**OPERATING SYSTEM - OVERVIEW**

An operating System (OS) is an intermediary between users and computer hardware. It provides users an environment in which a user can execute programs conveniently and efficiently.

In technical terms, It is a software which manages hardware. An operating System controls the allocation of resources and services such as memory, processors, devices and information.

**Definition**

An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.



An Os is a program that manages the computer hardware. It also provides a basis for application programs and acts as an intermediary between the computer user and the computer hardware. An amazing aspect of operating systems is how varied they are in accomplishing these tasks.

*Mainframe operating systems are designed primarily to optimize utilization of hardware.*  *Personal computer (PC) operating systems support complex games, business applications, and everything in between.* *Operating systems for handheld computers are designed to provide an environment in which a user can easily interface with the computer to execute programs.* Thus, some operating systems are designed to be *convenient,* others to be *efficient,* and others some *combination of the two.*

Because an operating system is large and complex, it must be created piece by piece. Each of these pieces should be a well-delineated portion of the system, with carefully defined inputs, outputs, and functions. To understand more fully the operating systems role, we explore operating systems from two viewpoints: that of the user and that of the system.

**1.1 User View**

The user's view of the computer varies according to the interface being used. Most computer users sit in front of a PC, consisting of a monitor/keyboard/ mouse, and system unit. Such a system is designed for one user to monopolize its resources. The goal is to maximize the work (or play) that the user is performing. In this case *the operating system is designed mostly for ease of use with some attention paid to performance and none paid to resource utilization various hardware and software resources are shared.* Performance is, of course, important to the user; but such systems are optimized for the single-user experience rather than the requirements of multiple users.

**1.1.2 System View**

From the computer's point of view, the operating system is the programs intimately involved with the hardware. In this context, we can view an *operating system as a resource allocator.* *A computer system has many resources that may be required to solve a problem: CP*U time, memory space, file-storage space, I/0 devices, and so on. *The operating system acts as the manager of these resources.* Facing numerous and possibly conflicting requests for resources, the operating system must decide how to allocate them to specific programs and users so that it can operate the computer system efficiently and fairly. As we have seen, resource allocation is especially important where many users access the same mainframe or minicomputer. A slightly different view of an operating system emphasizes the need to control the various I/0 devices and user programs. *An operating system is a control program*. *A control program manages the execution of user programs to prevent errors and improper use of the computer. It is especially concerned with the operation and control of I/O devices.*

1.2 **Defining Operating Systems**

We have looked at the operating system's role from the views of the user and of the system. How, though, can we define what an operating system is? In general, we have no completely adequate definition of an operating system. *Operating systems exist because they offer 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. Toward this goal, computer hardware is constructed. Since bare hardware alone is not particularly easy to use, application programs are developed. These programs require certain common operations, such as those controlling the I/ 0 devices. The common functions of controlling and allocating resources are then brought together into one piece of software: the operating system.

**1.3 Operating System** **Structure**

An operating system provides the environment within which programs are executed. Internally, operating systems vary greatly in their makeup, since they are organized along many different lines.

**Batch processing**

Batch processing is a technique in which Operating System collects one programs and data together in a batch before processing starts. Operating system does the following activities related to batch processing .

* OS defines a job which has predefined sequence of commands, programs and data as a single unit.
* OS keeps a number a jobs in memory and executes them without any manual information.
* Jobs are processed in the order of submission i.e. first come first served fashion.
* When job completes its execution, its memory is released and the output for the job gets copied into an output spool for later printing or processing .



**Advantages**

* Batch processing takes much of the work of the operator to the computer.
* Increased performance as a new job get started as soon as the previous job finished without any manual intervention.

**Disadvantages**

* Difficult to debug program.
* A job could enter an infinite loop.
* Due to lack of protection scheme, one batch job can affect pending jobs.

**Multiprogramming**

One of the most important aspects of operating systems is the ability to multi-program. A single program cannot, in general keep either the CPU or I/O devices busy at all times: Single users frequently have multiple programs running.

When two or more programs are residing in memory at the same time, then sharing the processor is referred to the multiprogramming . Multiprogramming assumes a sing le shared processor.  *Multiprogramming increases CPU utilization by organizing jobs(code and data) so that the CPU always has one to execute the idea is as follows:* The operating system keeps several jobs in memory simultaneously. Since, in generate main memory is too small to accommodate all jobs, the jobs are kept initially on the disk in the job poo1. This pool consists of all processes residing on disk awaiting allocation of main memory.

The set of jobs in memory can be a subset of the jobs kept in the job poo1. The operating system picks and begins to execute one of the jobs in memory. Eventually, the job may have to wait for some task, such as an I/O operation, to complete. In a non-multi-programmed system, the CPU would sit idle. In a multi-programmed system, the operating system simply switches to, and executes, another job. When thatjob needs to wait, the CPU is switched to *another* job, and so on. Eventually the first job finishes waiting and gets the CPU back. As long as at least one job needs to execute, the CPU is never idle.



Memory layout for a multiprogramming system.

Multi-programmed systems provide an environment in which the various system resources (for example, CPU, memory, and peripheral devices) are utilized effectively, but they do not provide for user interaction with the computer system.

Operating system does the following activities related to multiprogramming :

* The operating system keeps several jobs in memory at a time.
* This set of jobs is a subset of the jobs kept in the job pool.
* The operating system picks and beg ins to execute one of the job in the memory.
* Multiprogramming operating system monitors the state of all active programs and system resources using memory management programs to ensures that the CPU is never idle unless there are no jobs

**Advantages**

* High and efficient CPU utilization.
* User feels that many programs are allotted CPU almost simultaneously.

**Disadvantages**

* CPU scheduling is required.
* To accommodate many jobs in memory, memory management is required.

**Multitasking (Time sharing­)**

Multitasking refers to term where multiple jobs are executed by the CPU simultaneously by switching between them. Switches occur so frequently that the users may interact with each program while it is running .

Time sharing or (multitasking) is a logical extension of multi-programming. Time sharing systems, the CPU executes multiple jobs by switching among them, but the switches occur so frequently that the users can interact with each program while its running. Time sharing requires an interactive or (hands on) computer system, which provides direct communication between the user and the system. The user gives instructions to the operating system or to a program directly, using a input device such as a keyboard or a mouse, and waits for immediate results on an output device. Accordingly, the response time should be short typically less than one second. 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 the entire computer system is dedicated to his use, even though it is being shared among many users.

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 loaded into memory and executing is called a Process . When a process executes, it typically executes for only a short time before it either finishes or needs to perform I/0. I/0 may be interactive; that is, output goes to a display for the user, and input comes from a user keyboard, mouse, or other device. Since interactive I/0 typically runs at "people speeds," it may take a long time to complete. Input, for example, may be bounded by the user's typing speed; seven characters per second is fast for people but incredibly slow for computers. Rather than let the CPU sit idle as this interactive input takes place, the operating system will rapidly switch the CPU to the program of some other user.

Time sharing and multiprogramming require that several jobs be kept simultaneously in memory. If several jobs are ready to be brought into memory, and if there is not enough room for all of them, then the system must choose among them. Making this decision is job scheduling . When the operating system selects a job from the job pool, it loads that job into memory for execution. Having several programs in memory at the same time requires some form of memory management. In addition, if several Jobs are ready to run the same time, the system must choose among them. Making this decision is CPU scheduling. Finally, running multiple jobs concurrently requires that their ability to affect one another be limited in all phases of the operating system, including process scheduling, disk storage, and memory management. These considerations are discussed later.

In a time-sharing system, the operating system must ensure reasonable response time, which is sometimes accomplished through swapping where processes are swapped in and out of main memory to the disk. *A* more common method for achieving this goal is virtual memory, a technique that allows the execution of a process that is not completely in memory. The main advantage of the virtual-memory scheme is that it enables users to run programs that are larger than actual



Operating system does the following activities related to multitasking :

* The user gives instructions to the operating system or to a program directly, and receives an immediate response.
* Operating System handles multitasking in the way that it can handle multiple operations / executes multiple programs at a time.
* Multitasking Operating Systems are also known as Time-sharing systems.
* These Operating Systems were developed to provide interactive use of a computer system at a reasonable cost.
* A time-shared operating system uses concept of CPU scheduling and multiprogramming to provide each user with a small portion of a time-shared CPU.
* 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 very short time before it either finishes or needs to perform I/O.
* Since interactive I/O typically runs at people speeds, it may take a long time to completed. During this time a CPU can be utilized by another process.
* Operating system allows the 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 CPU rapidly from one user/program to the next, each user is given the impression that he/she has his/her own CPU, whereas actually one CPU is being shared among many users.

**Interactivity**

Interactivity refers that a User is capable to interact with computer system. Operating system does the following activities related to interactivity:

* OS provides user an interface to interact with system.
* OS managers input devices to take inputs from the user. For example, keyboard.
* OS manages output devices to show outputs to the user. For example, Monitor.
* OS Response time needs to be short since the user submits and waits for the result.

**Real Time System**

Real time systems represents are usually dedicated, embedded systems. Operating system does the following activities related to real time system activity.

* In such systems, Operating Systems typically read from and react to sensor data.
* The Operating system must guarantee response to events within fixed periods of time to ensure correct performance.

**Distributed Environment**

Distributed environment refers to multiple independent CPUs or processors in a computer system. Operating system does the following activities related to distributed environment:

* OS Distributes computation logics among several physical processors.
* The processors do not share memory or a clock.
* Instead, each processor has its own local memory.
* OS manages the communications between the processors. They communicate with each other through various communication lines.