Unit 1 Operating System – An Introduction

Structure:

1.1 Introduction

Dear student, you know that software is a collection of programs. The computer software can be divided into two main categories: application software and system software. Application software consists of the programs for performing tasks particular to the machine's utilization. Examples of application software include spreadsheets, database systems, desktop publishing systems, program development software and games. Application software is generally what we think of when someone speaks of computer programs. This software is designed to solve a particular problem for users.

On the other hand, system software is more transparent and less noticed by the typical computer user. This software provides a general programming environment in which programmers can create specific applications to suit their needs. This environment provides new functions that are not available

at the hardware level and performs tasks related to executing the application program. System software acts as an interface between the hardware of the computer and the application software that users need to run on the computer. The most important type of system software is the operating system.

Objectives:

After studying this unit, you should be able to:

- define and describe the functions of operating system
- explain the evolution of operating systems
- discuss the operating system structures

1.2 Definition and Functions of Operating System

Now let's see the definition of operating system and their functions.

Operating System – Definition

Operating System is a System Software (Set of system programs) which provides an environment to help the user to execute the programs. The Operating System is a resource manager which allocates and manages various resources like processor(s), main memory, input/output devices and information on secondary storage devices.

Functions of Operating System

Operating systems perform the following important functions:

i) **Processor Management:** It means assigning processor to different tasks which has to be performed by the computer system.

ii) **Memory Management:** It means allocation of main memory and secondary storage areas to the system programmes, as well as user programmes and data.

iii) **Input and Output Management:** It means co-ordination and assignment of the different output and input devices while one or more programmes are being executed.

iv) **File System Management:** Operating system is also responsible for maintenance of a file system, in which the users are allowed to create, delete and move files.

v) **Establishment and Enforcement of a Priority System**: It means the operating system determines and maintains the order in which jobs are to be executed in the computer system.

vi) Assignment of system resources, both software and hardware to the various users of the system.

An operating system has three main responsibilities:

1. 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.
2. Ensure that different programs and users running at the same time do not interfere with each other.
3. Provide a software platform on top of which other programs (i.e., application software) can run.

The first two responsibilities address the need for managing the computer hardware and the application programs that use the hardware. The third responsibility focuses on providing an interface between application software and hardware so that application software can be efficiently developed. Since the operating system is already responsible for managing the hardware, it should provide a programming interface for application developers.

1.3 Evolution of Operating Systems

Do you know that the present day operating systems have not been developed overnight? Just like any other system, operating systems also have evolved over a period of time, starting from the very primitive systems to the present day complex and versatile ones. A brief description of the evolution of operating systems has been described below.

1.3.1 Simple Batch Operating Systems

In the earliest days digital computers usually run from a console. I/O devices consisted of card readers, tape drives and line printers. Direct user interaction with the system did not exist. Users made a job consisting of programs, data and control information. The job was submitted to an operator who would execute the job on the computer system. The output appeared after minutes, hours or sometimes days. The user collected the output from the operator, which also included a memory dump. The operating system was very simple and its major task was to transfer control

from one job to another. The operating system was resident in memory (Figure 1.1).

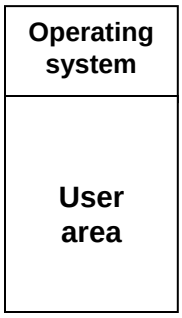

Fig. 1.1: Memory layout for simple batch system

To speed up processing, jobs with the same needs were batched together and executed as a group. For example, all FORTRAN jobs were batched together for execution; all COBOL jobs were batched together for execution and so on. Thus batch operating systems came into existence.

Even though processing speed increased to a large extent because of batch processing, the CPU was often idle. This is because of the disparity between operating speeds of electronic devices like the CPU and the mechanical I/O devices. CPU operates in the microsecond / nanosecond ranges whereas I/O devices work in the second / minute range. To overcome this problem the concept of SPOOLING came into picture. Instead of reading from slow input devices like card readers into the computer memory and then processing the job, the input is first read into the disk. When the job is processed or executed, the input is read directly from the disk. Similarly when a job is executed for printing, the output is written into a buffer on the disk and actually printed later. This form of processing is known as spooling an acronym for Simultaneous Peripheral Operation On Line. Spooling uses the disk as a large buffer to read ahead as possible on input devices and for storing output until output devices are available to accept them.

Spooling overlaps I/O of one job with the computation of other jobs. For example, spooler may be reading the input of one job while printing the output of another and executing a third job (Figure 1.2).

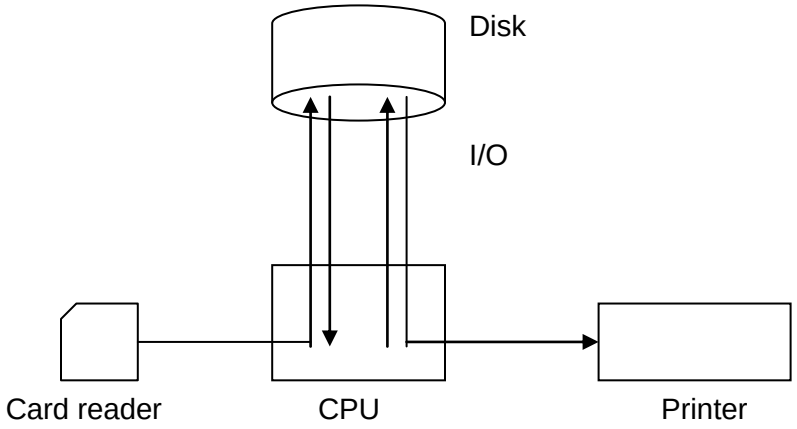

Fig. 1.2: Spooling

Spooling increases the performance of the system by allowing both a faster CPU and slower I/O devices to work at higher operating rates.

1.3.2 Multi-programmed Batched Operating Systems

The concept of spooling introduced an important data structure called the job pool. Spooling creates a number of jobs on the disk which are ready to be executed / processed. The operating system now has to choose from the job pool, a job that is to be executed next. This increases CPU utilization. Since the disk is a direct access device, jobs in the job pool may be scheduled for execution in any order, not necessarily in sequential order.

Job scheduling brings in the ability of multi-programming. A single user cannot keep both CPU and I/O devices busy. Multiprogramming increases CPU utilization by organizing jobs in such a manner that CPU always has a job to execute.

The idea of multi-programming can be described as follows. A job pool on the disk consists of a number of jobs that are ready to be executed (Figure 1.3). Subsets of these jobs reside in memory during execution. The operating system picks and executes one of the jobs in memory. When this job in execution needs an I/O operation to complete, the CPU is idle. Instead of waiting for the job to complete the I/O, the CPU switches to another job in memory. When the previous job has completed the I/O, it joins the subset of jobs waiting for the CPU. As long as there are jobs in memory waiting for the CPU, the CPU is never idle. Choosing one out of

several ready jobs in memory for execution by the CPU is called CPU scheduling.

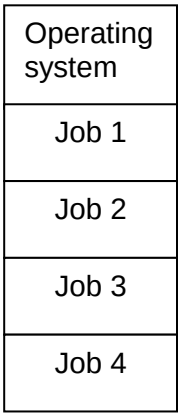

Fig. 1.3: Memory layout for multi-programmed system

1.3.3 Time-sharing Operating Systems

Multi-programmed batch systems are suited for executing large jobs that need very little or no user interaction. On the other hand interactive jobs need on-line communication between the user and the system. Time-sharing / multi tasking is a logical extension of multi-programming. It provides interactive use of the system.

CPU scheduling and multi-programming provide each user one time slice (slot) in a time-shared system. A program in execution is referred to as a process. A process executes for one time slice at a time. A process may need more than one time slice to complete. During a time slice a process may finish execution or go for an I/O. The I/O in this case is usually interactive like a user response from the keyboard or a display on the monitor. The CPU switches between processes at the end of a time slice. The switching between processes is so fast that the user gets the illusion that the CPU is executing only one user's process.

Time-sharing operating systems are more complex than multi-programmed operating systems. This is because in multi-programming several jobs must be kept simultaneously in memory, which requires some form of memory management and protection. Jobs may have to be swapped in and out of main memory to the secondary memory. To achieve this goal a technique called virtual memory is used.

Virtual memory allows execution of a job that may not be completely in a memory. The main logic behind virtual memory is that programs to be executed can be larger physical memory, its execution takes place.

Multi-programming and Time-sharing are the main concepts in all modern operating systems.

1.3.4 Personal Computer Operating Systems

The main mode of use of a PC (as its name implies) is by a single user. Thus Os for PCs were designed as a single user single task operating system, that is, it is assumed that only one user uses the machine and runs only one program at a time.

The operating system of PCs consists of two parts. One part is called the BIOS (Basic Input Output system) which are stored in a ROM (Read Only Memory). The other part called the DOS (Disk Operating System) is stored in a floppy disk or a hard disk.

When power is turned on BIOS takes control. It does what is known as power-on self test. The test sees whether memory is OK and all other relevant units function. Having done this, it reads from the disk a small portion of OS known as the boot and loads it into the main memory. This boot program then “pulls” the rest of the OS from the disk and stores it in the main memory. This is known as “booting the system”.

BIOS provides basic low level services where as DOS provides many user-level services. The major services provided by DOS are:

- File management which allows user to create, edit, read, write and delete files.
- Directory management which allows creation, change, search and deletion of directories.
- Memory management which allows allocation and de-allocation of memory.
- Command interpreter which interprets commands issued by the user and executes DOs functions, utility programs or application programs.
- Executive functions which provide programs to load and execute user programs, retrieve error codes, correct and rerun programs.
- Utility programs that do housekeeping chores such as COPY, ERASE, DIR etc.

1.3.5 Multi-processor Operating Systems (Parallel system)

Most systems to date are single-processor systems, which will be having only one CPU. However there is a need for multi-processor systems. Such systems will be having more than one processor and all these processor will share the computer bus, the clock and sometimes memory and peripheral devices. These systems are also called tightly coupled systems because the processors share the memory or a clock. Since all the processors function in parallel, they are also called parallel systems.

There are several reasons for building such systems. One advantage is increased throughput. By increasing the number of processors, we can get more work done in a shorter time.

Another advantage of multiprocessor Operating system is multi-processors can save money compared to multiple single systems because the processors can share peripherals, cabinets and power supplies. If several programs are to operate on the same set of data, it is cheaper to store those data on one disk and to have all the processor share them, rather than to have many computers with local disks and many copies of the data.

Another advantage of multi-processor Operating system is that they increase reliability. If functions can be distributed properly among several processors then the failure of one processor will not halt the system, but rather will only slow it down. If we have 10 processors and one fails, then each of the remaining nine processors must share the work of the failed processor. Thus the entire system runs only 10 percent slower, rather than failing together.

There are two most common types of multi-processor systems. One is symmetric multiprocessing model, in which each processor runs an identical copy of the operating system, and these copies communicate with one another as needed. Other is asymmetric multiprocessing model, in which each processor is assigned a specific task. Here one processor serves as master who controls the entire system and all the other processors serve as slaves. This model represents a master- slave relationship.

1.3.6 Distributed Systems

In this type of system all the computations are distributed among several processors. Distributed systems are also referred as loosely coupled

systems because here the processors do not share memory or a clock. Instead, each processor has its own local memory. The processors communicate with one another through various communication lines, such as high speed buses or telephone lines. The processors in distributed system vary in size, function and are referred as sites, nodes, and computers and so on depending on the context in which they are mentioned.

There are variety of reasons for building distributed systems, the major ones are:

Resource Sharing: If different users are connected to one another with different resources, then the user at one site may be able to use the resources available at another.

Computation Speedup: In distributed system**,** a particular computation will be partitioned into number of sub computations that can run concurrently. In addition to this, distributed system may allow us to distribute the computations among various sites. It is called load sharing.

Reliability: It means in distributed system, if one site fails, the remaining sites will share the work of failed site.

Communication: When many sites are connected to one another by a communication network, the processes at different sites have the opportunity to exchange information. A User may initiate file transfer or communicate with one another via electronic mail. A user can send mail to another user at the same site or at a different site.

1.3.7 Real - time Systems

Real-time operating systems are specially designed to respond to events that happen in real time. A real time operating system has well-defined, fixed time constrains. Processing must be done within the defined constraints, or else the system will fail. This feature is very useful in implementing systems such as an airline reservation system. In such a system, the response time should be very short because a customer's reservation is to be done while he/she waits.

There are two flavors of real-time systems. A hard real-time system guarantees that critical tasks complete at a specified time. A less restrictive type of real time system is soft real time system, where a critical real time task gets priority over other tasks, and remains that priority until it completes.

The several areas in which this type is useful are multimedia, virtual reality and advance scientific projects such as exploration and planetary rovers. Because of the expanded uses for soft real-time functionality, it is finding its way into most current operating systems, including major versions of Unix and Windows NT OS.

Self Assessment Questions

1. Operating System is a System Software which provides an environment to help the user to execute the programs. (True / False)
2. A form of processing known as spooling is an acronym for _____..
3. Disk Operating System (DOS) is an example for _____.
   a) PC OS
   b) Multi-processor OS
   c) Distributed OS
   d) Real Time OS

1.4 Operating System Structures

You can design an operating system as a huge, jumbled collection of processes without any structure. Any process could call any other process to request a service from it. The execution of a user command would usually involve the activation of a series of processes. While an implementation of this kind could be acceptable for small operating systems, it would not be suitable for large operating systems as the lack of a proper structure would make it extremely hard to specify code, test and debug a large operating system.

A typical operating system that supports a multiprogramming environment can easily be tens of megabytes in length and its design, implementation and testing amounts to the undertaking of a huge software project. In this section we discuss design approaches indeed to handle the complexities of today's large operating systems.

1.4.1 Layered approach

Dijkstra suggested the layered approach to lessen the design and implementation complexities of an operating system. The layered approach divides the operating system into several layers. The functions of operating system are divided among these layers. Each layer has well-defined functionality and input-output interfaces with the two adjacent layers.

Typically, the bottom layer is concerned with machine hardware and the top layer is concerned with users (Or operators).

The layered approach has all the advantages of modular design. In modular design, the system is divided into several modules and each module is designed independently. Likewise in layered approach each layer can be designed, coded and tested independently. Consequently the layered approach considerably simplifies the design, specification and implementation of an operating system. However, a drawback of the layered approach is that operating system functions must be carefully assigned to various layers because a layer can make use only of the functionality provided by the layer beneath it.

A classic example of the layered approach is the THE operating system, which consists of six layers. Figure 1.4 shows these layers with their associated functions.

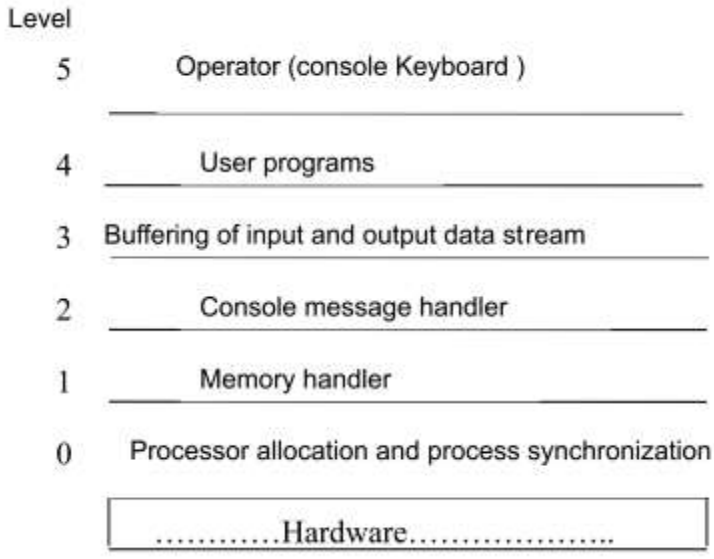

Fig. 1.4: Structure of the THE operating system

1.4.2 Kernel Based Approach

The Kernel-based design and structure of the operating system was suggested by Brinch Hansen. The Kernel (more appropriately called nucleus) is a collection of primitive facilities over which the rest of the operating system is built, using the functions provided by the kernel (see fig 1.5). Thus a kernel provides an environment to build operating system in

which the designer has considerable flexibility because policy and optimization decisions are not made at the kernel level. It follows that a kernel should support only mechanisms and that all policy decisions should be left to the outer layer. An operating system is an orderly growth of software over the kernel where all decisions regarding process scheduling, resource allocation, execution environment, file system, and resource protection and so on are made.

A kernel is a fundamental set of primitives that allows the dynamic creation and control of processes, as well as communication among them. Thus kernel only supports the notion of process and does not include the concept of a resource. However, as operating system has matured in functionality and complexity, more functionality has been relegated to the kernel. A kernel should contain a minimal set of functionality that is adequate to build an operating system with a given set of objectives. Including too much functionality in a kernel results in low flexibility at a higher level, whereas including too little functionality in a kernel results in low functional support at a higher level.

An outstanding example of a kernel is Hydra. Hydra is a kernel of an operating system for **c.mmp** a multiprocessor system. Hydra supports the notion of a resource and process, and provides mechanisms for the creation and representation of new types of resources and protected access to resources.

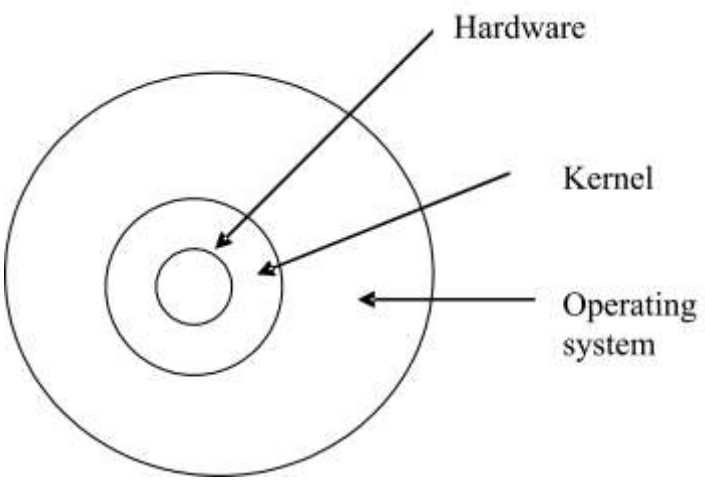

Fig. 1.5: Structure of a kernel- based operating system

1.4.3 Virtual Machine Approach

In the virtual machine approach, a virtual machine software layer on the bare hardware of the machine gives the illusion that all machine hardware (i.e. the processor, main memory, secondary storage, etc.) is at the sole disposal of each user. A user can execute the entire instruction set, including the privileged instructions. The virtual machine software creates this illusion by appropriately time multiplexing the system resources among all the users of the machine.

A user can also run a single user operating system on this virtual machine. The design of such a single user operating system can be very simple and efficient because it does not have to deal with the complications that arise due to multi-programming and protection. The virtual machine concept provides higher flexibility in that it allows different operating systems to run on different virtual machines. The efficient implementation of virtual machine software (e.g. VM/370), however, is a very difficult problem because virtual machine is huge and complex.

A classical example of this system is the IBM 370 system wherein the virtual machine software VM/370, provides a virtual machine to each user. When user logs on, VM/370 creates a new virtual machine (i.e. a copy of the bare hardware of the IBM 370 system) for the user.

The next unit gives you more explanation on Operating System Architecture.

Self Assessment Questions

4. A typical operating system that supports a multiprogramming environment will be less than 1MB in size. (True / False)
5. _____ suggested the layered approach to lessen the design and implementation complexities of an operating system.
6. The OS in IBM 370 is based on _____ approach.
   a) Kernel
   b) Virtual Machine
   c) Layered
   d) None of the above

1.5 Summary

Let's summarize the important points you learnt in this unit:

- An operating system is a collection of programs that is an intermediary between a user and the computer hardware.
- It performs various functions such as process management, input output management, file management, managing use of main memory, providing security to user jobs and files etc.
- In simple batch operating system, to speed up processing, jobs with the same needs were batched together and executed as a group.
- The primary objective of a multi-programmed operating system is to maximize the number of programs executed by a computer in a specified period and keep all the units of the computer simultaneously busy.
- A time shared operating system allows a number of users to simultaneously use a computer. The primary objective of a time shared operating system is to provide fast response to each user of the computer.
- The primary objective of Parallel System is to increase the throughput by getting done more jobs in less time.
- A real time system is used in the control of physical systems. The response time of such system should match the needs of the physical system.
- Since an operating system is large, modularity is important. The design of system as sequence of layers is considered first.
- The virtual machine concept takes the layered approach to heart and treats the kernel of the operating system and hardware as though they were all hardware.

1.6 Terminal Questions

1. Explain different functions of an operating system.
2. Explain the evolution of operating systems.
3. Write a brief note on operating system structures.

1.7 Answers

Self Assessment Questions

1. True
2. Simultaneous Peripheral Operation On Line

3. a
4. False
5. Dijkstra
6. b

Terminal Questions

1. Processor Management, Memory Management, Input Output Management, File System Management are some of the functions performed by Operating System. (Refer Section 1.2)
2. Operating systems have evolved over a period of time, starting from the very primitive systems to the present day complex and versatile ones. It all started with Batch OS and Multi-programmed Batch OS. Today we have Distributed OS and Real Time OS. (Refer Section 1.3)
3. There are different approaches to OS structures. Layered Approach, Kernel Based Approach and Virtual Machine Approach are some of them. (Refer Section 1.4)