CONTENTS

OPERATING SYSTEM

SYLLABUS

UNIT I

Operating systems objectives and functions – Different services of the operating system – Operating system as a user/computer interface, operating system as a resource manager, History of the operating systems – Serial processing, simple batch systems, multiprogrammed batch systems, Time Sharing systems. Process description and control – Process status – Process description – Process control – Processes and threads. Interprocess communication.

UNIT II

Memory Management – Memory management requirements single contiguous memory management, Fixed partitioned memory management, variable partitions, Non-contiguous allocation, paging, segmentation, virtual memory management systems.

UNIT III

Concurrency – Principles of concurrency, Mutual exclusion, software support, Dekkers algorithm, mutual exclusion, hardware support, mutual exclusion, operating system support, semaphore implementation, Deadlock – deadlock prevention, deadlock detection, deadlock avoidance.

UNIT IV

I/O Management and Disk scheduling – Disk scheduling algorithms. File management – Files – File Management systems – File system Architecture – Functions – File sharing – File directories – File allocation. Security policies and Mechanisms – Protection and access control.

UNIT V

Case studies the Unix operating system – Command language user's view of Unix, System call user's view of unix, implementation of unix. PC Dos operating system – Command language user's view of PC-Dos, system – Call user's view of PC-Dos, PC-Dos implementation, Netware – Communications Management in Netware, History of Netware, Netware 386 architecture, Netware features.

UNIT I

LESSON

1

OPERATING SYSTEM: AN INTRODUCTION

CONTENTS

1.0 AIMS AND OBJECTIVE

After studying this lesson, you should be able to:

- Evaluate objectives and functions of operating system
- An overview of OS services

- Operating systems as an user/computer interface and resource manager
- History of operating system
- Brief idea of batch systems
- Concept of multi programmed batch systems
- What is time sharing system

1.1 INTRODUCTION

The usage of an operating system with respect to a computer can easily be compared to the usage of the heart with respect to the human body. Without operating system a computer is simply of no use. The visible part of an operating system is the Graphical User Interface (GUI) – and yet most of what an operating system does is invisible.

Computer hardware provides us with the means of processing and storing information. However, the 'bare' machine on its own is virtually useless. In order to make the computer perform useful work for us, it is has to be 'driven' by means of programs-software which specify the tasks to be done. The combination of hardware and software provide a total usable system.

The unit covers most of the basic topics of operating systems. A brief idea about the operating system is discussed along with the different types of operating systems like serial processing, simple batched system, multiprogrammed batched system, time sharing system etc.

You know a computer does not do anything without properly instructed. Thus, for each one of the above power-on activities also, the computer must have instructions. These instructions are stored in a non-volatile memory, usually in a ROM. The CPU of the computer takes one instruction from this ROM and executes it before taking next instruction. ROMs are of finite size. They can store only a few kilobytes of instructions. One by one the CPU executes these instructions. Once, these instructions are over, the CPU must obtain instructions from somewhere. Where these instructions stored and what are their functions?

Usually these instructions are stored on a secondary storage device like hard disk, floppy disk or CD-ROM disk. These instructions are collectively known as operating system and their primary function is to provide an environment in which users may execute their own instructions. Once the operating system is loaded into the main memory, the CPU starts executing it. Operating systems run in an infinite loop, each time taking instructions in the form of commands or programs from the users and executing them. This loop continues until the user terminates the loop when the computer shuts down.

In order to exploit the most from a computer, therefore, a deep understanding of operating system is a must.

1.2 OBJECTIVES AND FUNCTIONS OF OPERATING SYSTEM

An operating system is the innermost layer of software which takes care of all technical aspects of a computer's operation. It shields the user of the machine from the low-level details of the machine's operation and provides frequently needed facilities.

1.2.1 Objectives of Operating System

The objective of the operating system is to complement the hardware by providing a layer of services which manage the resources of the hardware and permit the users to drive the system. In general, the user will not be aware of the union in effort. Indeed, many system facilities could be implemented by hardware or software, a common example being floating point processing. (In many computers, there is no in-built hardware available to perform floating-point computations, which therefore has to be implemented using software routines). The situation is further complicated by the existence of firmware which is a program encoded in a hardware form, usually in Read-only Memory (ROM). Firmware is often used to provide very basic services at a functional level just above the hardware (e.g., the bios in PC).

Operating systems are so ubiquitous in computer operations that one hardly realizes its presence. Most likely you must have already interacted with one or more different operating systems. The names like DOS, UNIX etc. should not be unknown to you. These are the names of very popular operating systems.

Again, the boundary between hardware and firmware is imprecise and in any case, is not important for most users.

Normally an operating system is the software which is already installed on a machine, before anything you add on your own.

Generally the operating system has a number of key elements:

(a) A technical layer of software for driving the hardware of the computer, like disk drives, the keyboard and the screen.

(b) A file system which provides a way of organizing files logically.

(c) A simple command language which enables users to run their own programs and to manipulate their files in a simple way. Some operating systems also provide text editors, compilers, debuggers and a variety of other tools.

Since the Operating System (OS) is in charge of a computer, the requests to use any of its resources and devices need to go through the operating system. An operating system provides the legal entry points into its code for performing basic operations e.g., writing to devices.

An operating system (sometimes abbreviated as "OS") is a program that, after being initially loaded into the computer by a boot program, manages all the other programs in a computer. The other programs are called applications or application programs. The application programs uses the operating system by making requests for services through API i.e. Application Program Interface. The users can interact directly with the operating system either through a user interface such as a command language (e.g., DOS) or a Graphical User Interface (GUI).

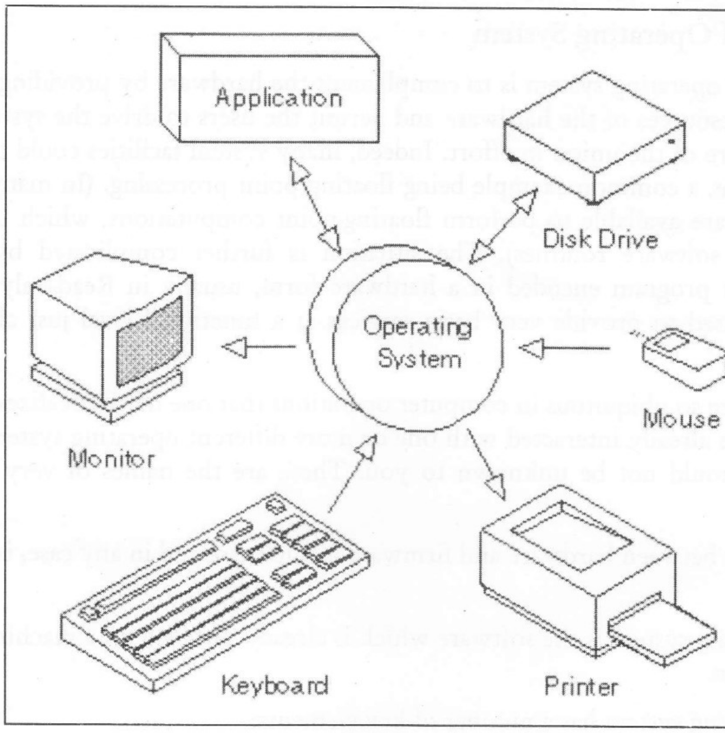

Figure 1.1: Operating System Interface

1.2.2 Functions of Operating System

An operating system manages all aspects of a complex system.

Imagine what will happen if three programs running on a computer all try to print their output simultaneously on the same printer. The first few lines might be from program one and the next few lines might be from program two and so on with the result resulting in chaos. The operating system can bring order to potential chaos by buffering all the output destined for the printer on the disk.

The primary task of operating system is to keep track of who is using which resource, to grant resource request, to account for usage, and to meditate conflicting request from different programs and users.

When a computer has multiple users, the needs for managing and protecting the memory, input/output devices and other resources are vary necessary. These needs arise because it is usually necessary to share expensive resource such as tapes drives and phototypesetters.

Operating systems perform basic tasks, such as recognizing 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 disk drives and printers.

For large systems, the operating system has even greater responsibilities and powers. It is like a traffic cop – it makes sure that different program and users running at the same time do not interfere with each other. The operating system is also responsible for security, ensuring that unauthorized users do not access the system.

An operating system performs the following services for applications:

1. In a multitasking system where multiple programs are run at the same time, the operating system determines which applications should run in what order and how much time should be allowed for each application before another application gets its turn.
2. It sends messages to each application or interactive user (or to a system operator) about the status of operation and any errors that may have occurred.
3. It handles input and output to and from attached hardware devices, such as hard disks, printers, and dial-up ports.
4. It arranges the sharing of internal memory among multiple applications.
5. It can off-load the management of what are called batch jobs (for example, printing) so that the initiating application is freed from this work.
6. In computers that can provide parallel processing, an operating system manages to divide the instructions of a program so that it runs on more than one processor at a time.

Other Operating System Functions

The operating system provides for several other functions including:

1. System tools (programs) used to monitor computer performance, debug problems, or maintain parts of the system.
2. A set of libraries or functions which programs may use to perform specific tasks especially relating to interfacing with computer system components.

The operating system makes these interfacing functions along with its other functions operate smoothly and these functions are mostly transparent to the user.

Operating System Concerns

As mentioned previously, an operating system is a computer program. Operating systems are written by human programmers who make mistakes. Therefore there can be errors in the code even though there may be some testing before the product is released. Some companies have better software quality control and testing than others so you may notice varying levels of quality from operating system to operating system. Errors in operating systems cause three main types of problems:

- *System crashes and instabilities:* These can happen due to a software bug typically in the operating system, although computer programs being run on the operating system can make the system more unstable or may even crash the system by themselves. This varies depending on the type of operating system. A system crash is the act of a system freezing and becoming unresponsive which would cause the user to need to reboot.
- *Security flaws:* Some software errors leave a door open for the system to be broken into by unauthorized intruders. As these flaws are discovered, unauthorized intruders may try to use these to gain illegal access to your system. Patching these flaws often will help keep your computer system secure. How this is done will be explained later.
- Sometimes errors in the operating system will cause the computer not to work correctly with some peripheral devices such as printers.

Classification

All operating systems contain the same components whose functionalities are almost the same. For example, all the operating systems perform the functions of storage management, process management, file management, securities i.e. protection of one user from one-another, etc. The procedures and methods that are used to perform these functions and the interface might be different but the fundamental concepts are same. Operating systems in general, perform similar functions but can be distinguished as per features. Hence, they can be classified into different categories on different bases. Let us study the different types of operating systems.

1.3 DIFFERENT SERVICES OF OPERATING SYSTEM

Every general-purpose computer must have an operating system to run other programs. Operating systems perform basic tasks, such as recognizing 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 disk drives and printers.

For large systems, the operating system has even greater responsibilities and powers. It is like a traffic cop - it makes sure that different program and users running at the same time do not interfere with each other. The operating system is also responsible for security, ensuring that unauthorized users do not access the system.

- *Program Execution:* The purpose of a computer system is to allow the user to execute programs. So the operating systems provides an environment where the user can conveniently run programs. The user does not have to worry about the memory allocation or multitasking or anything. These things are taken care of by the operating systems.

  Running a program involves the allocating and deallocating memory, CPU scheduling in case of multiprocess. These functions cannot be given to the user-level programs. So user-level programs cannot help the user to run programs independently without the help from operating systems.

- *I/O Operations:* Each program requires an input and produces output. This involves the use of I/O. The operating systems hides the user the details of underlying hardware for the I/O. All the user sees is that the I/O has been performed without any details. So the operating systems by providing I/O make it convenient for the users to run programs.

  For efficiently and protection users cannot control I/O so this service cannot be provided by user-level programs.

- *File System Manipulation:* The output of a program may need to be written into new files or input taken from some files. The operating system provides this service. The user does not have to worry about secondary storage management. User gives a command for reading or writing to a file and sees his/her task accomplished. Thus operating system makes it easier for user programs to accomplish their task.

  This service involves secondary storage management. The speed of I/O that depends on secondary storage management is critical to the speed of many programs and hence I think it is best relegated to the operating systems to manage it than giving individual users the control of it. It is not difficult for the user-level programs to provide these services but for above mentioned reasons it is best if this services left with operating system.

- *Communications:* There are instances where processes need to communicate with each other to exchange information. It may be between processes running on the same computer or running on the different computers. By providing this service the operating system relieves the user of the worry of passing messages between processes. In case where the messages need to be passed to processes on the other computers through a network it can be done by the user programs. The user program may be customized to the specifics of the hardware through which the message transits and provides the service interface to the operating system.
- *Error Detection:* An error is one part of the system may cause malfunctioning of the complete system. To avoid such a situation the operating system constantly monitors the system for detecting the errors. This relieves the user of the worry of errors propagating to various part of the system and causing malfunctioning.

  This service cannot allow to be handled by user programs because it involves monitoring and in cases altering area of memory or deallocation of memory for a faulty process. Or may be relinquishing the CPU of a process that goes into an infinite loop. These tasks are too critical to be handed over to the user programs. A user program if given these privileges can interfere with the correct (normal) operation of the operating systems.

1.4 OPERATING SYSTEM AS A USER/COMPUTER INTERFACE

An operating system is the most important program in a computer system. This is one program that runs all the time, as long as the computer is operational and exits only when the computer is shut down.

In general, however, there is no completely adequate definition of an operating system. Operating systems exist because they are a reasonable way to solve the problem of creating a usable computing system.

The fundamental goal of computer systems is to execute user programs and to make solving user problems easier. Hardware of a computer is equipped with extremely capable resources - memory, CPU, I/O devices etc. All these hardware units interact with each other in a well-defined manner. Bare hardware alone is not enough to solve a problem, particularly easy to use, application programs are developed. These various programs require certain common operations, such as those controlling the I/O devices. The common functions of controlling and allocating resources are then brought together into one piece of software: the operating system.

It is easier to define operating systems by their functions, i.e., by what they do than by what they are. The computer becomes easier for the users to operate, is the primary goal of an operating system. Operating systems exist because they are supposed to make it easier to compute with them than without them. This view is particularly clear when you look at operating systems for small personal computers.

Efficient operation of the computer system is a secondary goal of an operating system. This goal is particularly important for large, shared multi-user systems. These systems are typically expensive, so it is desirable to make them as efficient as possible.

Operating systems and computer architecture have had a great deal of influence on each other. To facilitate the use of the hardware, operating systems were developed. As operating systems were designed and used, it became obvious that changes in the design of the hardware could simplify them.

Without an operating system, the hardware of a computer is just an inactive electronic machine, possessing great computational power, but doing nothing for the user. All it can do is to execute fixed number of instructions stored into its internal memory (ROM: Read Only Memory), each time you switch the power on, and nothing else.

Operating systems are programs (fairly complex ones) that act as interface between the user and the computer hardware. They sit between the user and the hardware of the computer providing an operational environment to the users and application programs. For a user, therefore, a computer is nothing but the operating system running on it. It is extended machine.

Users do not interact with the hardware of a computer directly but through the services offered by operating system. This is because the language that users employ is different from that of the hardware. Whereas users prefer to use natural language or near natural language for interaction, the hardware uses machine language. It is the operating system that does the necessary translation back and forth and lets the user interact with the hardware. The operating system speaks users' language one hand and machine language on the other. It takes instructions in form of commands from the user and translates into machine understandable instructions, gets these instructions executed by the CPU and translates the result back into user-understandable form.

A user can interact with a computer if only he/she understands the language of the resident operating system. You cannot interact with a computer running UNIX operating system, for instance, if you do not know 'UNIX language' or UNIX commands. A UNIX user can always interact with a computer running UNIX operating system, no matter what type of computer it is. Thus, for a user operating system itself is the machine - an user/computer interface as shown in Figure 1.2

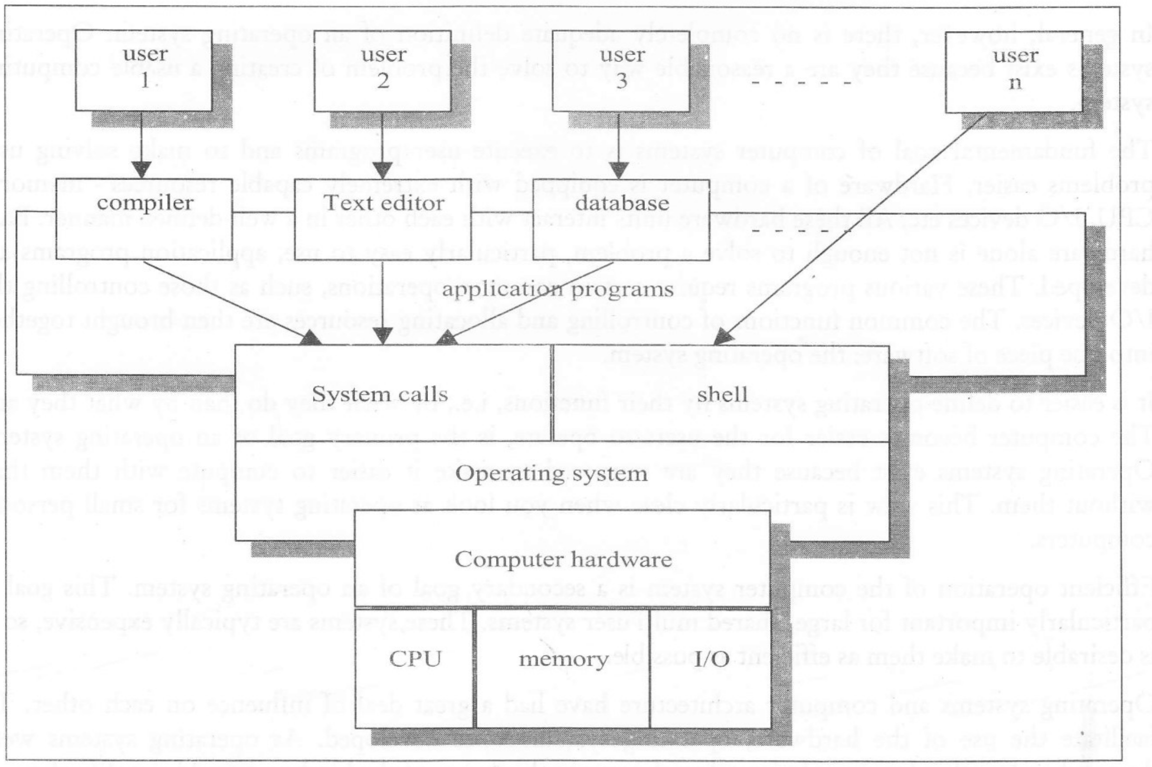

Figure 1.2: User/Computer Interface view of Operating System

1.5 OPERATING SYSTEMS AS A RESOURCE MANAGER

The computer hardware is made up of physical electronic devices, viz. memory, microprocessor, magnetic disks and the like. These functional components are referred to as resources available to computers for carrying out their computations. All the hardware units interact with each other in terms of electric signals (i.e. voltage and current) usually coded into binary format (i.e. 0 and 1) in digital computers, in a very complex way.

In order to interact with the computer hardware and get a computational job executed by it, the job needs to be translated in this binary form called machine language. Thus, the instructions and data of the job must be converted into some binary form, which then must be stored into the computer's main memory. The CPU must be directed at this point, to execute the instructions loaded in the memory. A computer, being a machine after all, does not do anything by itself. Which resource is to be allocated to which program, when and how, is decided by the operating system in such a way that the resources are utilized optimally and efficiently.

1.6 HISTORY OF OPERATING SYSTEM

It has been long recognized that the modern computer is an integrated system consisting of machine hardware, an instruction set, system software, application programs, and user interfaces. The use of a computer is driven by real-life problems demanding fast and accurate solutions. Depending on the nature of the problems, the solutions may require different computing resources.

An Operating System (OS) may be viewed as an organized collection of software extensions of hardware, consisting of control routines for operating a computer and for providing an environment for execution of programs. Programs usually invoke services of the operating system by means of operating-system calls. The user may interact with the operating system directly by means of operating-system commands. Thus, we can say that the operating system acts as an interface between users and the hardware of a computer system.

Internally, an operating system acts as a manager of resources of the computer system, such as processor, memory, files and I/O devices. The OS keeps track of each resource, and decides who gets a resource, for how long, and when, and is so doing attempts to optimize the resulting performance.

The primary objective of the operating system is to increase productivity of a processing resource, such as computer hardware or computer system users. User convenience and productivity were secondary considerations. In single user system the emphasis is on making the computer system easier to use by providing a graphical and hopefully more intuitively obvious user interface.

1.6.1 Computer Generations

You must be curious about knowing that when the first computer was built? How it was looking like? To know the history of computer we should start from very beginning, when the first calculating and computing device introduced.

Zero Generation: Mechanical Parts

1. *Abacus:* The first calculating device was built around 5000 years ago. It was known as Abacus. The Abacus was invented by Chinese but later on Japanese and Russians developed it with little bit changes.

2. *Napier Bones:* In 1617 Scottish Mathematician Sir John Napier made eleven rectangular rods of carved bones. This set of eleven rods was named Napier Bones. This was helpful in multiplying and dividing the numbers as well as addition and subtraction.

3. *Pascaline:* In 1642 Blaise Pascal invented first Mechanical calculator. His father was tax officer in Royan who was very tired daily due to his endless job. So Pascal decided to make a machine that could help his father. As a result he invented the Pascaline only at the age of 18 years.

   The Pascaline was a brass made rectangular box, this was very fast calculating machine of those days but it could only add and subtract but could not perform multiplication and division of the numbers.

4. *Leibnitz Calculator:* In 1694 a German Mathematician and Philosopher, Gottfried Wilhem Von Leibnitz improved the Pascaline and this new device, then called 'Leibnitz Calculator', became capable of multiplying and dividing also.

5. *Analytical and Difference Engine:* The real beginning of computers lies with an English Mathematician and Professor, Charles Babbage. Frustated, by many errors, he came across while examining calculations for the Royal Astronomical Society, Babbage declared, "I wish to God these calculations could be performed by Steam!." With these words the automation of computer begun.

6. *Hollerith's Tabulating Machine:* In 1889 Herman Hollerith of America, worked to find a faster way to compute the U.S. Census. The previous census of 1880 had taken about seven years to count and with an expanding population, the beauro feared it would take 10 years to count the latest census. Hollerith used cards to store data which he fed into a machine that compiled the results mechanically. Each punch on the card represented one number and combination of two punches represented one alphabet. A punch card could hold upto 80 variables.

   Instead of ten years this census data was compiled to find the report in just Six weeks due to Hollerith's machine. In addition to the fast speed working, the punch cards served as a data storage media.

   Hollerith introduced his punch card reader to the business world, founding tabulating machine company in 1896 which later turned to International Business Machines (IBM).

7. *Mark-I:* IBM encouraged to Howard Aikens to build a computer that could work on electricity. He started work in 1930 and made a monchester type 15 meter long computer that had wiring length of 800 kilometres. This computer was named as Mark-I and remained in use between 1948 to 1959.

First Generation (1945-1955): Vacuum Tubes

The first electronic computer was completed in 1946 by a team led by Professors Eckert and Mauchly at the University of Pennsylvania in U.S.A. This computer, called Electronic Numerical Integrator and Calculator (ENIAC) used high speed vacuum tubes. It had a very small memory and was designed primarily to calculate the trajectories of missiles. The ENIAC took about 200 microseconds to add two digits and about 2800 microseconds to multiply.

In 1946 John Von Neumann proposed an idea to store machine instructions in the memory of the computer along with data. The first computer using this principle was designed and commissioned at Cambridge University, U.K., under the leadership of Professor Maurice Wilkes. This computer called

EDSAC (Electronic Delay Storage Automatic Calculator) was completed in 1949 and used mercury delay lives for storage.

Commercial production of stored program electronic computers began in the early 50s. One of the early computer of this type was UNIVACI built by Univac division of Ramington Rand and delivered in 1951. This computer also used Vacuum tubes. As vacuum tubes used filaments as a source of electrons, they had a limited life. Each tube consumed about half a watt power. Computers typically used about ten thousand tubes. Power dissipation was very high.

With the advent of UNIVAC, the prospects of commercial application were perceived. The concept of operating system had not yet emerged. During this period one had to be a good electronics engineer, and understand the logical structure of a computer in great detail, and also know how to program an application in order to use a computer.

Second Generation (1955-1965): Transistors

A big revolution in electronics took place with the invention of transistors by Nardeen, Brattain and Shockley in 1946. Transistors made of germanium semiconductor material were highly reliable compared to vacuum tubes since transistors had no filament to burn. They occupied less space and used only a tenth of the power required by tubes. They could also switch from a 0 to a 1 state in a few microseconds, about a tenth of the time needed by vacuum tubes, thus computers made with transistors were about ten times reliable and faster, dissipated one-tenth the power, occupied about one-tenth the space and were ten times cheaper than those using vacuum tubes. Computer manufacturers thus changed over to transistors from tubes. The second generation computers emerged around 1955 with the use of transistors instead of vacuum tubes in computers. This generation lasted till 1965.

Magnetic cores were invented during this generation for storage. They were used to construct large random access memories. Memory capacity in the second generation was about 100 kilo bytes. Magnetic disk storage was also developed during this period.

The higher reliability of computers and large memory availability led to the development of high level languages. Good batch operating systems, particularly the ones on IBM 7000 series computers emerged during the second generation.

Third Generation (1965-1975): Integrated Circuits

The third generation began in 1965 with germanium transistors being replaced by silicon transistors. Integrating circuits, circuits consisting of transistors, resistors and capacitors grown on a single chip of silicon eliminating wired interconnection between components, emerged. From small scale integrated circuits which had about 10 transistors per chip/technology developed to medium scale integrated circuits with 100 transistors per chip. Switching speed of transistors went up by a factor of 10, reliability increased by a factor of 10. Power dissipation reduced by a factor of 10 and size was also reduced by a factor of 10. The cumulative effect of this was the emergence of extremely powerful CPUs with the capacity of carrying out 1 million instructions per second.

There were significant improvement in the design of magnetic core memories. The size of main memories reached about 4 megabytes. Magnetic disk technology improved rapidly 100 Megabytes/ drive became feasible.

The combined effect of high capacity memory, powerful CPU and large disk memories led to the development of time shared operating systems. Time shared systems increased programmer productivity.

The third generation probably ended by 1975. The improvements in the period 65-75 were substantial but no revolutionary new concept could be identified as heralding the end of third generation.

Fourth Generation (1975-1990): Large Scale Integration

- *First Decade (1976-85):* The fourth generation may be identified by the advent of the microprocessor chip. Medium scale integrated circuits yielded to large and Very Large Scale Integrated circuits (VLSI) packing about 50,000 transistors in a chip. Magnetic core memories were replaced by semiconductor memories. Semiconductor memory sizes of 16 megabytes with a cycle time of 200 n secs were in common use. The emergence of the microprocessor led to two directions in computer development. One direction was the emergence of extremely powerful personal computers. Computer cost came down so rapidly that professionals had their own computer to use in their office and home. Hard disks provided a low cost, high capacity secondary memory.

  Networks were developed. Disk memories became very large (1000 m bytes/drive). Another important development was interactive graphic devices and language interfaces to graphic systems. The emergence of graphics gave a great impetus to computer-aided engineering design.

  Fourth generation saw the coming of age of UNIX OS and time shared interactive systems. The systems became user friendly and highly reliable. The effective cost of computing came down. Computers also became all pervading.

- *Second Decade (1986-1995):* Microprocessors such as Pentium, Power PC, etc., are being used as the CPU of portable lap top and palm held computers (1995). Personal computers, Desk top workstations and powerful servers for numeric computing as well as file services use RISC microprocessors such as Alpha, MIPS and SUNSPARC.

  The area of hard disk storage also saw vast improvements. 1 GB of disk on workstations became common in 1994. For larger disks RAID technology (Redundant Array of Inexpensive Disks) was used to give storage of 100 GB. During the decade optical disks emerged as mass storage particularly for read only files. Optical storage sizes were of the order of 600 MB on a 5.25" disk. The availability of optical disks at low cost saw the emergence of multimedia applications. Multimedia workstations emerged as widely used systems with the emergence of distributed computers connected by networks considerable effort has gone into programming distributed systems. A number of parallel computers were built but no commonly accepted standard parallel programming language emerged.

Fifth Generation (Future Computers)

Even though computers in the last 40 years have become very fast, reliable and inexpensive, the basic logical structure proposed by Von Neumann has not changed. The basic block diagram of the CPU, memory and I/O is still valid today. With the improvements in integrated circuit technology, it is now possible to get specialized VLSI chips at a low cost. Thus, an architecture which makes use of the changes in technology and allows an easier and more natural problem solving is being sought.

1.6.2 Computer Languages

So far we have talked about one side of the computer system, i.e., the hardware, the machine itself. Now, we will talk about its usage. How this machine can be used with the help of various programs? This is called software. There are various types of software. Some are needed for some particular type of operations and others for running the computer. The basic difference between hardware and software can be explained by simple term that hardware is something which can be seen and touched whereas software cannot be seen or touched. Software is essential for running the computer.

- *Software:* As mentioned, it is needed to run the computer, but what is it? It is nothing but a set of instructions that directs the computer to process information. These instructions are called programs. A set of programs is called software. There are mainly two types of software. System software and Application software.
- *System Software:* As the name implies it is needed to run the system. It also coordinates the operations of the various hardware components of the computer. It is needed to run the computer system. It also acts as a coordinator between the computer and its user. Most of the system software are machine based so usually the manufacturer of the machine provides the software. There are two types of system software and they are:

  (a) The Operating System; and

  (b) The Language Translators

Operating System (OS)

It is a collection of programs that controls the overall operations of a computer. It helps the computer to supervise and manage its resources. Operating system calls the application program whenever needed, translates the programs and manages data needed to produce the output.

Language Translators

There are different type of translators available.

- *Assembler:* It is used to convert the program written in assembly language into machine language.
- *Compiler:* It translates a source program that is usually written in a high level language by a programmer into machine language, which the computer understands. It is capable of replacing source program with a series of machine – language instructions or with a subroutine. It is different for different languages. A compiler creates a unique object program, i.e., if a source program is compiled there is no need of that source program because output can be obtained by executing that object program.
- *Interpreter:* It translates each source program statement into a sequence of machine instructions and than executes these machine instructions before translating the next source language statement. Interpreters are also unique for each high level language. Basically, the function of the compiler and interpreter are the same; the only difference is that compiler provides a separate object program but the interpreter does not. At the time of using interpreter each and every time the source program is needed. Another difference is that interpreter translates the source program line wise, i.e., it would not execute the next line unless the first one has no error.

1.6.3 Application Software

It is a set of programs designed for specific uses or applications such as word processing, graphics or spreadsheet analysis. Infact, most of the commercial software fall under this category. Application software can be custom-made but is usually purchased off the shelf. Popular software packages are:

- *Word Processors:* This is the software where you play with the text. You enter the text by typing and manipulate it to create a good looking letter, brochure, thesis etc., and print them accordingly to your needs for example Microsoft MS Office etc.
- *Spread Sheets:* A spread sheet is a matrix of rows and columns. These columns and rows are used for entering text and numbers, which can then be used for calculations. An ideal software for accounting purposes and is very widely used for creating balance sheet for example Microsoft MS Excel, Lotus Spreadsheet etc.
- *Database System:* These software are used to create and extract data from a database in the form needed by you. Most of the information needed by you can be totalled too. It also helps you in arranging the data in a particular format too for example Microsoft MS Access, Microsoft Sql Server, Oracle 9i etc.

1.6.4 Accounting Packages

Various accounting packages are available in the market which would take care of all your accounting needs from ledgers to reports. There are provisions for creating exclusive reports too for example Tally 5.4 etc.

- *DTP Packages:* Without packages this book would not have been possible. It sort of acts as via media between computer and printing press. All the pre-press work can be done using these software. From designing a page with text and graphics to creating complicated art works, everything can be done using them for example Coral Draw etc.
- *Graphic Packages:* These are used for working with pictures and graphics instead of numbers and text. Computer accessible data can readily be converted to graphic form on the screen as well as on paper using dot-matrix printers, ink-jet printers or plotters for example AutoCad etc.

1.7 SERIAL PROCESSING

The earliest computer system has no OS at all, and is characterized as serial processing because users have to reserve times lots in advance, and during the allotted period, they occupy the computer exclusively. Thus the computer will be used in sequence by different users.

These early systems presented two major problems:

1. Users may finish their tasks earlier than you have expected, and unfortunately the rest.
2. Time is simply wasted. Or they may run into problems, cannot finish in the allotted time, and thus are forced to stop, which causes much inconvenience and delays the development.

In such systems, programs are presented by cards. Each card has several locations on it, where there may be a hole or not, respectively indicating 0 or 1.Programs are loaded into memory via a card reader. With no OS available, to compile their programs, users have to manually load the compiler program first with the user program as input. This involves mounting, or dismounting tapes or setting up card decks. If an error occurred, the user has to repeat the whole process from the very beginning. Thus much time is wasted.

1.8 SIMPLE BATCH SYSTEMS

A batch system is one in which jobs are bundled together with the instructions necessary to allow them to be processed without intervention. The main function of a batch processing system is to automatically keep executing jobs one after another as appears in the batch. The idea behind a batch processing system is to reduce the interference of the operator during the processing or execution of jobs by the computer.

The functions of a batch processing system are carried out by the batch monitor. The batch monitor resides in the low end of the main store. The current jobs out of the whole batch are executed in the remaining storage area. It can be said that, a batch monitor is responsible for controlling all the environment of the system operation. The batch monitor accepts batch initiation commands from the operator, processes a job, as well as performs the job of job termination and batch termination.

The basic physical layout of the memory of a batch job computer is shown below:

Monitor (permanently resident)
User Space (compilers, programs, data, etc.

The monitor is system software that is responsible for interpreting and carrying out the instructions in the batch jobs. When the monitor starts a job, the entire computer is dedicated to the job, which then controls the computer until it finishes.

A sample of several batch jobs might look like this:

```
$JOB user_spec; identify the user for accounting purpose
$FORTRAN; load the FORTRAN compiler
source program cards
$LOAD; load the compiled program
$RUN; run the program
data cards
$EOJ; end of job
$JOB user-spec; identify a new user
$LOAD application
$RUN
data
$EOJ
```

Often magnetic tapes and drums are used to store data and compiled programs, temporarily or permanent.

1.	Advantages of batch systems	Move much of the work of the operator to the computer increased performance since it was possible for job to start as soon as the previous job finished.
2.	Disadvantages	Turn-around time can be large from user standpoint more difficult to debug program due to lack of protection scheme, one batch job can affect pending jobs (read too many cards, etc) a job could corrupt the monitor, thus affecting pending jobs a job could enter an infinite loop.

One of the major shortcomings of early batch systems is that there's no protection scheme to prevent one job from adversely affecting other jobs.

The solution to this brought a simple protection scheme, where certain memory (e.g. where the monitor resides) were made off-limits to user programs. This prevented user programs from corrupting the monitor.

To keep user programs from reading too many (or not enough) cards, the hardware is changed to allow the computer to operate in one of two modes: one for the monitor and one for the user programs. IO can only be performed in monitor mode, so that IO requests from the user programs are passed to the monitor. In this way, the monitor can keep a job from reading past it's on $EOJ card.

To prevent an infinite loop, a timer is added to the system and the $JOB card is modified so that a maximum execution time for the job is passed to the monitor. The computer will interrupt the job and return control to the monitor when this time is exceeded.

Turn Around Time

In a batch processing system, we generally use the term 'turn around time'.
It is defined as the 'time' from which a user job is given to the 'time' when its output is given back to the user. This 'time' includes the batch formation time, time-limit to execute a batch, time taken to print results and the time required to physically sort the printed outputs that belong to different jobs. As the printing and sorting of the results are done for all the jobs of batch together, the turn around time for a job becomes the function of the execution time requirement of all jobs in the batch.

The turn around time can be reduced for different jobs by recording the jobs or faster input/output media like magnetic tape or disk surfaces. It takes very less time to read a record from these media. For example, it takes round about five milliseconds for a magnetic 'tape' and about one millisecond for a fast fixed-head disk in comparison to a card reader or printer that takes around 50-100 milliseconds. Thus, if a disk or tape is used, it reduces the amount of time the central processor has to wait for an input/output operation to finish before resuming processing. This would reduce the time taken to process a job which indirectly would bring down the turn-around times for all the jobs in the batch.

Job Scheduling

Another term that is mostly used in a batch processing system is Job Scheduling. Job scheduling is the process of sequencing jobs so that they can be executed on the processor. It recognizes different jobs on the basis of first-come-first-served (FCFS) basis, because of the sequential nature of the batch. The batch monitor always starts the next job in the batch. However, in some cases, you could also arrange the different jobs in the batch depending upon the priority of each batch. According to some criteria, sequencing of jobs required scheduling the jobs at the time of creating or executing a batch. On the basis of importance of jobs, certain 'priorities' could be set for each batch of jobs. Several batches could be formed on the same criteria of priorities. So, the batch having the highest priority could be made to run earlier than other batches. This would give a better turn around service to the selected jobs.

Storage Management

Let us discuss the concept of storage management. At any point of time, the main store of the computer is shared by the batch monitor program and the current user job of a batch. The big question that comes in our mind is how much storage has to be used by the monitor program and how much has to be provided for the user jobs of a batch. However, if too much main storage is provided to the monitor, then the user programs will not get enough storage. Therefore, an overlay structure has to be devised so that the unwanted sections of monitor code don't occupy storage simultaneously.

Sharing and Protection

The efficiency of utilization of a computer system is recognized by its ability of sharing the system's hardware and software resources amongst its users. Whenever, the idea of sharing the system resources comes, certain doubts also arise about the fairness and security of the system.

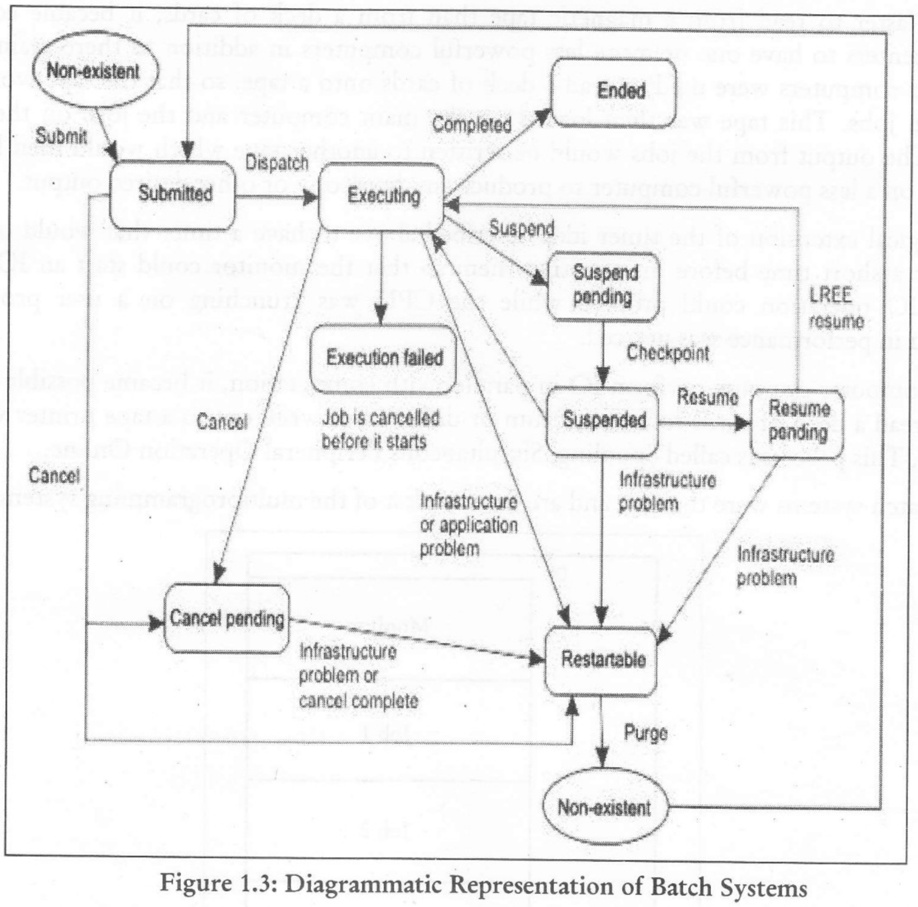

Figure 1.3: Diagrammatic Representation of Batch Systems

Every user wants that all his reasonable requests should be taken care of and no intentional and unintentional acts of other users should fiddle with his data. A batch processing system guarantees the fulfillment of these user requirements. All the user jobs are performed one after the other. There is no simultaneous execution of more than one job at a time. So, all the system resources like storage IO devices, central processing unit, etc. are shared sequentially or serially. This is how sharing of resources is enforced on a batch processing system.

Now, comes the question of protection. Though all the jobs are processed simultaneously, this too can lead to loss of security or protection. Suppose, there are two users A and B. User A creates a file of his own. User B deletes the file created by User A. There are so many other similar instances that can occur in our day to day life. So, the files and other data of all the users should be protected against unauthorized usage. In order to avoid such loss of protection, each user is bound around certain rules and regulations. This takes the form of a set of control statements, which every user is required to follow.

1.9 MULTI PROGRAMMED BATCH SYSTEMS

One difficulty with simple batch systems is that the computer still needs to read the deck of cards before it can begin to execute the job. This means that the CPU is idle (or nearly so) during these relatively slow operations.

Since it is faster to read from a magnetic tape than from a deck of cards, it became common for computer centers to have one or more less powerful computers in addition to there main computer. The smaller computers were used to read a deck of cards onto a tape, so that the tape would contain many batch jobs. This tape was then loaded on the main computer and the jobs on the tape were executed. The output from the jobs would be written to another tape which would then be removed and loaded on a less powerful computer to produce any hardcopy or other desired output.

It was a logical extension of the timer idea described above to have a timer that would only let jobs execute for a short time before interrupting them so that the monitor could start an IO operation. Since the IO operation could proceed while the CPU was crunching on a user program, little degradation in performance was noticed.

Since the computer can now perform IO in parallel with computation, it became possible to have the computer read a deck of cards to a tape, drum or disk and to write out to a tape printer while it was computing. This process is called Spooling: Simultaneous Peripheral Operation Online.

Spooling batch systems were the first and are the simplest of the multiprogramming systems.

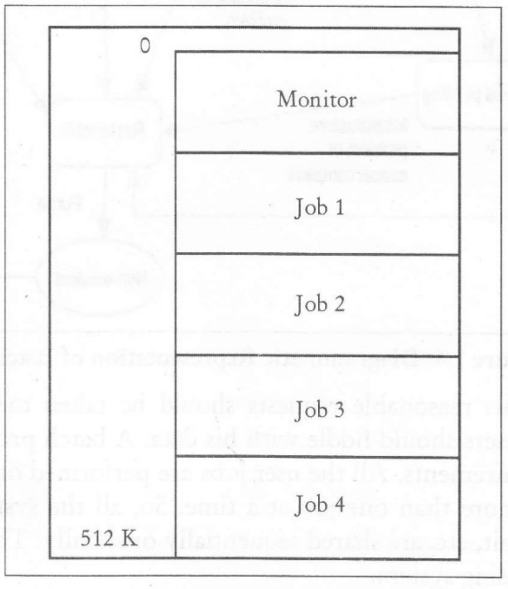

Figure 1.4: Spooling Batch Systems

As machines with more and more memory became available, it was possible to extend the idea of multiprogramming (or multiprocessing) as used in spooling batch systems to create systems that would load several jobs into memory at once and cycle through them in some order, working on each one for a specified period of time. At this point the monitor is growing to the point where it begins to resemble a modern operating system. It is responsible for:

- Starting user jobs
- Spooling operations
- IO for user jobs
- Switching between user jobs
- Ensuring proper protection while doing the above

As a simple, yet common example, consider a machine that can run two jobs at once. Further, suppose that one job is IO intensive and that the other is CPU intensive. One way for the monitor to allocate CPU time between these jobs would be to divide time equally between them. However, the CPU would be idle much of the time the IO bound process was executing.

A good solution in this case is to allow the CPU bound process (the background job) to execute until the IO bound process (the foreground job) needs some CPU time, at which point the monitor permits it to run. Presumably it will soon need to do some IO and the monitor can return the CPU to the background job.

In multiprogramming batch system several jobs are kept in main memory at the same time, and the CPU is multiplexed among them.

One advantage of multiprogramming batch systems was that the output from jobs was available as soon as the job completed, rather than only after all jobs in the current cycle were finished.

OS Features Needed for Multiprogramming

- I/O routine supplied by the system.
- Memory management – the system must allocate the memory to several jobs.
- CPU scheduling – the system must choose among several jobs ready to run.
- Allocation of devices.

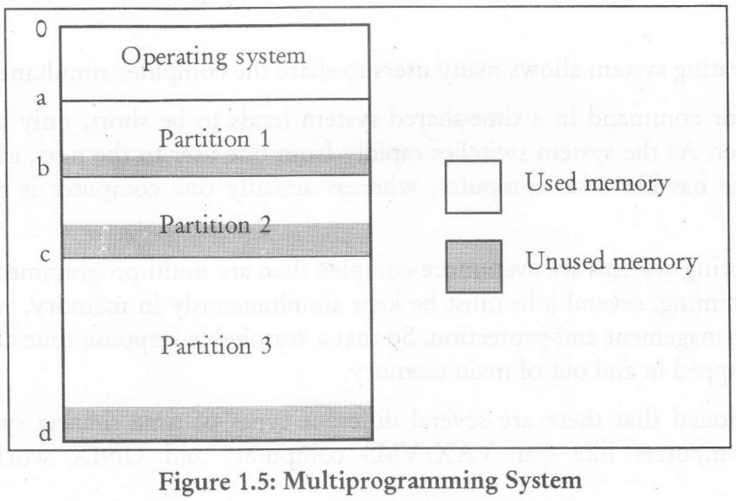

Figure 1.5: Multiprogramming System

Since multiple users have access to files, it is desirable to control by whom and in what ways files may be accessed. Batch systems are appropriate for executing large jobs that need little interaction. The user can submit jobs and return later for the results; it is not necessary for the user to wait while the job is processed.

Interactive jobs tend to be composed of many short actions, where the results of the next command may be unpredictable. The user submits the command and then waits for the results. Accordingly, the response time should be short-on the order of seconds at most.

Interactive System

An interactive system is used when a short response time is required. Earlier, computers with a single user were interactive systems. That is, the entire system was at the immediate disposal of the programmer/operator. This situation allowed the programmer great flexibility and freedom in program testing and development. But this arrangement resulted in substantial idle time while the CPU waited for some action to be taken by the programmer/operator. Because of the high cost of these early computers, idle CPU time was certainly undesirable. Batch operating systems were developed to avoid this problem. Batch systems improved the system utilization.

1.10 TIME SHARING SYSTEMS

Time-sharing systems were developed to provide interactive use of a computer system at a reasonable cost.

A time-shared operating system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time-shared computer. Each user has at least one separate program in memory. A program that is loaded into memory and is executing is commonly referred to as a process. When a process executes, it typically executes for only a short time before it either finishes or needs to perform
I/O. Now I/O can be interactive, i.e., the output is to a display for the user and input is from a user keyboard. Since interactive I/O typically runs at low speed (human speed), it may take a long time to complete. Input, for example, may be bounded by the user's typing speed; five characters per second is fairly fast for people, but is incredibly slow for computers. Rather than letting the CPU sit idle when this interactive input takes place, the operating system should rapidly switch the CPU to the program of some other user.

A time-shared operating system allows many users to share the computer simultaneously.

Since each action or command in a time-shared system tends to be short, only a little CPU time is needed for each user. As the system switches rapidly from one user to the next, each user is given the impression that she has her own computer, whereas actually one computer is being shared among many users.

Time-sharing operating systems are even more complex than are multi-programmed operating systems. As in multiprogramming, several jobs must be kept simultaneously in memory, which requires some form of memory management and protection. So that a reasonable response time can be obtained, jobs may have to be swapped in and out of main memory.

It should be mentioned that there are several different types of time sharing systems. One type is represented by computers like our VAX/VMS computers and UNIX workstations. In these