UNIT 1 OPERATING SYSTEM : AN OVERVIEW

Structure

1.1 INTRODUCTION

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. This software is designed to solve a particular problem for users. Examples of application software include spreadsheets, database systems, desktop publishing systems, program development software, and games.

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.

An **Operating System** (OS) is a collection of programs that acts as an interface between a user of a computer and the computer hardware. The purpose of an operating system is to provide an environment in which a user may execute the programs. Operating Systems are viewed as resource managers. The main resource is the computer hardware in the form of processors, storage, input/output devices, communication devices, and data. Some of the operating system functions are: implementing the user interface, sharing hardware among users, allowing users to share data among themselves, preventing users from interfering with one another, scheduling resources among users, facilitating input/output, recovering from errors, accounting for resource usage, facilitating parallel operations, organizing data for secure and rapid access, and handling network communications.

This unit presents the definition of the operating system, goals of the operating system, generations of OS, different types of OS and functions of OS.

1.2 OBJECTIVES

After going through this unit, you should be able to:

- understand the purpose of an operating system;
- describe the general goals of an operating system;
- discuss the evolution of operating systems;
- describe various functions performed by the OS;
- list, discuss and compare various types of OS, and
- describe various structures of operating system.

1.3 WHAT IS AN OPERATING SYSTEM?

In a computer system, we find four main components: the hardware, the operating system, the application software and the users. In a computer system the hardware provides the basic computing resources. The applications programs define the way in which these resources are used to solve the computing problems of the users. The operating system controls and coordinates the use of the hardware among the various systems programs and application programs for the various users.

We can view an operating system as a **resource allocator**. A computer system has many resources (hardware and software) that may be required to solve a problem: CPU time, memory space, files storage space, input/output devices etc. The operating system acts as the manager of these resources and allocates them to specific programs and users as necessary for their tasks. Since there may be many, possibly conflicting, requests for resources, the operating system must decide which requests are allocated resources to operate the computer system fairly and efficiently.

An operating system is a **control program**. This program controls the execution of user programs to prevent errors and improper use of the computer. Operating systems exist because: they are a reasonable way to solve the problem of creating a usable computing system. The fundamental goal of a computer system is to execute user programs and solve user problems.

While there is no universally agreed upon definition of the concept of an operating system, the following is a reasonable starting point:

A computer's operating system is a group of programs designed to serve two basic purposes:

- To control the allocation and use of the computing system's resources among the various users and tasks, and
- To provide an interface between the computer hardware and the programmer that simplifies and makes feasible the creation, coding, debugging, and maintenance of application programs.

An effective operating system should accomplish the following functions:

- Should act as a command interpreter by providing a user friendly environment.
- Should facilitate communication with other users.
- Facilitate the directory/file creation along with the security option.
- Provide routines that handle the intricate details of I/O programming.
- Provide access to compilers to translate programs from high-level languages to machine language
- Provide a loader program to move the compiled program code to the computer's memory for execution.
- Assure that when there are several active processes in the computer, each will get fair and non-interfering access to the central processing unit for execution.
- Take care of storage and device allocation.
- Provide for long term storage of user information in the form of files.
- Permit system resources to be shared among users when appropriate, and be protected from unauthorised or mischievous intervention as necessary.

Though systems programs such as editors and translators and the various utility programs (such as sort and file transfer program) are not usually considered part of the operating system, the operating system is responsible for providing access to these system resources.

The abstract view of the components of a computer system and the positioning of OS is shown in the *Figure 1.*

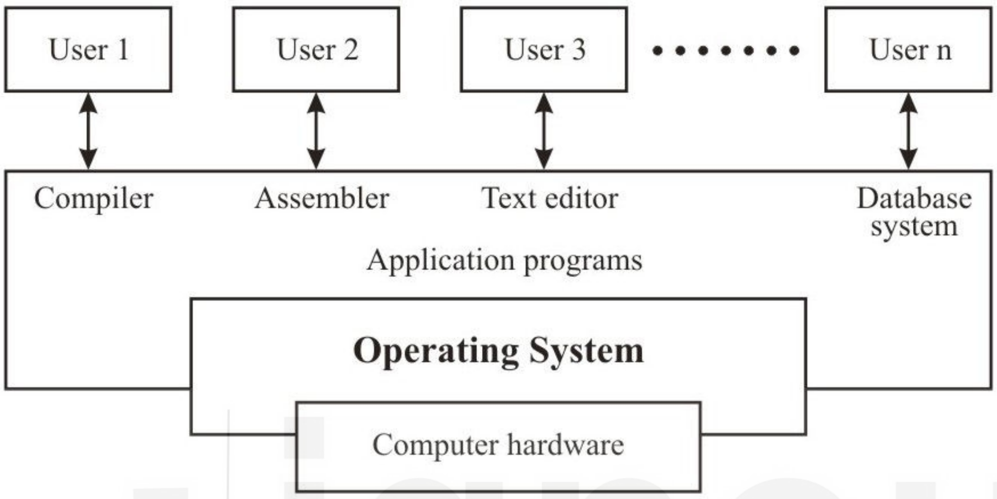

Figure 1: Abstract View of the Components of a Computer System

1.4 GOALS OF AN OPERATING SYSTEM

The primary objective of a computer is to execute an instruction in an efficient manner and to increase the productivity of processing resources attached with the computer system such as hardware resources, software resources and the users. In other words, we can say that maximum CPU utilisation is the main objective, because it is the main device which is to be used for the execution of the programs or instructions. We can brief the goals as:

- The primary goal of an operating system is to make the computer **convenient** to use.
- The secondary goal is to use the hardware in an **efficient** manner.

1.5 GENERATIONS OF OPERATING SYSTEMS

We have studied the definition and the goals of the operating systems.

Operating systems have been evolving over the years. We will briefly look at this development of the operating systems with respect to the evolution of the hardware / architecture of the computer systems in this section. Since operating systems have historically been closely tied with the architecture of the computers on which they run, we will look at successive generations of computers to see what their operating systems were like. We may not exactly map the operating systems generations to the generations of the computer, but roughly it provides the idea behind them.

We can roughly divide them into five distinct generations that are characterized by hardware component technology, software development, and mode of delivery of computer services.

1.4.1 0th Generation

The term 0th generation is used to refer to the period of development of computing, which predated the commercial production and sale of computer equipment. We consider that the period might be way back when Charles Babbage invented the Analytical Engine. Afterwards the computers by John Atanasoff in 1940; the Mark I, built by Howard Aiken and a group of IBM engineers at Harvard in 1944; the ENIAC, designed and constructed at the University of Pencylvania by Wallace Eckert and John Mauchly and the EDVAC, developed in 1944-46 by John Von Neumann, Arthur Burks, and Herman Goldstine (which was the first to fully implement the idea of the stored program and serial execution of instructions) were designed. The development of EDVAC set the stage for the evolution of commercial computing and operating system software. The hardware component technology of this period was *electronic vacuum tubes*.

The actual operation of these early computers took place **without the benefit of an operating system**. Early programs were written in machine language and each contained code for initiating operation of the computer itself.

The mode of operation was called "open-shop" and this meant that users signed up for computer time and when a user's time arrived, the entire (in those days quite large) computer system was turned over to the user. The individual user (programmer) was responsible for all machine set up and operation, and subsequent clean-up and preparation for the next user. This system was clearly inefficient and dependent on the varying competencies of the individual programmer as operators.

1.4.2 First Generation (1951-1956)

The first generation marked the beginning of commercial computing, including the introduction of Eckert and Mauchly's UNIVAC I in early 1951, and a bit later, the IBM 701 which was also known as the Defence Calculator. The first generation was characterised again by the vacuum tube as the active component technology.

Operation continued without the benefit of an operating system for a time. The mode was called "closed shop" and was characterised by the appearance of hired operators who would select the job to be run, initial program load the system, run the user's program, and then select another job, and so forth. Programs began to be written in higher level, procedure-oriented languages, and thus the operator's routine expanded. The operator now selected a job, ran the translation program to assemble or compile the source program, and combined the translated object program along with any existing library programs that the program might need for input to the linking program, loaded and ran the composite linked program, and then handled the next job in a similar fashion.

Application programs were run one at a time, and were translated with absolute computer addresses that bound them to be loaded and run from these reassigned storage addresses set by the translator, obtaining their data from specific physical I/O device. There was no provision for moving a program to different location in storage for any reason. Similarly, a program bound to specific devices could not be run at all if any of these devices were busy or broken.

The inefficiencies inherent in the above methods of operation led to the development of the ***mono-programmed operating system***, which eliminated some of the human intervention in running job and provided programmers with a number of desirable functions. The OS consisted of a permanently resident kernel in main storage, and a job scheduler and a number of utility programs kept in secondary storage. User application programs were preceded by control or specification cards (in those day, computer program were submitted on data cards) which informed the OS of what system resources (software resources such as compilers and loaders; and hardware resources such as tape drives and printer) were needed to run a particular application. The systems were designed to be operated as batch processing system.

These systems continued to operate under the control of a human operator who initiated operation by mounting a magnetic tape that contained the operating system executable code onto a "boot device", and then pushing the IPL (Initial Program Load) or "boot" button to initiate the bootstrap loading of the operating system. Once the system was loaded, the operator entered the date and time, and then initiated the operation of the job scheduler program which read and interpreted the control statements, secured the needed resources, executed the first user program, recorded timing and accounting information, and then went back to begin processing of another user program, and so on, as long as there were programs waiting in the input queue to be executed.

The first generation saw the evolution from hands-on operation to closed shop operation to the development of mono-programmed operating systems. At the same time, the development of programming languages was moving away from the basic machine languages; first to assembly language, and later to procedure oriented languages, the most significant being the development of FORTRAN by John W. Backus in 1956. Several problems remained, however, the most obvious was the inefficient use of system resources, which was most evident when the CPU waited while the relatively slower, mechanical I/O devices were reading or writing program data. In addition, system protection was a problem because the operating system kernel was not protected from being overwritten by an erroneous application program. Moreover, other user programs in the queue were not protected from destruction by executing programs.

1.4.3 Second Generation (1956-1964)

The second generation of computer hardware was most notably characterised by ***transistors*** replacing vacuum tubes as the hardware component technology. In addition, some very important changes in hardware and software architectures occurred during this period. For the most part, computer systems remained card and tape-oriented systems. Significant use of random access devices, that is, disks, did not appear until towards the end of the second generation. Program processing was, for the most part, provided by large centralised computers operated under ***mono-programmed batch processing operating systems***.

The most significant innovations addressed the problem of excessive central processor delay due to waiting for input/output operations. Recall that programs were executed by processing the machine instructions in a strictly sequential order. As a result, the CPU, with its high speed electronic component, was often forced to wait for completion of I/O operations which involved mechanical devices (card readers and tape drives) that were order of magnitude slower. This problem led to the introduction of the data channel, an integral and special-purpose computer with its own instruction set, registers, and control unit designed to process input/output operations separately and asynchronously from the operation of the computer's main CPU near the end of the first generation, and its widespread adoption in the second generation.

The data channel allowed some I/O to be buffered. That is, a program's input data could be read "ahead" from data cards or tape into a special block of memory called a buffer. Then, when the user's program came to an input statement, the data could be transferred from the buffer locations at the faster main memory access speed rather than the slower I/O device speed. Similarly, a program's output could be written another buffer and later moved from the buffer to the printer, tape, or card punch. What made this all work was the data channel's ability to work asynchronously and concurrently with the main processor. Thus, the slower mechanical I/O could be happening concurrently with main program processing. This process was called I/O overlap.

The data channel was controlled by a channel program set up by the operating system I/O control routines and initiated by a special instruction executed by the CPU. Then, the channel independently processed data to or from the buffer. This provided communication from the CPU to the data channel to initiate an I/O operation. It remained for the channel to communicate to the CPU such events as data errors and the completion of a transmission. At first, this communication was handled by polling – the CPU stopped its work periodically and polled the channel to determine if there is any message.

Polling was obviously inefficient (imagine stopping your work periodically to go to the post office to see if an expected letter has arrived) and led to another significant innovation of the second generation – the interrupt. The data channel was able to interrupt the CPU with a message – usually "I/O complete." Infact, the interrupt idea was later extended from I/O to allow signalling of number of exceptional conditions such as arithmetic overflow, division by zero and time-run-out. Of course, interval clocks were added in conjunction with the latter, and thus operating system came to have a way of regaining control from an exceptionally long or indefinitely looping program.

These hardware developments led to enhancements of the operating system. I/O and data channel communication and control became functions of the operating system, both to relieve the application programmer from the difficult details of I/O programming and to protect the integrity of the system to provide improved service to users by segmenting jobs and running shorter jobs first (during "prime time") and relegating longer jobs to lower priority or night time runs. System libraries became more widely available and more comprehensive as new utilities and application software components were available to programmers.

In order to further mitigate the I/O wait problem, system were set up to spool the input batch from slower I/O devices such as the card reader to the much higher speed tape drive and similarly, the output from the higher speed tape to the slower printer. In this scenario, the user submitted a job at a window, a batch of jobs was accumulated and spooled from cards to tape "off line," the tape was moved to the main computer, the jobs were run, and their output was collected on another tape that later was taken to a satellite computer for off line tape-to-printer output. User then picked up their output at the submission windows.

Toward the end of this period, as random access devices became available, tape-oriented operating system began to be replaced by disk-oriented systems. With the more sophisticated disk hardware and the operating system supporting a greater portion of the programmer's work, the computer system that users saw was more and more removed from the actual hardware-users saw a virtual machine.

The second generation was a period of ***intense operating system development***. Also it was the period for sequential batch processing. But the sequential processing of one job at a time remained a significant limitation. Thus, there continued to be low CPU utilisation for I/O bound jobs and low I/O device utilisation for CPU bound jobs. This was a major concern, since computers were still very large (room-size) and expensive machines. Researchers began to experiment with multiprogramming and multiprocessing in their computing services called the time-sharing system. A noteworthy example is the Compatible Time Sharing System (CTSS), developed at MIT during the early 1960s.

1.4.4 Third Generation (1964-1979)

The third generation officially began in April 1964 with IBM's announcement of its System/360 family of computers. Hardware technology began to use integrated circuits (ICs) which yielded significant advantages in both speed and economy.

Operating system development continued with the introduction and widespread adoption of multiprogramming. This marked first by the appearance of more sophisticated I/O buffering in the form of spooling operating systems, such as the HASP (Houston Automatic Spooling) system that accompanied the IBM OS/360 system. These systems worked by introducing two new systems programs, a system reader to move input jobs from cards to disk, and a system writer to move job output from disk to printer, tape, or cards. Operation of spooling system was, as before, transparent to the computer user who perceived input as coming directly from the cards and output going directly to the printer.

The idea of taking fuller advantage of the computer's data channel I/O capabilities continued to develop. That is, designers recognised that I/O needed only to be initiated by a CPU instruction – the actual I/O data transmission could take place under control of separate and asynchronously operating channel program. Thus, by switching control of the CPU between the currently executing user program, the system reader program, and the system writer program, it was possible to keep the slower mechanical I/O device running and minimizes the amount of time the CPU spent waiting for I/O completion. The net result was an increase in system throughput and resource utilisation, to the benefit of both user and providers of computer services.

This concurrent operation of three programs (more properly, apparent concurrent operation, since systems had only one CPU, and could, therefore execute just one instruction at a time) required that additional features and complexity be added to the operating system. First, the fact that the input queue was now on disk, a direct access device, freed the system scheduler from the first-come-first-served policy so that it could select the "best" next job to enter the system (looking for either the shortest job or the highest priority job in the queue). Second, since the CPU was to be shared by the user program, the system reader, and the system writer, some processor allocation rule or policy was needed. Since the goal of spooling was to increase resource utilization by enabling the slower I/O devices to run asynchronously with user program processing, and since I/O processing required the CPU only for short periods to initiate data channel instructions, the CPU was dispatched to the reader, the writer, and the program in that order. Moreover, if the writer or the user program was executing when something became available to read, the reader program would preempt the currently executing program to regain control of the CPU for its initiation instruction, and the writer program would preempt the user program for the same purpose. This rule, called the static priority rule with preemption, was implemented in the operating system as a system dispatcher program.

The spooling operating system in fact had multiprogramming since more than one program was resident in main storage at the same time. Later this basic idea of multiprogramming was extended to include more than one active user program in memory at time. To accommodate this extension, both the scheduler and the dispatcher were enhanced. The scheduler became able to manage the diverse resource needs of the several concurrently active used programs, and the dispatcher included policies for allocating processor resources among the competing user programs. In addition, memory management became more sophisticated in order to assure that the program code for each job or at least that part of the code being executed, was resident in main storage.

The advent of large-scale multiprogramming was made possible by several important hardware innovations such as:

- The widespread availability of large capacity, high-speed disk units to accommodate the spooled input streams and the memory overflow together with the maintenance of several concurrently active program in execution.
- Relocation hardware which facilitated the moving of blocks of code within memory without any undue overhead penalty.
- The availability of storage protection hardware to ensure that user jobs are protected from one another and that the operating system itself is protected from user programs.
- Some of these hardware innovations involved extensions to the interrupt system in order to handle a variety of external conditions such as program malfunctions, storage protection violations, and machine checks in addition to I/O interrupts. In addition, the interrupt system became the technique for the user program to request services from the operating system kernel.
- The advent of privileged instructions allowed the operating system to maintain coordination and control over the multiple activities now going on with in the system.

Successful implementation of multiprogramming opened the way for the development of a new way of delivering computing services-time-sharing. In this environment, several terminals, sometimes up to 200 of them, were attached (hard wired or via telephone lines) to a central computer. Users at their terminals, "logged in" to the central system, and worked interactively with the system. The system's apparent concurrency was enabled by the multiprogramming operating system. Users shared not only the system hardware but also its software resources and file system disk space.

The third generation was an exciting time, indeed, for the development of both computer hardware and the accompanying operating system. During this period, the topic of operating systems became, in reality, a major element of the discipline of computing.

1.4.5 Fourth Generation (1979 – Present)

The fourth generation is characterized by the appearance of the personal computer and the workstation. Miniaturization of electronic circuits and components continued and large scale integration (LSI), the component technology of the third generation, was replaced by very large scale integration (VLSI), which characterizes the fourth generation. VLSI with its capacity for containing thousands of transistors on a small chip, made possible the development of desktop computers with capabilities exceeding those that filled entire rooms and floors of building just twenty years earlier.

The operating systems that control these desktop machines have brought us back in a full circle, to the open shop type of environment where each user occupies an entire computer for the duration of a job's execution. This works better now, not only because the progress made over the years has made the virtual computer resulting from the operating system/hardware combination so much easier to use, or, in the words of the popular press "user-friendly."

However, improvements in hardware miniaturisation and technology have evolved so fast that we now have inexpensive workstation – class computers capable of supporting multiprogramming and time-sharing. Hence the operating systems that supports today's personal computers and workstations look much like those which were available for the minicomputers of the third generation. Examples are Microsoft's DOS for IBM-compatible personal computers and UNIX for workstation.

However, many of these desktop computers are now connected as networked or distributed systems. Computers in a networked system each have their operating systems augmented with communication capabilities that enable users to remotely log into any system on the network and transfer information among machines that are connected to the network. The machines that make up distributed system operate as a virtual single processor system from the user's point of view; a central operating system controls and makes transparent the location in the system of the particular processor or processors and file systems that are handling any given program.

☞ **Check Your Progress 1**

1) What are the main responsibilities of an Operating System?

....

....

....

....

1.5 TYPES OF OPERATING SYSTEMS

In this section we will discuss about the different types of operating systems.

Modern computer operating systems may be classified into three groups, which are distinguished by the nature of interaction that takes place between the computer user and his or her program during its processing. The three groups are called batch, time-sharing and real-time operating systems.

1.5.1 Batch Processing Operating System

In a batch processing operating system environment users submit jobs to a central place where these jobs are collected into a batch, and subsequently placed on an input queue at the computer where they will be run. In this case, the user has no interaction with the job during its processing, and the computer's response time is the turnaround time-the time from submission of the job until execution is complete, and the results are ready for return to the person who submitted the job.

1.5.2 Time Sharing

Another mode for delivering computing services is provided by time sharing operating systems. In this environment a computer provides computing services to several or many users concurrently on-line. Here, the various users are sharing the central processor, the memory, and other resources of the computer system in a manner facilitated, controlled, and monitored by the operating system. The user, in this environment, has nearly full interaction with the program during its execution, and the computer's response time may be expected to be no more than a few second.

1.5.3 Real Time Operating System (RTOS)

The third class is the real time operating systems, which are designed to service those applications where response time is of the essence in order to prevent error, misrepresentation or even disaster. Examples of real time operating systems are those which handle airlines reservations, machine tool control, and monitoring of a nuclear power station. The systems, in this case, are designed to be interrupted by external signals that require the immediate attention of the computer system.

These real time operating systems are used to control machinery, scientific instruments and industrial systems. An RTOS typically has very little user-interface capability, and no end-user utilities. A very important part of an RTOS is managing the resources of the computer so that a particular operation executes in precisely the same amount of time every time it occurs. In a complex machine, having a part move more quickly just because system resources are available may be just as catastrophic as having it not move at all because the system is busy.

A number of ***other definitions*** are important to gain an understanding of operating systems:

1.5.4 Multiprogramming Operating System

A multiprogramming operating system is a system that allows more than one active user program (or part of user program) to be stored in main memory simultaneously. Thus, it is evident that a time-sharing system is a multiprogramming system, but note that a multiprogramming system is not necessarily a time-sharing system. A batch or real time operating system could, and indeed usually does, have more than one active user program simultaneously in main storage. Another important, and all too similar, term is "multiprocessing".

1.5.5 Multiprocessing System

A multiprocessing system is a computer hardware configuration that includes more than one independent processing unit. The term multiprocessing is generally used to refer to large computer hardware complexes found in major scientific or commercial applications. More on multiprocessor system can be studied in Unit-1 of Block-3 of this course.

1.5.6 Networking Operating System

A networked computing system is a collection of physical interconnected computers. The operating system of each of the interconnected computers must contain, in addition to its own stand-alone functionality, provisions for handing communication and transfer of program and data among the other computers with which it is connected.

Network operating systems are not fundamentally different from single processor operating systems. They obviously need a network interface controller and some low-level software to drive it, as well as programs to achieve remote login and remote files access, but these additions do not change the essential structure of the operating systems.

1.5.7 Distributed Operating System

A distributed computing system consists of a number of computers that are connected and managed so that they automatically share the job processing load among the constituent computers, or separate the job load as appropriate particularly configured processors. Such a system requires an operating system which, in addition to the typical stand-alone functionality, provides coordination of the operations and information flow among the component computers. The networked and distributed computing environments and their respective operating systems are designed with more complex functional capabilities. In a network operating system, the users are aware of the existence of multiple computers, and can log in to remote machines and copy files from one machine to another. Each machine runs its own local operating system and has its own user (or users).

A distributed operating system, in contrast, is one that appears to its users as a traditional uni-processor system, even though it is actually composed of multiple processors. In a true distributed system, users should not be aware of where their programs are being run or where their files are located; that should all be handled automatically and efficiently by the operating system.

True distributed operating systems require more than just adding a little code to a uni-processor operating system, because distributed and centralised systems differ in critical ways. Distributed systems, for example, often allow program to run on several processors at the same time, thus requiring more complex processor scheduling algorithms in order to optimise the amount of parallelism achieved. More on distributed systems can be studied in Unit-2 of Block-3 of this course.

1.5.8 Operating Systems for Embedded Devices

As embedded systems (PDAs, cellphones, point-of-sale devices, VCR's, industrial robot control, or even your toaster) become more complex hardware-wise with every generation, and more features are put into them day-by-day, applications they run require more and more to run on actual operating system code in order to keep the development time reasonable. Some of the popular OS are:

- Nexus's Conix - an embedded operating system for ARM processors.
- Sun's Java OS - a standalone virtual machine not running on top of any other OS; mainly targeted at embedded systems.
- Palm Computing's Palm OS - Currently the leader OS for PDAs, has many applications and supporting companies.

- Microsoft's Windows CE and Windows NT Embedded OS.

1.6 DESIRABLE QUALITIES OF OS

The desirable qualities of an operating system are in terms of: Usability, Facilities, Cost, and Adaptability.

- Usability:
  - Robustness
  - *Accept all valid inputs and can handle them.*
  - Consistency
  - Proportionality
  - Convenience
  - Powerful with high level facilities.
- Facilities:
  - Sufficient for intended use
  - Complete
  - Appropriate.
- Costs:
  - Want low cost and efficient services.
  - Good algorithms.
    Make use of space/time tradeoffs, special hardware.
  - Low overhead.
    Cost of doing nothing should be low. E.g., idle time at a terminal.
  - Low maintenance cost.
    System should not require constant attention.
- Adaptability:
  - Tailored to the environment.
    Support necessary activities. Do not impose unnecessary restrictions. What are the things people do most — make them easy.
  - Changeable over time.
    Adapt as needs and resources change. e.g., expanding memory and new devices, or new user population.
  - Extendible-Extensible
    Adding new facilities and features - which look like the old ones.

1.7 OPERATING SYSTEMS: SOME EXAMPLES

In the earlier section we had seen the types of operating systems. In this section we will study some popular operating systems.

1.7.1 DOS

DOS (Disk Operating System) was the first widely-installed operating system for personal computers. It is a master control program that is automatically run when you start your personal computer (PC). DOS stays in the computer all the time letting you run a program and manage files. It is a single-user operating system from Microsoft for the PC. It was the first OS for the PC and is the underlying control program for Windows 3.1, 95, 98 and ME. Windows NT, 2000 and XP emulate DOS in order to support existing DOS applications.

1.7.2 UNIX

UNIX operating systems are used in widely-sold workstation products from Sun Microsystems, Silicon Graphics, IBM, and a number of other companies. The UNIX environment and the client/server program model were important elements in the development of the Internet and the reshaping of computing as centered in networks rather than in individual computers. Linux, a UNIX derivative available in both "free software" and commercial versions, is increasing in popularity as an alternative to proprietary operating systems.

UNIX is written in **C**. Both UNIX and C were developed by AT&T and freely distributed to government and academic institutions, causing it to be ported to a wider variety of machine families than any other operating system. As a result, UNIX became synonymous with "open systems".

UNIX is made up of the kernel, file system and shell (command line interface). The major shells are the Bourne shell (original), C shell and Korn shell. The UNIX vocabulary is exhaustive with more than 600 commands that manipulate data and text in every way conceivable. Many commands are cryptic, but just as Windows hid the DOS prompt, the Motif GUI presents a friendlier image to UNIX users. Even with its many versions, UNIX is widely used in mission critical applications for client/server and transaction processing systems. The UNIX versions that are widely used are Sun's Solaris, Digital's UNIX, HP's HP-UX, IBM's AIX and SCO's UnixWare. A large number of IBM mainframes also run UNIX applications, because the UNIX interfaces were added to MVS and OS/390, which have obtained UNIX branding. Linux, another variant of UNIX, is also gaining enormous popularity. More details can be studied in Unit-3 of Block-3 of this course.

1.7.3 WINDOWS

Windows is a personal computer operating system from Microsoft that, together with some commonly used business applications such as Microsoft Word and Excel, has become a *de facto* "standard" for individual users in most corporations as well as in most homes. Windows contains built-in networking, which allows users to share files and applications with each other if their PC's are connected to a network. In large enterprises, Windows clients are often connected to a network of UNIX and NetWare servers. The server versions of Windows NT and 2000 are gaining market share, providing a Windows-only solution for both the client and server. Windows is supported by Microsoft, the largest software company in the world, as well as the Windows industry at large, which includes tens of thousands of software developers.

This networking support is the reason why Windows became successful in the first place. However, Windows 95, 98, ME, NT, 2000 and XP are complicated operating environments. Certain combinations of hardware and software running together can cause problems, and troubleshooting can be daunting. Each new version of Windows has interface changes that constantly confuse users and keep support people busy, and Installing Windows applications is problematic too. Microsoft has worked hard to make Windows 2000 and Windows XP more resilient to installation of problems and crashes in general. More details on Windows 2000 can be studied in Unit-4 of Block -3 of this course.

1.7.4 MACINTOSH

The Macintosh (often called "the Mac"), introduced in 1984 by Apple Computer, was the first widely-sold personal computer with a graphical user interface (GUI). The Mac was designed to provide users with a natural, intuitively understandable, and, in general, "user-friendly" computer interface. This includes the mouse, the use of icons or small visual images to represent objects or actions, the point-and-click and click-and-drag actions, and a number of window operation ideas. Microsoft was successful in adapting user interface concepts first made popular by the Mac in its first Windows operating system. The primary disadvantage of the Mac is that there are fewer Mac applications on the market than for Windows. However, all the fundamental applications are available, and the Macintosh is a perfectly useful machine for almost everybody. Data compatibility between Windows and Mac is an issue, although it is often overblown and readily solved.

The Macintosh has its own operating system, Mac OS which, in its latest version is called Mac OS X. Originally built on Motorola's 68000 series microprocessors, Mac versions today are powered by the PowerPC microprocessor, which was developed jointly by Apple, Motorola, and IBM. While Mac users represent only about 5% of the total numbers of personal computer users, Macs are highly popular and almost a cultural necessity among graphic designers and online visual artists and the companies they work for.

In this section we will discuss some services of the operating system used by its users. Users of operating system can be divided into two broad classes: command language users and system call users. Command language users are those who can interact with operating systems using the commands. On the other hand system call users invoke services of the operating system by means of run time system calls during the execution of programs.

More can be read from Block-4 of MCS-203 Operating Systems.

☞ **Check Your Progress 2**

1) Discuss different views of the Operating Systems.

....

....

2) Summarise the characteristics of the following operating systems:

- Batch Processing

- Time Sharing
- Real time

....

....

1.8 FUNCTIONS OF OS

The main functions of an operating system are as follows:

- Process Management
- Memory Management
- Secondary Storage Management
- I/O Management
- File Management
- Protection
- Networking Management
- Command Interpretation.

1.8.1 Process Management

The CPU executes a large number of programs. While its main concern is the execution of user programs, the CPU is also needed for other system activities. These activities are called processes. A process is a program in execution. Typically, a batch job is a process. A time-shared user program is a process. A system task, such as spooling, is also a process. For now, a process may be considered as a job or a time-shared program, but the concept is actually more general.

The operating system is responsible for the following activities in connection with processes management:

- The creation and deletion of both user and system processes
- The suspension and resumption of processes.
- The provision of mechanisms for process synchronization
- The provision of mechanisms for deadlock handling.

  More details can be studied in the Unit-2 of this course.

1.8.2 Memory Management

Memory is the most expensive part in the computer system. Memory is a large array of words or bytes, each with its own address. Interaction is achieved through a sequence of reads or writes of specific memory address. The CPU fetches from and stores in memory.

There are various algorithms that depend on the particular situation to manage the memory. Selection of a memory management scheme for a specific system depends upon many factors, but especially upon the hardware design of the system. Each algorithm requires its own hardware support.

The operating system is responsible for the following activities in connection with memory management.

- Keep track of which parts of memory are currently being used and by whom.
- Decide which processes are to be loaded into memory when memory space becomes available.
- Allocate and deallocate memory space as needed.

Memory management techniques will be discussed in great detail in Unit-1 of Block-2 of this course.

1.8.3 Secondary Storage Management

The main purpose of a computer system is to execute programs. These programs, together with the data they access, must be in main memory during execution. Since the main memory is too small to permanently accommodate all data and program, the computer system must provide secondary storage to backup main memory. Most modem computer systems use disks as the primary on-line storage of information, of both programs and data. Most programs, like compilers, assemblers, sort routines, editors, formatters, and so on, are stored on the disk until loaded into memory, and then use the disk as both the source and destination of their processing. Hence the proper management of disk storage is of central importance to a computer system.

There are few alternatives. Magnetic tape systems are generally too slow. In addition, they are limited to sequential access. Thus tapes are more suited for storing infrequently used files, where speed is not a primary concern.

The operating system is responsible for the following activities in connection with disk management:

- Free space management
- Storage allocation
- Disk scheduling.

More details can be studied in Unit-3 of Block-2 of this course.

1.8.4 I/O Management

One of the purposes of an operating system is to hide the peculiarities or specific hardware devices from the user. For example, in UNIX, the peculiarities of I/O devices are hidden from the bulk of the operating system itself by the I/O system. The operating system is responsible for the following activities in connection to I/O management:

- A buffer caching system
- To activate a general device driver code
- To run the driver software for specific hardware devices as and when required.

More details can be studied in the Unit-3 of Block-2 of this course.

1.8.5 File Management

File management is one of the most visible services of an operating system. Computers can store information in several different physical forms: magnetic tape, disk, and drum are the most common forms. Each of these devices has it own characteristics and physical organisation.

For convenient use of the computer system, the operating system provides a uniform logical view of information storage. The operating system abstracts from the physical properties of its storage devices to define a logical storage unit, the file. Files are mapped, by the operating system, onto physical devices.

A file is a collection of related information defined by its creator. Commonly, files represent programs (both source and object forms) and data. Data files may be numeric, alphabetic or alphanumeric. Files may be free-form, such as text files, or may be rigidly formatted. In general a files is a sequence of bits, bytes, lines or records whose meaning is defined by its creator and user. It is a very general concept.

The operating system implements the abstract concept of the file by managing mass storage device, such as types and disks.Also files are normally organised into directories to ease their use. Finally, when multiple users have access to files, it may be desirable to control by whom and in what ways files may be accessed.

The operating system is responsible for the following activities in connection to the file management:

- The creation and deletion of files.
- The creation and deletion of directory.
- The support of primitives for manipulating files and directories.
- The mapping of files onto disk storage.
- Backup of files on stable (non volatile) storage.
- Protection and security of the files.

1.8.6 Protection

The various processes in an operating system must be protected from each other's activities. For that purpose, various mechanisms which can be used to ensure that the files, memory segment, CPU and other resources can be operated on only by those processes that have gained proper authorisation from the operating system.

For example, memory addressing hardware ensures that a process can only execute within its own address space. The timer ensures that no process can gain control of the CPU without relinquishing it. Finally, no process is allowed to do its own I/O, to protect the integrity of the various peripheral devices. Protection refers to a mechanism for controlling the access of programs, processes, or users to the resources defined by a computer controls to be imposed, together with some means of enforcement.

Protection can improve reliability by detecting latent errors at the interfaces between component subsystems. Early detection of interface errors can often prevent contamination of a healthy subsystem by a subsystem that is malfunctioning. An unprotected resource cannot defend against use (or misuse) by an unauthorised or incompetent user. More on protection and security can be studied in Unit-4 of Block-3.

1.8.7 Networking

A distributed system is a collection of processors that do not share memory or a clock. Instead, each processor has its own local memory, and the processors communicate with each other through various communication lines, such as high speed buses or telephone lines. Distributed systems vary in size and function. They may involve microprocessors, workstations, minicomputers, and large general purpose computer systems.

The processors in the system are connected through a communication network, which can be configured in the number of different ways. The network may be fully or partially connected. The communication network design must consider routing and connection strategies and the problems of connection and security.

A distributed system provides the user with access to the various resources the system maintains. Access to a shared resource allows computation speed-up, data availability, and reliability.

1.8.8 Command Interpretation

One of the most important components of an operating system is its command interpreter. The command interpreter is the primary interface between the user and the rest of the system.

Many commands are given to the operating system by control statements. When a new job is started in a batch system or when a user logs-in to a time-shared system, a program which reads and interprets control statements is automatically executed. This program is variously called (1) the control card interpreter, (2) the command line interpreter, (3) the shell (in Unix), and so on. Its function is quite simple: get the next command statement, and execute it.

The command statements themselves deal with process management, I/O handling, secondary storage management, main memory management, file system access, protection, and networking.

The *Figure 2* depicts the role of the operating system in coordinating all the functions.

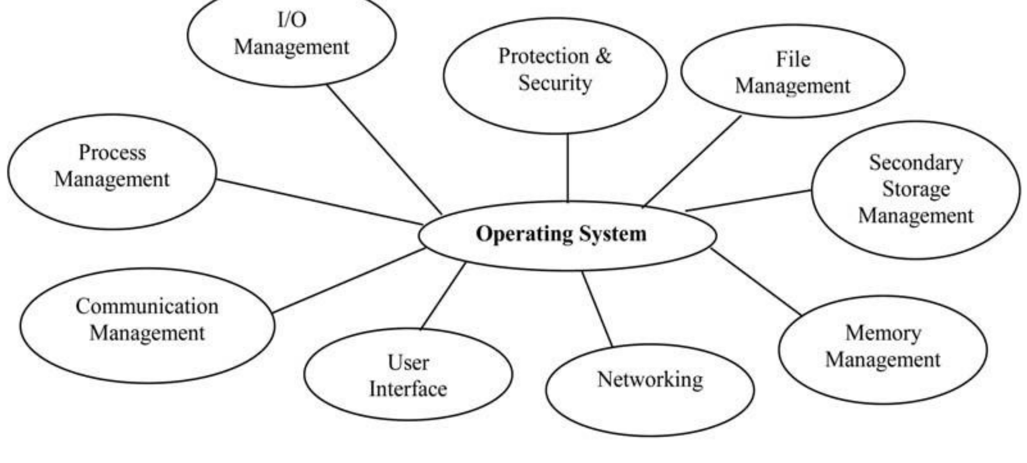

Figure 2: Functions Coordinated by the Operating System

☞ **Check Your Progress 3**

1) Mention the advantages and limitations of Multiuser Operating Systems

....

....

....

2) What is a Multitasking system and mention its advantages.

....

....

....

3) Illustrate a simple operating system for a security control system.

....

....

....

1.9 SUMMARY

This Unit presented the principle operation of an operating system. In this unit we had briefly described about the history, the generations and the types of operating systems.

An operating system is a program that acts as an interface between a user of a computer and the computer hardware. The purpose of an operating system is to provide an environment in which a user may execute programs. The primary goal of an operating system is to make the computer **convenient** to use. And the secondary goal is to use the hardware in an **efficient** manner.

Operating systems may be classified by both how many tasks they can perform "simultaneously" and by how many users can be using the system "simultaneously". That is: *single-user* or *multi-user* and *single-task* or *multi-tasking*. A multi-user system must clearly be multi-tasking. In the next unit we will discuss the concept of processes and their management by the OS.

1.10 SOLUTIONS/ANSWERS

Check Your Progress 1

1) The most important type of system software is the operating system. An operating system has three main responsibilities:

   - Perform basic tasks, such as recognising input from the keyboard, sending output to the display screen, keeping track of files and directories on the disk, and controlling peripheral devices such as disk drives and printers.
   - Ensure that different programs and users running at the same time do not interfere with each other.

- 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.

Check Your Progress 2

1) The following are the different views of the operating system.

   As a scheduler/allocator:

   - The operating system has resources for which it is in charge.
   - Responsible for handing them out (and later recovering them).
   - Resources include CPU, memory, I/O devices, and disk space.

   As a virtual machine:

   - Operating system provides a "new" machine.
     – This machine could be the same as the underlying machine. Allows many users to believe they have an entire piece of hardware to themselves.
     – This could implement a different, perhaps more powerful, machine. Or just a different machine entirely. It may be useful to be able to completely simulate another machine with your current hardware.

   As a multiplexer:

   - Allows sharing of resources, and provides protection from interference.
   - Provides for a level of cooperation between users.
   - Economic reasons: we have to take turns.

2) **Batch Processing**: This strategy involves reading a series of jobs (called a batch) into the machine and then executing the programs for each job in the batch. This approach does not allow users to interact with programs while they operate.

   Time Sharing: This strategy supports multiple interactive users. Rather than preparing a job for execution ahead of time, users establish an interactive session with the computer and then provide commands, programs and data as they are needed during the session.

   Real Time: This strategy supports real-time and process control systems. These are the types of systems which control satellites, robots, and air-traffic control. The dedicated strategy must guarantee certain response times for particular computing tasks or the application is useless.

Check Your Progress 3

1) The advantage of having a multi-user operating system is that normally the hardware is very expensive, and it lets a number of users share this expensive resource. This

means the cost is divided amongst the users. It also makes better use of the resources. Since the resources are shared, they are more likely to be in use than sitting idle being unproductive.

One limitation with multi-user computer systems is that as more users access it, the performance becomes slower and slower. Another limitation is the cost of hardware, as a multi-user operating system requires a lot of disk space and memory. In addition, the actual software for multi-user operating systems tend to cost more than single-user operating systems.

2) A multi-tasking operating system provides the ability to run more than one program at once. For example, a user could be running a word processing package, printing a document, copying files to the floppy disk and backing up selected files to a tape unit. Each of these tasks the user is doing appears to be running at the same time.

   A multi-tasking operating system has the advantage of letting the user run more than one task at once, so this leads to increased productivity. The disadvantage is that more programs that are run by the user, the more memory is required.

3) An operating system for a security control system (such as a home alarm system) would consist of a number of programs. One of these programs would gain control of the computer system when it is powered on, and initialise the system. The first task of this initialise program would be to reset (and probably test) the hardware sensors and alarms. Once the hardware initialisation was complete, the operating system would enter a continual monitoring routine of all the input sensors. If the state of any input sensor is changed, it would branch to an alarm generation routine.

1.11 FURTHER READINGS

1) Madnick and Donovan, *Operating systems – Concepts and design,* McGrawHill Intl. Education.

2) Milan Milenkovic, *Operating Systems, Concepts and Design*, TMGH, 2000.

3) D.M. Dhamdhere, *Operating Systems, A concept-based approach*, TMGH, 2002.

4) Abraham Silberschatz and James L, *Operating System Concepts*. Peterson, Addition Wesley Publishing Company.

5) Harvay M. Deital, *Introduction to Operating Systems,* Addition Wesley Publishing Company.

6) Andrew S. Tanenbaum, *Operating System Design and Implementation,* PHI.