocode?

Pseudocode is an informal high level description of the operating principle of a computer program or other algorithm.

Programming is a complicated process. Initially, the programmer has to understand the program specifications. Then he needs to organise his thoughts and finally construct the program. This can be a difficult task when the program requirements are complex in nature. Pseudocode provides a simple

method of developing the program logic as it uses everyday language to prepare a brief set of instructions in the order in which they will appear in the finished program. It allows the programmer to focus on the steps required to solve a program rather than on how to use the computer language. An advantage of pseudocode is that it can be used to describe a program to a non-technical user and can still provide guidelines for the writing of program code. Moreover, writing pseudocode saves lot of time later during the construction and testing phase of program's development.

2.1.4.2 Pseudocode Guidelines

Writing pseudocode is not that difficult task. Even if you do not know anything about the computers or computer languages, you can still develop effective and efficient pseudocodes, provided that you are writing in an organised manner. The main purpose of pseudocode is to help the programmer to write efficient program code. You need *not* to be as specific as in a computer language. Even longish, yet simple statements can generate ample amount of ideas for the programmer about how to go about writing an actual code. However, you must honestly attempt to add enough detail and analysis to the pseudocode.

Although there are no established standards for pseudocode instruction, here are a few general guidelines for checking your pseudocode:

1. Statements should be written in simple English (or whatever you prefer) and should be language independent. Remember that you are only describing the logic plan to develop a program, you are not coding.
2. Steps must be clear, that is, unambiguous and when the steps (that is, instructions) are followed, they must produce a solution to the specified problem. If the pseudocode is difficult for a person to read or translate into code, then something is wrong with the level of detail you have chosen to use.
3. Be concise, you do not need to worry about programming language syntax, but can still be precise.
4. Each instruction should be written in a separate line and each statement in pseudocode should express just one action for the computer. If the task list is properly drawn, then in most cases each task will correspond to one line of pseudocode.
5. Capitalize keywords such as READ, WRITE, IF, ELSE, etc.
6. Use only three structures: Sequence, Selection and Repetition and apply proper indentation to signify particular control structures.

End multi-line structures with ENDIF, ENDCASE, ENDDO etc.

7. Each set of instructions is written from top to bottom, with only one entry and one exit.
8. It should allow for easy transition from design to implementation in programming language.

Advantages of using pseudocode

- Since it is language independent, it can be used by most programmers. It allows the developer to express the design in plain natural language.
- It is easier to develop a program from a pseudocode than with a flowchart. Programmers do not have to think about syntaxes or symbols; they simply have to concentrate on the underlying logic. The focus is on the steps required to solve a problem rather than on how to use the computer language.
- Often, it is easy to translate pseudocode into a programming language, a step which can be accomplished by less experienced programmers.
- The use of words and phrases in the pseudocode, which are in line with basic computer operations, simplifies the translation from the pseudocode algorithm to a specific programming language.
- Unlike flowcharts, pseudocode is compact and does not tend to run over many pages. Its simple structure and readability makes it easier to modify as well.
- Since they are inclined more towards natural languages, pseudocodes are self-documenting. They are easy to generate and maintain using simple word processors. In addition, pseudocode can be reviewed by groups more easily than real code.
- Pseudocode allows programmers who work in different computer languages to talk to each other.

Disadvantages of using pseudocode

- The main disadvantage of using pseudocode is that it does not provide visual representation of the program's logic.
- There are no accepted standards for writing pseudocodes. Different programmers use their own style of writing pseudocode.

2.1.5 Program Testing and Debugging

With the tools described above, it should be possible to develop a program that will

implement the solution of any problem on the computer. However, the fact that a program is capable of producing results is no guarantee that these answers are correct. Furthermore, it is often found that many newly written programs at first refuse to execute at all. Thus, all computer programs need to be tested to ensure that they are running properly and giving desired results. If they fail the test procedure, the reasons for this failure must be detected and removed. **The process of detecting, isolating and correcting bugs in a program is known as *debugging*.**

Debuggin is the routine process of locating and removing computer program errors, bugs or abnormalities which is methodically handled by software tester by using debugging or testing tools.

Debugging checks, detects and corrects errors or bugs to allow proper program operation according to its specifications.

The errors in a program can be classified into two categories:

1. **Syntax Errors: These errors occur when the rules of the programming language are not followed.** Syntax errors are the most common errors and typically, they involve incorrect punctuation, incorrect word sequence, undefined terms or misuse of terms. For example, DO WILE X < 10, contains a syntax error. In this example, the problem is of incorrect spelling of WHILE. These errors are easier to correct because they provide useful error messages which gives an idea about what is wrong with the program.
2. **Logical Errors: A logical error is an error in planning the program's logic.** These errors are the most difficult to detect, because they do not produce any error messages. Even the program might run successfully, but the desired output may be compromised. For example, if an instruction such as 'X = Y * Z' is needed to multiply two numbers, but has been coded as 'X = Y + Z', then the program will run successfully but the output produced will not be correct. To remove logical errors, the developer has to 'walk through' the entire logic again, testing each process in the way.

Debugging and testing are often carried out in tandem. For example, as a fIrst debugging exercise, we might scan through a newly written program to remove obvious typographical errors. Then, a first test of the program will likely yield incorrect answers, indicating the presence of less obvious errors. Hence, the debugging process must be carried out in greater depth followed by a re-testing

of the program. This two-part process should then be repeated until we are convinced that the program is reliable. Remember testing of a program must be done to the extremes, that is, all sorts of data (normal, unusual, and unexpected) should be checked so that the program can struggle through all the hardships.

2.1.5.1 Testing Approaches

Testing is a process of analyzing a software item to detect the differences between existing and required conditions and to evaluate the features of the software items. Normally, the testing and debugging process takes as much as half of the program development schedule. Even if utmost care has been taken while analyzing, designing and coding the program, some effort must be done to verify that there are no errors. On the other hand, if an imperfect job was done, then the testing becomes iterative, that is, the first round of testing exposes the presence of errors and subsequent rounds of testing checks if the corrected program is now operating correctly. Hence, a thorough and comprehensive set of test cases, based on the essential model and user implementation model of the system should be initiated.

Usually, two types of approaches are applied to test a program:

1. **Black Box Approach:** Black box testing does not explicitly require the knowledge of internal code and structure. It is usually described as focusing on testing functional requirements, external specifications or interface specifications of the program. The ultimate goal for black box testing is to test every possible combination and value of input.
2. **White Box Approach:** White box testing allows one to peek inside the 'box', and it focuses specifically on using internal knowledge of the program like its logic to guide the selection of test data. This type of testing is initiated to test the code or its logic with little or no regard to its specifications. The ultimate goal is to test every execution path through the program logic.

Black box testing: The *black box* testing method, also known as *functional testing* is a testing technique where the internal architecture of the item being tested is not necessarily known by the tester (the person responsible for testing the program). This type of testing should not be performed by the developer of the program. Rather it should be performed by a tester who has no prior knowledge about the program. The tester should only know about the inputs and the expected outcomes. He should not know how the program arrives at those outputs. For this reason, the tester and the programmer can be independent of

one another, avoiding programmer bias toward his own work.

Advantages of black box testing:

- The test is unbiased because the designer and the tester are independent of each other. The test is done from the point of view of the user, not the designer.
- Due to the nature of black box testing, the test planning can begin as soon as the specifications are written.
- The tester does not need knowledge of any specific programming languages.
- It is more effective on larger units of code than white box testing.
- It helps in exposing any ambiguities or inconsistencies in the specifications.
- The assumptions made while coding does not propagate to test data and the test data need not be changed when code is changed.

Disadvantages of black box testing

- Testing every possible input stream is unrealistic because it would take inordinate amount of time. Therefore, many program paths may go untested.
- Without clear and concise specifications, black box test cases are hard to design.
- There may be unnecessary repetition of test inputs if the tester is not informed of test cases the programmer has already tried.
- This type of testing cannot be directed toward specific segments of code, which may be very complex (and therefore more error prone).

White box testing Also known as *glass box, structural, clear box* and *open box* testing, *white box* testing is one of the most popular testing approaches. In this testing technique, explicit knowledge of the internal workings of the item being tested are used to select the test data. Unlike black box testing, white box testing uses specific knowledge of programming code to examine outputs. The tester should read and use the source code of the applications being tested. He must apply test cases and various inputs in order to test branches, conditions, loops and the logical sequence of statements being executed. To test the program using this approach is beneficial as it helps in finding bugs that are hard to find through normal usage of the application, which usually needs an expert eye.

Advantages of white box testing

- In some cases, testing the code in real environment could be costly, thus white box testing provides a good bug detecting approach before giving the software to the customer.
- It forces test developer to reason carefully about implementation.
- In some cases, the developer is asked to test his own code using a white box test approach. This is good since the best person to know the code is the one who actually wrote it.

Disadvantages of white box testing

- White box testing requires a considerable time to finish; it is somehow a slow approach.
- It also requires a good programmer to do it; a regular user will not be able to do the testing since it requires familiarity with the programming language used.
- The developer will always have the same perspective towards the approach of solving the problem, thus, the special cases that he missed when actually building the program will probably stay unnoticed.

2.1.6 Program Documentation

Documentation is an integral part of the program development process. The programming process is incomplete if it is not properly documented. ***Documentation* refers to all written material that helps to explain the use and maintenance of a system or program.** Proper documentation enables the user to understand the program and acts as a reference tool for programmers, in case modifications are required. It is easier to understand the logic of a program, from the documented records rather than the code. Most programmers spend more of their working time modifying old code than they do writing new code. If a program is difficult to change, it will be clumsily modified, which may result in an unreliable application. Hence, it is essential to make the programs easy to modify and this is eased considerably by documentation. Documentation also acts as a communication medium between members of the development team. It provides information to management to help them plan, budget, and schedule the software development process.

Every stage of program development involves documentation. However, the process of documentation never ends throughout the life of the program. It has to be carried out at regular intervals as and when the program is modified during its maintenance phase. Documentation involves collecting, organising,

storing and maintaining a complete record of programs and other documents used or prepared during various phases of program development. At the time of implementation, the programmer should supply all the necessary documents to the end user. Without the documentation, it would be very difficult for the user to use the program efficiently. If he does not understand how to use the program, then it will defeat the purpose of program development.

2.1.6.1 Types of Documentation

Earlier we have discussed that documentation is useful for the developer as well as the user. The user can use the supplied documents to understand the functionality of the application and the developer can use it for reference purposes. On this basis, we can classify the documentation into two categories:

- For the user
- For the developers

Documentation for the user

Generally, programming is initiated to solve a problem for the end user. Usually, the end user has nothing to do with how the program was created but he is interested in how it can be used to his advantage. Since the user does not know the internal details of the program, he must be provided with proper documents so that he can use the application efficiently. Some of useful documenting techniques from the user's point of view are discussed under following sub-sections.

1. **System manuals:** A system manual contains the detailed technical information for each application system. It describes each of the major functions performed by the application. These manuals usually include:
    1. General system narrative describing the purpose of the application.
    2. Application data describing all files, databases and records used within the application.
    3. Screen displays such as screen prints or their equivalent, displaying the format of each screen and their purposes.
    4. A sample of each report to show their format along with their purposes. The security classification of the report and the information concerning sequences, page breaks and summary.
2. **User manuals:** User manuals assist the person to understand the proper use of the system. It provides detailed instructions concerning the procedure(s) to be followed to enter data into the application for processing, as well as to request nonrecurring reports and other

information. The contents and meaning of each field or data element found on each screen, and the contents of each report, will be described in detail. A description of the data contained in the master file or database, which may take the form of a database repository report(s), will also be provided.

3. **Online help:** Online help is now an established media for providing user documents, for example, WinHelp or HTML Help provided by Microsoft. Readers may access help text as a whole document, may use hypertext links to navigate around the text and may have a table of contents and index entries to find particular items of interest. It is often provided as a 'context-sensitive' structure where the users go directly to the relevant topic by pressing a help button or the Fl key. Online documentation can be revised by reissuing the whole document on CD or similar media. Usually updated versions of the help are posted on the company's web site to download for registered users.

Documents for developers

Developers need to keep proper documentation for reference purposes. We know that programming is an ongoing process and most of the applications require modification from time to time. Modifying an application, if it is a complex one, is not very easy. Some of useful documenting techniques from the programmer's point of view are discussed under following sub-sections.

1. **Comments:** Comments are used within a program to assist the reader in understanding the function or purpose of a single or multiple instructions. Note that although comments are incorporated in the program itself, they are not the part of actual code. They are just there to guide the reader and not for executing any process.
2. **Visual appearance of code:** The readability of a program is greatly improved by the consistent use of indents. When indenting is carried out consistently, it becomes immediately clear where each structure begins and ends. Apart from the indentation, variable, process and module should be named appropriately so that their purpose and functionality can be easily understood by simply looking at their names
3. **Programming tools:** Algorithms, flowcharts, and pseudocodes are useful means of program documentation. They help in making the program logic easier to understand. In case, the program needs certain modification, the programmer can fall back on these tools.
4. **Various reports:** Programming involves many steps. In all these steps,

certain reports are generated such as error reports, maintenance reports and testing reports. These reports are particularly useful when the programmer tries to reconstruct the original program.

*** Importance of software documents**

1. It gives overview of product
2. difficult codes are explained in documents or easy understanding
3. Any other developer can work on previously designed codes
4. Helps in proper communication

2.1.7 Characteristics of a Good Program

Every computer requires appropriate instruction set (programs) to perform the required task. The quality of the processing depends upon the given instructions. If the instructions are improper or incorrect then it is obvious that the result will also be superfluous. Therefore, proper and correct instructions should be provided to the computer so that it can fulfil its duties appropriately. Hence, a program should be developed in such a way that it ensures proper functionality of the computer. Even from the human point of view, a program should be written in such a manner that it is easier to understand the underlying logic. A few important characteristics that a computer program should possess are:

1. **Portability:** Portability refers to the ability of an application to run on different platforms (operating systems) without or with minimal changes. Due to rapid development in hardware and the software, nowadays platform change is a common phenomenon.
2. **Readability**: The program should be written in such a way that it makes other programmers or users follow the logic of the program without much effort. If a program is written structurally, it even helps the programmer himself to follow his own program in a better way if he is away from it for some time.
3. **Efficiency:** Every program requires certain processing time and memory to process the instructions and data. As you must have realised, processing power and memory are the most precious resources of a computer, a program should be laid out in such a manner that it utilises the least amount of memory and processing time.
4. **Structural:** To develop a program, the task must be broken down into a number of subtasks. These subtasks should be developed independently, that is, each subtask should be able to perform the assigned job without the help of any other subtask. If a program is developed structurally, the program not only becomes more readable, but the testing and

documentation process also gets easier.

5. **Flexibility:** A program should be flexible enough to handle most of the changes without having to rewrite the entire program. Most of the programs are developed for a certain period and they require changes from time to time.
6. **Generality:** Apart from flexibility, the program should also be general. By generality, we mean that if a program is developed for a particular task then it should also be used for all similar tasks of the same domain.
7. **Documentation:** Documentation is one of the most important components of an application development. Even if a program is developed following the best programming practices, it will be rendered useless if the end user is not able to fully utilise the functionality of the application. A well-documented application is also useful for programmers because even in the absence of the author, other programmers can understand it.

2.1.8 Summary

A computer needs to be instructed to perform all its tasks. These instructions are provided in the form of computer program. The person who develops the program is known as the *programmer.* Programmers are categorized in two groups: *system programmer* (deals with hardware) and *application programmer* (develops application software). Programming involves many stages such as task analysis, flowcharting, coding, program testing, implementation, documentation and maintenance.

Algorithm is a precise rule (or set of rules), which specifies how to solve some problem. An algorithm must be unambiguous and it should reach a result after a finite number of steps. Algorithms are represented into programs (when the algorithm is represented in a programming language), flowcharts and pseudocodes.

The *flowchart* is a means of visually presenting the flow of data through an information processing system, the operations performed within the system and the sequence in which they are performed. By visualising the process, a flowchart can quickly help identify bottlenecks or inefficiencies where the process can be streamlined or improved.

Pseudocode is a generic way of describing an algorithm without the use of any specific programming language-related notations. It is an outline of a program, written in a form, which can easily be converted into real programming statements. Pseudocode uses plain English statements rather than symbols to represent the processes in a computer program.

Before implementing the program on the user's system, it should be tested thoroughly to ensure the reliability of the application. The process of validating a program is known as *testing,* and the process of detecting, isolating and correcting bugs (errors) in a program is known as *debugging* There are two approaches of testing, namely, *black box testing* and *white box testing.* Using black box testing, the tester is not concerned about the internal behaviour and structure of the program whereas in the case of white box testing, the tester derives test data from an examination of the program's logic and structure.

Documentation refers to all written material that helps to explain the use and maintenance of a system or program. It helps the programmers to keep in touch with the program while the users use the documentation to understand the functionality of the program.

2.1.9 *Review Questions:*

Q.1 Define a flowchart. List some important reasons for using flowcharts. Briefly mention all the steps involved in the program development cycle. What are the advantages and disadvantages of using a pseudocode?

Q2 Define the following:

- Debugging
- Algorithm
- Procedural programming
- Modular programming

Q.3 Define an algorithm. List the characteristics of a good algorithm.

Q.4 What are the advantages and disadvantages of

- Black Box Testing
- White Box Testing

Q.5 Explain program development.

Q.6 Discuss the structure of a flowchart? What guidelines should be followed while making a flowchart?

Q.7 Discuss the two types of errors. Explain the various approaches which should be followed to correct these errors.

Q.8 Why is documentation necessary in the development of program? Explain various documenting techniques which are useful for programmers and users.

Q.9 Explain the characteristics of a good program.

2.1.10 *Suggested Readings:*

1. Computer Fundamentals By Pradeep K. Sinha and Priti Sinha (BPB Publications)
2. Fundamentals of Information Technology By Shiv Kumar Anand and Harmohan Sharma (Kalyani Publishers)
3. "Fundamentals of Computers", by V.Rajaraman, PHI, New Delhi

Lesson No. 2.2 **Author : Arun Sharma**

COMPUTER LANGUAGES

Chapter Outline:

2.2.0 Objectives

- Learn concept of Computer Languages
- Types of Computer Languages
- Familiarized with working of Language Translators, namely Assembler, Compiler and Interpreter
- Good Programming Features

We, as human beings, use natural languages such as English, Hindi, French or Spanish to communicate. Similarly, a user communicates with the computer in a language understood by it. We also know that a computer cannot think. It needs to be instructed to perform all the tasks. The instructions are provided in the form of computer programs, which are developed by a programmer using a programming language. **The language, which the user employs to interact with computer, is known as *computer or programming language*. The process of using such languages to instruct the computer is known as *programming* or *coding*.**

A programming language consists of a set of characters, symbols and usage rules that allow the user to communicate with computers. A

programming language has to follow syntax rules to create an accurate program so that the communication with the computer can yield desired results.

2.2.1 Classification of Programming Languages

You would be amazed to know that computers understand only one language and that is binary language or the language of 0s and 1s. **Binary language is also known as *machine* or *low-level language.*** Although these programs were easily understood by the computer, it proved too difficult for a normal human being to remember all the instructions in the form of 0s and 1s. Therefore, the computer remained a mystery to a common man until other languages such as assembly and high-level languages were developed which were easier to learn and understand. These languages use commands that have some degree of similarity with English (such as 'if else', 'exit'). Programming languages can be divided into three major categories: *machine language, assembly language* and *high-level languages.*

1. **Machine Language:** Machine language is the native language of computers. It uses only 0s and 1s to represent data and the instructions written in this language, consist of series of 0s and 1s.
2. **Assembly Language:** Assembly language correspondences symbolic instructions and executable machine codes and was created to use letters instead of 0s and 1s to run a machine.
3. **High-level Language:** These languages are written using a set of words and symbols following some rules similar to a natural language such as English. The programs written in high-level languages are known as *source programs* and these programs are converted into machine-readable form by using compilers or interpreters.

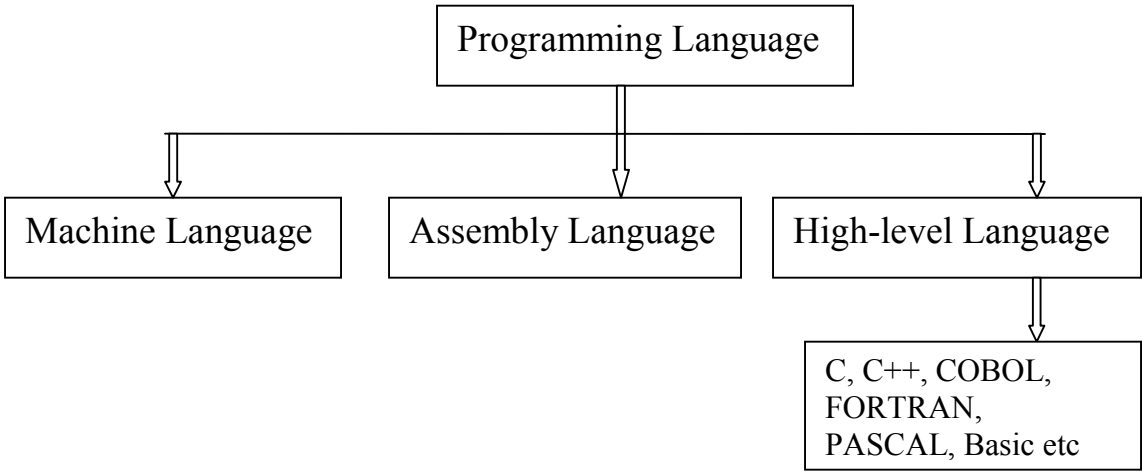


Figure 2.2.1 Types of Programming Languages

Note: *Together, machine and assembly language are also known as low-level languages.*

2.2.2 Generations of Programming Languages

Since early 1950s, programming languages have evolved tremendously. This evolution has resulted in the development of hundreds of different languages. With each passing year, the languages become user-friendly and more powerful than their predecessors. We can illustrate the development of all the languages in five generations.

2.2.2.1 First Generation: Machine Language

The first language was binary, also known as machine language, which was used in the earliest computers and machines. We know that computers are digital devices, which have only two states, ON and OFF (1 and 0). Hence, computers can understand only two binary codes, that is, 1 and 0. Therefore, every instruction and data should be written using 0s and 1s. Machine language is also known as the computer's 'native' language because this system of codes is directly understood by the computer.

OPCODE (Operation Code)	OPERAND (Memory Location)

Figure 2.2.2 Machine Language Instruction Format

Usually, instruction in machine language consists of two-part form as

shown in Figure 2.2.2. The first part is the command or an operation, which tells the computer what functions are to be performed. The second part of the instruction is the operand, which tells the computer where to find or store the data on which the desired operation is to be performed.

Advantages of machine language : Even though machine language is not a human friendly language, it offers certain advantages as listed below:

- **Translation Free**: Machine language is the only language that computers can directly execute without the need for conversion. In fact, it is the only language that computer is able to understand. Even an application using high-level languages, has to be converted into machine-readable form so that the computer can understand the instructions.
- **High Speed:** Since no conversion is needed, the applications developed using machine language are extremely fast..

Disadvantages of machine language : There are many disadvantages in using machine language to develop programs:

- **Machine Dependent**: Every computer type differs from the other, based on its architecture. Hence, an application developed for a particular type of computer may not run on the other type of computer.
- **Complex Language:** Machine language is very difficult to read and write. Since all the data and instructions must be converted to binary code, it is almost impossible to remember the instructions.
- **Error Prone:** Since the programmer has to remember all the opcodes and the memory locations, it is bound to be error prone.
- **Tedious:** Machine language poses real problems while modifying and correcting a program. Sometimes the programming becomes too complex to modify and the programmer has to re-program the entire logic again. Therefore, it is very tedious and time-consuming and since time is a precious commodity, programming using the machine language tends to be costly.

Due to its overwhelming limitations, machine language is rarely used nowadays.

2.2.2.2 Second Generation: Assembly Language

The complexities of machine language led to the search of another language called assembly language. It was developed in the early 1950s and its main developer was IBM.

Assembly language allows the programmer to interact directly with the hardware. This language assigns a mnemonic code to each machine language instruction to

make it easier to remember or write.

An assembly language provides a mnemonic instruction, usually three letters long, corresponding to each machine instruction. The letters are usually abbreviated indicating what the instruction does. For example, *ADD* is used to perform an addition operation, *MULT* for multiplication, etc. Assembly languages make it easier for humans to remember how to write instructions to the computer, but an assembly language is still a representation of the computer's native instruction set. Since each type of computer uses a different native instruction set, assembly languages cannot be standardised from one machine to another and instructions from one computer cannot be expected to work on another.

Assembler: Assembly language is nothing more than a symbolic representation of machine code, which also allows symbolic designation of memory locations. However, no matter how close assembly language is to machine code, the computer still cannot understand it. **The assembly language program must be translated into machine code by a separate program called an *assembler***. The assembler program recognises the character strings that make up the symbolic names of the various machine operations and substitutes the required machine code for each instruction. At the same time, it also calculates the required address in memory for each symbolic name of a memory location and substitutes those addresses for the names resulting in a machine language program that can run on its own at any time. In short, an assembler converts the assembly codes into binary codes and then it assembles the machine understandable code into the main memory of the computer, making it ready for execution.

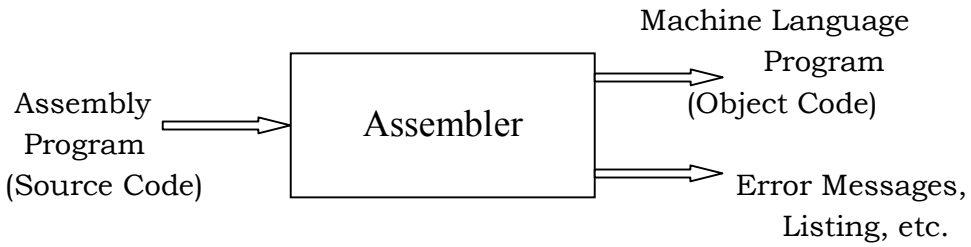


Figure 2.2.3 Working of an Assembler

The original assembly language program is also known as the *source code,* while the final machine language program is designated the *object code*. The functions of an assembler are given below:

1. It allows the programmer to use mnemonics while writing source code programs, which are easier to read and follow.
2. It allows the variables to be represented by symbolic names, not as

memory locations.

3. It translates mnemonic operations codes to machine code and corresponding register addresses to system addresses.
4. It checks the syntax of the assembly program and generates diagnostic messages on syntax errors.
5. It assembles all the instructions in the main memory for execution.
6. In case of large assembly programs, it also provides linking facility among the subroutines.
7. It facilitates the generation of output on required output medium.

Advantages of assembly language: The advantages of using assembly language to develop a program are:

- **Easy to Understand and Use:** The programs written in assembly language are much more easier to understand and use as compared to its machine language. Assembly programs are easier to modify.
- **Less Error Prone:** Since mnemonic codes and symbolic addresses are used, the programmer did not have to keep track of the storage locations of the information and instructions. Hence, there are bound to be less errors while writing an assembly language program.
- **Efficiency:** Assembly programs can run much faster and use less memory and other resources than a similar program written in a high-level language. Apart from speed, the memory requirement of a program is usually smaller than a similar program written in high-level language.

Disadvantages of assembly language: The disadvantages in using assembly language to develop a program are:

- **Machine Dependent:** Different computer architectures have their own machine and assembly languages, which means that programs written in these languages are not portable to other incompatible systems. This makes it a low-level language.
- **Harder to Learn:** The source code for an assembly language is cryptic and in a very low machine-specific form. Being a machine-dependent language, every type of computer architecture requires different assembly language, making it nearly impossible for a program to remember and understand every dialect of assembly.
- **Slow Development Time:** Even with highly skilled programmers, assembly generated applications are slower to develop as compared to high-level language based applications. The development time can be 10 to 100 times as compared to high-level language generated application.

- **Less Efficient:** A program written in assembly language is less efficient as compared to an equivalent machine language program because every assembly instruction has to be converted into machine language. Therefore, the execution of assembly language program takes more time than its equivalent machine language program
- **No Standardisation:** Assembly languages cannot be standardised because each type of computer has a different instruction set and therefore, a different assembly language.

2.2.2.3 Third Generation: High-level Language

Since assembly language required deep knowledge of computer architecture, it demanded programming as well as hardware skills to use computers. Due to computer's widespread usage, early 1960s saw the emergence of the third generation programming languages (3GL). Languages such as COBOL, FORTRAN, BASIC and C are examples of 3GLs and are considered high-level languages.

High-level languages are vague because they are similar to English language. In addition, programs written using these languages can be machine independent. A single high-level statement can substitute several instructions in machine or assembly language. Unlike assembly and machine programs, high level programs may be used with different types of computers with little or no modification, thus reducing the re-programming time and expense.

Translating high-level language to machine language: Since computers understand only machine language, it is necessary to convert the high-level language programs into machine language codes. This is achieved by using language translators or language processors, generally known as compilers, interpreters or other routines that accepts statements in one language and produces equivalent statements in another language.

Compiler: A compiler is a kind of translator that translates a program into another program, known as *target language.* Usually, the term compiler is used for language translator of high-level language into machine language. The compiler replaces single high-level statement with a series of machine language instruction. A compiler usually resides on a disk or other storage media. When a program is to be compiled, its compiler is loaded into main memory. The compiler stores the entire high-level program, scans it and translates the whole program into an equivalent machine language program. During the translation process, the compiler reads the source program and checks the syntax (grammatical) errors. If there is any error, the compiler generates an error message, which is usually displayed on the screen.

In case of errors, the compiler will not create the object code until all the errors are rectified.

Once the program has been compiled, the resulting machine code is saved separately and can be run on its own at any time, that is, once the object code is generated, there is no need for the actual source code. However, if the source code is modified then it is necessary to recompile the program again to affect the changes.

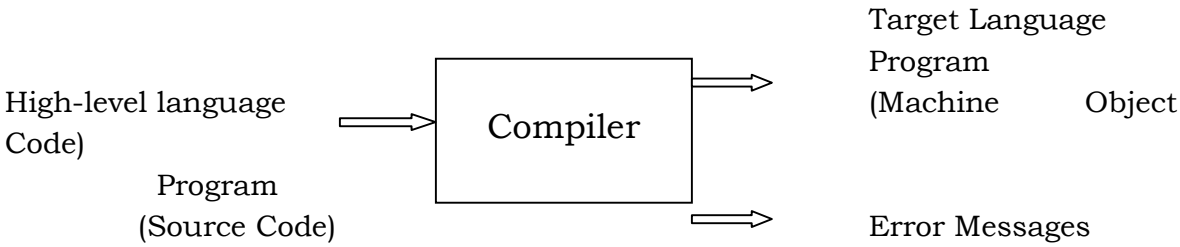


Figure 2.2.4 Working of a Compiler

Note: *For each high-level language, a separate compiler is required. For example, a compiler for* C *language cannot translate a program written in FORTRAN. Hence, to execute both language programs, the host computer must have the compilers of both languages.*

Interpreter: An interpreter is also a language translator and translates high-level language into machine language. However, unlike compilers, it translates a statement in a program and executes the statement immediately, that is, before translating the next source language statement. When an error is encountered in the program, the execution of the program is halted and an error message is displayed. Similar to compilers, every interpreted language such as BASIC and LISP has its own interpreters.

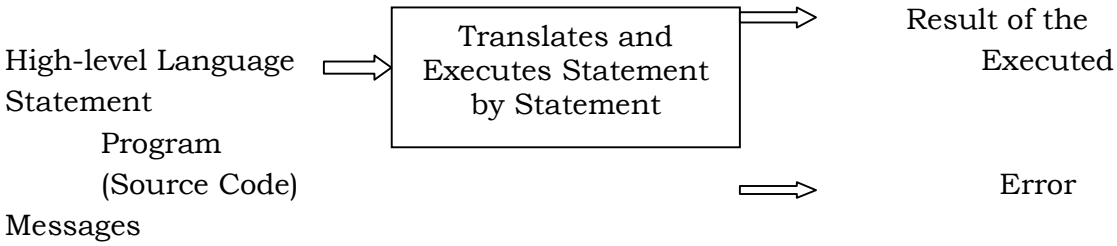


Figure 2.2.5 Working of an Interpreter

There are fundamental similarities in the functioning of interpreter and compiler. However, there are certain dissimilarities also, as given in Table 2.2.2.

Table 2.2.2 Differences between Compiler and Interpreter

Basis	Compiler	Interpreter
Object Code	A compiler provides is a separate object program.	An interpreter does not generate a permanent saved object code file.
Translation Process	A compiler converts the entire program into machine code at one go.	An interpreter translates the source code line-wise, that is, it would execute the current statement before translating the next statement.
Debugging Ease	Removal of errors (debugging) is slow.	Debugging becomes easier because the errors are pointed out immediately.
Implementation	By nature, compilers are complex programs. Hence, they require hard-core coding. They also require more memory to execute a program.	Interpreters are easier to write because they are less complex programs. They also require less memory for program execution.
Execution Time	Compilers are faster as compared to interpreters because each statement is translated only once and saved in object file, which can be executed anytime without translating again.	Interpreters are slower as compared to compilers because each statement is translated every time it is executed from the source program.

Nowadays, many languages use a hybrid translator having the characteristics of compiler as well as interpreter. In such a case, the program is developed and debugged with the help of interpreter and when the program becomes bug free, the compiler is used to compile it.

Linker : An application usually consists of hundreds, thousands or even millions of lines of codes. The codes are divided into logical groups and stored in different modules so that the debugging and maintenance of the codes becomes easier. When a program is broken into several modules, each source program (module) can be modified and compiled independently. In such a case, these independent source programs have to be linked together to create a complete application. This job is done by a tool known as *linker.*

Loader : *Loaders* are a part of the operating system that brings an

executable file residing on disk into memory and starts it running. It is responsible for loading, linking, and relocation.

A. loader performs four basic tasks as follows:

1. **Allocation:** It allocates memory space for the programs.
2. **Linking:** It combines two or more separate object programs and supplies the information needed to allow references between them.
3. **Relocation:** It prepares a program to execute properly from its storage area.
4. **Loading:** It places data and machine instructions into the memory.

Advantages of high-level languages: High-level languages (HLL) are useful in developing complex software. Unlike assembly language, the programmer does not need to learn the instruction set of each computer being worked with. The various advantages of using high-level languages are discussed below:

- **Readability:** Since high-level languages are closer to natural languages, they are easier to learn and understand. In addition, a programmer does not need to be aware of computer architecture; even a common man can use it without much difficulty. This is the main reason of HLL's popularity.
- **Machine Independent:** High-level languages are machine independent in the sense that a program created using HLL can be used on different platforms with very little or no change at all.
- **Easy Debugging:** High-level languages include the support for ideas of abstraction so that programmers can concentrate on finding the solution to the problem rapidly, rather than on low level details of data representation, which results in fewer errors. Moreover, the compilers and interpreters are designed in such a way that they detect and point out the errors instantaneously. Hence, the programs are free from all syntax errors.
- **Easier to Maintain:** As compared to low-level languages, the programs written in HLL are easily modifiable because HLL programs are easier to understand.
- **Low Development Cost:** High-level languages permit faster development of programs. Although a high-level program may not be as efficient as an equivalent low-level program but the savings in programmer time generally outweighs the inefficiencies of the application. This is because the cost of writing a program is nearly constant for each line of code, regardless of the language. Thus, a high-level language, where each line of code translates to 10 machine instructions, costs only a fraction as compared to program

developed in a low-level language.

- **Easy Documentation:** Since the statements written in HLL are similar to natural languages, they can be easily understood by human beings. As a result, the code is obvious, that is, there is few or no need for comments to be inserted in programs.

Disadvantages of high-level languages: There are two main disadvantages of high-level languages.

- **Poor Control on Hardware:** High-level languages are developed to ease the pressure on programmers so that they do not have to know the intricacies of hardware. As a result, sometimes the applications written in high-level languages cannot completely harness the total power available at hardware level.
- **Less Efficient:** The HLL applications are less efficient as far as computation time is concerned. This is because, unlike low-level languages, high-level languages must be created and sent through another processing program known as a compiler. This process of translation increases the execution time of an application. Programs written in high-level languages take more time to execute and require more memory space. Hence, critical applications are generally written in low-level languages.

Some popular high-level languages: Although a number of languages evolved in the last five decades, only few languages were considered worthwhile to be marketed as commercial products. In the next section, we will discuss the commonly used high-level languages.

- **FORTRAN:** FORTRAN, or FORmula TRANslator, was developed by John Backus for IBM 704 mainframes in 1957. FORTRAN gained immense popularity as compared to any of its counterpart and is still extensively used to solve scientific and engineering problems. A compiler for FORTRAN-I was released in 1957 including many features like arithmetic if variables with implicit data types and passing subgroups as parameters to other subprograms. In 1966 and 1977, ANSI (American National Standard Institute) versions of FORTRAN were introduced named FORTRAN-I and FORTRAN 77, respectively.

  The main feature of FORTRAN is that it can handle complex numbers very easily. However, the syntax of FORTRAN is very rigid. A FORTRAN program is divided into sub-programs, each sub-program is treated as a separate unit, and they are compiled separately. The compiled programs are linked together

at load time to make a complete application. It is not suitable for a large amount of data as well and, hence, it is not often used for business applications.

- **COBOL:** COBOL or COmmon Business Oriented Language, has evolved after many design revisions. Grace Murray Hopper, on behalf of US Defense was involved in the development of COBOL as a language. The first version was released in 1960 and later revised in 1974 and 1984. COBOL was standardised with revisions by ANSI in 1968.

  COBOL is considered a robust language for the description of Input/Output formats. It could cope with large volumes of data. Due to its similarity with English, COBOL programs are easy to read and write. Since, it uses English words rather than short abbreviations, the instructions are self-documenting and self-explanatory. However, due to its large vocabulary, the programs created using COBOL are difficult to translate. COBOL helped companies to perform accounting work more effectively and efficiently. Even today COBOL programs are used on mainframes.
- **BASIC:** BASIC or Beginner's All-Purpose Symbolic Instruction Code, was developed by John Kemeny and Thomas Kurtz at Dartmouth College in the year 1960. It was the first interpreted language made available for general use. It was standardised by ANSI in the year 1978. Presently many advanced versions of BASIC are available and used in a variety of fields as business, science and engineering.

  BASIC programs were traditionally interpreted. This meant that each line of code had to be translated as the program was running. BASIC programs, therefore, ran slower than FORTRAN programs. In BASIC program, each statement is prefixed by a line number, which serves a dual purpose - to provide a label for every statement and to identify the sequence in which the statements will be executed. BASIC is easy to learn as it uses common English words; therefore, it is a good language for beginners to learn their initial programming skills.
- **PASCAL:** Named after Blaise Pascal, a French philosopher, mathematician and physicist, PASCAL was specifically designed as a teaching language. This language was developed by Niklaus Wirth at the Federal Institute of Technology of Zurich in early 1970s.

  PASCAL is a highly structured language, which forces programmers to design programs very carefully. Its object was to force the student to

correctly learn the techniques and requirements of structured programming. PASCAL was designed to be platform-independent, that is, a PASCAL program could be compiled on any computer and the result would run correctly on any other computer.

- **C:** C was initially developed as an experimental language called A. Later on, it was improved, corrected, and expanded until it was called B. This language, in turn was improved, upgraded and debugged and was finally called C. C was developed by Dennis Ritchie at Bell Labs in the mid-1970s. C can be thought of as a 'low-level' high-level language because of its excellent reach to low-level hardware components. It was originally designed for systems programming. The Unix operating system, for example, was written in C. However, it can also be used for applications programming. The compact nature of the compiled code, plus its speed, made it useful for early PC applications.

  C consists of rich collection of standard functions useful for managing system resources. It is flexible, efficient, and easily available. Having syntax close to English words, it is an easy language to learn and use. The applications generated using C are portable, that is, the programs written in C can be executed on multiple platforms. C works on a data structure, which allows a simple data storage. It has the concept of pointers, the memory addresses of variables and files.
- **C++:** This language was developed by Bjarne Stroustrup in early 1980s. It is the superset of C and supports object oriented features. This language is used effectively in developing system software as well as application software. As it was an extension of C, C++ maintained the efficiency of C and added the power of inheritance. C++ works on classes and objects as a backbone of object oriented programming. Being a superset of C, it is an extremely powerful and efficient language. However, C++ is much harder to learn and understand than its predecessor C.

  Salient features of C++ are:
  - Strongly typed.
  - Case-sensitive.
  - Compiled and faster to execute.
- **Java:** This language was developed by Sun Microsystems of USA in 1991. It was originally called 'Oak'. Java was designed for the development of software for consumer electronic devices. As a result, Java came out to be a simple, reliable, portable, and powerful language. This language truly

implements all the object-oriented features. Java was developed for Internet and contributed a lot to its development. It handled certain issues like portability, security, networking and compatibility with various operating systems. It is immensely popular on web and is used for creating scientific and business applications.

Features of Java are:

- Simple and robust language.
- Secured and safe.
- Truly object-oriented language.
- Portable and platform independent.
- Multithreaded, distributed, dynamic, and architecture neutral.

- **PROLOG:** PROLOG is derived from PROgramming in LOGic. PROLOG was developed in Marseilles (France), in 1972 and implemented the same year using ALGOL W compiler. The development work related to PROLOG started in 1970 by Alain Coulmeraurer and Philippe Roussel. The aim was to develop a language for making deductions from text. PROLOG is the major language in the logic programming languages category. Prolog is not considered a general programming language.

  Features of PROLOG are:

  - Successful language for solving database related query processing.
  - Simple syntax and semantics.
  - Supports the use of built-in functions.
  - Facts, rules and queries are the basic components of PROLOG. Allocation of values to variables is simplified.
  - Does not provide solutions to general-purpose problems so that it cannot be considered a general-purpose programming language.
  - Does not support abstraction.
  - If the relationship between database and mathematical relations is not explicitly represented then the implementation becomes difficult.

- **LISP:** In 1950s some efforts were made to include list processing with IPL, that is, Information Processing Language. In 1958, John McCarthy working for IBM on an algebraic expression differentiator project felt the need of a list processing. A system should be capable of handling dynamically allocated lists as well as de-allocating them. The concept was implemented by McCarthy by combining best features of FORTRAN and IPL and LISP came into existence. LISP or List Processing is a functional

programming language based on recursive data structure known as a list. In a functional programming language, computations are purely carried out through the application of functions.

LISP is different from most other languages in numerous ways. One of the important differences is that LISP can execute data structures as programs and the program itself can be used as data. It forces a user to think of the equivalence of programs and data. The other important feature is too heavily bent on recursion as a control structure in LISP. In LISP, each value is either an atom or a list.

Features of LISP are:

- Permits the use of recursion, that is, a program can call itself.
- Readily writable because of the regularity of syntax and easy usability.
- Supports garbage collection, that is, de-allocating the memory to variable not in use.
- Works on interaction, that is, user can intermix program writing, compilation, testing, debugging, and running a single session.
- The core language of AI (Artificial Intelligence).
- Not considered a reliable language.
- The programs written in LISP lack in readability.
- Cannot be used as a general purpose programming language

2.2.2.4 Fourth Generation: 4GL

Fourth generation languages (4GLs) have simple, English-like syntax rules, commonly used to access databases. 4GLs are non-procedural languages. The non-procedural method is simply to state the needed output instead of specifying each step one after another to perform a task. In other words, the computer is instructed WHAT it must do rather than HOW a computer must perform a task.

The non-procedural method is easier to write, but has less control over how each task is actually performed. When using non-procedural languages, the methods used and the order in which each task is carried out is left to the language itself; the user does not have any control over it. In addition, 4GLs sacrifice computer efficiency in order to make programs easier to write. Hence, they require more computer power and processing time. However, with the increase in power and speed of hardware and with diminishing costs, the use of 4GLs have spread.

Fourth generation languages have a minimum number of syntax rules. Hence, people who have not been trained as programmers can also use such languages to write application programs. This saves time and allows professional

programmers for more complex tasks. The 4GLs are divided into three categories:

1. **Query Languages**: They allow the user to retrieve information from databases by following simple syntax rules.
2. **Report Generators:** They produce customised reports using data stored in a database. The user specifies the data to be in the report, the report's format and whether any subtotals and totals are needed.
3. **Application Generators:** With application generators, the user writes programs to allow data to be entered into the database. The program prompts the user to enter the needed data. It also checks the data for validity.

Advantages of 4GLs : The main advantage of 4GLs is that a user can create an application in a much shorter time for development and debugging than with other programming languages. The programmer is only interested in what has to be done and that too at a very high level. Being nonprocedural in nature, it does not require the programmers to provide the logic to perform a task. Therefore, a lot of programming effort is saved as compared to 3GLs.

Disadvantages of 4GLs : Since programs written in a 4GL are quite lengthy, they need more disk space and a large memory capacity as compared to 3GLs. These languages are inflexible also because the programmers control over language and resources is limited as compared to other languages. These languages cannot directly utilise the computer power available at hardware level as compared to other levels of languages.

2.2.2.5 Fifth Generation: Very High-level Languages

Fifth generation languages actually is a future concept. They are just the conceptual view of what might be the future of programming languages. These languages will be able to process natural languages. The users will be freed from learning any programming language to communicate with the computers. The programmers may simply type the instruction or simply tell the computer via microphones what it needs to do.

2.2.3 Features of a Good Programming Language

To begin the language selection process, it is important to establish some criteria for what makes a language good. A good language choice should provide a path into the future in a number of important ways.

1. **Ease of use**: The language should be easy writing codes for the programs and executing them. The vocabulary of the language should resemble

English. Simplicity helps in the readability of the language. A simple language is easier to grasp and code.

2. **Portability** Computer languages should be independent of any particular hardware or operating system, that is, programs written on one platform should be able to be tested or transferred to any other computer or platform.
3. **Reliability:** Reliability is concerned with making a system failure free, and thus is concerned with all possible errors. The language should have the support of error detection as well as prevention.
4. **Safety:** The system must always do what is expected and be able to recover from any situation that might lead to a mishap or actual system hazard.
5. **Performance:** By performance, we mean that the language should not only be capable of interacting with the end users, but also with the hardware. Nowadays, the hardware has become very sophisticated and quiet fast. Hence, the application developed using a good language should tap the maximum resources of the available hardware power in terms of speed and memory efficiency.
6. **Cost:** Cost component is a primary concern before deploying a language at a commercial level. It includes several costs such as:
   - Program execution and translation cost.
   - Program creation, testing and use.
   - Program maintenance.
7. **Promote structural programming:** A good language should be capable of supporting structural programming. A structured program also helps programmers to visualise the problem in a logical way, thereby reducing the probability of errors in the code.
8. **Compact code:** A good language should also promote compact coding, that is, the intended operations should be coded in a minimum number of lines.
9. **Maintainability:** Creating an application is not the end of the system development. It should be maintained regularly so that it can be easily modified to satisfy new requirements.
10. **Reusability:** The language should facilitate the adaptation of code for use in other applications. Code is reusable when it is independent of other codes.
11. **Standardisation:** Standardisation means the extent to which the language

definition has been formally standardised (by recognized bodies such as ANSI and ISO) and the extent to which it can be reasonably expected that this standard will be followed in a language translator.

2.2.4 Summary

Programming language is a language which a user employs to interact with the computer. Programming languages can be divided into three major categories, namely *machine language, assembly language* and *high-level language.*

First generation language is the *machine language* in which instructions are in the *form* of 0s and 1s. Second generation language is the assembly language in which mnemonic code is assigned to each machine language instruction to make it easier to remember and write. Third generation languages are closer to the natural languages. They are machine independent and use language translators (compiler, interpreter, and assembler) to translate the high-level code into machine code.

Fourth generation languages, also called as non-procedural languages, use minimum syntax rules and are categorised into query languages, report generators, and application generators. Fifth generation language is in the development stage. The computer will be able to accept, interpret and execute the instruction in the natural language of the user.

FORTRAN language was developed by John Backus in mid 1950s to solve scientific an engineering problems. *COBOL* language was developed for use in business and accounting applications. This language is robust and mainly runs on the mainframe computers. C language was developed in Bell Labs for system programming. It is portable, flexible, and have collection of built-in functions to make the language powerful. *C++* language was developed in early 1980s to enhance the features of existing C language. It includes object-oriented features, that is, classes, polymorphism and inheritance to solve the real world problems. Sun Microsystems developed Java in 1991 to implement all object-oriented features. Java supports many features, which include portability, platform independence, robustness, and multithreading. *PASCAL* was named after French philosopher, Blaise Pascal. It was used to design a program in a structured manner. John Kemeny and Thomas Kurtz developed BASIC in 1960 for those who could learn initial programming skills. It was used in the area of business, science, and engineering. *LISP* was designed in 1950s to write artificial intelligence programs. It executes the data structures as program and the program itself can be used as a data. *PROLOG* was developed in Marseilles in 1972. It was used for solving database related query processing.

Glossary :

1.	COBOL	:	Common business-oriented language
2.	FORTRAN	:	Formula Translation
3.	BASIC	:	Beginner's All Purpose Symbolic Instruction Code
4.	ALGOL	:	Algorithmic Language
5.	PROLOG	:	Programming in Logic
6.	LISP	:	List Programming

2.2.5 *Review Questions:*

Q.1 Briefly describe the classification of programming languages?

Q.2 Differentiate between a compiler and interpreter.

Q.3 Write down the salient features of C++.

Q.4 Explain how a high level language is translated into machine language?

Q.5 Explain the features of a good programming language.

Q.6 Discuss the advantages and disadvantages of the following:
a) Machine language b) High-level language

Q.7 Explain in detail about any four popular high-level languages.

2.2.6 *Suggested Readings:*

1. Computer Fundamentals By Pradeep K. Sinha and Priti Sinha (BPB Publications)
2. Fundamentals of Information Technology By Shiv Kumar Anand and Harmohan Sharma (Kalyani Publishers)
3. “Fundamentals of Computers”, V.Rajaraman, PTI, New Delhi.

COMPUTER SOFTWARE

Chapter Outline:

2.3.0 Objectives

- What is software?
- Relation between Software and Hardware
- Learn different Types of Software
- What is System Software?
- Different type of Application Packages
- Concept of Firmware

A computer system consists of ***hardware***, the electronic devices that are capable of computing and manipulating information and *software* (set of instructions) that carry out predefined tasks to complete a given job. As we know, a computer cannot think or perform on its own. It performs operations like addition, subtraction, multiplication and division only when the user instructs it to do so. The user issues instructions and the CPU acts in accordance with the instructions. The sets of instructions, which control the sequence of operations, are known as *programs* and collectively, programs are called ***software***.

It is the combination of physical equipment (hardware) and logical

instructions (software) that gives modem computing systems their power and versatility. In this lesson, we will focus on the logical or intangible part of the computer system, that is, the software.

2.3.1 Software: Definition

Software is a generic term for organised collection of computer data and instructions. It is responsible for controlling, integrating and managing the hardware components of a computer system and to accomplish specific tasks. In other words, the software tells the computer system what to do and how to do it. For example, software instructs the hardware what to display on the user's screen, what kinds of input to take from the user and what kind of output to generate. Thus, software communicates with the hardware, it organises the control sequences and the hardware carries out the instructions defined by the software. Software is an intangible commodity, that is, it is that part of the computer system that user cannot touch. It turns a computer into a valuable device.

As discussed previously, a computer needs to be instructed to perform any task. These instructions are given in the form of *computer programs,* which are written in computer programming languages. A program controls the activity of processing done by the processor and it performs exactly what a program instructs. The moment at which hardware (processor, memory, etc.) is acting as per the instructions of a program, the program is said to be in *running* or *executing* state.

A set of programs, which are specifically written to provide user a precise functionality like solving a specific problem is termed as a *software package.* For example, word processing software package provides functionality to the computer so that it can act like a typewriter for writing text. Similarly, an image processing software package assists a user in drawing and manipulating graphics.

2.3.2 Relationship between Software and Hardware

Software refers to the computer programs that are loaded into a computer system and hardware refers to all the visible devices, which are assembled together to build a computer system. It is the blending of software and hardware that gives life to a computer system. Thus, hardware and software share a special relationship. If the hardware is the 'heart' of a computer system, then the software is its 'soul'. Both are complimentary to each other. An analogy can be taken of a video game system, which comprises a console, games cassettes, joystick and display screen as the hardware. The games in the cassettes can be considered as the software. To play a particular game, firstly, the cassette of

that game has to be loaded on the console and then played.

Figure 2.3.1 Playing Computer Game using Hardware and Software

Similarly, to get a particular job done by a computer, firstly, the relevant software is loaded in the hardware. It is apparent, therefore, that software is vital. Another inference from this analogy is that different software can be used on the same hardware to perform different jobs, just as different games can be played on the same console by using different cassettes.

2.3.3 Software Categories

Software can be broken into two major categories: *System software* that provides the basic non-task-specific functions of the computer and *Application software* utilised by users to accomplish specific tasks. System software is the software that is essential for computer to function. Application software is the additional software that a user chooses to use.

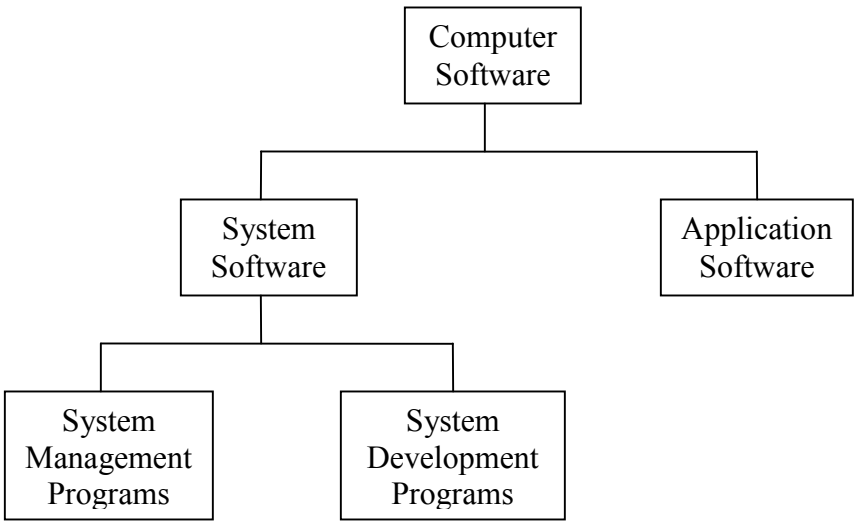


Figure 2.3.2 Software Categories

2.3.4 System Software

Software that contributes to the control and performance of the computer system and permits the user to use the system more conveniently is termed as *System software.* You must have noticed while purchasing a new computer system that it is always accompanied by software, either stored in a floppy or CD, which is supplied by the manufacturer. This software manages and supports the computer system and its information processing activities. System software is transparent and less noticed by a typical user.

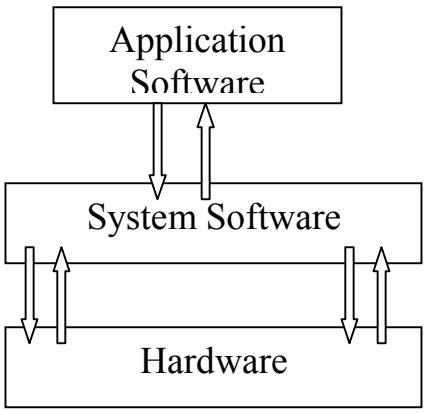


Figure 2.3.3 System Software

System software comprises programs written in low-level languages, which interact with the hardware at a very basic level. They are the basic necessity of a computer system for its proper functioning. System software serves as the interface between hardware and the end users. System software not only controls the hardware but also provides a platform for other programs to run onto them.

Depending upon the functionality, the system software can be further divided into two major categories, namely, system management programs and system development programs.

2.3.4.1 System Management Programs

System management programs, as the name implies, are responsible for the management and accurate functioning of the computer system. It includes an integrated system of programs that manage the operations of the processor, control the Input/Output, manage storage resources and provide various support services as the computer executes application programs. These softwares are commercially available in the market as *operating system, device drivers* and *system utilities.*

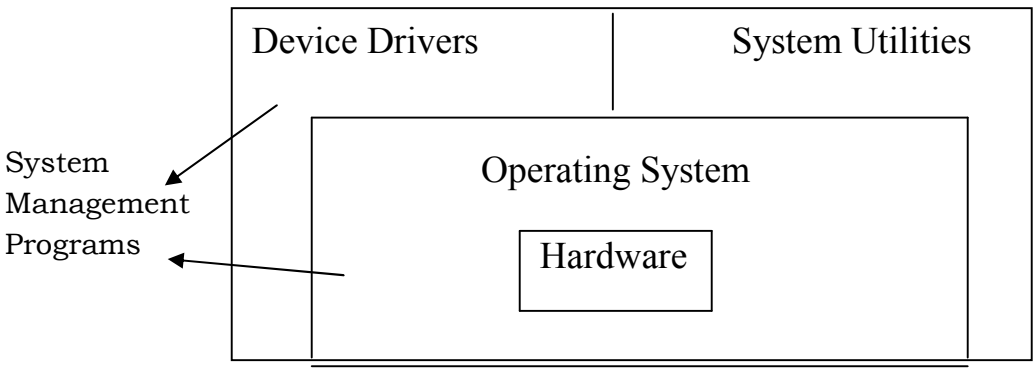


Figure 2.3.4 System Management Programs

1. **Operating System:** The most important type of system software is the operating system. It is responsible for performing basic tasks such as recognising input from the keyboard, sending output to the display screen, keeping track of files and directories on the hard disk and controlling peripheral devices such as printers and modems. In addition, the operating system ensures that different programs executing at the same time do not interfere with each other. It provides a software

platform on top of which other programs can run. In simpler words, the operating system organises and controls the hardware.

The basic functions of an operating system are listed below.

a. It handles the creation, deletion, suspension, resumption, scheduling and synchronisation of processes.

b. It handles allocation and deallocation of memory space as required by various programs.

c. It is responsible for creation and deletion of files and directories. It also organises, stores, retrieves, names and protects all the files.

d. It manages all the devices of the computer system such as printers and modems. If any device fails, it detects the device failure and notifies the same to the user.

e. It protects system resources and information against destruction and unauthorised use.

f. It provides the interface between the user and the hardware.

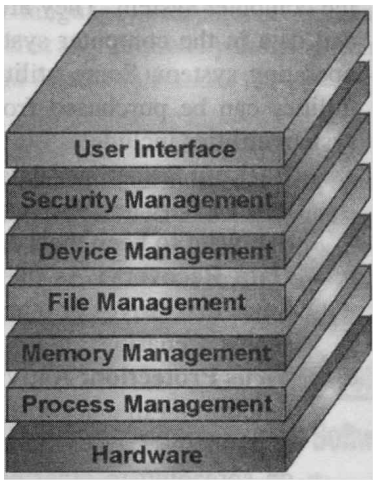


Figure 2.3.5 Operating System

The examples of an operating system are MS-DOS, Microsoft Windows *98/2000/ME/XP,* UNIX, Linux and Mac OS.

2. Device Drivers: Device drivers are system programs, which are responsible for proper functioning of devices. Every device or hardware, whether it is a printer, monitor, mouse or keyboard, has a driver program for support. Whenever a new device is added to the computer system, a new device driver must be installed before the device can be used. In

modern operating systems, most hardware drivers, such as the keyboard driver, comes with the operating system. For other devices like a printer, a user must load the device driver of that particular printer. A driver acts like a translator between the device and programs that use the device. For example, when a user prints a document, the processor issues a set of generic commands to the printer driver and the driver translates those commands into the specialised instructions that the printer understands. Note that each device has its own set of specialised commands that only its driver understands.

Depending upon how data is fetched by the devices, device drivers can be divided into two categories:

a. **Character Device Drivers**: Character device drivers are intended for those devices, which have a simple character based interface such as the keyboard. Such devices do not require any buffering as they send and receive data in single character chunks.

b. **Block Device Drivers:** Block device drivers are intended for those devices that have a data block based interface such as the hard disk. In order to improve speed and efficiency, these devices transfer the whole data buffer at one time.

Note: *Device drivers are not independent programs; they assist and are assisted by the operating system for the proper functioning of the device.*

3. **System Utilities: System utility programs perform day-to-day tasks related to the maintenance of the computer system.** They are used to support, enhance, expand and secure existing programs and data in the computer system. Certain utility programs are usually bundled along with the operating system. Some utility programs are available for free and as per requirement, other utilities can be purchased from third party commercial vendors. Most common functions of system utilities include:

   a. **Backup:** Sometimes data files can get corrupted, or get accidentally deleted. In such a case, data backups become very useful. A backup system utility is essential for those organisations that want to keep their data intact.

   b. **Data Recovery:** As the name implies data recovery programs are used to recover data. Since disk drives or other hardware may fail, these utilities are essential to recover data in such a scenario.

   c. **Virus Protection:** Antivirus programs are essential system software for a computer system functioning in a network. Viruses

are small programs written with malicious intent, which copy themselves to the local system hard drives from other infected systems. Viruses keep on spreading to other computers through the network or exchange of infected floppies. Once installed on a system, Antivirus scans the hard disk for any kind of virus and, if found, removes them. In addition, they monitor the clean (virus free) computer for any activity of viruses.

d. Disk Management: Disk management programs include various system softwares like *de fragmenting disks, data compression software and formatting disk tools.* De-fragmentation implies putting fragments of files in a sequential order onto the disk. This reduces the time to access the file. Data compression programs squeeze out the slack space generated by the formatting schemes. Formatting tools format the hard drive in tracks and sectors for orderly storing of the data in the drive.

2.3.4.2 System Development Programs

System development programs consist of system software, which are associated with the development of computer programs. These program development tools allow programmers to write and construct programs that the operating system can execute.

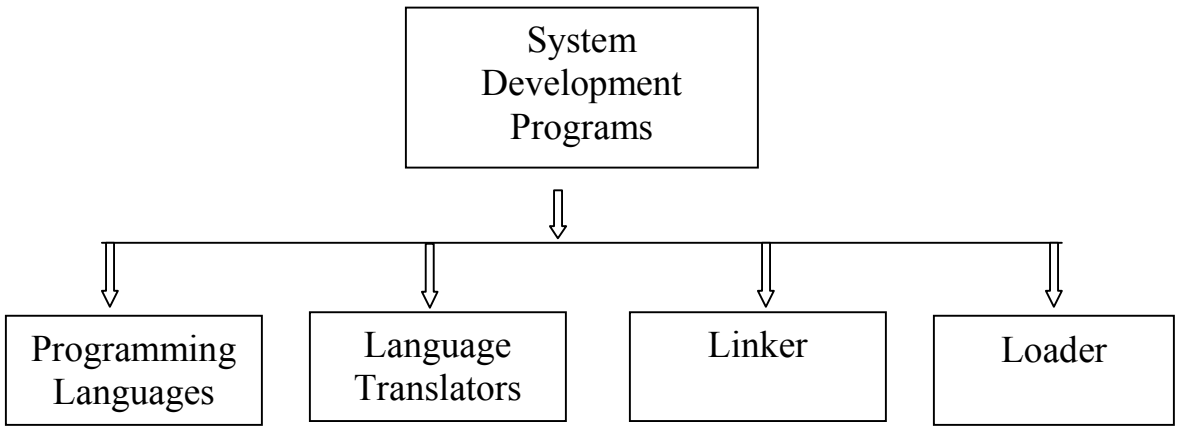


Figure 2.3.6 System Development Programs

A software program is developed to accomplish a particular task. The program developer needs certain tools to built a software, which include an appropriate computer language, translator to translate a particular language to machine language, etc. All the necessary tools that are required by the programmer, namely,

programming languages, language translators, linker and *loader* fall under the category of system development programs.

1. **Programming Languages:** A programming language is a primary interface of a programmer with a computer. It is a set of vocabulary and set of grammatical rules for instructing a computer or computing device to perforce specifically taste. A program is an ordered list of instructions that, when executed, causes the computer to behave in a predetermined manner. A programming language includes a series of commands, which are used in development of software. Programming languages are further divided into three categories, namely, *machine language, assembly language,* and *high-level language.* Hundreds of programming languages have been developed since the invention of a computer system. The success and strength of a programming language is judged with respect to standard features.

   The choice of computer language to be used depends on what kind of program is to be written, for example, a system program or an application program. Choosing a particular language has important consequences; for example, whether it will be easy to write and maintain programs. Figure 2.3.7 illustrates some of the most commonly used languages and what tasks they are usually used for

Languages	Used for
C	System Software
JAVA C++	Application and Components
Fortran Visual Basic Xbase, Powerbuilder	Applications
Java Script Word Basic	Scripts and Macros

Figure 2.3.7 Various Programming Languages

2. **Language Translators:** Computers only understand a language consisting of 0s and 1s called machine language. To ease the burden of programming entirely in 0s and 1s, special programming languages called high-level programming languages were developed that resembled natural languages like English. Therefore, a tool was required which could translate a program written in a programming language

to machine language. Along with every programming language developed, a *language translator* was also developed, which accepted the programs written in a programming language and executed them by transforming them into a form suitable for execution.

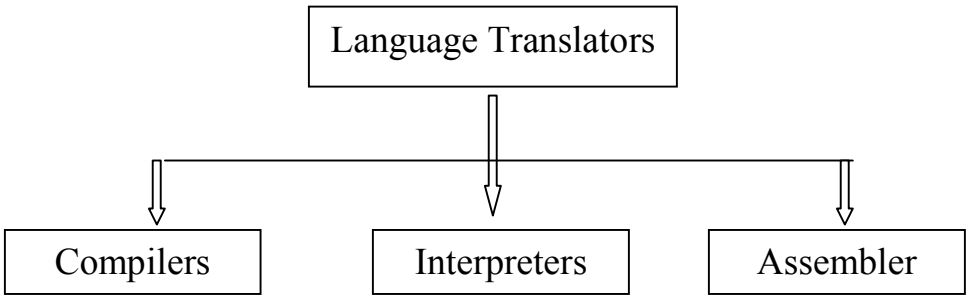


Figure 2.3.8 Language Translators

In other words, language translators help in converting programming languages to machine language, that is, they convert programming statements into the 0s and 1s that the computer is able to process.

Depending on the programming language used, language translators are divided into three major categories:

a. **Compiler:** A compiler is a type of language translator that translates a program code into machine language. The programs written in any programming language needs to be converted to binary form. Therefore, in order to execute the programs, a programmer needs to compile the written programs. As a system program, a compiler translates source code (user written program) into object code (binary form). The compiler looks at the entire piece of source code and reorganises the instructions. Figure 2.3.9 illustrates how a source program is compiled, resulting in an executable program (object code). The user then furnishes the required inputs to get the desired output. Examples: Borland C++ compiler and Microsoft's VC++ compiler

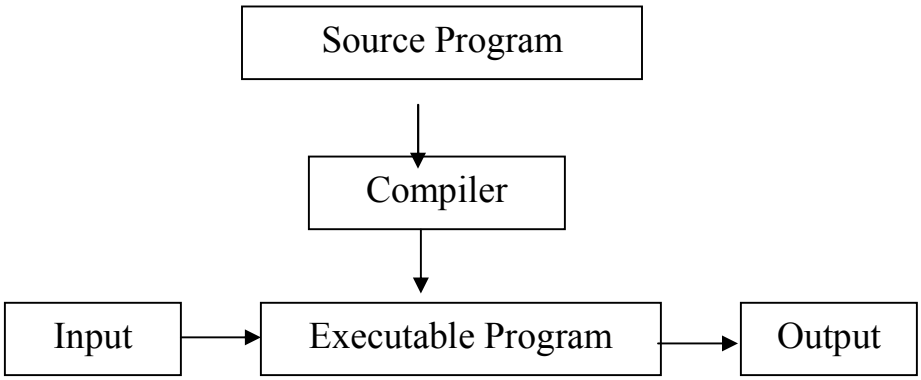


Figure 2.3.9 A Compiler

b. **Interpreter:** An interpreter is another type of language translator, which analyses and executes the source code in line-by-line manner, without looking at the entire program. In other words, an interpreter translates a statement in a program and executes the statement immediately, that is, before translating the next source language statement. The advantage of interpreters is that they can execute a program spontaneously. Compilers require some time before an executable program is made because it looks at the whole source code. However, programs produced by compilers run much faster than the same programs executed by an interpreter.

Figure 2.3.10 illustrates how an interpreter directly interprets the program statements, requiring the user to input data to get the desired output. Examples: GW-BASIC Interpreter and LISP Interpreter

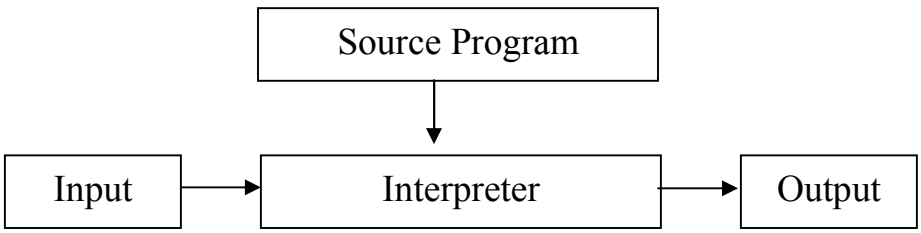


Figure 2.3.10 An Interpreter

c. **Assembler:** Compared to all the types of programming languages, assembly language is closest to the machine code. It is fundamentally a symbolic representation of machine code However, no matter how close assembly language is to machine code, the computer is still unable to understand it. The assembly language program must be translated into

machine code by a separate program called an *assembler.* The assembler program recognises the character strings that make up the symbolic names of the various machine operations and substitutes the required machine code for each instruction. In short, an assembler converts the assembly codes into machine codes, making the assembly program ready for execution

3. **Linker:** A typical software generally comprises hundreds, thousands or even millions of lines of programming statements or code. The code is divided into logical groups and stored in different independent modules so that the debugging and maintenance of the code becomes easier. Before execution, different object codes resulting from the independent modules have to be linked together to create a executable program. This job is performed by a tool known as *linker.* A linker is a system program that links together several object modules and libraries to form a single, coherent, program (executable).
4. **Loader:** Loader is a kind of system software, which is responsible for loading and relocation of the executable program in the main memory. The functions of a loader include assigning load time space for storage, that is, storage allocation and to assist a program to execute appropriately.

2.3.5 Application Software

The most often seen software by a general user is the application software. It is used to accomplish specific tasks rather than just managing a computer system. For a user, without application software, the computer system has no specific use. Application software may consist of a single program, such as a Microsoft's Notepad for writing and editing simple text. It may also consist of a collection of programs, often called a software package, which work together to accomplish a task, such as a spreadsheet package. Application software may also include a larger collection of programs (a software suite), related but independent programs and packages, which have a common user interface or shared data format, such as Microsoft Office suite.

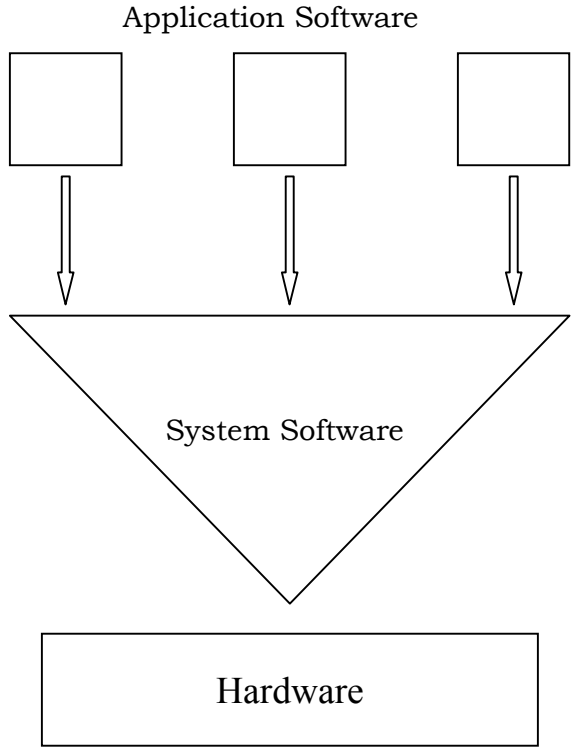


Figure 2.3.11 Relationship of Application and System Software

Application software may be used for a variety of reasons:

- As a business tool.
- To assist with graphics and multimedia projects.
- To support home, personal and educational activities.
- To facilitate communications.
- As aid in entertainment industry

Application software are in turn controlled by system software, which manages hardware devices and performs background tasks for them. Figure 2.3.11 illustrates that application layer executes on the system software layer, which lies on the hardware layer.

Typical types of application software packages include:

Table 2.3.1 Application Software Packages

Type of Software	What Does it Do?	Examples
Word Processors	Virtually all personal computers are equipped with a word processing program, which has the same function as a typewriter for writing letters, reports or other documents and printing.	Microsoft Word, WordPerfect
Spreadsheets	A table containing text and figures, which is used to perform calculations. Spreadsheets are usually used for budgets, statistics etc.	Microsoft Excel, Lotus 1-2-3
Database Management System	Used for storing information, for example, the names and addresses of all your clients.	Microsoft SQL Server, Oracle
Accounting Programs	They generate extensive financial reports, produce invoices and statements to customers, handle accounts payable and receivable, print payroll checks and payroll reports and track inventory.	TALLY
Presentation Tools	To create presentations by allowing you to produce slides or handouts.	Microsoft PowerPoint
Desktop Publishing	For creating magazines, newsletters, books etc.	Quark Express, Adobe PageMaker
Multimedia Applications	Used for creating multimedia presentations, for example, Web sites, animations, videos etc.	Dreamweaver, Flash, Premier
Telecommunications Software	A program that helps a user to connect and transfer information and files to and from the Internet. It is often part of your operating system or systems software.	DialUp Networking, Open Transport
Web Browser Software	With an Internet connection, this type of software enables a user to visit from one Web site to another by following hyperlinks, to search locations and view web documents. You can also	Google Chrome, Mozila, Opera etc., Microsoft Internet

	access pictures, sounds, videos, animations, and more.	Explorer

Firmware: Firmware is a combination of software (generally system software) permanently stored in the memory (hardware). As the name suggests, firmware is the program or data that has been written onto read-only memory (ROM). BIOS is an example of firmware, which is installed inside a computer on a chip to check different parts of the system when it is started before loading the operating system. BIOS being a firmware ensures that it will always be available and will not be damaged in case of a power failure. ROMs, PROMs, and EPROMs that have data or programs recorded on them are firmware.

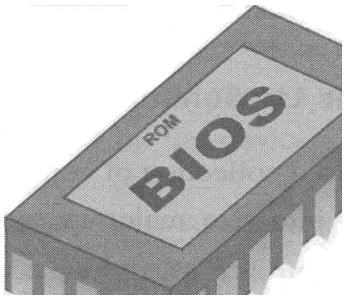


Figure 2.3.12 BIOS

2.3.6 Summary

Software is a generic term for organised collection of computer data and instructions. Computer software can be divided into two major categories, namely, *system software* and *application software.* Software that contributes to the control and performance of the computer system and permits the user to use the system more conveniently is termed as *system software. Application software* is the most often seen software by a general user, which is utilised to accomplish specific tasks other than just running on the computer system.

The system software can be further divided into two major categories, namely, *system management programs* and *system development programs.* System management programs take the responsibility of managing and accurate functioning of the computer system. For example, operating system, device drivers, and system utilities. System development programs comprise system softwares associated with the development of computer programs. For example, computer languages, language translators, linkers and loaders.

System utility programs perform day-to-day tasks related to the maintenance of the computer system like Antivirus software and data recovery

software. Device drivers can be divided into two categories, namely, *character device drivers* and *block device drivers.* Depending on the programming language used, language translators are divided into three major categories, namely, *compilers, interpreters,* and *assemblers.*

Application software are controlled by system software, which manage hardware devices and perform background tasks for them. Typical types of application software packages include word processors, spreadsheets, database management systems, presentation tools, accounting programs, telecommunications software, and web browser software.

Firmware is a combination of software (generally system software) permanently stored in the memory (hardware).

2.3.7 *Review Questions:*

Q.1 What do you understand by term software?

Q.2 Briefly explain few system utilities.

Q.3 What do you mean by:

a. Complier b. interpreterc. Assembler

Q.4 What is firmware? What is its importance in a computer system?

Q.5 What is the relationship between software and hardware?

Q.6 Write in detail about the categories in which software can be divided.

Q.7 Why are application software important? In which areas are application software used? Give relevant software names and their use.

2.3.8 *Suggested Readings:*

1. Computer Fundamentals By Pradeep K. Sinha and Priti Sinha (BPB Publications)
2. Fundamentals of Information Technology By Shiv Kumar Anand and Harmohan Sharma (Kalyani Publishers)
3. “Fundamentals of Compters”, by V.Rajaraman, PHI, New Delhi.

Lesson No. 2.4 **Author : Arun Sharma**

OPERATING SYSTEM

Chapter Outline:

2.4.0 Objectives

- Basic components of an operating system
- Types of Operating system
- Important functions provided by Operating System

In the early days of computer use, computers were huge machines, which were expensive to buy, run and maintain. The user at that time interacted directly with the hardware through machine language. A software was required which could perform basic tasks, such as recognising input from the keyboard, sending output to the display screen, keeping track of files and directories on the disk, and controlling peripheral devices such as printer and scanner. The search for such software led to the evolution of modern day *Operating System (OS).* This software is loaded onto the top of memory and performs all the aforesaid basic tasks. Initially, the operating system's interface was only character-based. This interface provides the user with a command prompt and the user has to type all the commands to perform various functions. As a result, the user had to memorise many commands.

With the advancement in technology, operating system became user-friendly by providing graphical user interface (GUI). The GUI based operating system allows manipulation of software by using visual objects such as windows, pull-down menus, mouse pointers and icons. Consequently, operating the computer became easy.

2.4.1 Operating System

An ***operating system*** is a collection of system programs that together control the operation of a computer system. It is the most important software that funs an computer. Operating system along with hardware, application and other system software and users constitute a computer system. It is the most important part of any computer system. It acts as an intermediate between a user and the computer hardware. The operating system has two objectives. Firstly, an operating system controls the computer's hardware. The second objective is to provide an interactive interface to the user and interpret commands so that he can communicate with the hardware.

1. **Managing the computer's hardware:** The prime objective of the operating system is to manage and control the various hardware resources of a computer system. These hardware resources include processor, memory, disk space etc. It receives the user's input from the keyboard and then outputs the data to the monitor. Operating system supervises which input device's data is requesting for being processed and which processed data is ready to be displayed on the output device. In addition to communicating with hardware, the operating system provides an error handling procedure and displays an error notification. If a device is not functioning properly, the operating system tries to communicate with the device again. If it is still unable to communicate with the device, it provides an error message notifying the user about the problem. Figure 2.4.1 illustrates how operating system manages the hardware resources of a computer system.

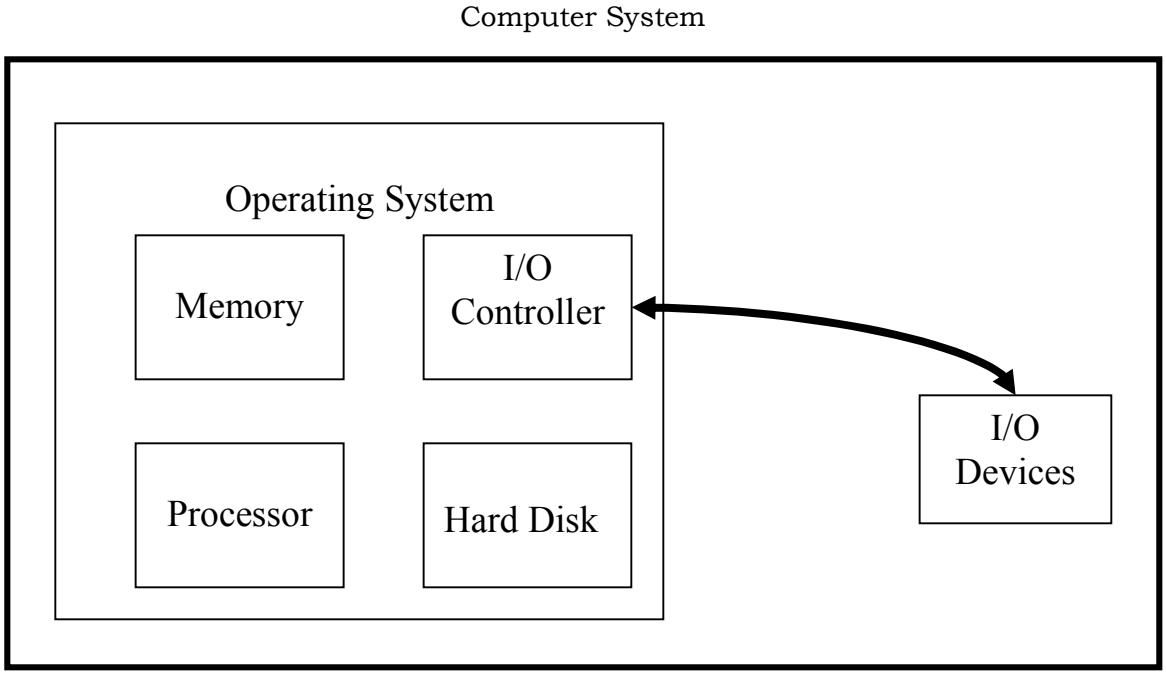


Figure 2.4.1 Operating System Managing Hardware Resources

2. **Providing an Interface:** The operating system organises applications so that users can easily access, use and store them. When an application is opened, the operating system assists the application to provide the major part of the user interface. It provides a stable and consistent way for applications to deal with the hardware without the user having to know all the details of the hardware. If the program is not functioning properly, the operating system again takes control, stops the application and displays appropriate error message.

2.4.2 Evolution of Operating System

In the early days, computers lacked any form of operating system. The user would arrive at the machine armed with his program and data, often on punched paper tape. The program would be loaded into the machine, and the machine set to work, until the program stopped. Then came machines with libraries of support code (initial operating systems), which were linked to the user's program to assist in operations such as input and output. At this stage, operating systems were very diverse, with each vendor producing one or more operating system specific to its particular hardware. Typically, every time new hardware architecture was introduced, there was a need of new operating system compatible with the new

architecture. This state of affairs continued until 1960s when IBM developed the S/360 series of machines. Although there were enormous performance differences across the range, all the machines ran essentially the same operating system called S/360.

Then came the small 4 bit and 8 bit processors known as *microprocessors.* The development of microprocessors provided inexpensive computing for the small businesses. This led to the widespread use of interchangeable hardware components using a common interconnection and thus an increasing need for standardised operating system to control them. The most important of the early operating system was CP/M-80 for the 8080/8085/Z-80 microprocessors. With the development of microprocessors like 386, 486, and the PENTIUM series by Intel, the whole computing world got a new dimension. AT&T and Microsoft came up with character-based operating system, namely, Unix and DOS (Disk Operating System), which supported the then prevalent hardware architectures. After the character-based operating systems, Microsoft and Apple Macintosh came with their Windows 3.1 and MAC, which were GUI based operating systems and well suited for the Desktop PC market. Today operating systems such as Windows XP and Red Hat Linux have taken the driver's seat in personal desktops. These operating systems can handle diverse hardware devices.

2.4.3 Types of Operating System

The operating system has evolved immensely from its primitive days to the present digital era. From batch processing systems to the latest embedded systems, the different types of operating system can be classified into six broad categories:

1. **Batch Processing Operating System:** This type of operating system was one of the first to evolve. Batch processing operating system allowed only one program to run at a time. These kinds of operating systems can still be found on some mainframe computers running batches of jobs. Batch processing operating system works on a series of programs that are held in a queue. The operating system is responsible for scheduling the jobs according to priority and the resources required. Batch processing operating systems are good at churning through large numbers of repetitive jobs on large computers. For example, this operating system would be best suited for a company wishing to automate their payrolls. List of employees will be entered, their monthly salary will be calculated and corresponding pay slips would be printed. Batch processing is useful for this purpose since these procedures are repeated for every employee each month.
2. **Time-sharing or Multi-user Operating System:** A multi-user operating system is used in computer networks which allow different users to access the same data and application programs on the same network. It also allows users to communicate with each other. The multi-user operating system

builds a user database account, which defines the right that users have on a particular resource of the system.

3. **Multi-tasking Operating System:** In multi-tasking operating system, more than one process (task) can be executed concurrently. The processor is switched rapidly between the processes. Hence, a user may run more than one process at a time. It is quite common that a user on his computer can have a word processor open and running, an audio CD player playing and he might be browsing the Internet, all at the same time. This type of operating system allows a user to switch between the applications and even transfer data between them. For example, it allows a user to copy a picture from an Internet site, opened in the browser application and paste it into image editing application.
4. **Real-time Operating System:** Real-time operating systems (RTOS) are designed to respond to an event within a predetermined time. This kind of operating system is primarily used in process control, telecommunications, etc. The operating system monitors various inputs which affect the execution of processes, changing the computers model of the environment, thus affecting the outputs, within a guaranteed time period (usually less than 1 second). As the real-time operating systems respond quickly, they are often used in applications such as air flight or railway reservation booking.
5. **Multi-processor Operating System:** A multi-processor operating system can incorporate more than one processor dedicated to running processes. This technique of using more than one processor is often called *parallel processing.*
6. **Embedded Operating System:** An embedded operating system refers to the operating system that is self-contained in the device and resident in ROM (Read Only Memory). Since embedded systems are usually not general-purpose systems, these operating systems are lighter or less resource intensive as compared to general purpose OS. Most of these operating systems also offer real-time operating system qualities. Typical systems that use embedded operating systems are household appliances, car management systems, traffic control systems and energy management systems.

2.4.4 Functions of an Operating System

The main functions of a modem operating system are as follows:

1. **Process Management:** As a process manager, the operating system handles the creation and deletion of processes, suspension and resumption of processes and scheduling and synchronisation of processes.
2. **Memory Management:** As a memory manager, the operating system handles allocation and deallocation of memory space as required by various

programs.

3. **File Management:** The operating system is responsible for creation and deletion of files and directories. It also takes care of other file-related activities such as organising, storing, retrieving, naming and protecting the files.
4. **Device Management:** Operating system provides input/output subsystem between process and device driver. It handles the device caches, buffers and interrupts. Operating system also detects device failures and notifies the same to the user.
5. **Security Management:** The operating system protects system resources and information against destruction and unauthorised use.
6. **User Interface:** Operating system provides the interface between the user and the hardware. The user interface is the layer that actually interacts with the computer operator. The interface consists of a set of commands or menus through which a user communicates with a program.

2.4.4.1 Process Management

A ***process*** is an execution of sequence of instructions or program by the Central Processing Unit (CPU). It can also be referred as the basic unit of program that the operating system deals with the help of the processor. For example, a text editor program running on a computer is a process. A program may cause several other processes to begin. For instance, a text editing program can furnish a request for printing while editing the document. Here, the text editor is a program that initiates two processes - one for editing the text and second for printing the document. Hence, process is initiated by the program to perform an action, which can be controlled by a user or by the operating system. A process in order to accomplish a task needs certain resources like CPU time, memory allocation and I/O device. Therefore, the idea of process management in an operating system is to accomplish the process assigned by the system or a user in a manner that the resources are utilised in a proper and efficient manner.

Life cycle of a process : The operating system is responsible for managing all the processes that are running on a computer and allocates each process a certain amount of time to use the processor. In addition, the operating system also allocates various other resources that processes will need such as computer memory or disk space. To keep track of the state of all the processes, the operating system maintains a table known as the *process table.* Inside this table, every process is listed along with the resources it is using and its current state. Processes at any particular moment can be present in one of the states: *running, ready* or *waiting.* The running state means that the process has all the resources it needs for execution and it has been given permission by the operating system to use the processor. Note that only one process can be in the running state at

any given time. The remaining processes are in a waiting state or a ready state for being processed. The action of holding the current state of the process, which is temporarily stopped (waiting) from the running state, is done through interrupts generated by the operating system. The change of the state of the process from one form to another is called *context change* and this course of action is known as *context switching.*

Let us consider the steps in an example of two processes, a text editor and a calculator, running simultaneously on a computer system.

Step 1: The operating system receives a request to open a text editor.

Step 2: A new process for the text editor is initiated by the operating system.

Step 3: Resources such as keyboard, memory and hard disk space are made available and the process enters in ready state.

Step 4: The scheduler than dispatches the text editor in running state.

Step 5: In the meantime, another process, a calculator is initiated by the user.

Step 6: A new process is created by the operating system for the calculator.

Step 7: Resources such as keyboard and memory are made available and this process enters in ready state.

Step 8: Now as the calculator process is ready for the processor to work upon, the text editor process, if not terminated (finished), is kept in either waiting state or ready state.

Step 9: The calculator process then attains the running state by the scheduler and when finished is terminated.

Step 10: Then the text editor process again gets the running through ready state by the scheduler.

Threads: A *thread* is a task that runs concurrently with other tasks within the same process. Also known as *lightweight process,* a thread is the simplest unit of a process. The single thread of control allows the process to perform only one task at one time.

Note: *Writing a program in which a process creates multiple threads* is *called multithreading programming.*

Uniprogramming and multiprogramming: As the name implies, uniprogramming means only one process at a time. In uniprogramming, users can perform only one activity at a time. In multiprogramming system, multiple processes can be initiated at the same time. Multiprogramming system supports the resource sharing features like which processes will get to use the physical resources of the machine and when. At the same time, an operating system must ensure that all processes get their fair share of the CPU time.

Process scheduling: In a multiprogramming system, at any given time, several processes will be competing for the CPU's time. At any instance, only one process will be in the running state while others will be in ready or waiting state. The operating system determines how to allocate processor time among all the ready processes. This allocation is referred as *scheduling.* The prime objective of scheduling is to switch the CPU among processes so frequently that users can interact with each running program.

There are different levels of scheduling:

- **High-level Scheduling**: In this level of scheduling, decisions are made on whether a new process should be initiated in the system. Once initiated, this scheduler takes the newly submitted jobs and converts them into processes, which are further put in the ready state for processing.
- **Medium-level Scheduling:** In medium-level scheduling, the operating system decides whether to introduce a process to the waiting state from running state or to put back a process from waiting state to the ready state. This scheduler relieves the processor from the processes for a short duration of time. The processes are then sent to the waiting or blocked state. This helps the operating system to give each process a fair chance to run and improves efficiency of the system.
- **Low-level Scheduling:** In this level of scheduling, decisions are made on which processes in the ready state should be dispatched to the CPU for execution. This scheduler is the most important and complex among all the scheduling levels. The reason being that it comes into picture every, time a current process surrenders its control over the CPU. Different operating systems have adopted different procedures to implement low-level scheduling.

Preemptive and non Preemptive scheduling CPU scheduling may take place under the following four circumstances:

1. When a process switches from the running state to waiting state.
2. When a process switches from the running state to ready state.
3. When a process switches from the waiting state to ready state.
4. When a process terminates.

When scheduling takes place only under 'first' and 'fourth' circumstance, it is said to be *nonpreemptive* scheduling and if the scheduling takes place for 'second' and 'third' circumstances, it is said to be *preemptive.* In preemptive scheme, the scheduler can remove a process from the running state to other state (waiting, blocked, etc.) in order to allow other processes to run. In case of a nonpreemptive scheme, a process when gets into running state shall relinquish its control over the

processor until it is terminated.

The difference between both the schemes is that in preemptive the operating system has the control over the process current states, whereas in case of nonpreemptive scheme the process once entered in running state gets the full control of the processor. Both the schemes have their advantages as well as disadvantages.

Deadlock : In a multiprogramming environment, several processes may compete for a limited number of resources. It is a situation where a set of process are blocked because each process is holding a rescuer and waiting for each other rescuers acquired by some other process. A process requests for the required resource and if it is not available then the process has to wait for it. There might be a situation when the process has to wait endlessly because the requested resource may be held by other waiting processes. This type of situation is known as *deadlock.*

A deadlock situation arises if the following four conditions hold simultaneously on the system:

1) **Mutual Exclusion**: Only one process can use a resource at a time. If another process requests for the resource, the requesting process has to wait until the requested resource is released.
2) **Hold and Wait:** In this situation, a process might be holding some resource while waiting for additional resource, which are currently being held by other processes.
3) **No Preemption:** Resources cannot be preemptive, that is, resources cannot be forcibly removed from a process. A resource can only be released voluntarily by the holding process, after that process has completed its task.
4) **Circular Wait:** This situation may arise when a set of waiting processes say PI and P2 exist in such a way that PI is waiting for resource that is held by P2, and likewise P2 is waiting for the resource held by PI.

To ensure that deadlocks never occur, the system can use either a deadlock-prevention or a deadlock avoidance scheme.

- **Deadlock Prevention:** As we discussed earlier, deadlock can occur only when all the four deadlock causing conditions hold true. Hence, the system should ensure that at least one of the four deadlock causing factors would not hold true so that deadlock can be prevented.
- **Deadlock Avoidance:** Additional information concerning which resources a process will require and use during its lifetime, should be provided to the operating system beforehand.

2.4.4.2 Memory Management

In addition to managing processes, the operating system also manages the primary memory of the computer. The part of the operating system that handles this job is

called the *memory manager.* Since every process must have some amount of primary memory to execute, the performance of the memory manager is crucial to the performance of the entire system. As the memory is central to the operation of any modern operating system, its proper use can make a huge difference. The operating system's memory manager is responsible for allocating primary memory to processes and for assisting the programmer in loading and storing the contents of the primary memory. Managing the primary memory, sharing and minimising memory access time are the basic goals of the memory manager. The major tasks accomplished by the memory manager, so that all the processes function in harmony are:

- **Relocation:** Each process must have enough memory to execute.
- **Protection and Sharing:** A process should not run into another process's memory space.

Relocation : When a process is to be executed, it has to be first loaded from the secondary storage (like hard disk) to the main memory (RAM). This is called *process loading.* In addition, the process has to be loaded back to the secondary memory after its execution. Since, memory is limited and the other processes need it for their use, an operating system swaps the two processes, which is called *swapping.* Once the process is 'swapped out', it is uncertain to tell when it will be 'swapped in' because of the number of processes running concurrently. Therefore, when the process is swapped back into main memory, it must be placed back to the same memory space as before which is done by the memory manager.

Protection and sharing: The memory manager should also make sure that each process accesses memory location, which is not being used by any other process. It is done to protect the already used memory location from being allocated to some other processes running concurrently. Any process should not access the memory location allocated to any other process without permission. It is the duty of the memory manager to check all the memory references at the run time, so that it references only the memory location allocated to that particular process. At the same time, the memory protection program should be flexible enough to allow concurrent processes to share the same proportion of the main memory.

Memory allocation : In uniprogramming systems, where only one process runs at a time, memory management becomes very simple. The process to be executed is loaded into the part of memory space that is unused. Early MS-DOS systems operated in the same manner.

The main challenge of efficiently managing memory comes when a system has multiple processes running at the same time. The memory manager can allocate a portion of primary memory to each process for its own use. However, the memory manager must keep track of which processes are running in which memory

locations and must also determine how to allocate and de-allocate available memory when new processes are created and old processes have finished execution.

While different strategies are used to allocate space to processes competing for memory, three of the most popular are *Best fit, First fit* and *Worst fit.*

1. **Best Fit:** In this case, the memory manager places a process in the smallest block of unallocated memory in which it will fit. For example, a process requests 12KB of memory and the memory manager currently has a list of unallocated blocks of 6KB, 14KB, 19KB, 11KB and 13KB blocks. The best fit strategy will allocate 12KB of the 13KB block to the process.
2. **First Fit:** There may be many unallocated blocks in the memory, so the memory manager, to reduce the amount of time it spends analyzing the available blocks, begins at the start of primary memory and allocates memory from the first location it encounters large enough to satisfy the request. Using the same example to fulfil 12KB request, first fit will allocate 12KB of the l4KB block to the process.
3. **Worst Fit:** The memory manager places a process in the largest block of unallocated memory available. To furnish the l2KB request again, worst fit will allocate 12KB of the 19KB block to the process, leaving a 7KB block for future use.

2.4.4.3 Concept of Virtual Memory

A process executes only in the main memory, which is also referred to as real memory. Today with the advent of graphic-oriented applications like 3D video games, business applications, etc., a user requires large memory for running such applications than the real memory installed on the system. Note that it is not essential that the whole program must be loaded in the main memory for processing, as only the referenced page needs to be present in the memory at the time of execution. Therefore, the rest of the program can lie in the hard disk portion allocated as virtual memory. Later, the fetching of the referenced page can be accomplished by a memory map. The process of swapping the pages between virtual memory and real memory is called *page-in* or *swap-in.* By using this approach, the system can run programs that are actually larger than the primary memory of the system. Virtual memory allows for very effective multiprogramming and relieves the user of the unnecessarily tight constraints of main memory.

Virtual memory is a storage of location scheme in which secondary memory can be addressed of through it were post of main memory. Virtual memory, in other words, is a way of making the real memory of a computer system effectively larger than it really is. The system does this by determining which parts of its memory are often sitting idle and then makes a command decision to empty their contents onto a disk, thereby freeing up precious RAM.

2.4.4.4 File Management

File system is one of the most visible aspects of the operating system. The file system provides a uniform logical view of the information storage, organised in terms of files, which are mapped onto the underlying physical device like the hard disk. While the memory manager is responsible for the maintenance of primary memory, the file manager is responsible for the maintenance of the file system. In the simplest arrangement, file system contains a hierarchical structure of data. This file system maintains user data and metadata (the data describing the files of user data). The hierarchical structure usually contains the metadata in the form of directories of files and sub-directories. Each file is a named collection of data stored on the disk. The file manager implements this abstraction and provides directories for organising files. It also provides a spectrum of commands to read and write the contents of a file, to set the file read/write position, to set and use the protection mechanism, to change the ownership, to list files in a directory and to remove a file. The file manager provides a protection mechanism to allow machine users to administer how processes executing on behalf of different users can access the information contained in different files.

The file manager also provides a logical way for users to organise files in the secondary storage. To assist users, most file managers allow files to be grouped into a bundle called a *directory* or a *folder.* This allows a user to organise his or her files according to their purpose by placing related files in the same directory. By allowing directories to contain other directories called *sub-directories,* a hierarchical organisation can be constructed. For example, a user may create a directory called Games that contains sub-directories called Cricket, Football, and Tennis. Within each of these sub-directories could be files that fall within that particular category. A sequence of directories within directories is called a *directory path.*

2.4.4.5 Device Management

Device management in an operating system refers to the process of managing various devices connected to the computer. The device manager manages the hardware resources and provides an interface to hardware for application programs. A device communicates with the computer system by sending signals over a cable. The device communicates with the machine through a connection point called *port.* The communication via port is done through rigidly defined protocols, like when to send the data and when to stop. These ports are consecutively connected to a bus (a set of wires) which one or more device uses to communicate with the system. The operating system communicates with the hardware with standard software provided by the hardware vendor called *device drivers.* Device driver works as a translator between the electrical signals from the hardware and the application programs of

the operating system. Drivers take data that the operating system has defined as a file and translate them into streams of bits placed in specific locations on storage devices. There are differences in the way that the driver program functions, but most of them run when the device is required and function much the same as any other process. The operating system will frequently assign processes based on priority to drivers so that the hardware resources can be released and set free for further use.

Broadly, managing input and output is a matter of managing queues and buffers. Buffers are special storage facilities that take a stream of bits from a device like keyboard to a serial communication port. Buffers hold the bits and then release them to the CPU at a convenient rate so that the CPU can act on it. This task is important when a number of processes are running and taking up the processor's time. The operating system instructs a buffer to continue taking the input from the device. In addition, it also instructs the buffer to stop sending data back to the CPU while the process, using the input is suspended. When the process, requiring input, is made active once again, the operating system will command the buffer to send data again. This process allows a keyboard to deal with external users at a higher speed.

2.4.4.6 Security Management

Security in terms of a computer system covers every aspect of its protection in case of a catastrophic event, corruption of data, loss of confidentiality, etc. Security requires not only ample protection within the system, but from the external variables also, in which the system operates. In this section, we will be covering security in terms of internal protection, which is one of the most important functions of the operating system. This involves protecting information residing in the system from unauthorised access. Various security techniques employed by the operating system to secure the information are *user authentication* and *backup of data.*

User authentication : The process of authenticating users can be based on user's possession like a key or card, user information like the username and password, and user attributes like fingerprint and signature. Apart from these techniques, user information is often the first and most significant line of defence in a multi-user system. After the user identifies himself by a username, he is prompted for a password. If the password supplied by the user matches the password stored in the system, the system authenticates the user and gives him access to the system. A password can also be associated with other resources (files, directories, etc.), which when requested upon prompts the user for password. Unfortunately, passwords can often be guessed, illegally transferred or exposed. To avoid such situations, a user should keep the following points in mind:

- Password should be at-least six characters in length.
- The system should keep track of any event about any attempt to break the password.
- The system should allow limited number of attempts for submitting a password on a particular system.
- Password based on dictionary words should be discouraged by the system. Alphanumeric passwords, such as *PASS001,* should be used.

Backup of data: No matter what kind of information a system contains, backup of data is of utmost importance to its users. Backup or archiving is an important issue for a user and especially for business organisations. Typically, a computer system uses hard drives for online data storage. These drives may sometimes fail, or can be damaged in case of a catastrophic event, so care must be taken to ensure that the data is not lost. To ensure this, operating system should provide a feature of backing of data, say from a disk to another storage device such as a floppy disk or an optical disk. The purpose of keeping backups is to be able to restore individual files or complete file system. Recovery from the loss of an individual file or of an entire disk, may be done from backup. Operating system usually provides some system software that is used for taking backups of the data.

2.4.4.7 User Interface

Operating system organises applications so that users can easily access them, use them and store application data. When an application is opened, the operating system lets the application provide the majority of the user interface. The operating system still has the responsibility of providing access to the hardware for whatever the application needs. If the program cannot function properly, the operating system again takes control, stops the application and displays an error message. An effective interface of an operating system does not concern the user with the internal workings of the system. A good user interface should attempt to anticipate the user's requirements and assist him to gather information and use necessary tools. Common interfaces provided by different operating systems can be categorised as *Command Line Interface* (CLI) and *Graphical User Interface* (GUI).

Command Line Interface (CLI): In early days of computing, operating sytems provided the user with the facility of entering commands via an interactive terminal. Those were the only means of communication between a program and its user, based solely on textual input and output. Commands were used to initiate programs, applications and so on. A user had to learn many commands for proper operation of the system.

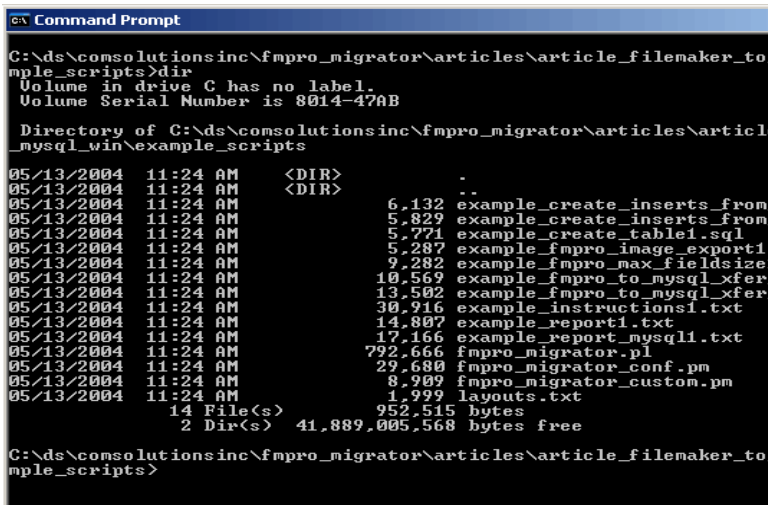


Figure 2.4.2 Command Line Interface

Graphical User Interface (GUI) With the development in chip designing technology, computer hardware became quicker and cheaper, which led to the birth of graphical user interface based operating system. These operating systems provide users with pictures rather than just characters to interact with the machine. The operating system displays icons, buttons, dialog boxes, etc. on the screen. The user sends instructions by moving a pointer on the screen (generally mouse) and selecting certain objects by pressing buttons on the mouse while the mouse pointer is pointing at them. MS-Windows

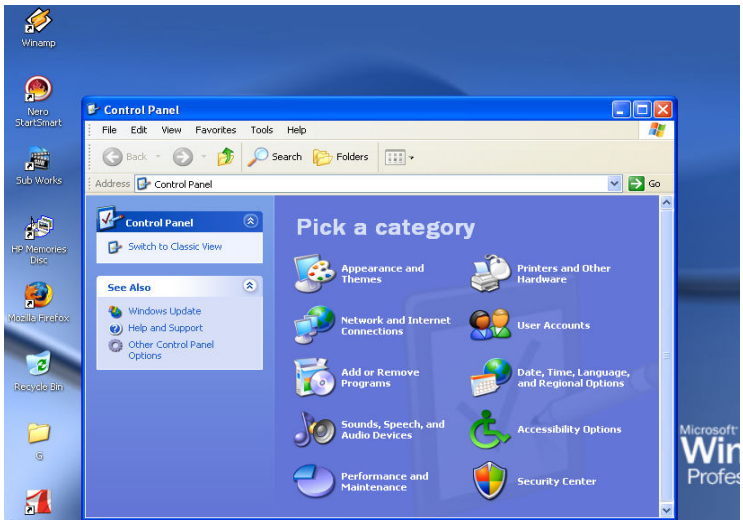


Figure 2.4.3 Graphical User Interface

2.4.5 Summary

Operating system is a type of software that controls and coordinates the operation of the various types of devices in a computer system. The two main objectives of an operating system are controlling the computer's hardware and providing an interactive interface

between the user and the machine. Operating system has six major roles to perform, namely, process management, memory management, file management, device management, security management and providing user interface.

A *process* or task is a portion of a program in some stage of execution. A program can consist of several processes, each working on its own. A process in a computer system may be in one of a number of different possible states, such as new, running, waiting, ready or terminated. The part of the operating system that manages the primary memory of the computer among different processes is called the *memory manager. Virtual memory* is an imaginary main memory supported by operating system in conjunction with the secondary memory. The system that an operating system uses to organise and keep track of files is known as *file management system.* A program that controls a device is called the *device driver.* Operating system's device manager uses this program to let a user use the specific device. Every device, whether it is a printer, disk drive, or keyboard must have a driver program. A user interface is a set of commands or menus through which a user communicates with the system.

2.4.6 *Review Questions:*

Q.1 What is an operating system? Explain the various types of services provided by the operating system.

Q.2 Differentiate between
- Uniprogramming and Multiprogramming
- Preemptive and Non-preemptive Scheduling
- Deadlock Avoidance and Deadlock Prevention

Q.3 Define a process. How many types of process scheduling exist?

Q.4 Discuss various types of interfaces in the operating system. What is a deadlock ? How can it be eliminated from the system?

Q.5 How many types of operating system are there? Explain them.

Q.6 Explain the life cycle of a process.

Q.7 Explain how memory protection and process allocation is done by an operating system.

Q.8 Discuss the evolution of operating system.

2.4.7 *Suggested Readings:*

1. Computer Fundamentals By Pradeep K. Sinha and Priti Sinha (BPB Publications)
2. Fundamentals of Information Technology By Shiv Kumar Anand and Harmohan Sharma (Kalyani Publishers)
3. “Fundamentals of Computers”, by V.Rajaraman, PHI, New Delhi

PGDCA **PAPER-I**

INTRODUCTION TO

INFORMATION TECHNOLOGY

LESSON NO. 2.5 **AUTHOR : ARUN SHARMA**

DATA COMMUNICATION

Chapter Outline:

2.5.0 Objectives

- ✓ *What are the basic components in data communication system?*
- ✓ *Types of transmission modes*
- ✓ *Different types of transmission media through which communication signals are transmitted*
- ✓ *Modulating Techniques*
- ✓ *How multiplexer and switch are helpful in communication*

The term communication in simple words means sending or receiving information. In ancient times, people used the beating of drums, smoke signals, mirrors reflecting sunlight and so on to send messages. Another way of long distance communication was the use of homing pigeons to carry messages. With the advancement in science and technology, various devices were invented for communication. Telegraph, which originated in Greece was one of the most prevalent ways of data communication in the 19th century. In 1870, Alexander Graham Bell invented the telephone, which revolutionised the way long distance communication used to take place. Voice communication became common after the invention of the telephone. The invention of computer and a major breakthrough in 1950s in setting up a network of computers was just the beginning of the electronic transfer of information or data communication. The advancement of computer networks and later the internet acted as a boost for Data communication.

2.5.1 Data Communication

A *communication system* can be defined as the collection of hardware and software that facilitates intersystem exchange of information between different devices. When we communicate, we are sharing information. This sharing can be *local* (face to face communication) or it may be *remote* (communication over distance).

***Data communication* is the exchange of data between two devices via some form of wired or wireless transmission medium**. It includes the transfer of data, the method of transfer and the preservation of the data during the transfer process. To initiate data communication, the communicating devices should be a part of an existing communication system. For data communication to be effective, the following three fundamental characteristics should be considered:

- **Delivery:** The system must deliver data to the correct or the intended destination.
- **Accuracy:** The system must deliver data accurately (error free).
- **Timeliness:** The system must deliver data in a timely manner without enough time lags.

2.5.1.1 Data Communication Components

There are five basic components in data communication system:

1. **Message:** It is the information that is to be communicated.
2. **Sender:** The sender is the device that sends the message.
3. **Receiver:** The receiver is the device that receives the message.
4. **Medium:** The transmission medium is the physical path that communicates the message from sender to receiver.

5. **Protocol:** Protocol refers to a set of rules that coordinates the exchange of information. Both the sender and receiver should follow the same protocol to communicate data. Without the protocol, the sender and receiver cannot communicate with each other; just as a person speaking English cannot be understood by a person who speaks only Hindi. In data communication, TCP/IP protocol is followed by computer systems to communicate,

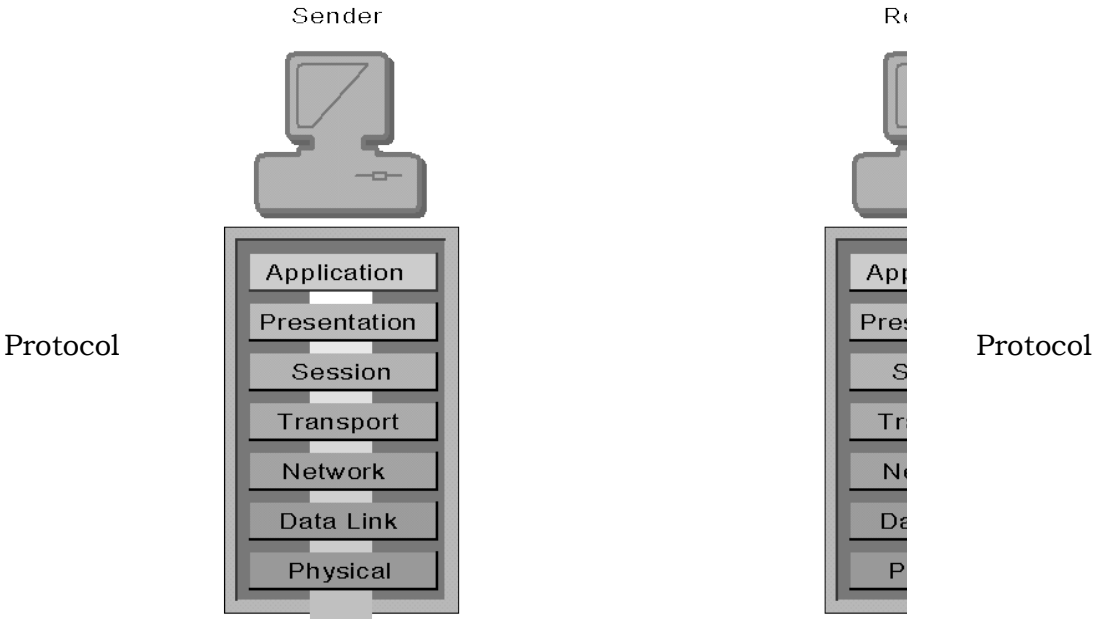


Figure 2.5.1 Data Communication Components

e.g. suppose you want to convey your final marks to your mother. You call her via telephone and inform her about your marks. Here your marks is the message, you are the sender, your mother is the receiver, the telephone line is the medium and the language in which you are conversing is the protocol.

2.5.1.2 Data Transmission Mode

Data transmission mode refers to the direction of signal flow between two linked devices. There are three types of transmission modes: *simplex, half-duplex,* and *full-duplex.*

Simplex: Simplex transmission is unidirectional. The information flows in one direction across the circuit, with no capability to support response in the other direction. Only one of the communicating devices transmits information, the other can only receive it. Television transmission can be considered as an

example of simplex mode of transmission where the satellite only transmits the data to the television, vice versa is not possible. Other examples are Radio broadcasting, computer to printer communication, keyboard to computer etc.

Half-duplex : In half-duplex mode, each communicating device can receive and transmit information, but not at the same time. When one device is sending, the other can only receive at that point of time. In half-duplex transmission mode, the entire capacity of the transmission medium is taken over by the device, which is transmitting at that moment. The most common example of half-duplex transmission is the wireless handsets (generally used by military personnel) where one user talks at a time and another listens.

Full-duplex: Full-duplex transmission mode, also known as the duplex mode, allows both communicating devices to transmit and receive data simultaneously. A full-duplex mode can be compared to a two-way road with traffic flowing in both directions. A common example of full-duplex transmitting mode is the telephone network, where two people communicate over a telephone line; both can talk as well as listen at the same time.

2.5.1.3 Data Communication Measurement

The measurement of the quantity of data that can be passed down a communication link in a given time is done in terms of bandwidth. Fundamentally, *bandwidth* refers to the maximum volume of information that can be transferred over any communication medium. The more the information needed to transmit in a given period, the more the bandwidth required. On digital circuits, bandwidth is measured in bits per second (bps), which refers to the number of binary data bits transmitted per second. A thousand (1,000) bps is one kilobit per second or Kbps.

In the popular digital context, the level of bandwidth falls into three categories:

- **Narrowband**: In narrowband, there is a single transmission channel of 64Kbps or less. There can also be a number of 64Kbps transmission *(N x* 64Kbps) but not more that 1.544 million bits per second (Mbps).
- **Wideband:** In wideband, the bandwidth capacity lies between 1.544Mbps (also called T1 line) and 45Mbps (T3 line).
- **Broadband:** The bandwidth capacity in broadband is equal to 45Mbps or a T3 line.

2.5.2 Transmission Media

Transmission media refers to the physical media through which communication signals (data and information) are transmitted. The information or a signal transmitted from one device to another is through electromagnetic signals. An

electromagnetic signal is the combination of electric and magnetic fields, vibrating in conjugation with each other. Electromagnetic signals include power, voice, radio waves, infrared light, visible light, ultraviolet light, X rays and gamma rays. All these together constitute an electromagnetic spectrum. These signals can travel through vacuum, air or any other transmission medium. Voice signals are generally transmitted as current over metal cables. Radio frequencies are generally transmitted through air or space. Third type of electromagnetic energy is the visible light, which is currently being used for communication through fiber optic cable. Transmission media can be divided into two broad categories: *guided* media and *unguided* media.

2.5.2.1 Guided Media

Guided transmission media use a cabling system that guide the data signals along a specific path. The data signals are bound by the cabling system. *Cabling* refers to transmission medium that consists of cable of various metals like copper, tin or silver. Guided medium is also known as bound medium. There are four basic types of guided media: **open wire, twisted pair, coaxial cable and optical fiber.**

Open wire Open wire is traditionally used to describe the electrical wire system or power transmission wires strung along power poles. No shielding or protection from noise interference is used. This can include multi-conductor cables or single wires. This medium is susceptible to a large degree of noise and interference. In addition, it also suffers from loss of energy problem and it can be easily tapped. Consequently, it is not recommended for long data transmission distances.

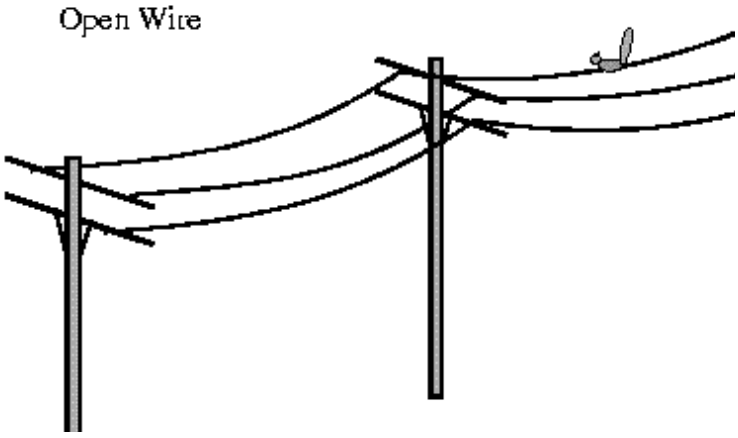


Figure 2.5.2 Open Wire

Twisted pair : In this kind of cabling, pairs of wires are twisted together which are surrounded by an insulating material and an outer layer called jacket. Each pair consists of a wire, used for receiving data signal and a wire used for transmitting data signal. The wires are twisted in order to reduce noise"(unwanted signal) and interference from external sources. Twisted pairs are used in a short distance communication (less than 100 metres) and they are available in two forms: *unshielded* and *shielded.*

Unshielded Twisted Pair (UTP) Cable: UTP cable is the most common type of telecommunication medium in use today. It is most suited for both data and voice transmission and hence is commonly used in telephone systems. The twisted pair consists of two metal conductors (usually copper) that are insulated separately with their own colored plastic insulation. UTP cables have a maximum transmission speed of up to 9600 bps.

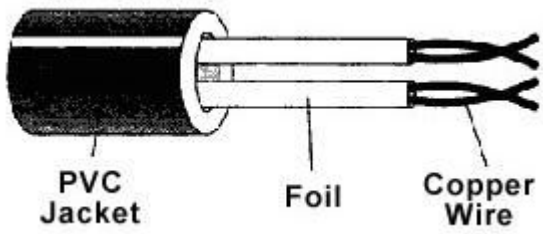


Figure 2.5.3 Unshielded Twisted Pair Cable

Shielded Twisted Pair (STP) Cable: STP cable has a metal foil or braided-mesh covering that covers each pair of insulated conductors. The metal foil is used to prevent infiltration of electromagnetic noise. This shield also helps to eliminate crosstalk, a phenomenon that can be experienced during telephone conversation when one can hear another conversation in the background.

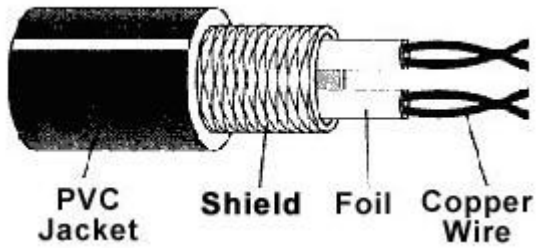


Figure 2.5.4 Shielded Twisted

Coaxial cable Unlike twisted pairs that have two wires, coaxial cables have a

single central conductor, which is made up of solid wire (usually copper). This conductor is surrounded by an insulator over which a sleeve of metal mesh is woven. This metal mesh is again shielded by an outer covering of a thick material (usually PVC) known as jacket. Coaxial cable is very robust and is commonly used in Cable TV network. As compared to twisted pairs, it also offers higher bandwidth. A coaxial cable is capable of transmitting data at a rate of l0 Mbps.

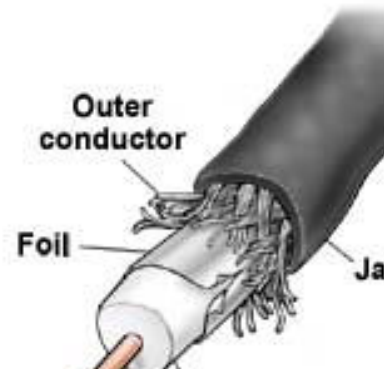


Figure 2.5.5 Coaxial Cable

Optical fiber So far, we have studied twisted pair and coaxial cable; both of which transmit data in the form of current. Optical fiber, on the other hand, consists of thin glass fibers that can carry information in the form of visible light. The typical optical fiber consists of a very narrow strand of glass called the core. Around the core is a concentric layer of glass called the cladding. A typical core diameter is 62.5 microns

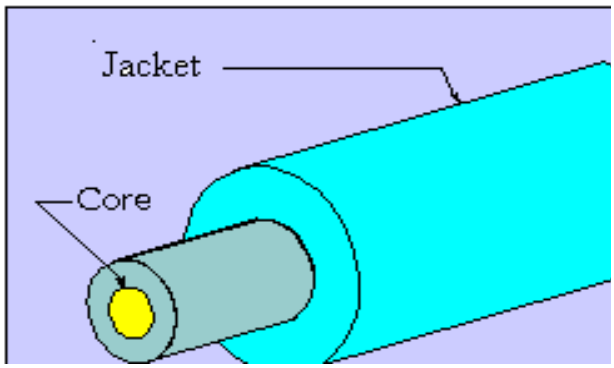


Figure 2.5.6 Optical Fiber

(1 micron = 10^{-6} metres). Cladding generally has a diameter of 125 microns. The cladding is covered by a protective coating of plastic known as jacket.

Advantages of Optical Fiber

- Since transmission is light-based rather than electricity, it is immune to noise interference.
- Transmission distance is greater than other guided media because of less signal attenuation (degradation in quality over distance).
- It is more secure because cable cannot be tapped.
- They are smaller and lighter than copper wire and are free *from* corrosion as well.
- Fiber optic offers, by far, the greatest bandwidth of any transmission system.

Disadvantages of Optical Fiber

- Fiber optic is expensive as it is costly to produce, maintain and instal.
- They are more fragile as fiber optic tends to break easily as compared to copper wires.

2.5.2.2 Unguided Media

Unguided transmission media is data signals that flow through the air. They are not guided or bound to a fixed channel to follow. One of the common unguided media of transmission is *radio frequency propagation.*

Radio frequency propagation In radio frequency propagation, the signal is carried over carrier waves (waves which carry signals over them), which have frequencies in the range of radio frequency spectrum. There are three types of RF (radio frequency) propagation, namely, *ground wave, ionospheric* and *line of sight.*

Ground wave propagation follows the curvature of the earth. They have carrier frequencies of up to 2MHz. AM radio is an example of ground wave propagation.

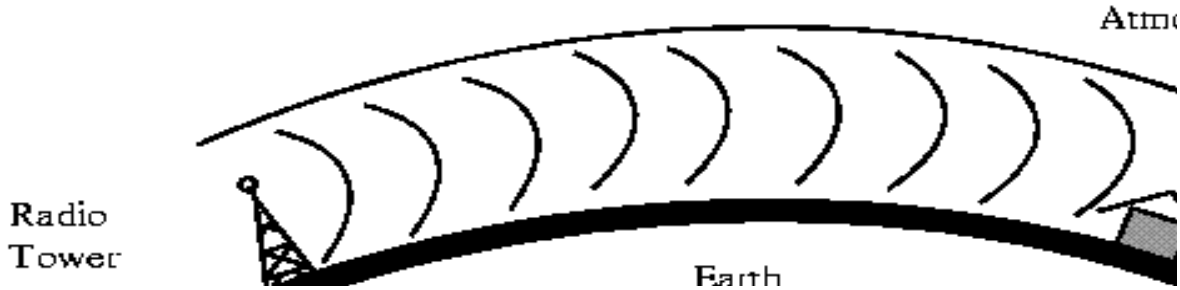


Figure 2.5.7 Ground Wave Propagation

In ionospheric propagation, the signal wave bounces off the earth's ionosphere layer in the upper atmosphere. It operates in the frequency range of 30-85MHz. As this type of propagation depends on the earth's ionosphere, it changes with the day timings and weather.

Figure2.5.8 Ionospheric Propagation

Line of sight propagation transmits exactly in the line of sight. The receiving station must be in view of the transmitting station. It is sometimes called space waves or tropospheric propagation. It is limited by the curvature of the earth for ground-based stations (50 km). Examples of line of sight propagation are FM radio, microwave and satellite.

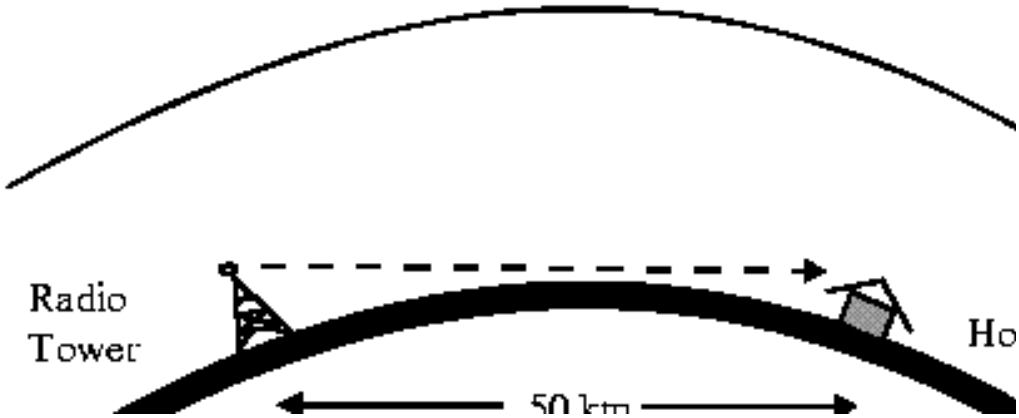


Figure 2.5.9 Line of Sight Propagation

Microwave: Microwave transmission is line of sight transmission. The transmit station must be in visible contact with the receive station. This sets a limit on the distance between stations depending on the local geography. Typically, the line of sight due to the earth's curvature is only 50 km to the horizon. Therefore,

repeater stations must be placed so the data signal can travel farther than the distance limit.

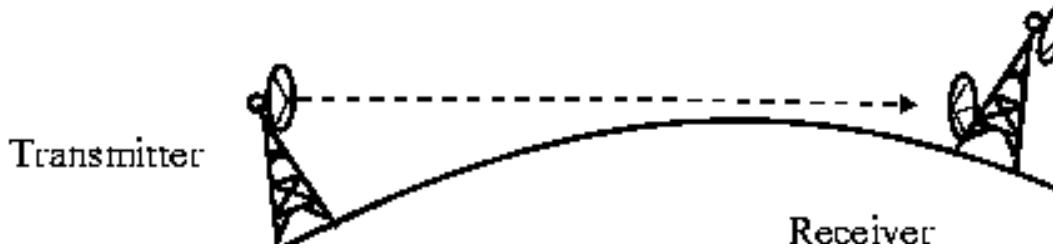


Figure 2.5.10 Microwave Transmission..

Satellite: Satellite transmission is also a kind of line of sight transmission. Satellites are set in geostationary orbits directly over the equator, which rotates in synchronisation to earth and hence look stationary from any point on earth. These geostationary orbits are placed 36,000 km above the earth's surface. The communication is carried through uplinks and downlinks. The uplink transmits the data to the satellite and downlink receives the data from the satellite. Uplinks and downlinks are also called *earth stations* because they are located on the earth. The area shadowed by the satellite in which the information or data can be transmitted and received is called the *footprint.*

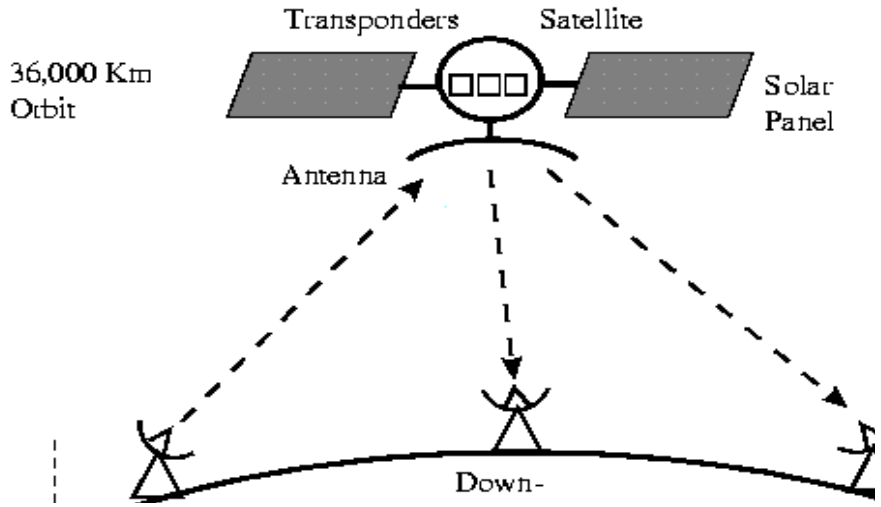


Figure 2.5.11 Satellite Communication

2.5.2.3 Analog and Digital Data Transmission

The major role of the physical medium is to move information from one communicating device to another. However, information to be transmitted should be first transformed into electromagnetic signals. For example, you cannot just write a letter on the piece of paper, insert it into a wire and transmit it across town. Information over any medium is transmitted by two main methods called **analog and digital.**

Analog signals An analog signal is a continuous waveform that changes smoothly over time. The sine wave is the most fundamental form of an analog signal. Sine waves can be described by three characteristics, namely, amplitude, frequency and wavelength.

Figure 2.5.12 An Analog Signal

Amplitude is the value of the signal at any point on the wave. The maximum amplitude of a sine wave is the highest value it reaches on the vertical axis. The unit for amplitude depends on the type of the signal. For electrical signals, the unit is normally volts and amperes. *Frequency* refers to the number of cycles a signal completes in one second. In other words, frequency means the number of times a signal wave goes up and down in a second and it is measured in Hertz (Hz). For example, if a signal wave completes one cycle in one second, its frequency is one Hz. **Wavelength** refers to the distance between successive similar points of a given wave, that is, one cycle of the waveform.

Analog signals are perfect for carrying data such as voice or sound. However, these signals are prone to errors or noise, which are caused from an outside source. Attenuation is another problem with analog signals because the amplitude of the wave naturally changes over distance.

Digital signals Digital data is the data stored in the form of 0s and 1s. When the signal is at a high point, its value is 1 and when it is low, its value is 0. A signal in digital format has precise voltages that are not affected by noise or attenuation as compared to analog signals, which are very prone to noise. Digital signals can be represented by a graph similar to a bar graph. In Figure 2.5.13, 1 can be encoded as a positive voltage and 0 as zero voltage.

To transmit data over analog phone lines, a modem is required to convert

the digital data signals to analog signals. When transmitted over long distances, analog signals require to be amplified, which can possibly distort the value of the data transmitted. When analog data is converted to digital data, it can be transmitted over digital signals faster and without distortion. Digital data is precise but can never transmit the range of information available which is possible in case of analog data transmission.

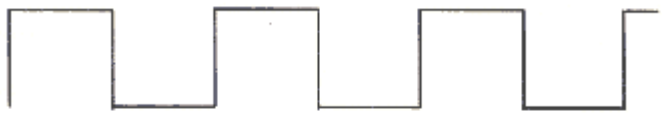


Figure 2.5.13 A Digital Signal Wave

2.5.2.4 Modulation Techniques

Consider a scenario where a boy standing over the roof of his house has to deliver a paper to the boy standing over the roof of the house in front. Due to the distance between the two roofs, the first boy cannot just hand over the paper to the other. In addition, paper alone cannot travel the distance (if thrown) because it is lightweight. Therefore, the first boy wraps the paper onto a stone and throws it, towards the other boy, who catches the stone wrapped in the paper. The boy takes off the paper from the stone, reads the message and discards the stone. In this manner, the first boy is able to deliver a message to the other boy. In the same way, before a signal is transmitted in a wide communication system, the signals (paper) are superimposed on a carrier signal (stone), which propagates by means of an electromagnetic wave. This process is called **modulation**. Modulation is the addition of information (or the signal) to a signal carrier wave. These carrier waves carry the signals to travel over long distances. Generally, there are two forms of modulation - *amplitude* and *frequency*.

Amplitude modulation In this modulation, the amplitude of a carrier wave is varied in accordance with a characteristic of the modulating signal. The frequency of the carrier remains the same, only the amplitude changes to follow variations in the signal. In simpler words, the two discrete binary digits (0 and 1) are represented by two different amplitudes of the carrier signal. Figure 2.5.14 depicts how the modulating signal is superimposed over the carrier signal that results in an amplitude-modulated signal.

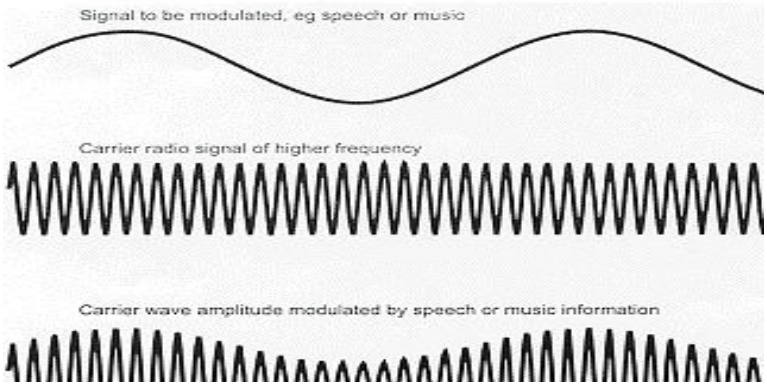


Figure 2.5.14 Amplitude Modulation

Frequency modulation In this modulation, the instantaneous frequency of carrier wave is caused to depart from the centre frequency by an amount proportional to the instantaneous value of the modulating signal. In simple words, frequency modulation is the method of impressing modulating signal onto a carrier signal wave by varying its instantaneous frequency rather than its amplitude (see Figure 2.5.15).

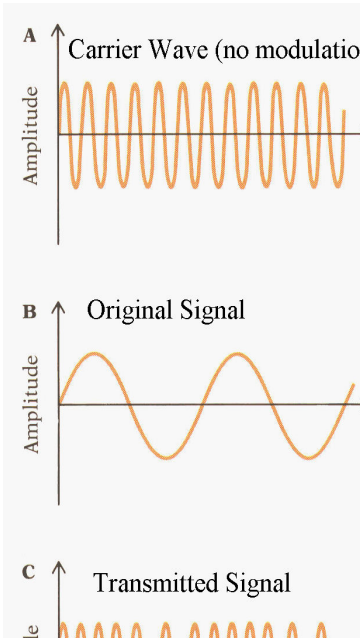


Figure 2.5.15 Frequency Modulation

2.5.3 Multiplexing

In a network environment, it is common that the transmission capacity of a medium linking two devices is greater than the transmission needs of the

connected devices. Hence, the medium can be shared so that it can be used fully. This can be done by multiplexing. **Multiplexing** is a technique used for sending several signals simultaneously over a common medium. An analogy of multiplexing can be made with a multilane highway. Just as a multilane highway can carry increased volumes of traffic in multiple lanes at higher speeds and at relatively low incremental cost ,per lane, higher-capacity circuit can carry multiple conversations in multiple channels at relatively low incremental cost per channel.

2.5.3.1 Multiplexers

In a multiplexed system, several devices share the capacity of one link called the common medium. Figure 2.5.16 shows the three devices on the left communicating to the devices on the right through the common medium. The communication device that multiplexes (combines) several signals from the devices on the left for transmission over the common medium is called a *multiplexer (MUX).* At the receiving end, a *demultiplexer (DEMUX)* completes the communication process by separating multiplexed signals from a transmission line and distributing it to the intended receiver.

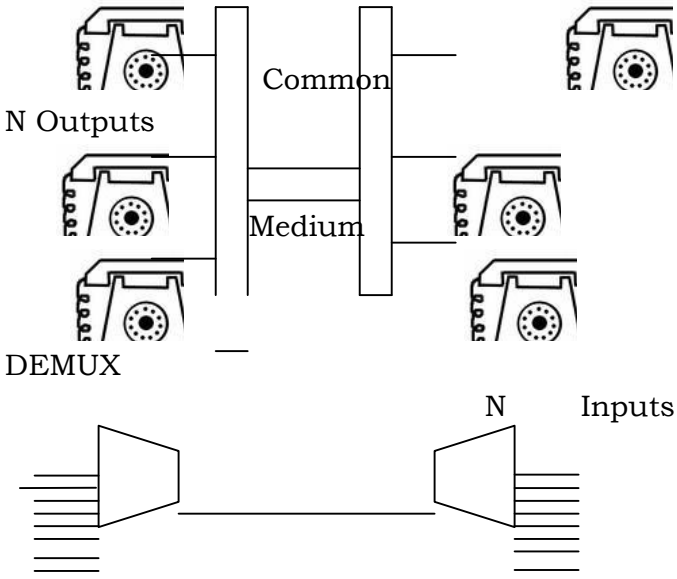


Figure 2.5.16 Multiplexing

Signals are multiplexed using two basic techniques: *frequency-division multiplexing* (FDM) and *time division multiplexing* (TDM).

2.5.3.2 Asynchronous and Synchronous Transmission

The primary concern while considering the transmission of data from one device to another is to decide whether to send the data one bit at a time (serial mode) or to send a group of bits into a large group (parallel mode). In serial mode, one bit is sent at a given instant, hence, only one communicating channel is needed to transmit data. Since only one communication channel is required the cost is largely reduced. On the other hand, in parallel mode, multiple bits are sent at any given instance, therefore, more than one channel is needed to transmit data. There is only one way of sending data in parallel mode but there are two subclasses of serial transmission, namely, *asynchronous* and *synchronous.*

Asynchronous transmission Asynchronous transmission is so called because the timing of the signal is not important. The information that is received or transmitted follows a predefined pattern. As long as the patterns are followed, the receiving device can retrieve the information without any regard to the timing of the signal sent. However, a synchronising pulse is necessary for the receiver to know when the data is coming and when it is ending. Hence, each byte of information is preceded by a *start bit* (denoted by 0) and ended by a *stop bit* (denoted by 1).

Synchronous transmission Synchronous mode of transmission works on the same media as the asynchronous transmission but the transmitter does not send start and stop bits to the receiver. The receiver's clock is synchronised with the transmitter's clock. In other words, data is transmitted as an unbroken string of 1s and 0s and the receiver, on the basis of clock timings, separate the string into bytes. Timing becomes very important in synchronous transmission because without start and stop bits, there is no in-built mechanism to help the receiving device accessing the incoming information. The advantage of synchronous transmission is *speed.* With no extra start and stop bits, overhead is lessened, increasing the speed of transmission. Therefore, synchronous transmission is useful for high-speed application like transfer of large data from one computer to another.

2.5.4 Switching

Consider a scenario of a small office having four telephone sets used by the four employees for communication. If direct lines were to be used for all the people, 6 duplex lines are required *{n(n-1)12* lines, where *n* is the number of telephone

sets}. This is called point -to-point connection. This method, however, is impractical and wasteful when applied to very large networks. The number and length of the links require too much of infrastructure, in addition, majority of these links would remain idle and wasted most of the time. A better solution is *switching.* On a network, switching means routing traffic by setting up temporary connections between two or more network points. This is done by devices located at different locations on the network called *switches* (or exchanges). In a switched network, some switches are directly connected to the communicating devices while others are used for routing or forwarding information.

Switching traditionally employs three methods, namely, *circuit switching, packet switching* and *message switching.* Out of these only circuit and packet switching are in use nowadays, message switching has been phased out in general communications.

2.5.4.1 Circuit Switching

When a device wants to communicate with another device, circuit switching technique creates a fixed bandwidth channel called a circuit, between the source and the destination. This circuit is reserved exclusively for a particular information flow, and no other flow can use it. A common example of a circuit witched network is Public Switched Telephone Network (PSTN).

In circuit switching, data is transmitted with no delay (except for negligible propagation delay). This method is simple and requires no special facilities. Therefore, it is well suited for low speed data transmission.

2.5.4.2 Packet Switching

Packet switching introduces the idea of breaking data into *packets,* which are discrete units of potentially variable length blocks of data. Apart from data, these packets also contain a header with control information like the destination address, priority of the message, etc. These packets are passed by the source point to its local packet switching exchange (PSE). When the PSE receives a packet, it inspects the destination address contained in the packet. Each PSE contains a navigation directory specifying the outgoing links to be used for each network address. On receipt of each packet, the PSE examines the packet header information and then either removes the header or forwards the packet to another system. If the channel is not free, then the packet is placed in a queue until the channel becomes free. As each packet is received at each transitional PSE along the route, it is forwarded on the appropriate link mixed with other packets. At the destination PSE, the packet is finally passed to its destination. Note that not all packets travelling between the same two points,

even those from a single message, will necessarily follow the same route. Therefore, after reaching their destination, each packet is put into order by a packet assembler and disassembler (PAD). Packet switching is widely used in data networks like the Internet.

2.5.4.3 Message Switching

Message switching technique employs the 'store and forward' mechanism. In this mechanism, a special device (usually a computer system with large memory storage) in the network receives the message from a communicating device and stores it into its memory. Then it finds a free route and sends the stored information to the intended receiver. A message in this kind of switching is always delivered to one device where it is stored and then rerouted to its destination. Message switching is one of the earliest types of switching techniques which was common in 1960s and 1970s. As the delays in such switching are inherent (time delay in storing and forwarding the message) and large capacity of data storage is required, this technique has virtually become obsolete.

2.5.5 Summary

Communication system refers to the collection of hardware and software that supports intersystem exchange between different devices. *Data communication* is the exchange of data between two devices via some form of wired or wireless transmission medium. There are five basic components in data communication, namely, *message, sender, receiver, medium* and *protocol.* The direction of signal flow between two communicating devices is defined by the data transmission modes. There are thee types of transmission modes: *simplex* (unidirectional data flow), *halfduplex* (bi-directional data flow, but one at a time) and *full-duplex* (simultaneous bi-directional data flow).

Bandwidth refers to the maximum volume of information that can be transferred over any communication medium. The greater the amount of information needed to transmit in a given period, the more the bandwidth required. The physical or wireless medium through which two communicating devices communicate is known as *transmission media.* The wired (physical) transmission mediums are known as *guided* mediums and the wireless transmission mediums are known as *unguided* mediums.

There are four basic types of guided media (also known as bound media): *open wire, twisted pair, coaxial cable* and *optical fiber.* Unguided transmission media is data signals that flow through the air. One of the common unguided media of transmission is *radio frequency propagation* (microwave and satellite). Information over any medium is transmitted by two main methods called *analog*

and *digital*. An analog signal is a continuous waveform that changes smoothly over time. Digital data refers to the data stored in form of 0s and 1s.

Modulation refers to the process of impressing information on a carrier wave by changing some of the wave's characteristics (such as amplitude, frequency or phase) so that it is more suitable for transmission over the medium between transmitter and receiver. Generally, there are two forms of modulation: *amplitude* and *frequency.*

Multiplexing refers to the process of transmitting more than one signal over a single link, route or channel. There are two basic multiplexing techniques: *frequency-division multiplexing* (FDM) and *time-division multiplexing* (TDM).

Asynchronous transmission refers to the data transmission of one character at a time, with intervals of varying lengths between transmittals and with start bits at the beginning and stop bits at the end of each character to control the transmission. *Synchronous transmission* is a method of communication in which data is sent in blocks, without the need for start and stop bits between each byte. Synchronisation is achieved by sending a clock signal along with the data.

Switching refers to routing traffic by setting up temporary connections between two or more network points. A temporary connection is achieved by devices located at different locations on the network, called switches. There are three methods of switching: *circuit switching, packet switching,* and *message switching.*

15.6 Self Check Exercise

Q1. What is the difference between analog signal and digital signal ?

Q2. What is difference betweeen Asynchronous and Synchronous transmission ?

Q3. Explain the various types of guided and unguided media.

Q.4 What are the various switching methods ? Discuss briefly.

2.5.7 Suggested Readings:

1. Computer Networks by Andrew S.Tanenbaum (PHI, New Delhi).
2. Data Communication and Newworking by Behrouz A. Forouzam (Tata McGraw Hill)
3. Introduction to Data Communications and Networking by Wayne Tomasi (Pearson)
4. Data Communication and Networks by Naveen Kumari (Katson Books)

LESSON NO. 2.6 **AUTHOR : ARUN SHARMA**

COMPUTER NETWORK

Chapter Outline:

2.6.0 Objectives

- ✓ *Types of Computer Network*
- ✓ *Learn types of Network Topologies and their advantages & disadvantages*
- ✓ *OSI Model and its different Layers*
- ✓ *Hardware used* to communicate over a network *(Network Devices)*

2.6.1 Introduction

A *computer network* is a collection of two or more computers, which are connected together to share information and resources. Computers in a network are interconnected by telephone lines, coaxial, cables, satellite links, radio and/or some other communication technique. A network can be as few as

several personal computers on a small network or as large as the Internet, a worldwide network of computers.

Today, when talking about networks, we are generally referring to three primary categories: *Local Area Network (LAN), Metropolitan Area Network (MAN)* and *Wide Area Network (WAN).* These categories are defined depending upon various factors like the size of the network, the distance it covers and the type of link used in interconnection.

Advantages of Computer Network :

- File sharing
- Resource Sharing
- Increased Storage Capacity
- Internet Services
- Interactive entertainments
- e-mail (person to person communication)
- Video Conferencing
- Remote Login
- Flexible Access

2.6.1.1 Local Area Network (LAN)

A LAN, or Local Area Network, is a computer network that spans only a small geographical area (usually within a square mile or less), such as an office, home or building. In a local area network, the connected computers have a network operating system installed onto them. One computer is designated as the *file server,* which stores all the software that controls the network along with the software that can be shared by the computers attached to the network. Other computers connected to the file server are called *workstations.* On most LAN s, cables are used to connect the computers.

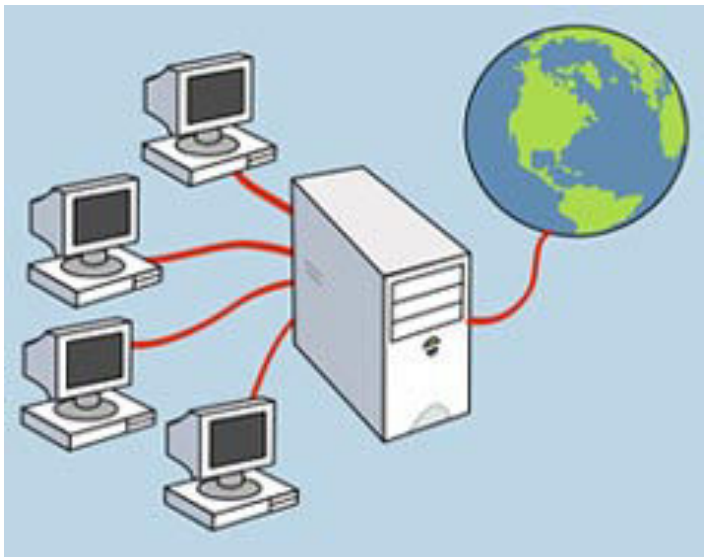


Figure 2.6.1 Local Area Network

Usually, LAN offers a bandwidth of 10 to 100 Mbps. They are generally limited to a maximum distance of only a few kilometers, although they are usually much shorter.

2.6.1.2 Metropolitan Area Network (MAN)

A MAN, or Metropolitan Area Network, is a network of computers spread over a 'metropolitan' area such as a city and its suburbs. As the name suggests, this sort of network is usually reserved for metropolitan areas where the city bridges its local area networks with a series of backbones, making one large network for the entire city. It may be a single network such as a cable television network or it may be a means of connecting a number of LANs. Note that, MAN may be operated by one organisation (a corporate with several offices in one city) or be shared resources used by several organisations in the same city.

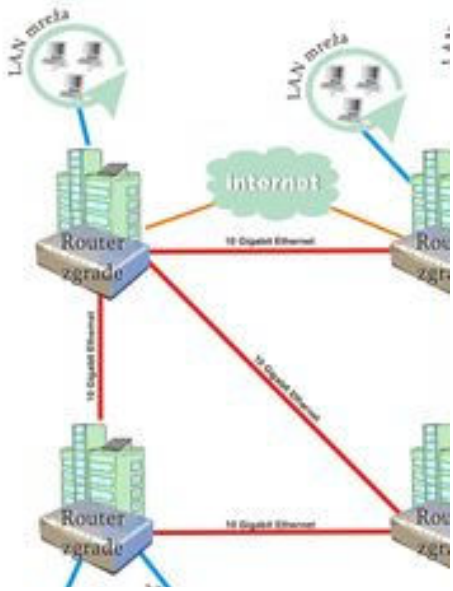


Figure 2.6.2 Metropolitan Area Network

2.6.1.3 Wide Area Network (WAN)

A WAN, or Wide Area Network, is a system of interconnecting many computers over a large geographical area such as cities, states, countries or even the whole world. These kinds of networks use telephone lines, satellite links and other long-range communications technologies to connect. Such networks are designed to serve an area of hundreds or thousands of miles such as public and

private packet switching networks and national telephone networks. For example, a company with offices in New Delhi, Chennai and Mumbai may connect the LANs for each of those locations to each other through a WAN. Although a WAN may be owned or rented by private business, it is usually a public network designed to connect small and intermediate sized networks together. The largest WAN in existence is the Internet.

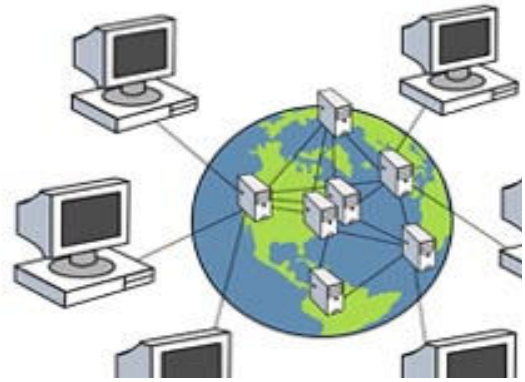


Figure 2.6.3 Wide Area Network

Difference Between LAN & WAN

	LAN	WAN
1.	It is used within a limited geographical area	It is spread over a large area
2.	LAN can be private	It involves authorities like telephone companies i.e. Public and Private
3.	Data transmission speed is high	Data transmission speed is low as compared to LAN
4.	Data transmission is cheaper	It is expensive to transmit data through WAN
5.	Less complex network	More complex network
6.	Data transmission errors are less making it more reliable	More data transmission error making it less reliable
7.	Topologies used : star, bus and ping	Hybrid topology
8.	It operates on principle of broadcasting	Works on principle of switching
9.	Communication medium used for networking is co-arial cable	Sattelite links are used as communication medium

2.6.2 Network Topologies

Network topology is the logical arrangement of a computer in a network. The term *topology* refers to the way a network is laid out, either physically or logically. A topology can be considered as the network's shape. It is the geometric representation of the relationship of all the links. There are five basic topologies: *Bus, Ring, Star, Tree* and *Mesh.*

2.6.2.1 Bus Topology

Bus topology uses a common bus or backbone (a single cable) to connect all devices with terminators at both ends. The backbone acts as a shared communication medium and each node (file server, workstations and peripherals) is attached to it with an interface connector. Whenever a message is to be transmitted on the network, it is passed back and forth along the cable, past the stations (computers) and between the two terminators from one end of the network to the other. As the message passes each station, the station checks the message's destination address. If the address in the message matches the station's address, the station receives the message. If the addresses do not match, the bus carries the message to the next station, and so on. Figure 2.6.4 illustrates how devices such as file servers, workstations, and printers are connected to the linear cable or the backbone.

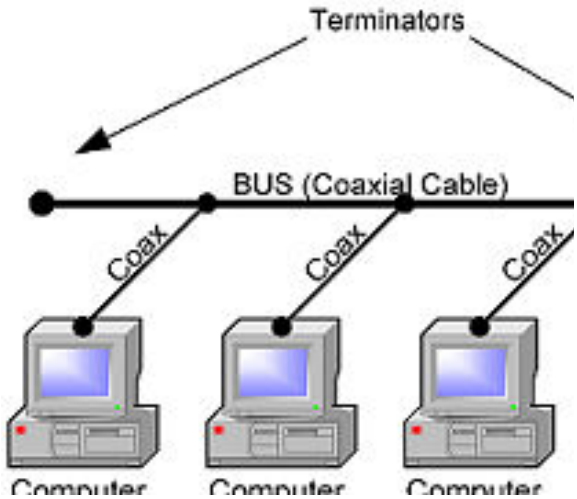


Figure 2.6.4

Advantages of Bus Topology

- Connecting a computer or peripheral to a linear bus is easy.
- This topology requires least amount of cabling to connect the computers and, therefore, less expensive than other cabling arrangement.

- It is easy to extend a bus since two cables can be joined into one longer cable with a connector.

Disadvantages of Bus Topology

- Entire network shuts down if there is a failure in the backbone.
- Heavy traffic can slow down a bus because computers on such networks do not coordinate with each other to reserve time to transmit.

2.6.2.2 Ring Topology

In ring topology, computers are placed on a Circle of cable without any terminated ends since there are no unconnected ends. Every node has exactly two neighbours for communication purposes. All messages travel through a ring in the same direction (clockwise or counterclockwise) until it reaches its destination. Each node in the ring incorporates a repeater. When a node receives a signal intended for another device, its repeater regenerates the bits and passes them along the wire.

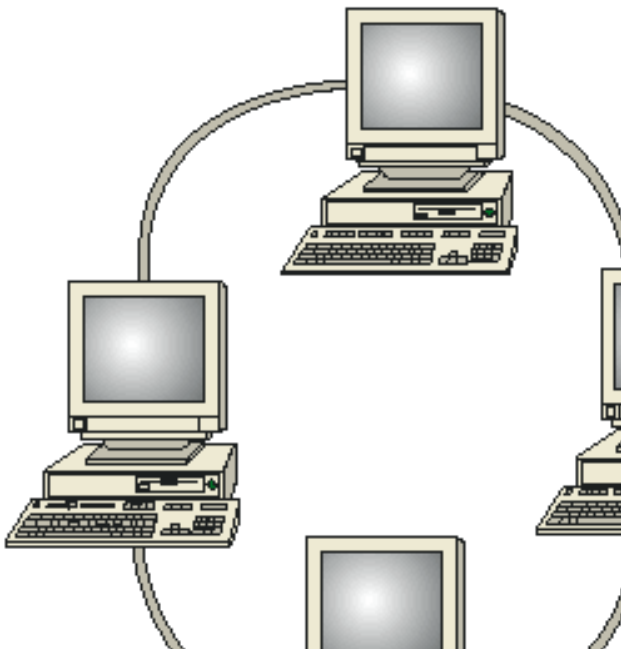


Figure 2.6.5 Ring Topology

Advantages of Ring Topology

- Ring topology is easy to install and reconfigure.
- Every computer is given equal access to the ring. Hence, no single computer can monopolise the network.

Disadvantages of Ring Topology

- Failure in any cable or node breaks the loop and can take down the entire network.
- Maximum ring length and number of nodes are limited.

2.6.2.3 Star Topology

In star topology devices are not directly linked to each other but they are connected via a centralised network component known as *hub* or *concentrator.* The hub acts as a central controller and if a node wants to send data to another node, it boosts up the message and sends the message to the intended node. This topology commonly uses twisted pair cable, however, coaxial cable or fiber optic cable can also be used.

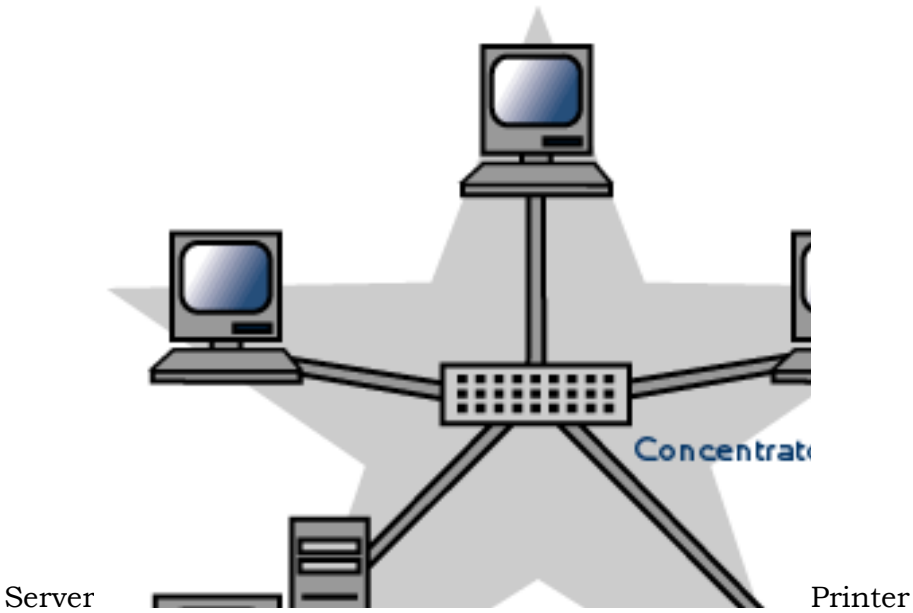


Figure 2.6.6 Star Topology

Advantages of Star Topology

- Star topology is easy to install and wire.
- The network is not disrupted even if a node fails or is removed from the network.
- Fault detection and removal of faulty parts is easier in star topology.

Disadvantages of Star Topology

- It requires a longer length of cable.
- If the hub fails, nodes attached to it are disabled.
- The cost of the hub makes the network expensive as compared to bus and ring topology.

2.6.2.4 Tree Topology

A tree topology combines characteristics of linear bus and star topologies. It consists of groups of star-configured workstations connected to a bus backbone cable. Not every node plugs directly to the central hub. The majority of nodes connect to a secondary hub that in turn is connected to the central hub. Each secondary hub in this topology functions as the originating point of a branch to which other nodes connect.

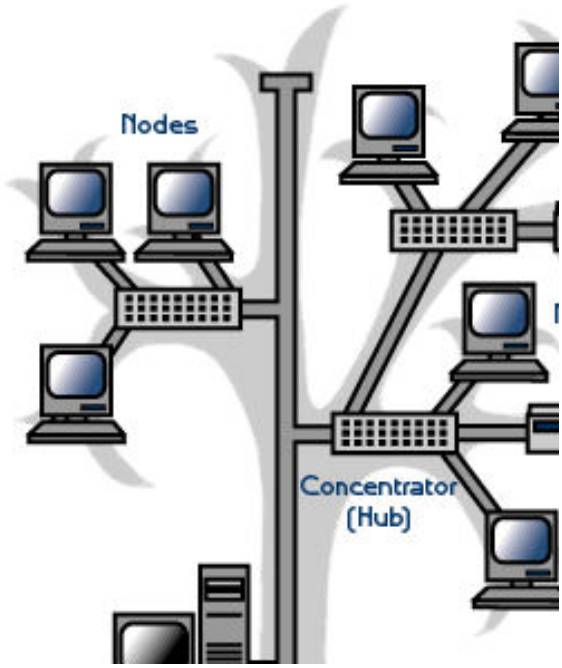


Figure 2.6.7 Tree Topology

Advantages of Tree Topology

- The distance to which a signal can travel increases as the signal passes through a chain of hubs.
- Tree topology allows isolating and prioritising communications from

different nodes.

- Tree topology allows for easy expansion of an existing network, which enable organisations to configure a network to meet their needs.

Disadvantages of Tree Topology

- If the backbone line breaks the entire segment goes down.
- It is more difficult to configure and wire than other topologies.

2.6.2.5 Mesh Topology

In a mesh topology, every node has a dedicated point-to-point link to every other node. Messages sent on a mesh network can take any of several possible paths from source to destination. A fully connected mesh network has *n(n-l)/2* physical links to link *n* devices. For example, if an organisation has 5 nodes and wants to implement a mesh topology, 5(5-1)/2, that is 10 links are required. In addition, to accommodate that many links, every device on the network must have *n-l* communication (input/output) ports.

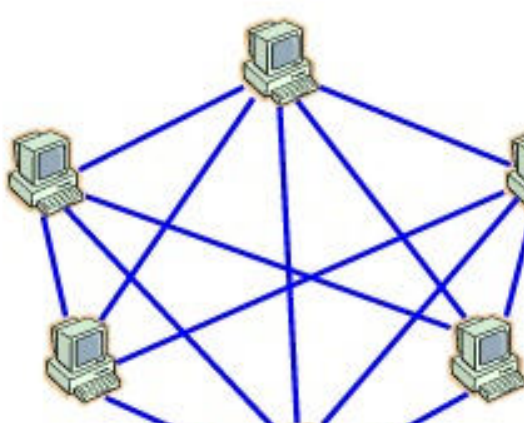


Figure 2.6.8 Mesh Topology

Advantages of Mesh Topology

- The use of large number of links eliminates network congestion.
- If one link becomes unusable it does not disable the entire system.

Disadvantages of Mesh Topology

- The amount of required cabling is very large.
- As every node is connected to the other, installation and reconfiguration is very difficult.
- The amount of hardware required in this type of topology can make it expensive to implement.

2.6.3 Communication Protocols

Imagine yourself standing near a traffic crossing. You can notice that for the smooth movement of the traffic, a functioning traffic light is essential. The *red* signal sends a message to stop, a *yellow* signal to wait and a *green* signal to cross. This set of rules, which tell a driver when to move and when to stop are traffic protocols. Similarly, computers adhere to certain protocols that define the manner in which communication take place. A computer protocol is a set of rules that coordinates the exchange of information. If one computer is sending information to another and they both follow the same protocol, the message gets through; regardless of what types of machines they are and on what operating systems they are running. As long as the machines have software that can manage the protocol, communication is possible.

2.6.3.1 The OSI Model

Open Systems Interconnection (OSI) is a standard reference model for communication between two end users in a network. It describes seven year that computer system use to communicate ever a network. It is used in developing products and understanding networks. In 1983, the International Organisation for Standardisation (ISO) published a document called 'The Basic Reference Model for Open Systems Interconnection, which visualise network protocols as a seven-layered model. The model lays a framework for the design of network systems that allow for communication across all types of computer systems. It consists of seven separate but related layers, namely, *Physical, Data Link, Network, Transport, Session, Presentation* and *Application.*

In developing the model, the designers refined the process of transmitting data to its most fundamental elements. They identified which networking functions had related uses and collected those functions into discrete groups that became the seven layers.

2.6.3.2 Categories of the OSI Layers

The seven layers of the OSI reference model can be divided into two categories:

1. **Upper Layers:** The upper layers of the OSI model consist of the *application, presentation* and *session* layers. Primarily, these layers deal with application issues, and are implemented only in the software. The highest layer, application is closest to the end user.
2. **Lower Layers:** The lower layers of the OSI model comprises the *transport, network, data link* and *physical* layers. These layers handle the data transport issues. The physical layer and data link layer are implemented in

both hardware and software. The other lower layers that is network and transport, are generally implemented only in software. The lowest layer, the physical layer, is closest to the physical network medium (the network cabling) and is responsible for actually placing the information on the medium.

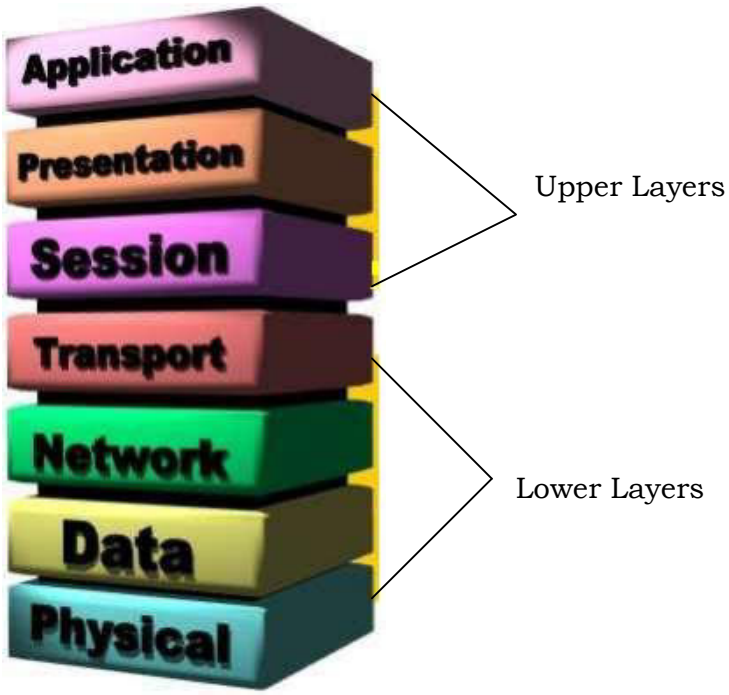


Figure 2.6.9 Categories of OSI Model

A layer in the OSI model communicates with two other OSI layers, the layer directly above it and the layer directly below it. For example, the data link layer in System X communicates with the network layer and the physical layer. When a message is sent from one machine to another, it travels down the layers on one machine and then up the layers on the other machine. This route is illustrated in Figure 2.6.10. As the message travels down the first stack, each layer (except the physical layer) adds a header information to it. These headers contain control information that are read and processed by the corresponding layer on the receiving stack. At the receiving stack, the process happens in reverse. As the message travels up the other machine, each layer strips off the header added by its peer layer.

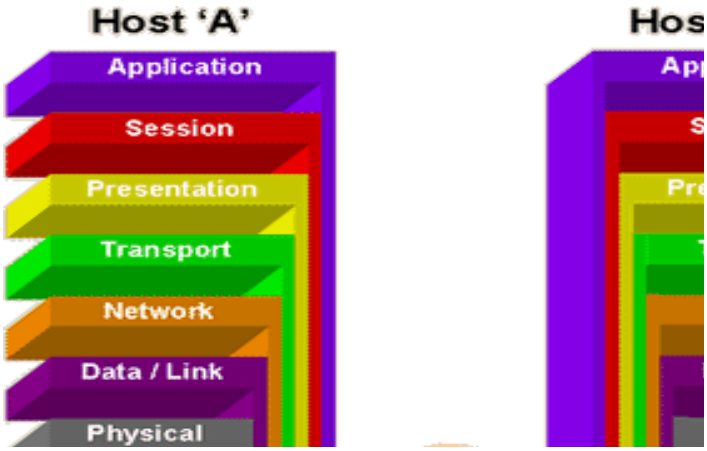


Figure 2.6.10 Message Transfer in Layered Architecture

Physical layer The physical layer is the lowest layer of the OSI model and it defines the physical and electrical characteristics of the network. This layer also defines what kind of network interface card must be installed in each computer and what kind of hubs to be used. In other words, the physical layer is the conduit between the computer's networking hardware and its networking software. Physical layer communication media include various types of copper or fiber optic cable as well as many different wireless solutions. This layer communicates with data link layer and regulates the transmission of a stream of bits over a physical medium. This layer also defines which transmission technique is used to send data over the cable.

Data link layer The function of the data link layer is to transform the data into a line that is free of transmission errors and is responsible for node-to-node delivery. On the sender side, the data link layer divides the stream of bits from the network layer into manageable form known as *frames.* These data frames are then transmitted sequentially to the receiver. Data link layer also performs flow control of the frames in order to match the data transfer speed of the sender to the data processing speed of the receiver. On the receiver end, the data link layer detects and corrects any errors in the transmitted data, which it gets from the physical layer.

Network layer The network layer provides the physical routing of the data, that is, it determines the path between the sender and the receiver. The outbound data is passed down from the transport layer is encapsulated in the network

layer's protocol and then sent to the data link layer for segmentation and transmission. This layer organises frames from the data link layer into packets, that is, the inbound data is de-fragmented in the correct order and then the assembled packet is passed to the transport layer. Network layer also manages traffic problems such as switching, routing and controlling the congestion of data packets. Network layer provides a uniform addressing mechanism so that more than one network can be interconnected.

Transport layer The basic function of the transport layer is to handle error recognition and recovery of the data packets. The transport layer establishes, maintains, and terminates communications between the sender and the receiver. At the receiving end, transport layer rebuilds packets into the original message, and to ensure that the packets arrived correctly, the receiving transport layer sends receipt acknowledgments.

Session layer The session layer organises and synchronises the exchange of data between the sending and the receiving applications. This layer establishes the control between the two computers in a session, regulating which side transmits, when and for how much duration. The session layer lets each application at one end know the status of the other at the other end. An error in sending application is handled by the session layer in such a manner so that the receiving application may know that the error has occurred. The session layer can resynchronise applications that are currently connected to each other. This may be necessary when communications are temporarily interrupted or when an error has occurred that results in loss of data.

Presentation layer The function of presentation layer is to ensure that information sent from the application layer of one system would be readable by the application layer of another system. This is where application data is packed or unpacked, ready for use by the running application. This layer also manages security issues by providing services such as data encryption and compresses data so that fewer bits need to be transferred on the network.

Application layer The application layer is the entrance point that programs use to access the OSI model and utilise network resources. This layer represents the services that directly support applications. This OSI layer is closest to the end user. Application layer includes network software that directly serves the user, providing such things as the user interface and application features such as electronic mail, USENET newsreaders, etc.

2.6.4 Network Devices

In the above section, we discussed that network, hardware, and software

follow rules, called protocols, in order to convey information in an orderly fashion. Now, we shall focus on a particular set of network devices - *Network Interface Card (NIC), Hub, Repeater, Switch, Bridge, Router* and *Gateway.* These devices interconnect individual computers and ensure that they communicate efficiently.

Network interface card Network interface card is the first contact between a machine and the network. It connects clients, servers and peripherals to the network via a port. Most network interfaces come as small circuit board that can be inserted onto one of the computer motherboard's slots. Alternatively, modem computers sometimes include the network interface as part of their main circuit boards (motherboards). Each network interface is associated with a unique address called its *media access control* (MAC) address. The MAC address helps in sending information to its intended destination.

Figure 2.6.11 Network Interface Card

Hub A hub is a small box that connects individual devices on a network so that they can communicate with one another. The hub operates by gathering the signals from individual network devices, optionally amplifying the signals and then sending them onto all other connected devices. Amplification of the signal ensures that devices on the network receive reliable information. A hub can be thought of as the centre of a bicycle wheel, where the spokes (individual computers) meet. Also known as a concentrator, a hub works on the physical layer of the OSI model.

Repeater A repeater is an electronic device that operates on the physical layer of the OSI model. Its job is to regenerate the signal over the same network

because the signals became too weak or corrupted so as to extend the length to which signal can be transmitted over the same network. Signals that carry information within a network can travel a fixed distance before attenuation endangers the integrity of the data.

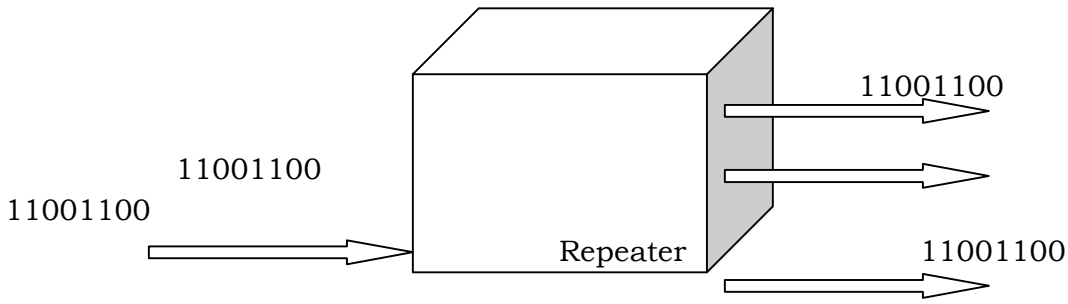


Figure2.6.12 Repeater

A repeater installed on the link receives signal, regenerates it and sends the refreshed copy back to the link Nowadays, the terms repeater and hub are used synonymously, but they are actually not the same. Although at its very basic level, a hub can be thought of as a multiport repeater.

Switch Like a hub, a switch too connects individual devices on a network so that they can communicate with one another. Switches work on the data link layer of the OSI model. Although a switch looks nearly the same as the hub, it generally is more 'intelligent'. Unlike hubs, network switches are capable of inspecting the data packets as they are received, determining the source and destination device of that packet, and forwarding that packet appropriately. Hubs, in contrast, broadcast all information to each connected computer, whether or not that computer is the intended recipient. In this manner, switches help reduce overall network traffic.

Bridge A bridge filters data traffic at a network boundary. It reduces the amount of traffic on a LAN by dividing it into two segments. Bridges operate at the data link layer of the OSI model. It inspects incoming traffic and decides whether to forward or discard it. An Ethernet bridge, for example, it inspects each incoming Ethernet frame (including the source and destination addresses and sometimes the frame size) in making individual forwarding decisions. When a frame enters a bridge, the bridge not only regenerates the signals but also

checks the address of the destination and forwards the new copy only to the segment to which the address belongs.

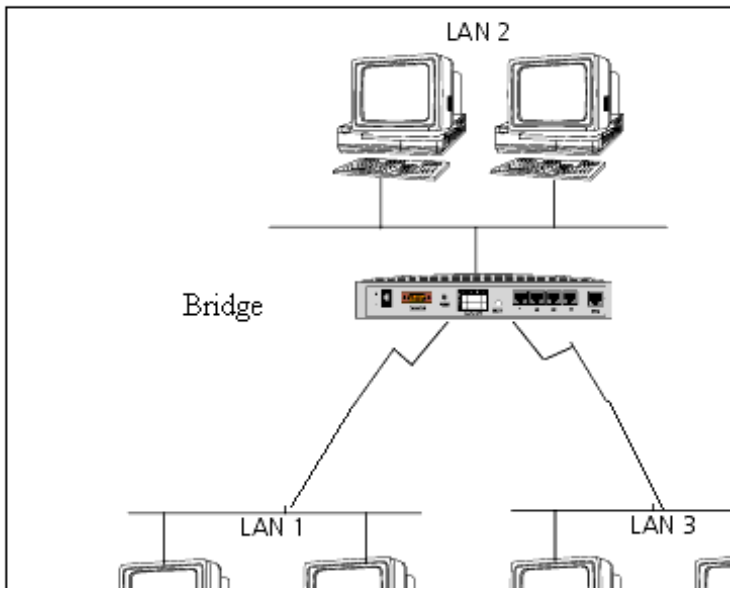


Figure 2.6.13 Bridging Multiple LANs

Router A router is an essential network device for interconnecting two or more networks. Router's sole aim is to trace the best route for information to travel. As network traffic changes during the day, routers can redirect information to take less congested routes. A router creates and/or maintains a table called a 'routing table' that stores the best routes to certain network destinations. Routers are generally expensive and difficult to configure and maintain. They are critical components of a network, if they fail, the network services will be significantly impaired. Most routers operate by examining incoming or outgoing signals for information at the network layer. In addition, they can permit or deny network communications with a particular network.

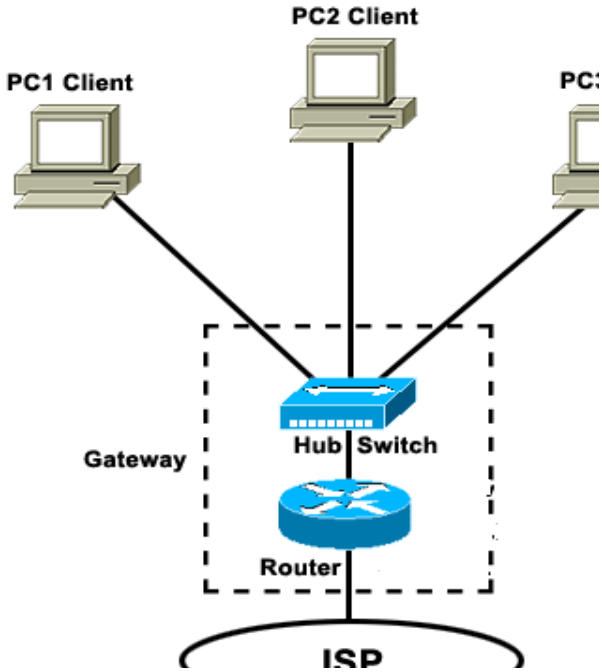


Figure 2.6.14 Router

Gateway A gateway is an internetworking device, which joins two different network protocols together. It works on all the seven layers of the OSI model. Gateways are also known as protocol converters. A gateway accepts the packet formatted for one protocol and converts the formatted packet into another protocol. A network gateway can be implemented completely in software, hardware or as a combination of both.

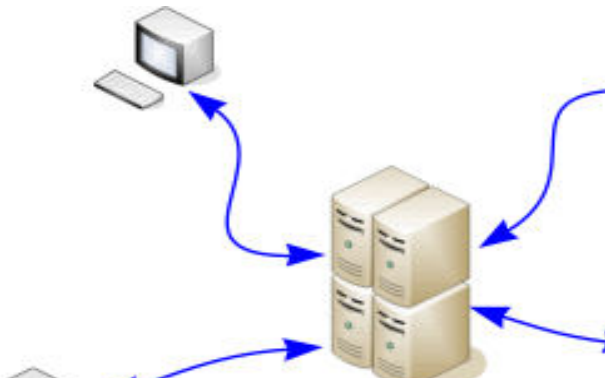


Figure 2.6.15 Gateway

2.6.5 Summary

A *computer network* is a collection of two or more computers, which are connected together to share information and resources. A network can be classified into three categories: *local area network* (LAN), *metropolitan area network* (MAN), and *wide area network* (WAN). *Local area network* (LAN) is a computer network that spans only a small geographical area (usually within a square mile or less), such as an office, home or building. *Metropolitan area network* (MAN) is a network of computers spread over a 'metropolitan' area such as a city and its suburbs. *Wide area network* (WAN) is a system of interconnecting many computers over a large geographic area such as cities, states, countries or even the whole world.

Topology refers to the way a network is laid out, either physically or logically. It is the geometric representation of the relationship of all the links. There are five basic topologies: *bus, ring, star, tree,* and *mesh. Bus topology* network uses a common backbone (a single cable) to connect all devices with terminators at both the ends. In *ring topology,* every node has exactly two neighbours connected to form a ring for communication purposes. In *star topology,* devices are not directly linked to each other but are connected through a hub forming the shape of a star. *Tree topology* consists of groups of star-configured workstations connected to a bus backbone cable. In a mesh topology, every node has a dedicated point-to-point link to every other node.

Open Systems Interconnection (OSI) is a standard reference model for communication between two end users in a network. It consists of seven separate but related layers, namely, *physical, data link, network, transport, session, presentation* and *application.* The seven layers of the OSI reference model can be divided into two categories: *upper layers* (comprises the application, presentation, and session layer) and *lower layers* (comprises the transport, network, data link, and physical layer). Primarily, upper layers deal with application issues, and are implemented only in the software. The highest layer, application, is closest to the end user. The lower layers handle the data transport issues. The physical layer and data link layer are implemented in both hardware and software. The other lower layers, that is, network and transport, are generally implemented only in software.

To communicate over a network, a particular set of network devices such as *network interface card* (NIC), *hub, switch, repeater, bridge, router* and *gateway* are used. These devices interconnect individual computers and ensure that they

communicate efficiently.

16.6 Self Check Exercise :

Q.1 What is a Computer Network ? Explain various types of Computer Networks.

Q.2 What is a Topology ? Explain the various types of Topologies ?

Q.3 Write a detailed note on the OSI model.

Q.4 Write short note on the following :

a.	Hub	b.	Repeater	c.	Switch
d.	Bridge	d.	Router	f.	Gateway

2.6.7 Suggested Readings:

1. Computer Network by Andrew S. Tanenbaum (PHI, New Delhi).
2. Data Communication and Networking by Behrouz A. Forouzam (Tata Mcgraw Hill Publishing Company Limited)
3. Introduction to Data Communications and Networking by Wayne Tomasi (Pearson)
4. Data Communication and Networks by Naveen Kumari (Katson Books)

LESSON NO. 2.7 **AUTHOR : ARUN SHARMA**

DATABASE FUNDAMENTALS

Chapter Outline:

2.7.0 Objectives

- ✓ *Meaning of Data, Information and Knowledge*
- ✓ *Qualities and Need of information*
- ✓ *What is database? Learn terms related to databases.*
- ✓ *Learn Logical Data Concepts*
- ✓ *Learn Physical Data Concepts*

Mankind has always relied upon data and information for its survival and growth. Numerous techniques and devices have been developed to manage and organise the information. Before the development of computers people used to store and process data with the help of papers, file folders, microfilms, etc. Nowadays, computers are commonly used to perform such tasks. They have

replaced millions of pieces of paper, file folders and file cabinets as the principal media for storing important information. Computers have replaced typewriters for creating and modifying documents. They are preferred over electromechanical calculators as the best way to perform mathematical calculations. The advantage of computers over the other tools is that computers accomplish much more, with much faster speeds and with greater accuracy.In today's information society, it is often said that information is power. Due to the rapid growth of information technology in the last few decades, all the organisations and the institutions have realised the value of information as a resource and the importance of the speed and ease with which this resource can be managed. Because of this eternal quest for data management, database technology came into existence. The term database is made up of two separate words, *data* and *base*. Hence, database means that it is a base for data, that is, an assembled group of data. In this lesson, we will learn about databases, the terminology used in the world of databases and various other aspects of database technologies.

2.7.1 Data, Information, and Knowledge

Before defining database, we should know about two terms, *data* and *information* which are used frequently with databases. Data can be anything such as a number, a person's name, images, sounds and so on. Hence, *data* can be defined as a set of isolated and unrelated raw facts, represented by values, which have little or no meaning. They have little or no meaning because they lack a context for evaluation. Usually, the values are represented in the forms of characters, numbers or any symbol such as 'Monica', '35', and 'Chef'. Note that although these words and numbers have certain meaning, it is difficult to figure out exactly what these values signify. However, when the data is processed and converted into a meaningful and useful form, it is known as information. Hence, *information* can be defined as a set of organised and validated collection of data. For example, 'Monica is 35 years old and she is a chef'.

In the strict sense, data refers to the values physically recorded in the database whereas information refers to the conclusion or meaning drawn out from it. With respect to database, these terms are synonymous. In reality, what is information in one sense may be data in another. Sometimes, it is hard to tell the difference between the two. For example, the bill at the grocery store can be used as data as well as information. As information, the bill represents the amount you owe to the store. However, when the store manager uses his

computer to calculate total sales for the day based on all purchases made, it is treated as data. In summary, whether something is data or information depends on how it is used in relation to given context.

Other than data and information, one more term, *knowledge* is frequently used with database technology. *Knowledge* is the act of understanding the context in which the information is used. It can be based on learning through information, experience, guessing and/or intuition. Based on the knowledge, the information can be used in a particular context, for example, if an hotelier uses the information about Monica *(she is a chef)* to hire her, he is using his knowledge. Hence, knowledge can also be referred to as a person's capability and wisdom and how much that person knows about a particular subject. Consequently, it can be said that data constitutes information and information constitutes knowledge.

Figure 2.7.1 Data, Information and Knowledge

Note: The term data processing means the process of collecting all items of data together to produce meaningful information. It can be done either manually or by the use of computers. If data processing is done with the help of computers, it is known as EDP (Electronic Data Processing).

2.7.1.1 Prerequisites of Information

Information is the processed data, on which decisions are taken and the subsequent actions are performed thereafter. For the decisions to be meaningful and useful, the information must possess the following qualities:

1. **Accurate:** To be useful, information must be accurate. The information should be accurate at all levels because all further developments are based on the available information. Accurate information provides a reliable and valid representation of raw facts. The cost of inaccurate or distorted information can be extremely high.
2. **Timely:** Information is appreciated only if it is available on time. If the information is available ahead of the time, its value may be diminished because the information might get archaic or the user may simply forget it. Obviously, any availability after the due time simply has no significance.
3. **Complete:** Complete information tends to be comprehensive in covering the

issue or topic of interest. Without complete information, a decision-maker will get a distorted view of reality. For example, incomplete market information can lead businesses to introduce products and services that customers do not want. As a result, the organisation may incur huge losses.

4. **Precise:** Apart from being complete, information must also be precise, that is, it should be to the point, containing all the essential elements of the relevant subject areas. In order to provide complete information, usually large amount of data is gathered. Important information may be buried in the stacks of such data. You need to work hard to get precise information out of it.
5. **Relevant:** Information is relevant if it can be applied to a specific situation, problem or issue of interest. For example, operation managers need information on costs and productivity whereas marketing managers need information on sales projections and advertising rates. In contrast, product inventory information is not relevant to a payroll manager.

2.7.1.2 Need for Information

Information is an important part of our day-to-day life. Almost all activities of our life are affected by the quantity as well as the quality of information. If you are well informed then the chances of your success increases manifold. Some of the common usage of information is discussed below.

- **Information and Decision-Making:** Every job or task involves decision-making. Decision making is the process of identifying, selecting and implementing the best possible alternative. The right information, in the right form and at the right time, is essential to make correct decisions. For example, based on information about customers, competitors, and production capabilities, a manager may decide to inform top executives that a strategic decision needs to be made. Top executives would use this information to identify alternatives for consideration. Each alternative would then be evaluated based on feasibility, cost, time and other criterions. Based on their assessment, top executives would select the alternative that makes the best business sense and begin its implementation.
- **Information and Communication:** Information is vital for communication and a critical resource for performing work in organisations. Business managers spend most of their day in communicating with other managers, subordinates, customers, vendors, etc. Indeed, management in itself is an information process, which

involves collecting, processing, and distributing information. A manager must keep track of the information flow from sources inside and outside the organisation. He has to process the information and disseminate it. Decisions are taken based on this information.

- **Information and Knowledge:** Information plays a vital role in the accumulation of knowledge. Within a corporation, the importance of efficient use of information (or knowledge), for normal functioning of the corporation, has gained such an importance that it has become a major constituent for the growth and survival of corporation in a competitive environment. The future is shaped by our actions today and our actions today are based upon our knowledge. Therefore, for achieving higher levels of success, one must be well informed and should have clarity of knowledge.
- **Information and Productivity:** Information helps in making sense of our environment, which assists in achieving the performance objectives. In fact, productivity is directly related to the availability and value of the information and its application in the related context. Individuals and organisations need to ensure that their information-processing systems are properly integrated. They should also ensure that the necessary information is being supplied to the right people at the right time in the organisation. For example, it makes sense to share customer complaint data about a specific product with the members of the product development team responsible for redesign, thereby increasing the efficiency as well as productivity.

2.7.2 Database: Definition

A *database* is a logically coherent, organised collection of similar (related) data and database management system is a collection of porgrams that enables its users to access databases, manipulate data, report and represent data. Similar data refers to the collection of data which is stored for the same context. For example, an employee database contains 'similar' data for all employees and every employee's entry contains similar type of information. The organised information (that is, database) serves as a base from which the desired information can be retrieved, conclusions can be drawn and' decisions can be made, by further reorganising or processing this data.

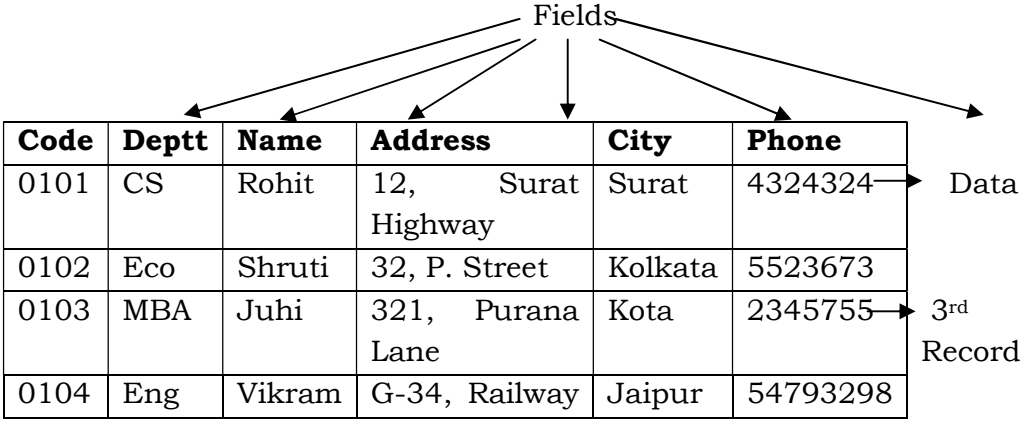
You must have used a database in your everyday life. Dictionary is one of the most common examples of a database, where words are arranged alphabetically. Similarly, a telephone directory is another example of a

database, where the names of telephone subscribers are listed in alphabetical order. Many people keep a list of the addresses of their friends and relatives, which is usually written down in an address book. This list is, in fact, a database of the addresses. Based on this address database, several operations are performed such as if a person moves to a new location, his/her address is changed (that is, editing the database) in the address book. Sometimes new addresses are also added in the address book. When one wants to send a letter to someone, he/she has to look into the book to find the address (that is, searching in the database). If the list is too large (hundreds of addresses are added), finding out a particular address may become cumbersome. Therefore, the addresses are usually classified, based on the initials of the person on different pages for names beginning with different letters.

2.7.2.1 Fundamentals of Database

Databases are used to manage information. Within the database, the data is organised into storage containers called *tables*. Tables are made up of columns and rows. In a table, columns represent individual *fields* and rows represent *records* of data. Various parts of a database are given below.

- **Field:** The smallest unit of data, which has meaning to its users, is called a field. It is also known as data element or elementary data. A field is a specific category of information in a database. It is generally used for a group of alphanumeric characters. In Figure 2.7.2, *Code, Deptt, Name, Address, City* and *Phone* are termed as fields.
- **Record:** A record is a collection of multiple related fields that can be treated as a unit. For example, Figure 2.7.2 contains nine records (0101 to 0109) and each record has six fields. In database terminology, sometimes records are also known as *tuples*.



			Colony		
0105	Pun	Naman	63, Paradise colony	Patiala	9065633 → 5th Record

Figure 2.7.2 Fields and Records in a File

- **File:** A file is a named collection of logically related multiple records. For example, a collection of all the employee records of a company would be an employee file. Note that every record in a file has the same set of fields. Depending on the database software, a file can also be referred to as a *table.*

The database software can save each collection of records (table) in a separate file or they may be saved in a single database, which is logically separated in tables. A collection of employee details file, salary file, job file, and so on constitutes an employee's database. Therefore, database is a collection of multiple related files (tables). A simple database might be a single file containing many records, each of which contains the same set of fields as shown in Figure 2.7.2.

Imagine the employee list as a grid or table, as shown in Figure 2.7.2. Each row in the table contains information about a single employee. The rows are broken up by columns. Each column contains a single part of the employee record. The *Code* column contains employee's code, and every employee's code is listed in this column, one in each row. Similarly, the *Name* column contains employee's name, The employee list contains 'similar' data for all employees. Every employee's record, or row, contains the same type of information. Each has a code, department, name, address, city and phone. The data is also 'structured' so that it can be broken into logical columns, or fields, that contain a single part of the employee record.

2.7.2.2 Some Important Database Related Terms

So far, we have discussed some basic database terminology such as fields, records, and tables. Now let us learn some other frequently used database terms.

Data types A data type determines the type of data that can be stored in a column (field). Although many data types are available, four of the most commonly used data types are *Alphanumeric, Numeric, Boolean* and *DateTime.* Alphanumeric data types are used to store characters, numbers, special characters or combination of any of these. Note that if a numeric value is stored

in an alphanumeric field, the value is treated as a character, not a number. Numeric data types are used to store only numeric values. Boolean data types store only logical value, either true or false. DateTime data type is used to store date and time values. The values for this data type vary widely depending on the database management software being used.

Type of Field ---	Alphanumeric	Numeric	Boolean	DataTime
Field Name------	Name	Salary	Is_Married	Joining_Date
Data ----------	Sachin	12000	False	02/1 0/98

- **Key** A key or key field is a column value in a table that is used to either uniquely identify a row of data in a table, or establish a relationship with another table. For example, the *Code* field in Figure 2.7.2 can be used to uniquely identify a record. A key is normally correlated with one column in table, although it might be associated with multiple columns. Usually a key is used to sort data, that is, arranging the records in ascending or descending order. It is also referred as *sort key, index* or *key word.* Such as, to sort the records based on age, the age field is considered as a key. Most database systems allow more than one key so that the records can be sorted in various ways. Keys can be of three types *primary, foreign,* and *candidate* keys.
- **Primary Key:** The term primary key denotes a key that is chosen by the database designer as the principal means of identifying unique records within a table. The primary key should be choser in such a way that its values must not (or rarely) change. For example, *EmpCode* field can be designated as the primary key because all *EmpCodes* are unique and the value once entered will never changed until the person is in the organisation.
- **Foreign Key:** A foreign key is the combination of one or more columns in a table (parent table; that references a primary key in another table (child table). Figure 2.7.3 illustrates how a foreign key constraint is related to a primary key constraint. The *Item_Code* column in the ITEM table references the *Item_Code* in the PURCHASE table. Here PURCHASE is the parent table and ITEM is the child table.

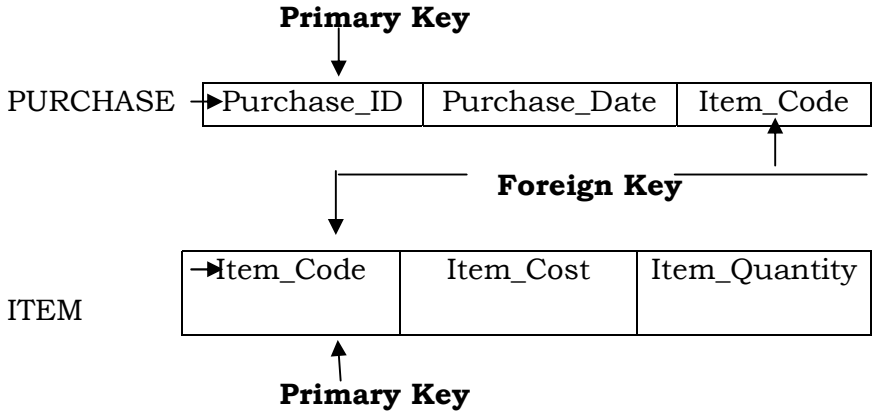


Figure 2.7.3 Foreign Key

Candidate Key: A combination of one or more fields whose value uniquely identifies a record in a table, that is, no two records in a table can have the same key value. For example, usually in an organisation, every employee has a unique *Code.* This field *(Code)* can be used to uniquely identify that employee's record. Since any two employees cannot have the same code, there record can be fetched simply by using their code. Note that every key field is a candidate key but there cannot be more than one primary key in a table.

Apart from the three main keys, other keys such as alternate key, composite key, and secondary key are also used commonly. Any candidate key(s) other than the one chosen as primary, is known as *alternate key. Composite key* is a key composed of more than one column. Sometimes, it is also known as concatenated key or structured key. *Secondary key* is a key used to speed up the searches and retrieval. Contrary to primary key, a secondary key does not necessarily contain unique values. For example, the *Name* column in a CUSTOMERS table may be used for searching records, but the *Name* field can contain duplicate values.

Data dictionary Apart from the data, the database also stores *metadata,* which describes the tables, columns, indexes, constraints and other items that make up the database. In simple words, metadata is data about data. This metadata is stored in an area called the *data dictionary.* Hence, a data dictionary defines the basic organisation of a database. It contains the list of all files in the database, the number of records in each file and the names and types of each field. Data dictionaries do not contain any actual data from the database, but only the book-keeping information for managing it.

Most database systems keep the data dictionary hidden from users to prevent them from accidentally destroying its contents. Different users use the

dictionary in different ways. The database administrator needs a dictionary to help ensure consistency among the data items, to educate users about the database content and to help ensure that different department defines the same data in the same way. The programmers may use it to ensure that they have the name and coding of the data items or segments correct in their programs. Managers may use it as a guide to what data could be made available to them.

2.7.3 Logical Data Concepts

Logical data description refers to the manner in which data is viewed by the programmer or end user. The logical data model is conceptual and it reflects the way the user describes the reality. This model is also known as *entity relationship data model* and it is based on the perception of a real world, which consists of a set of basic objects called *entities, attributes* and *relationships* among these entities.

2.7.3.1 Entity

An entity is an object which has its existence in the real world. It includes all those 'things' about which data is collected. An entity may be a tangible object such as a student, a place or a part. It may also be non-tangible such as an event, a job title or a customer account. For example, if we say that customer buys goods, it means customer and goods are entities. Diagrammatically, entities are represented in rectangles (see Figure 2.7.4).

Figure 2.7.4 Entities

2.7.3.2 Attributes

Attributes are units that describe the characteristics or properties of entities. In a database, entities are represented by tables and attributes are columns or fields. For example, a customer entity may have numerous attributes such as code, name, and address. Similarly, the goods entity may have attributes like code and price (see Figure 2.7.5). They are drawn in elliptical shapes along with the entity rectangles.

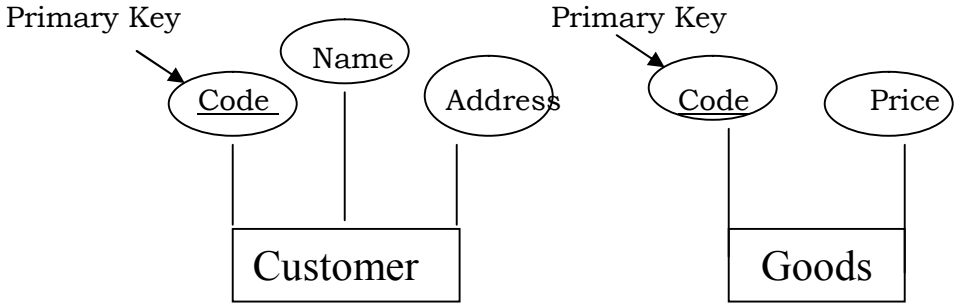


Figure 2.7.5 Entities and Attributes

2.7.3.3 Relationship

Relationship is an association, dependency or link between two or more entities and is represented by diamond symbol. For example, the statement *customer buys goods* is defining the relationship (goods bought) between the two entities, namely, *customer* and *goods.* All relations have three components:

1. **Name:** It is the title or entity identifier such as Goods Bought relation as in Figure 2.7.6.
2. **Degree: It** represents the number of attributes (fields) associated with the table or relation.
3. **Cardinality:** It can be thought of as the maximum number of records in one entity that are linked to a single row in another entity and vice versa. Tuples are the number of records in a relation.

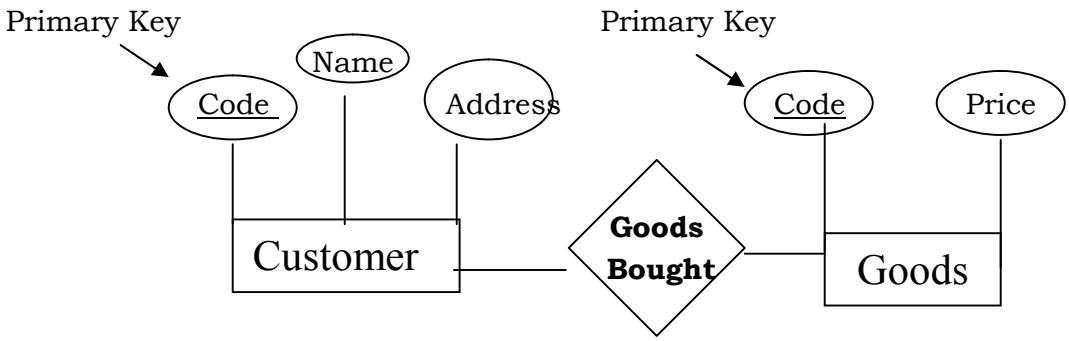


Figure 2.7.6 Entities, Attributes and Relationship

2.7.3.4 Types of Relationship

Even though a relationship may involve more than two entities, the most commonly encountered relationships are binary, involving exactly two entities. Generally, such binary relationships are of three types: *one-to-one, one-to-many,* and *many-to-many.*

One-to-one relationship (1:1) In one-to-one relationship, one record in a table is related to only one record in another table. For example, a department cannot be headed by more than one department head.

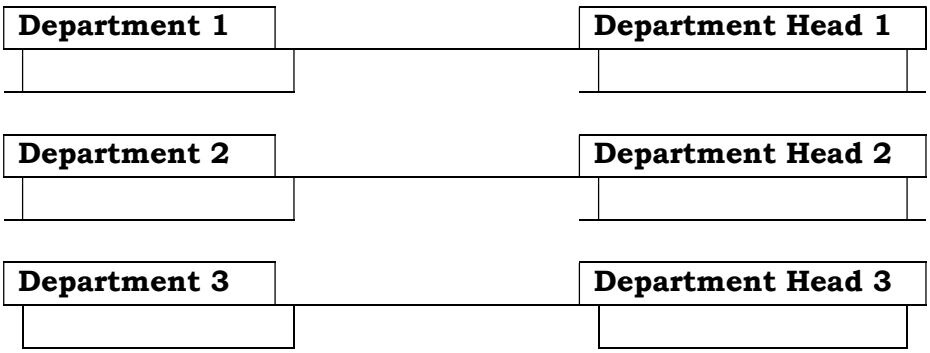


Figure 2.7.7 One-to-One Relationship

One-to-many relationship (1 :M) In one-to-many relationship, one record in a table (parent table) can be related to many records in another table (child table). For example, a father may have more than one child but the child has only one father.

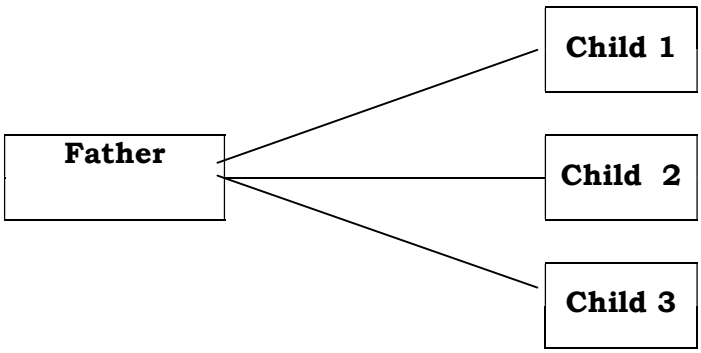


Figure 2.7.8 One-to-Many Relationship

Many-to-many relationship (M:M) In many-to-many relationship, one record in a table can be related to one or more records in another table and one or more records in the second table can be related to one or more records in the first table. For example, a customer can buy many goods and many customers can buy the same good.

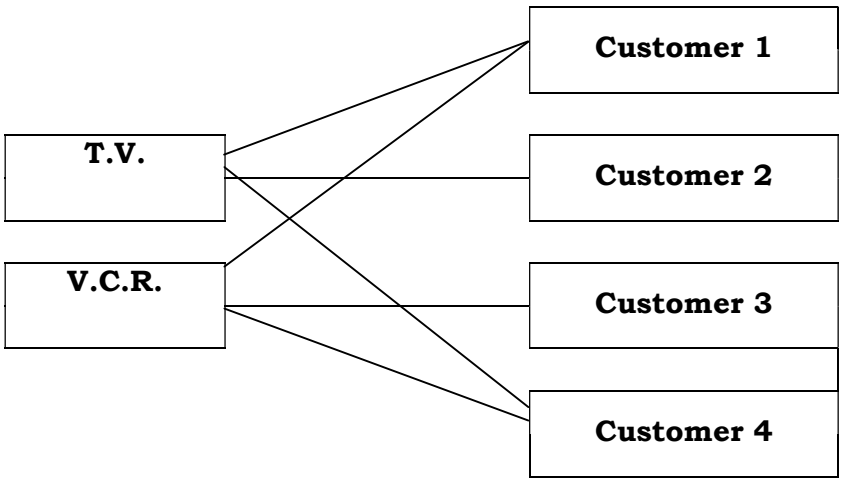


Figure 2.7.9 Many-to-Many Relationship

2.7.4 Physical Data Concepts

Physical concepts of data refer to the manner in which the data is physically stored on the hardware (like hard disk). Fundamentally, it involves the physical organisation of the records of a file for the convenience of storage and retrieval of data. The physical organization is much more related with optimising the use of the storage medium when a particular logical structure is stored on or in it. Usually, the files are organised in three fashions - *sequential, direct* and *indexed sequential*

Sequential files In sequential files, the data is stored and/or retrieved in a logical order, that is, in a sequence. The records are stored one after another in an ascending or descending order, based on the key field of the records. Generally, these files are stored on sequential storage devices such as magnetic tapes and punched cards. In such files, to retrieve a record, all the records must be traversed sequentially before reaching to the desired record. An analogy to sequential files may be taken as an audio cassette. If you are at the first song and want to listen to the fourth song, you will have to traverse through the second and third songs, respectively.

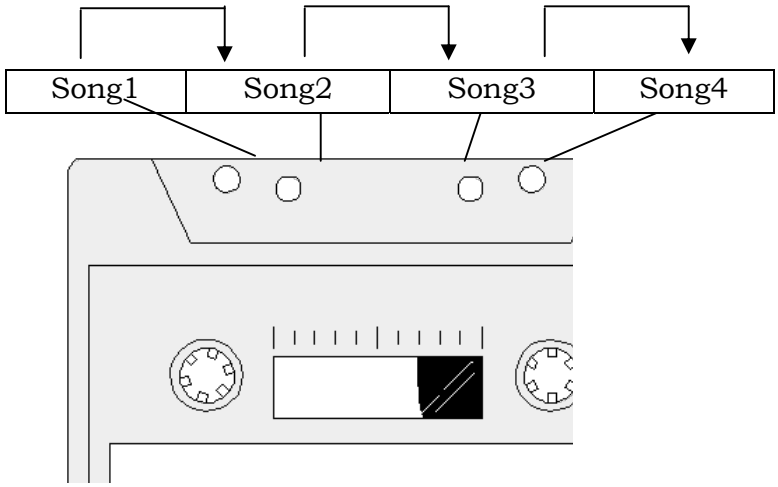


Figure 2.7.10 Sequential Access

Sequential files are easy to organise, maintain, and understand. The hardware, associated with sequential organisation, is also relatively cheaper, compared to other file organisations. However, changing sequential file (such as adding, deleting, and updating records) is a difficult operation. For example, to add a record in the middle of a sequential file, the entire file has to be sorted, rewritten, and stored on the storage device again.

Direct files Direct files facilitate accessing any record directly or randomly without having to traverse the sequence of the records. These files are also known as *random* or *relative* files. Even though only one item can be accessed at a time, but that item may be stored anywhere in the file. For example, in case of CDs (compact discs) any desired song can be played randomly. Generally, these files are stored on direct access storage devices such as hard disks and CDs. However, note that apart from direct access devices, the file itself must have random access facility. Hence, a sequential file stored on direct access device like CD-ROM will process the data sequentially irrespective of the storage medium.

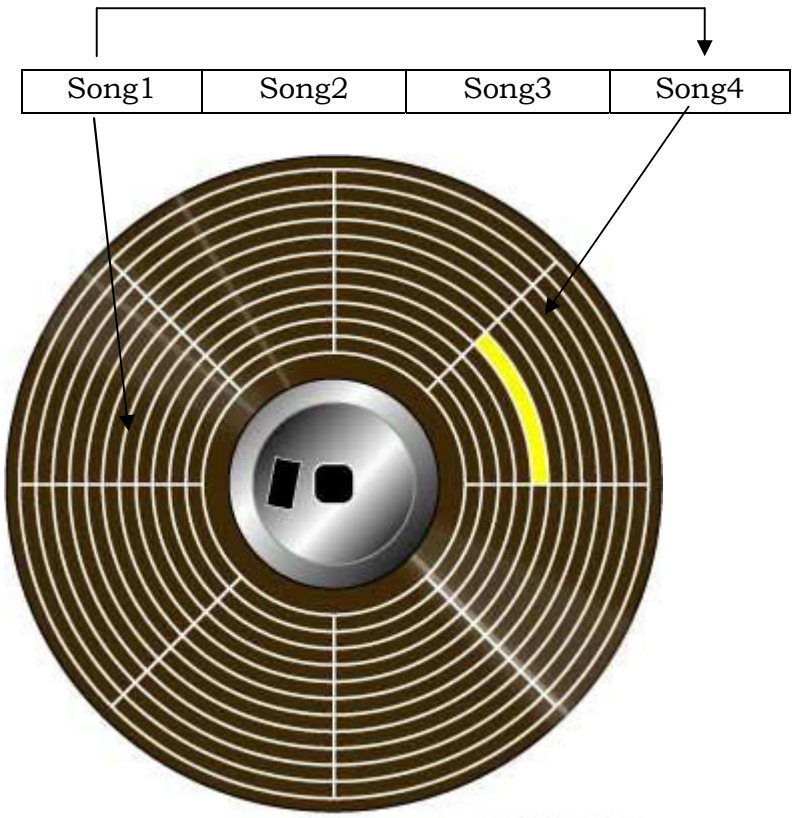


Figure 2.7.11 Direct Access

When the records are stored, the program itself generates the address for the record by applying certain techniques to its key field, which is unique for every record. The relation between key record and location on storage device can be given in form of equation as F (key value) = Record Address. The same formula is used again to retrieve the desired record.

Direct access method facilitates the speedy and direct retrieval of information. Any record can be accessed within a fraction of time because unlike sequential files, this method does not have to traverse a sequence of records. As a result, updation, addition and deletion of record becomes much faster. However, this method is expensive than sequential, because usually the direct files are stored on direct access devices, which are expensive as compared to sequential storage devices. Moreover, direct access is less storage efficient because they tend to generate wide gaps in between any two records. In addition, address generation

overhead is also involved for accessing records due to addressing functions.

Indexed sequential files Before discussing the indexed sequential file organisation, let us first know about indexes. You might have used indexing many times while going through a book. At the back pages of most of the books, certain key words (topics), along with the page numbers in which these topics are discussed are given. In similar fashion, an indexed file includes an index table (also known as reference table) that relates key field values to storage locations of the corresponding records. Hence *indexing* can be defined as a technique of ordering the records in a table without making any modification in the table.

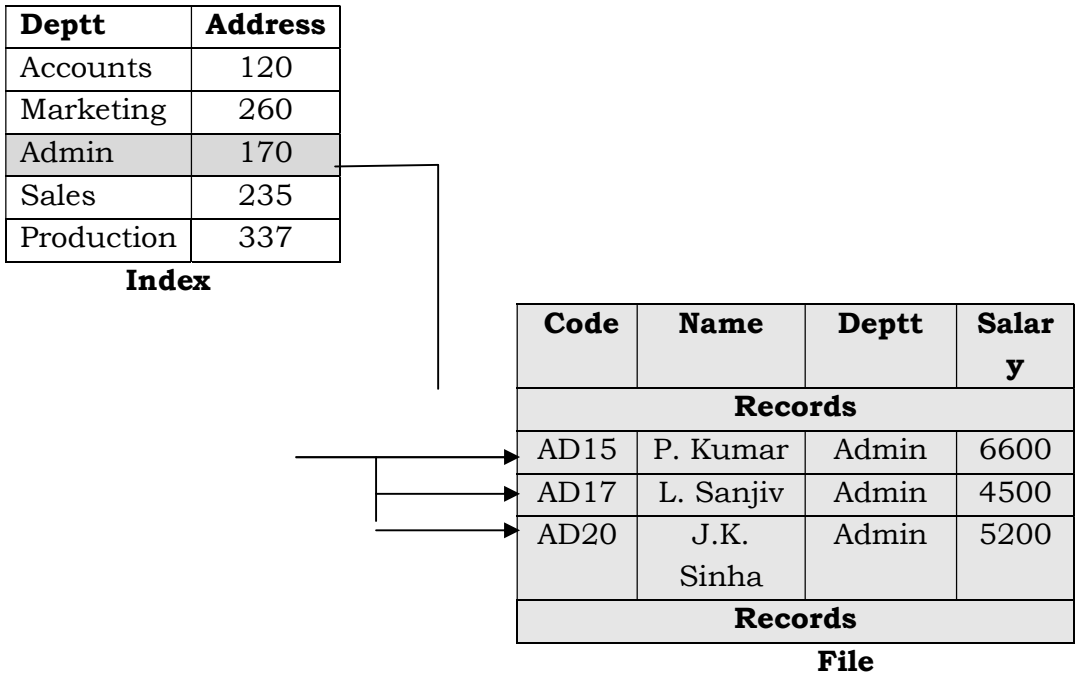


Figure 2.7.12 Indexed Sequential Access

Essentially, indexed sequential technique is a hybrid of sequential and direct file organisation. It provides a combination of access types that are supported by a sequential and a direct file. The indexed file organisation uses a separate index file, which contains the key values and the location of the corresponding record. The records are organised in an orderly sequence and the index table is used to access the records without searching through the entire file. The records may be in random sequence but the index table is stored in sorted sequence on the key values. Since the index table is in sorted sequence, the file

management system simply accesses the records in the order of the index values. To access a particular record, a search is made in the index table to determine the address of the first record of the segment in which the data is placed, then a direct access is made to that address. After that, a sequential search is invoked until the desired record is located.

Index file requires considerably less space than the data file as it requires only two fields - the key field and the address of record's location, even if the record may have numerous fields. However, it is still an overhead, as other file organisations do not use any extra file. Indexes do not affect the file organisation of records physically, that is, various indexes can use the same file without rearranging its records. Indexing speeds up the data retrieval, but it may slow down the update process because indexes also need to be maintained along with the data file while adding and deleting. Since the index sequential files require direct access storage devices, they are an expensive means of file organisation.

2.7.5 Summary

Data is a collection of raw items such as words, numbers, images and sounds that have not been organised and arranged into understandable form. When the data is processed and converted into a meaningful and useful form, it is known as *information.* Hence, information is a set of organised and validated collection of data. *Knowledge* is the act of understanding the context in which the information is used. It can be based on learning through information, experience, guessing and/or intuition.

A *database* is a collection of related data organised in a manner that allows access, retrieval, use and maintenance of that data. Within the database, the data is organised into storage containers called *tables,* which are made up of columns and rows. In a table, columns represent individual *fields* and rows represent *records* of data. A *field* is the smallest unit of data, which has meaning to its users. It is also known as *data element* or *data item.* A *record* is a collection of multiple related fields and a collection of related records is *a file* or *table.*

A *key field* is a column value in a table that is used to either uniquely identify a row of data in a table or establish a relationship with another table. They can be of three types: primary, foreign and candidate keys.

Primary key denotes a key that is chosen by the database designer as the principal means of identifying records within a table. There cannot be more than one primary key in a table. *A foreign key* is the combination of one or more

fields in a table that references a primary key in another table.

Candidate key refers to a combination of one or more fields whose value uniquely identifies a record in a table, that is no two records in a table can have the same key value.

An *entity* is a person, place, thing, activity or event for which data is collected, stored and maintained. Each characteristic describing a particular entity is called an *attribute.* In a database, entities are represented by tables and attributes are columns or fields.

Relationship is an association, dependency or link between two or more entities. Generally, relationships are of three types: one-to-one, one-to-many and many-to-many. The relationship in which one record in a table is related to only one record in another table is known as *one-to-one* relationship. In *one-to-many* relationship, one record in a table can be related to many records in another table. When one record in a table can be related to one or more records in another table, and one or more records in the second table can be related to one or more records in the first table, the relationship is said to be *many-to-many.*

In a sequential file, the data is stored and/or retrieved in a sequence. The records are stored one after another in an ascending or descending order, based on the key field of the records. Usually, these files are stored on sequential storage devices like magnetic tapes.A direct file allows accessing any record directly or randomly without having to traverse the sequence of the records. These files are stored on direct access storage devices like hard disks. *Indexing* is a technique of ordering the records in a table without making any modification in the original table. This technique is used by indexed sequential files, which uses a separate index file containing the key values and the location of the corresponding record. The records are organised in an orderly sequence and the index table is used to access the records without searching through the entire file.

17.6 Self Check Exercise :

Q.1 What is database ?

Q.2 Define Key. Explain different types of keys alongwith suitable examples .

Q.3 What is the difference between unique key and primary key ?

Q.4 Define entity, entity set, attribute, relationship and relationship set.

17.7 Suggested Readings :

1. Fundamentals of Database System by Elmasri & Navathe (Pearson Edition
2. Database System Concepts by Silbwerschatz, Korth and Sudarshwan (Tata McGraw Hill).
3. An Introduction to Database System by C.J.Date (Addison Wesley)

LESSON NO. 2.8 **AUTHOR : ARUN SHARMA**

DATABASE MANAGEMENT SYSTEM

Chapter Outline:

2.8.0 Objectives

- ✓ *How the Database Management System evolved.*
- ✓ *What are the disadvantages of Traditional file–processing system?*
- ✓ *What is Database Management System?*
- ✓ *Advantages and disadvantages of Database Management System.*

A database management system is a software package designed to define, manipulate, retrieve and manage data in a database. Theory of database systems plays an extremely important role in the fields of computer science and information technology. The study is important for applications as simple as storing and retrieving addresses, in which case we store names, addresses and contact information persons and for complex systems like multimedia databases where we store and retrieve images, sounds and video clips etc or geographical information system in which we store and analyze information about maps.

Even in our day-to-day life we encounter several activities that involve some interaction with a database though we may or may not be aware of it. For

example, when we go to bank and make some transaction; when we make reservation for railways, airline or in hotel; or when we order some magazine's subscription or obtain membership of some club or forum, chance are that our activities may involve database in one or the other way.

Therefore, before starting straightaway talking about the concept of Database Management System, it would be more appropriate to discuss how the need for an electronic data management system arose and how the Database Management System evolved.

2.8.1 Evolution of Database System

With the advent of civilization, advancement of living standards, the spread of education, sharing of knowledge and industrial revolution, the need of record keeping also evolved, which included areas of monetary and non-monetary transactions. The records were kept either in simple plain text or in some formatted manner, for example, records of business transactions in business organizations, students' records in educational institute, research data in areas of research etc.

We, in our daily routine, maintain records of one or the other of our activities though we may or may not be putting it on papers. Since our memory is constrained by its capacity, we tend to put, on paper, those records, which are important and shall be required for future referencing or for analysis or otherwise. This method of record keeping is being followed in business organizations, institutes and elsewhere and all data is kept in records on paper, resulting in very huge piles of paper records in the organizations stored for future references.

For example, consider a business organization maintaining its every day transaction vouchers relating to trading of goods, purchase of items, contingency payments and some other routine transactions. The record keeping of the transaction involving money is a statutory requirement also, so as to keep track of sales and income of the business organization for imposing various types of taxes. Otherwise also the business organizations keep records of all their monetary transactions for their future reference and for computing profits and then analyzing the financial position of the organization.

All this information is kept on papers and the resulting pile of records on papers becomes very huge for just one year and normally, business and all other organizations maintain records for many years depending on statutory and organizational policy requirements.

2.8.2 Disadvantages of Traditional file–processing system:

A traditional file processing system has the following major disadvantages:

2.8.2.1 Redundancy of data

In traditional file processing system each and every subsystem maintains its own set of files for storage and retrieval of data and these subsystems maintains a lot of duplicate data, which is basically unnecessary redundancy wasting storage space and access time. For example, name, fathers name, age, address and telephone number belonging to students may be stored both in the office subsystem, for maintaining admission records, and in the library subsystem also for maintaining information about the students. Similarly, in the bank system a customer may be having his name, address and telephone number appearing in two different subsystems, namely, Saving Account and Loan Account. Which is unnecessary redundancy of the data. This redundancy leads to higher storage and access cost. This redundancy in storing the same data many times leads to several problems. First there is duplication of effort of storing the same data at more than one place. Second, storage space is wasted when the same data is stored repeatedly and this problem may be serious for large databases. Third, files that represent the same data may become inconsistent. This may happen because an update is applied to some of the files but not to others.

2.8.2.2 Inconsistency in data

Since, because of redundancy the same data may be lying at more then one location, there are chances of inconsistency in the same data lying at different places. Because the copies of the same data may be having same values because of updation of data at one place without reflecting the updation at the other places. For example, the address of the customer may have changed and updated accordingly in the Saving Account subsystem of the bank but the Loan Account subsystem may be showing the unchanged address of the customer, which is clearly the case of data inconsistency. Even if the updation has been reflected at all the places for the same piece of data, there may still be inconsistency because different persons operate on different parts of the system. For example, date of birth of the student in office subsystem may have been entered as 03-Feb-1972 whereas in library subsystem it may have been entered as 02-Mar-1972.

2.8.2.3 Problem of Security

All the users of the database should not be able to access the whole data. For example, in a banking system, persons operating on Saving Account need to see only that part of the database that has information about the saving Accounts of the customers. They do not need access to information about customer loans. Since application programs are added to the system in an ad hoc manner, it is difficult to enforce such security constraints. It becomes very difficult to check that right persons are getting right type of access on the right data, which is basically the theme of security of database.

2.8.2.4 Problem of Integrity

The data values stored in the database must satisfy certain types of *consistency constraints.* In case of traditional file processing system since applications programs written in some high level language operate directly on data, it becomes difficult to enforce integrity constraint on the data. For example, if there are constraint in the banking system that minimum balance for Saving Account should never be below a certain limit or Loan Account should contain a maximum limit of loan only then these constraints are very difficult to enforce in traditional file processing system. 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.

2.8.2.5 Problem of Concurrency

Concurrency means simultaneous access of the data. Concurrency helps in improvement of the overall performance of the system and facilitates a faster response time. For this many systems allow multiple users to update the data simultaneously. In such an environment, interaction of concurrent updates may result in inconsistent data. For example, if two customers, A and B, of the bank having a joint account with balance Rs. 3500, withdraw amount, say Rs. 2000/- and Rs. 1000/- respectively from the joint account at about the same time, the result of the concurrent executions may leave the account in an incorrect (or inconsistent) state. Suppose that the programs executing on behalf of each withdrawal read the old balance, reduced that value by the amount being withdrawn and wrote the result back. Depending on which one

writes the value last, the account may contain Rs. 1500 or Rs. 2500, rather than the correct value of Rs. 500. To guard against this possibility, the system must maintain some form of supervision.

2.8.2.6 Difficulty in accessing data

Some new queries, which may not have been anticipated and not have been incorporated in the application programs operating on the database, may need to be incorporated in the existing system. For example, if one of the bank officers needs to find out the names of all customers who live within the city's 147001 zip code. The officer asks the data-processing department to generate such a list. Because this request was not anticipated when the original system was designed, 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 (i) obtain the list of all customers and have the needed information extracted manually, or (ii) ask the data-processing department to have a system programmer 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 a Saving Account balance of Rs. 10,000 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 must be developed for general use.

2.8.3 Lack of proper backup and recovery

Traditional file processing systems do not provide proper facilities for backup and recovery of database. In case of some failure the entire database is either corrupted or becomes inaccessible. Consequently recovery of the data becomes very difficult. A computer system, like any other mechanical or electrical device, is subject to failure. In many applications, it is crucial to ensure that, once a failure has occurred and has been detected, the data are restored to the consistent state that existed prior to the failure. For example, consider a program to transfer Rs. 5000 from account A to *B.* If a system failure occurs during the execution of the program, it is possible that Rs. 5000 were withdrawn from account A but were not credited to account B, resulting in an

inconsistent database state. Clearly, it is essential for database consistency that either both the credit and debit occur or neither of them occurs. That is, the funds transfer must be *atomic* it must happen in its entirety or not at all. It is difficult to ensure this property in a conventional file-processing system. And it leads data to an inconsistent state.

2.8.4 Isolated Data

Because data are scattered in various files, and files may be in different formats, it is difficult to write new application programs to retrieve the appropriate data.

These difficulties, among others, have prompted the development of DBMS.

2.8.3 Database Management System

In late 1950s some companies proposed the notion of generalized access to electronically stored data. Here, it becomes relevant to define this notion of *generalized access*. Generalized access means the method by which all sorts of data may be read and manipulated. By that time non-sequential access methods had got evolved because of emergence of magnetic disks, which provided direct access to data.

In 1960s two basic models of data storage and retrieval evolved. First was the Network Data Model developed by Conference on Data Systems Language Database Task Group (CODASYL DBTG) and the Hierarchical Data Model pioneered by IBM.

By this time, simultaneously, the concept of Database Management System also emerged, which is defined in parts and in composite as follows:

2.8.3.1 Database

Data, as already has been defined, is a collection of known facts and figures that can be recorded and have implicit meaning.

Field: Data can be stored in data items and the atomic unit of storing data is called a field. Field is the smallest unit of named data. It may consist of any number of bits or bytes. It's like elementary data types of most of the programming languages. And can be used for storing number and/or characters.

For example, Roll Number can be stored in an integer data item, which may be named meaningfully to "Rollno". Name can be stored in a string or a character array and named accordingly, fee can be stored in real numbers and so on.

Data Aggregate: It is a collection of data items with in a record, which is given a name and is referred to as a whole.

For example, Date may be composed of the data items, month, day and year, similarly, address can be composed of data items Number, Street, City, State, Pincode, Country etc.

Record: A record is a named collection of related data items or data aggregates. When an application program reads data from the database, it may read on complete (logical) record.

For example, A student record may be composed of data items like Roll Number, Name, Father's Name, Permanent Address, Address for Correspondence, Data of Birth, Sex, Data of Admission, Class, etc. Similarly an employee record may be composed of data items like Employee Number, Name, Permanent Address, Address for Correspondence, Date of Birth, Sex, Date of Joining, Department, Category, Basic Pay etc.

Segment: A segment contains one or more data items and is the basic quantum of data which passes to and from the application programs under control of the database management software.

For example, a set of records stored in or read from a database.

File: A file is a basic unit of (logical) storage in the computer system. It is a named collection of all occurrences of a given type of (logical) records.

For example, a file is containing records of all employees of an organization etc.

Database: It is collection of the occurrences of multiple, but related files, containing the relationships among records, data aggregates and data items.

Database System: In most systems the term database does not refer to all record types but to a specified collection of them. There can be several databases in one system. However, the contents of the different databases are assumed to be separate and disjoint. Such collection of databases is called database system.

For example, Students database and employees database constitutes a database system of an educational institute.

2.8.3.2 Management

Management in general means, planning, organizing and controlling. Means plan the tasks to be performs, organize the available resources and control the

working of the all the components of the system. In database system theory, by management we mean management of the database. Which include services like data description, data manipulation and data control. The database manager performs most of the activities in coordination with the database administrator.

2.8.3.3 System

The word "SYSTEM" covers a very broad spectrum of concepts. This is derived from the Greek word systema, which means an organised relationship among the functioning units or components. In our daily life, we come into contact with the transportation system, the communication system, the accounting system, the production system, the economic system and for over four decades, the computer system. Similarly, business systems are the means by which business organisations achieve their predetermined goals. A business system combines policies, personnel, equipment and computer facilities to co-ordinate the activities of a business organisation. Essentially, a system represent, an organised way of achieving the predetermined objective.

There are various definitions of the word system, but most of them seem to have a common idea that suggests that *a system is an orderly grouping of interdependent components linked together according to a plan to achieve a specific goal*. The word component may refer to physical parts (engines, wheels of car), managerial steps (planning, organising, controlling or a subsystem in a multi-level structure). The components may be simple or complex, basic or advanced. They may be a single computer with a keyboard, memory and printer or a series of intelligent terminals linked to a mainframe. In either case, each component is part of the total system and has to do its own share of work for the system to achieve the desired goal. In database theory system means set of software modules which facilitate management of the database.

18.3.4 Database Management System

Having defined the Database Management System in parts now we are ready to combine all the parts and derive one definition for Database Management system or DBMS.

> ***Database Management System is a software system, which is used for defining, constructing, manipulating and controlling database. Defining a database involves specifying the data types, structures and constraints for the data to be stored in the database. Constructing the database is the process of storing the data itself on some storage medium that is controlled by the DBMS.***

Manipulating a database includes such functions as querying the database to retrieve specific data, updating the database to reflect changes and generating reports from the data. Controlling the database mean defining authorization for users and defining their access right.

Databases and database technology is having a major impact on the growing use of computers. It is fair to say that databases will play a critical role in almost all areas where computers are used, including business, engineering, medicine, law, education and library science, to name a few.

A database has the following implicit properties:

- A database represents some aspect of the real world, some times called the Universe of Discourse (UoD). Changes in the mini world are reflected in the database.
- A database is a logically coherent collection of data with some inherent meaning. A random assortment of data cannot correctly be referred to as a database.
- A database is designed, built and populated with data for a specific purpose. It has an intended group of users and some preconceived applications in which these users are interested.

A database can be of any size and varying complexity. For example, the list of names and addresses may consist of only a few hundred records, each with a simple structure. On the other hand, the card catalog of a large library may contain half a million cards stored under different categories - by last name of the primary author, by subject, by book title with each category organized in an alphabetic order.

A database may be generated and maintained manually or by machine. The library card catalog is an example of database that may be manually created and maintained. A computerized database may be created and maintained either by a group of application programs written specifically for that task or by a Database Management System.

A Database Management System (DMBS) is a collection of programs that enables users to create and maintain a database. The DBMS is hence a general purpose software system that facilitates the processes of defining, constructing and manipulating databases for various applications. Defining a database involves specifying the data types, structures and constraints for the data to be stored in the database. Constructing the database is the process of storing the data itself on some storage medium that is controlled by the DBMS. Manipulating a database

includes such functions as querying the database to retrieve specific data, updating the database to reflect changes in the mini world and generating reports from the data.

2.8.4 Advantages of a Database System

One of the main advantages of using a database system is that the organization can exert, via the DBA, centralized management and control over the data. The database administrator is the focus of the centralized control. Any application requiring a change in the structure of a data record require an arrangement with the DBA, who makes the necessary modifications. Such modifications do not affect other applications or users of the record in question. Therefore, these changes meet another requirement of the DBMS: data independence, the advantages of which are discussed in coming sections.

2.8.4.1 Removal of Unnecessary Redundancies

Centralized control of data by DBA avoids unnecessary duplication of data and effectively reduces the total amount of data storage required. It also eliminates the extra processing necessary to trace the required data in a large mass of data. Another advantage of avoiding duplication is the elimination of the inconsistencies that tend to be present in redundant data files. Any redundancies that exist in the DBMS are controlled and the system ensures that these multiple copies are consistent.

In the database approach, the views of different user groups are integrated during database design. For consistency, we should have a database design that stores each logical data item such as a student's name or date of birth in only one place in the database. This does not permit any inconsistency and it saves storage space as well. In some cases, controlled redundancy may be useful. For example, we may store StudentName and Class redundantly in a GRADE_REPORT file (Figure 2.8.1(b)), because, whenever we retrieve a GRADE_REPORT record, we want to retrieve the student name and class along with the grade and Roll number identifier. By placing all the data together, we do not have to search multiple files to collect this data. In such cases the DBMS should have the capability to control this redundancy so as to prohibit inconsistencies among the files. This may be done by automatically checking that the StudentName RollNumber values in any GRADE_REPORT record in Figure 2.8.1(b) match one of the Name-StudentNumber values of a STUDENT record (Figure 2.8.1(a)). Such checks can be specified to the DBMS during database design and automatically enforced by the DBMS whenever the GRADE_ REPORT file is updated. Figure 2.8.1(c) shows a GRADE_REPORT

record that is inconsistent with the STUDENT file of Figure 2.8.1(a), which may be entered erroneously if the redundancy is not controlled.

Student

RollNumber	Name	FatherName	Class	DateofBirth
1101	Rahul	Ramesh	MCA-I	10-10-1981
1102	Mahesh	Puran	MCA-I	12-12-1980
1103	Piyush	Rajesh	MCA-I	11-11-1979

(a)

GRADE_REPORT

Roll Number	Name	Class	Grade
1101	Rahul	MCA-I	A
1102	Mahesh	MCA-I	B
1103	Piyush	MCA-I	A

(b)

GRADE_REPORT

Roll Number	Name	Class	Grade
1102	Ramesh	MCA-I	A

(c)

Figure 2.8. 1 Redundant storage of data items among files.

(a) Student data including RollNumber, Name, FatherName, Class and Age.

(b) Controlled redundancy: Including RollNumber, StudentName and Class fields in the GRADE_REPORT file.

(c) Inconsistency: A GRADE_REPORT record that is inconsistent with the STUDENT records in Figure 2.8.1(a) (the Name of student number 1102 is Mahesh not Ramesh).

18.4.2 Integrity

Centralized control can also ensure that adequate checks are incorporated in the DBMS to provide data integrity. Data integrity means that the data contained in the database is both accurate and consistent. Therefore, data values being entered for storage could be checked to ensure that they fall within a specified range and are of the correct format. For example, the value for the age of an employee may be in the range of 16 and 75. Another integrity check that should be incorporated in the database is to ensure that if there is a reference to certain object, those objects must exist. In the case of an automatic teller machine, for example, a user is not allowed to transfer funds from a nonexistent savings account to a checking account.

Most database applications have certain integrity constraints that must hold for the data. A DBMS should provide capabilities for defining and enforcing these constraints. The simplest type of integrity constraint involves specifying a data type for each data item. For example, in Figure 2.8.1(a), we may specify that the value of the Class data item within each STUDENT record must be an integer between 1 and 5 and that the value of Name must be a string of no more than 30 alphabetic characters. A more complex type of constraint that occurs frequently, involves specifying that a record in one file must be related to records in other files. For example, in Figure 2.8.1(a) and 2.8.1(b), we can specify that "every Grade_Report record must be related to a Student record." Another type of constraint specifies uniqueness on data item values, such as "every Student record must have a unique value for RollNumber." These constraints are derived from the meaning or semantics of the data and of the mini-world it represents. It is the database designers' responsibility to identify integrity constraints during database design. Some constraints can be specified to the DBMS and automatically enforced. Other constraints may have to be checked by update programs or at the time of data entry.

A data item may be entered erroneously and yet still satisfy the specified integrity constraints. For example, if a student receives a grade of A but a grade of C is entered in the database, the DBMS cannot discover this error automatically, because C is a valid value for the Grade data type. Such data entry errors can only be discovered manually (when the student receives the grade and complains) and corrected later by updating the database. However, a grade of X can be automatically rejected by the DBMS, because X is not a valid value for the Grade data type.

2.8.4.3 Security

Data is of vital importance to an organization and may be confidential. Unauthorized persons must not access such confidential data. The DBA who has the ultimate responsibility for the data in the DBMS can ensure that proper access procedures are followed, including proper authentication schemes for access to the DBMS and additional checks before permitting access to sensitive data. Different levels of security could be implemented for various types of data and operations. The enforcement of security could be data-value dependent (e.g., a manager has access to the salary details of employees in his or her department only), as well as data-type dependent (but the manager cannot access the medical history of any employees, including those in his or her department).

When multiple users share a database, it is likely that some users will not be authorised to access all information in the database. For example, financial data is often considered confidential and hence only authorised persons are allowed to access such data. In addition some users may he permitted only to retrieve data, whereas others are allowed both to retrieve and to update. Hence, the type of access operation-retrieval or update must also be controlled. Typically, users or user groups are given account numbers protected by passwords, which they can use to gain access to the database. A DBMS should provide a security and authorisation subsystem, which the DBA uses to create accounts uses to specify account restrictions. The DBMS should then enforce these restrictions automatically. Notice that we can apply similar controls to the DBMS software. For example, only the DBA'S staff may be allowed to use certain privileged software, such as the software for creating new accounts.

2.8.4.4 Conflict Resolution

Since the database is under the control of the DBA, he should resolve the conflicting requirements of various users and applications. In essence, the DBA chooses the best file structure and access method to get optimal performance for the response-critical applications, while permitting less critical applications to continue to use the database, albeit with a relatively slower response.

2.8.4.5 Data Independence

Data independence is defined as the immunity of applications to change in storage structure and access strategy - which implies that the applications concerned do not depend on any one particular storage structure and access method. Data Independence has been fully explained later.

2.8.4.6 Sharing of Data

A database allows the sharing of data under its control by any number of application programs or users. Therefore, new applications can be developed to operate against that same stored data. In other words, the data requirements of new applications may be satisfied without having to create any new stored files.

2.8.4.7 Persistent Storage for Program Objects and Data Structures

A recent application of databases is to provide persistent storage for program objects and data structures. This is one of the main reasons for the emergence of the object oriented DBMS. Programming languages typically have complex data structures, such as record types in PASCAL or class definitions in C++. The values of program variables are discarded once a program terminates, unless the programmer explicitly stores them in permanent files, which often involves converting these complex structures into a format suitable for file storage. When the need arises to read this data once more, the programmer must convert from the file format to the program variable structure. Object-oriented database systems are compatible with programming languages such as C++ and SMALLTALK and the DBMS software automatically performs any necessary conversions. Hence, a complex object in C++ can he stored permanently in an object-oriented Database Management System. Such an object is said to be persistent, since it survives the termination of program execution and can later be directly retrieved by another C++ program.

The persistent storage of program objects and data structures is an important function of database systems. Traditional database systems often suffer from the so-called impedance mismatch problem, since the data structures provided by the DBMS were incompatible with the programming language's data structures. Object-oriented database systems typically offer data structure compatibility with one or more object-oriented programming languages.

2.8.4.8 Database Inferencing Using Deduction Rules

Another recent application of database systems is to provide capabilities for defining deduction rules for inferencing new information from the stored database facts. Such systems are called deductive database systems. For example, there may be complex rules in the mini-world application for determining when a student is on probation. These can be specified declarative as deduction rules, which when executed can determine all students on probation. In a traditional DBMS, an explicit procedural program code would have to be written to support such applications. But if the mini-world rules

change, it is generally more convenient to change the declared deduction rules than to re-code procedural programs.

2.8.4.9 Providing Multiple User Interfaces

Because many types of users, with varying levels of technical knowledge, use a database, a DBMS should provide a variety of user interfaces. These include query languages for casual users, programming language interfaces for application programmers, forms and command codes for parametric users and menu-driven interfaces and natural language interfaces for stand-alone users.

2.8.4.10 Representing Complex Relationships among Data

A database may include numerous varieties of data that are interrelated in many ways. A DBMS must have the capability to represent a variety of complex relationships among the data as well as to retrieve and update related data easily and efficiently.

2.8.4.11 Providing Backup and Recovery

A Database Management System must provide facilities for recovering from hardware or software failures. The backup and recovery subsystem of the DBMS is responsible for recovery. For example, if the computer system fails in the middle of a complex update program, the recovery subsystem is responsible for making sure that the database is restored to the state it was in before the program started executing. Alternatively, the recovery subsystem can ensure that the program is resumed from the point at which it was interrupted so that its full effect is recorded in the database.

2.8.5 Disadvantages of Database Management System

In spite of the above advantages, there are some situations where using a DBMS may incur unnecessary overhead costs as compared to traditional file processing, which is a significant disadvantage of the Database Management System. In addition to the cost of purchasing or developing the software, the hardware has to be upgraded to allow for the extensive programs and the workspaces required for their execution and storage. The processing overhead incurred by Database Management System to implement security, integrity and sharing of the data causes a degradation of the response and through-put times. An additional cost is that of migrating from a traditionally separate application environment to an integrated one.

While centralization reduces duplication, the lack of duplication required that the database be adequately backed-up, so that in case of failure, the data can be recovered. Backup and recovery operations are fairly complex in a Database Management System environment and this is exacerbated in a concurrent

multi-user database system. Furthermore, a database system requires a certain amount of controlled redundancies and duplication to enable access to related data items.

Centralization also means that the data is accessible from a single, source, namely the database. This increases the potential severity of security breaches and disruption of the operation of the operations of the organization because of downtime and failures. The replacement of a monolithic centralized database by a federation of independent and cooperating distributed databases, resolves some of the problems resulting from failure and downtimes.

The overhead costs of using a Database Management System are due to the following:

- High initial investment in hardware, software and training.
- Generality that a Database Management System provides for defining and processing data.
- Overhead for providing security, concurrency control, recovery and integrity functions.

Additional problem may arise if the database designer and DBA do not properly design the database or if the database system applications are not implemented properly. Because of the overhead costs of using a Database Management System and the potential problems of improper administration, it may be more desirable to use regular files under the following circumstances:

- The database and applications are simple, well defined and not expected to change.
- There are stringent real-time requirements for some programs that may not be met because of Database Management System overhead.
- Multi user access to data is not required.

2.8.6 Need of Databases

Having discussed advantages and disadvantage of the DBMS approach now we are able to analyze the need of the database. From the above discussion it is clear that advantages of the DBMS approach clearly outweigh disadvantage of the approach. Therefore, if the cost of installing DBMS is affordable for the organization, then using DBMS for data management can be of great help. Though in single user environment, where database is very small, the advantages may not be so obvious. But consider a case of multi user environment and a very large database, the advantages of a database system over traditional file processing system will surely be more readily apparent. Therefore in case of large databases and multi user environment the DBMS

approach should be followed to over come the problems arising out of the use of traditional file processing system.

2.8.7 Summary

With the advent of civilization, advancement of living standards, the spread of education, sharing of knowledge and industrial revolution, the need of record keeping also evolved. A traditional file processing system has major disadvantages like redundancy of data, inconsistency of data, difficulty in accessing data, and lack of proper recovery and backup. A Database Management System (DMBS) is designed to remove all these problems. A Database Management System (DMBS) is a collection of programs that enables users to create and maintain a database. Therefore in case of large databases the DBMS approach should be followed to over come the problems arising out of the use of traditional file processing system.

2.8.8 Self Check Exercise :

Q.1 Explain the various disadvantages of Traditional File Processing System ?

Q.2 What is Database and Database System ?

Q.3 Explain the components of a Database System.

Q.4 Define DBMS. What are the advantages of DBMS ?

2.8.9 Suggested Readings :

1. Fundamentals of Database System by Elmasri and Navathe (Pearson Education)
2. Database System Concepts by Silberschatz, Korth and Sudarshan (Tata McGraw Hill)
3. An Introductin to Database System by C.J.Date (Addison Wesley)