UNIT 3 COMPUTER SOFTWARE

Structure

3.1 INTRODUCTION

The word **software** collectively refers to various kinds of programs used to operate computers and related devices. A **program** is a sequence of instructions that a computer can interpret and execute. Programs can be built into the hardware itself, or they may exist independently in a form known as software. Hardware describes the physical components of computers and related devices.

Software may be distributed on floppy disks, CD-ROMs, and on the Internet. It is usually stored on an external long-term memory device, such as a hard drive or magnetic diskette. When the program is in use, the computer reads it from the storage device and temporarily places the instructions in random access memory (RAM). The process of fetching and then performing the instructions is called “running,” or “executing,” a program. Software programs and procedures that are permanently stored in a computer’s read-only memory (ROM) are called firmware.

The two main types of software are system software and application software. Application software consists of programs that are aimed to help users in solving particular computing problems. Microsoft Internet Explorer for web browsing, Adobe Photoshop for developing computer graphics, Yahoo Messenger for instant messaging all lies in the application software category. The other class of software is the system software, which encompasses the programs that heavily interact with computer resources and provide services to other programs. Popular examples in this are Operating Systems (OS), hardware drivers, compiler etc.

3.2 OBJECTIVES

After reading this unit, you should be able to:

- Describe about the different types of operating systems and their functions;
- State the characteristics of system software and application software;
- Differentiate between compiler and interpreters;
- Enumerate the advantages and disadvantages of compilers and interpreters;
- Define and explain the concepts and philosophy of open-source software; and
- Explain the process of software acquisition.

3.3 SYSTEM SOFTWARE

System software co-ordinates the various parts of computer system and mediates between the application software and computer hardware. Operating system is system software, which manages and controls the computers activities. The other system software consists of computer language translation programs that convert programming languages into machine language and utility programs that perform common processing tasks.

3.3.1 Operating Systems

An **operating system** is a set of computer programs that controls the computer hardware and acts as an interface with the application programs. The operating system plays a central role in the functioning of a computer system. It is usually stored on disk, after the computer system is started or booted up portions of operating system are transferred to memory as required. The **kernel** as the name suggests is the heart of the operating system and controls the most critical processes. Windows by Microsoft, Linux, UNIX, and the Macintosh are the commonly used operating systems.

In some specialized or embedded computers the operating instructions are contained in their circuitry; common examples are the microcomputers found in calculators, automobile engines, mobile phones and microwave ovens.

Functions of Operating System

An operating system performs allocation and assignment of system resources, schedules the use of computer resources, monitors the computer system activities etc. The various activities performed by a typical operating system are:

- Performing common computer hardware functions.
- Providing a user interface
- Providing a degree of hardware independence
- Managing system memory
- Managing processing tasks
- Providing networking capability
- Controlling access to system resources
- Managing files

Common Hardware Functions

All application programs must perform certain tasks. For example

- Getting input from the keyboard or some other input devices
- Retrieving data from disks
- Storing data on disks
- Displaying information on a monitor or printer

Each of these basic functions requires a more detailed set of instructions to complete. The operating system converts a simple, basic instruction into the set of detailed instructions required by the hardware. In effect, the operating system acts as intermediary between the application program and the hardware. The typical OS performs hundreds of such functions, each of which is translated into one or more instructions for the hardware. The OS notifies the user if input/output devices need attention, if an error has occurred, or if anything abnormal has happened in the system.

One of the most important functions of any operating system is providing a user interface. A user interface allows individuals to access and command the computer system. The first user interfaces for mainframe and personal computer systems were command based. A command-based user interface requires that text commands be given to the computer to perform basic activities. For example, the command ERASE 00TAXRTN would cause the computer to erase or delete a file called 00TAXRTN. RENAME and COPY are other examples of commands used to rename files and copy files from one location to another. Many mainframe computers use a command-based user interface. In some cases, a specific job control language (JCL) is used to control how jobs or tasks are to be run on the computer system.

A graphical user interface (GUI) uses pictures (called icons) and menus displayed on screen to send commands to the computer system. Many people find that GUIs are easier to use because user intuitively grasp the functions. Today, the most widely used graphical user interface is Windows by Microsoft. Alan Kay and others at Xerox PARC (Palo Alto Research Center, located in California) were pioneers in investigating the use of overlapping windows and icons as an interface. As the name suggests, Windows is based on the use of a window, or a portion of the display screen dedicated to a specific application. The screen can display several windows at once. The use of GUIs has contributed greatly to the increased use of computers because users no longer need to know command-line syntax to accomplish tasks.

Hardware Independence

The applications make use of the operating system by making requests for services through a defined application program interface (API). Programmers can use APIs to create application software without having to understand the inner workings of the operating system.

Suppose a computer manufacturer designs new hardware that can operate much faster than before. If the same operating system for which an application was developed can run on the new hardware, minimal (or no) changes are needed to the application to enable it to run on the new hardware. If APIs did not exist, the application software developers might have to completely rewrite the application program to take advantage of the new, faster hardware.

Memory Management

The purpose of memory management is to control how memory is accessed and to maximize available memory and storage. The memory management feature of many operating systems allows the computer to execute program instructions effectively and with speed.

Memory controller allows the computer system to efficiently and effectively store and retrieve data and instructions and to supply them to the CPU. Memory management programs convert a user’s request for data or instructions (called a logical view of the data) to a physical location where the data or instructions are stored. A computer understands only the physical view of data –that is, the specific location of the data in storage or memory and the techniques needed to access it. Memory controller converts a logical address to a physical address.

Memory management is important because memory can be divided into different segments or areas. Some computer chips provide “rings” of protection. An operating system can use one or more of these rings to make sure the application programs do not penetrate an area of memory and disrupt the functioning of the operating system, which could cause the computer system to crash. Memory management features of

today's operating system are needed to make sure that application programs can get the most from available memory without interfering with other important functions of the operating system or with other application programs.

Most operating systems support **virtual memory,** which allocates space on the hard disk to supplement the immediate, functional memory capacity of RAM. Virtual Memory works by swapping programs or parts of programs between memory and one or more disk drives using a concept call **paging.** This reduces CPU idle time and increases the number of jobs that can run in a given time span.

Processing Task

Managing all processing activities is accomplished by the task management features of operating systems. Task management allocates computer resources to make the best use of system assets. Task management software may permit one user to run several programs or tasks at the same time (multitasking) and allow several users to use the same computer at the same time (time-sharing).

An operating system with multitasking capabilities allows a user to run more that one application at the same time. Without exiting a program, user may start another application and switch to newly started application, and then jump back to the first program, picking up where it was left off. Better still, while user is working in the *foreground* on one program, one or more other applications can be churning away, unseen, in the *background*, sorting a database, printing a document, or performing other lengthy operations. In the absence of starting background processes, a foreground process would monopolize the computer and if the process happens to be non-interactive such as a file print, then the user will just have to sit and stare at the screen till the process completed. Multitasking can save users a considerable amount of time and effort.

Time-sharing allows more than one person to use a computer system at the same time. For example, 15 customer service representatives may be entering sales data into a computer system for a mail-order company at the same time. In another case, thousands of people may be simultaneously using an on-line computer service to get stock quotes and valuable business news.

Time-sharing works by dividing time into small CPU processing time slices, which can be a few milliseconds or less in duration. During a time slice, some tasks for the first user are done. The computer then goes from that user to the next. During the next time slice, some tasks for the next user are completed. This process continues through each user and cycles back to the first user. Because the CPU processing time slices are small, it appears that all jobs or tasks for all users are being completed at the same time. In reality, each user is sharing the time of the computer with other users.

The ability of a computer to handle an increasing number of concurrent users smoothly is called scalability. This is a critical feature for systems expected to handle a large numbers of users such as a mainframe computer or a Web server. Because personal computer operating systems usually are orientated toward single users, the management of multiple-user tasks often is not needed.

Networking Capability

The operating system can provide features and capabilities that aid users in connecting to a computer network. For example, Apple computer users have built-in network access through AppleShare feature, and the Microsoft Windows operating systems come with the capability to link users to the Internet.

Access to System Resources

Computers often handle sensitive data that can be accessed over networks. The operating system needs to provide a high level of security against unauthorized access to the users' data and programs. Typically, the operating system establishes a log-on procedure that requires users to enter an identification code and a matching password. If the identification code is invalid or if password does not go with identification code, the user cannot gain access to the computer. The operating system also requires that user passwords be changed frequently. If a user is successful in logging onto the system, the operating system records the details of the user and system usage. In some, organizations, these records are also used to bill users for system and resource usage. The operating system also reports any attempted breaches of security.

File Management

An operating system performs file management functions to ensure that the files are available to CPU when needed and that they are protected from access by unauthorized users. Many computers support multiple users who store files on centrally located disks or tape drives. The operating system must be able to resolve what to do if more than once user requests access to the same file at the same time. Even on stand-alone personal computers with only one user, file management is needed to keep track of where files are located, what size they are, when they were created, and who created them.

3.3.2 Language Translators

The CPU (also called processor) of a computer understands commands in machine language, where each instruction is a series of binary digits. Programming in machine language is not easy, as programmers have to remember the machine codes, which are in binary format. To help programmers, other high level programming languages have been developed whose instructions are easy to remember for programmers as these languages use English words. C, Java, SQL are examples of high level programming languages. Programming languages can be divided into assembly languages and high-level programming languages.

For any program to be executed, it has to be first converted into its equivalent machine language program and then loaded into the memory of computer. To perform the translations of programs, language translators are used. As the process of programming language translations are machine dependent, the translators fall in the system software category.

Assemblers: The computer software that translates the assembly language programs into corresponding machine language programs are known as *assemblers*. Assembly language uses mnemonics instead of binary codes used in machine language. For example ADD R1 R2 is an assembly language instruction for adding the contents of register R1 with the contents of register R2 and store the result in R1. The use of mnemonics helps programmers to remember programming codes. But still to write big programs like a word processing software can be very cumbersome in assembly language.

Compiler and Interpreter: Compiler and interpreter are used to translate a high level programming language program into a machine language program. As the translation process is very cumbersome, some compilers first translate the source code (the program in high-level language) into the equivalent assembly language program and then use the assemblers for the next step. To define, a compiler is a program that translates a source text written in a language A into a target program in language B, whereas, interpreter is a program which directly executes the program in a given programming language A.

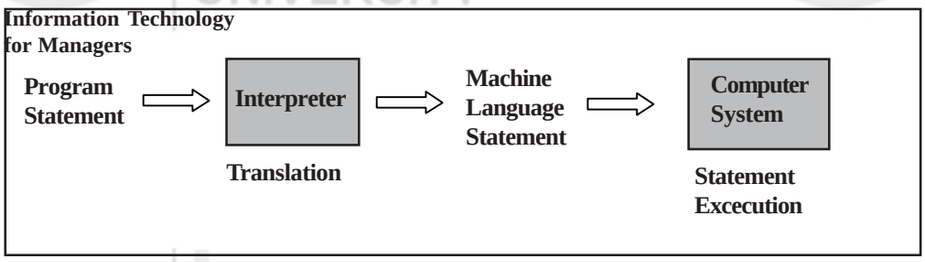


Fig. 3.1 : Working of Interpreter

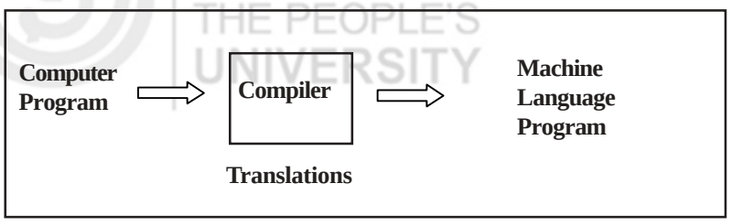
Difference between Compiler and Interpreter

An interpreter is a language translator that reads the source code line-by-line and executes them one by one. On the other hand, a compiler first reads the complete source code and then generates its object code (the equivalent machine language program).

Interpreter has the advantage that the process of translation takes less time as compared to the compiler, but the program generated is less efficient as compared to the compiler in terms of the time is to takes to execute the program.

Compilers are generally used when the efficiency of the generated binary (machine language) program is desired. Interpreters are mostly used for educational purposes, where the programmer makes frequent translations of the program as the changes to the program are frequent. Also, as the interpreter generates the binary program on the fly, platform independence can be achieved. Platform independence means the capability of the program to be translated on any platform and executed on any platform. By platform we mean the hardware and the operating system running on that system. For example, interpreted languages such as HTML, Perl, and Lisp are platform independent. Java, which is a half-compiled half-interpreted language, has benefits of both. As Java programs are compiled to byte-code, instead of binary code, it can be ported to any platform which has the Java Virtual Machine software, which interprets the byte-code. By compiling the program to byte code certain level of optimization that are possible in compilers are achieved.

Step 1 : Program Execution:



Step 2 : Program Execution:

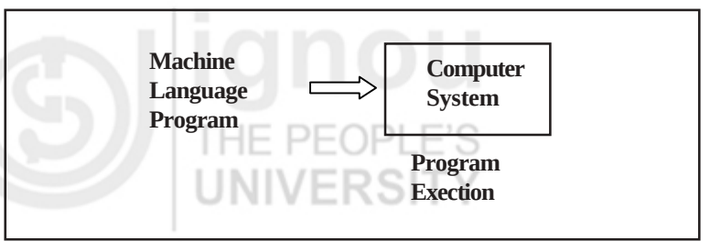


Fig. 3.2. Working of Compiler

Advantages and Disadvantages of Interpreter over Compiler

Advantages

- As compared to compiler no synthesis phase is required in interpreter: Neither there is a need to learn target language B nor the target code is to be generated. Thus interpreters don't have synthesis phase.
- Direct Execution: There is no intermediate compilation phase so the code is directly executed.

Disadvantages

- Efficiency Loss: As the code is executed on the fly, the efficiency of the program is low. In compilers, there is a separate phase for optimization of the program code.
- Interpreter must be available on target machine: The compiled code can be executed on any similar machine. The code needs not to be compiled every time. For languages, which are interpreter based, the interpreter must be available on each machine where the code is to be executed.

Activity A

Why is there a difference between High-Level and low level languages?

.....

.....

.....

.....

.....

3.3.3 Utility Programs

A utility program is designed for general support to the processes of a computer. They are usually for routine, repetitive tasks and many users share them. Examples of utility programs include diagnostic programs, trace programs, input routines, and programs used to perform routine tasks, *i.e.,* perform everyday tasks, such as copying data from one storage location to another. Utility programs are also available commercially; for example, Norton Utilities package is a set of utility programs for checking disks for computer viruses, checking hard drive for bad locations and removing them and for performing disk compression.

3.4 APPLICATION SOFTWARE

Application software is a complete, self-contained program that performs a specific function directly for the user. This is in contrast to system software, which exists to support application programs. Application software may consist of a single program, such as an image viewer; a small collection of programs (often called a software package) that work together to accomplish a task, such as a spreadsheet or text processing system; a larger collection (often called a software suite) of related but independent programs and packages that have a common user interface or shared data format, such as Microsoft Office, which consists of closely integrated word processor, spreadsheet, database manager, etc.; or a software system, such as a database management system, which is a collection of fundamental programs that may provide some service to a variety of other independent applications. Some of the example application software according to their types are given below in the table.

Table 3.1: Examples of Application Software

Type	Software
Word Processing	Microsoft Word, Corel Word Perfect
Spreadsheet	Microsoft Excel, Lotus 1-2-3
Graphics	Adobe Illustrator, Macromedia FreeHand
Desk Top Publishing	Quark Express, Adobe Page Maker, Corel Ventura Publisher

Activity B

Explain the difference between Applications / Utilities with the help of examples.

.....

.....

.....

.....

.....

3.4.1 Programming Languages

A programming language is an artificial language (as opposed to natural languages such as Hindi, English etc.) that is used to generate or to express computer programs. Both system software and application software are developed using one or many programming languages.

Generations of Programming Languages

The programming languages have been divided into different generations according to their characteristics and capabilities.

- **Machine Language** The first generation of computer programming languages is machine language. Programs in machine language consist of instructions coded in of 0s and 1s, thus the alphabets of machine language are 0 and 1. The storage locations and data items are also specified using 0s and 1s. These languages are machine dependent. There is a machine language corresponding to each microprocessor available.

- **Assembly Language** The second generation of computer programming language started using mnemonics (like ADD, SUB) to represent machine language instructions and storage locations. Assembly language is also machine-dependent. System software or at least part of it is usually developed in assembly languages.

- **Third Generation Language (3GL)** 3GL are English-like languages. They use statements and commands, which are similar to the words used in English. 3GLs are easier to learn, but less efficient in the use of computer resources as compared to machine and assembly languages. Typically, a statement in 3GL is translated into many instructions of machine language. C, BASIC, FORTRAN, COBOL and Pascal are the popular third generation languages.

FORTRAN (FORmula TRANslation): Fortran was developed in 1956 by John Backus. It was developed keeping in mind the scientific and engineering application.

COBOL (Common Business Oriented Language): COBOL was developed in the early 1960s under the auspices of the U.S. Department of Defense in cooperation with computer manufactures, users, and universities. It was designed to be a language for writing programs for business problems. Another design objective was to keep it machine independent. The language was designed in such a manner that it could evolve and grow to take care of changing program development requirements. Many standards for COBOL have been published since then.

BASIC (Beginners All-purpose Symbolic Instruction Code): BASIC was developed in 1964 by John Kemeny and Thomas Kurtz to teach students as Dartmouth College to use computers. It was meant to be a very simple language to learn and also one that would be easy to translate. Furthermore, the designers wished it to be a foundation language for students who wished to learn more powerful languages such as FORTRAN or ALGOL.

Pascal: It was developed in the late 1960s by Niklaus Wirth of Zurich. He named it after the great mathematician and philosopher, Blaise Pascal. Both Pascal and BASIC have been used extensively for teaching to the beginners.

C was developed at Bell Laboratories in 1972 by Dennis Ritchie. Many of its principles and ideas were taken from the earlier language B and B's earlier ancestors BCPL and CPL. C was developed with the purpose of creating a high level language that could be used for writing machine independent programs and would still allow the programmer to control the behavior of individual bits of data. The bit processing features have made C a popular system software development language.

- **Fourth Generation Languages (4GL) :** Fourth generation languages are less procedural and even more English-like then third generation languages. The emphasis is more on the output format than the procedure applied to achieve the results. Some fourth generation languages are:

Table 3.2: Fourth Generation Languages

4GLs	Salient Feature
Natural, Power Builder, Visual C++, Visual Basic	Application Generators
SQL, RPG-III	Query Languages/ Report Generators
Systat, SAS Graph	Graphics Languages
People Soft HRMS, SAP R/3	Application Software Packages

- **Object Oriented Languages :** Object Oriented Programming is a type of programming in which programmers define the data types and structure of the data, in addition, the operations (functions) that can be applied to the data is also defined. The data and functions together are known as objects. Programmers can create relationships between objects. For example, objects can inherit

characteristics from other objects. One of the principal advantages of object-oriented programming techniques over procedural programming techniques is that they enable programmers to create modules that do not need to be changed when a new type of object is added. A programmer can simply create a new object that inherits many of its features from existing objects. This makes object-oriented programs easier to modify. The object-oriented programming languages are usually referred to by the acronym OOPL, Java, C++ and Smalltalk are the popular languages.

3.5 OPEN SOURCE SOFTWARE

Open Source Software (OSS) is primarily defined as software, which is freely re-distributable and includes the source code. The licenses under which OSS is released vary greatly. The complete Open Source Definition can be found at http://www.opensource.org/osd.html. OSS is vastly different from the mainstream software industry where source code is highly guarded and programs are only distributed in their binary form, which is non-modifiable format.

The most important aspect of the open source movement is the participation of users. When a user wants a feature or a bug fix for a commercial program, the user is at the mercy of the software vendor. However, with open source, the user can modify the program according to his needs or fix a bug. Many users will help develop the program for free, simply to improve the product and for the benefit of the community.

These are a few of the most common and popular licenses for OSS.

- GNU Public License (GPL)
- Limited GNU Public License (LGPL)
- BSD-Style License
- The Artistic License
- The Netscape Public License (NPL) and the Mozilla Public License (MPL)
- Apple Public Source License (APSL)

A few advantages of OSS are:

1) **Cost Effective:** Open source software often comes free. The individual or organization users can save the software cost.
2) **Customizable:** Since Open source software comes with the source; one can customize existing software to suit one's needs. Closed source software *may* be customizable, but you need to negotiate and/or pay for customization. Open source licenses typically *guarantee* you the right to be able to customize the software.
3) **More Secure:** Since the source code is open, more people scrutinize the source code, and hence more flaws are found and corrected. The end result is that the code produced is more secure compared to similar closed-source code.

The following terms are synonymous with Open Source Software: Freeware, Free/Libre/Open Source Software (FLOSS).

Activity C

Explain the differences between commercial software, shareware, open source software, freeware, and public domain software.

.....

.....

.....

3.6 ACQUIRING APPLICATION SOFTWARE

A company can either develop or purchase software for its use. In some cases, the purchased software can be modified / customized according to the needs of the company. The different options available are summarized in the diagram below.

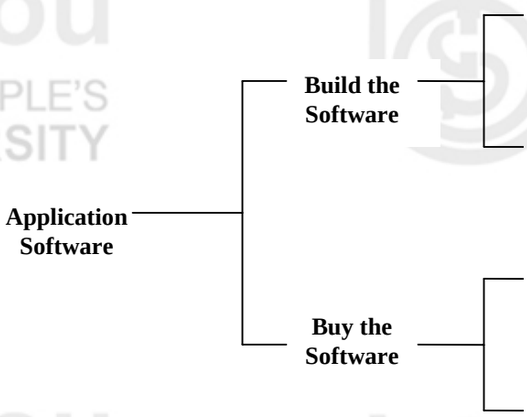


Fig. 3.3 : Sources of Acquiring Application Software

Build the software: If the requirements of the company are unique or specific, then the decision to build the software may be taken. If the organization has the required talent and time, it may be built by the company itself. This kind of development is known as *in-house development.* Also the company may obtain customized software from software vendors. Such software developed for particular companies are called *contract software.*

Buy the software: The Company has another option of purchasing, leasing, or renting software from software companies, who develop programs and sell them to many computer users and organizations. The software developed for the general market is called *off-the-shelf software.* They are readily available and many companies use them to support their business processes.

Customized Software: The Company can also opt to go for a mix of both buy and build decision. In that case, the company can purchase some off-the-shelf available software, and customize it to its needs by in-house or external personnel. There are software vendors in the market who provide a range of services like installing, modifying software, training end users, etc. They can be contracted to do the customization.

3.7 SUMMARY

Computer software have developed so much over the past years that it is very difficult to cover all aspects of the same. System software and application software represents two broad levels of categorization. System software encompasses of the operating system, language translators, and the utility programs. Application software is aimed to solve particular user computing problems. Open source software is distributed with the source code and freely available at a fraction of cost as compared to proprietary software. Acquiring application software is an important business activity and requires to be managed carefully.

3.8 UNIT END EXERCISES

1) What are the main types of software?

2) What are the main functions of operating system?

3) Define multi-tasking and time-sharing system?

4) Difference between compiler and interpreter?

5) Name different generations of programming languages and their characteristics?

6) What are the advantages of open-source software?

7) Describe the decision making process of acquiring application software?

3.9 REFERENCES AND SUGGESTED FURTHER READINGS

Laudon C K & Laudon J P, *Management Information Systems*, Pearson education, Asia.

Stair R, Reynolds G, *Principles of Information Systems; Course Technology*.

Turban, T., Ephraim, M. and Wetherbe J., *Information Technology for Management*, John Wiley and Sons, Inc., 1998.