Introduction to Operating System

Without its Software, a computer basically is useless. With its software, a computer can store, process and retrieve information etc and can be engaged in many other valuable activities.

Computer Software can be roughly divided into two kinds

i) System Software
ii) Application Software

Application Software is designed for a particular Application and solves the problems of the users. Application software's are Business Software, Engineering & Scientific Software, Embedded Software, artificial Intelligence Software etc.

System Software manages the operation of the computer and is a set of programs written to operate and control the operation of a computer system. Objectives of this type of software are to control and operate hardware components of computer system in such a way that they yield maximum efficiency.

Operating System is a kind of System Software which controls and operates computer resources i.e. memory and I/O Devices etc can be utilized up to their full potential. So, an Operating System is a program that acts as an Interface between a computer user and computer hardware, and the purpose of an Operating System is to provide an environment in which a user can work in a convenient and efficient manner.

Operating Systems available are from Text-Based Operating Systems to GUI Operating Systems. Similarly, variety in Operating Systems is from Single user to Multi user to Network Operating Systems. Goal of an Operating System is Convenience and Efficiency. Along with these two goals, another thing that emerged is the ability to evolve i.e. What was before and what is now, and all the Modern Operating Systems of now a days work like this.

Services provided by Operating System

The Operating System provides some services to programs and to the users of those programs, in order to make programming task easier.

Although services provided by Operating System differ from one Operating System to another, but there are some Services that are provided by almost every Operating System.

Services provided by Operating System are

1. Program Execution.

The Operating System should be able to load a program into memory and execute it.

2. I/O Devices.

Normally, a user program cannot execute I/O operations directly. So when a running program needs an I/O, the Operating System should be able to fulfill the need of that program.

3. File System.

Programs need to read and write files. Similarly, new files can be created and unwanted existing files are deleted. Operating System should be able to manage the activities related to files.

4. Communication.

Sometimes, programs need to exchange information with each other. Exchange of information can be implemented using a Shared Memory (Same Machines) or Message Passing in the form of Packets (On a Network). So, Operating System should be able to communicate between programs.

5. Error Detection.

The Operating System should always be aware of possible errors. Errors occur due to Hardware i.e. CPU, Memory or I/O Devices etc or due to Software i.e. User Programs etc. The Operating System should take necessary steps to ensure correct and constant computing.

Following are the Services that do not help User Programs but are used for Sharing Computer Resources and giving Reliability/Stability and Efficiency.

6. Resource Allocation.

Operating System is the manager of resources and when resources are requested by different users at the same time, then Operating System Allocates and Deallocates these resources to many Users/Programs in an efficient manner.

7. Accounting.

Accounting is used for two purposes.

i) To keep track of resources, i.e. which resources are used by which users so that they can be billed.

ii) Usage Statistics of resources may be a valuable tool for researchers in order to reconfigure the system for improving Computing Services.

8. Protection & Security.

Protection & Security is a very important service provided by Operating System in Multi user Environment.

Protection ensures that all access to System Resources is controlled and several jobs executed simultaneously should not interferee with each other.

Security ensures that outsiders wil not gain access to the system. Security is implemented on users to specify passwords in order to gain access to System Resources.

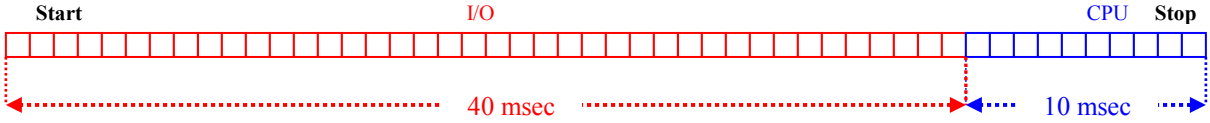
Multiprogramming

Multiprogramming is used for improving CPU utilization. We know that in uni programming or mono programming only one process is executed at one time i.e. program loaded and executed at the command typed by the user and after the completion of process Operating System gives control back to the user by prompting on the terminal and wait for the next command.

Although mono programming is sometimes used on small computers but on large computers with multiple users it is not used. Reasons for Multiprogramming

- Applications are made easier to program if we split them into two or more processes.
- Large computers are used by several people which leads to presence of more than one process in the memory.
- Large computers are very expensive, so if CPU remains idle, precious time of CPU is wasted. To utilize the CPU more efficiently more processes need to be in the memory, so that if one process waits another may be given the control of CPU.

In mono programming if a process is waiting for an I/O to complete and it takes 40msec to complete the I/O operation and the computation takes 10msec then the CPU was idle waiting for I/O operation to complete 80 percent of the time.

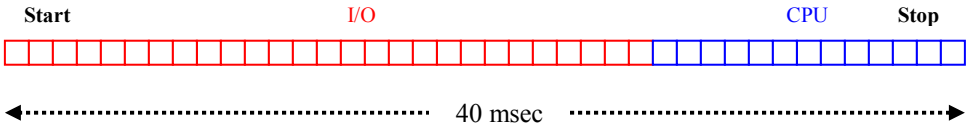


CPU Utilization = [(CPU Time / Total time) x 100]
CPU Utilization = [(10 / 50) x 100]
CPU Utilization = 20 %

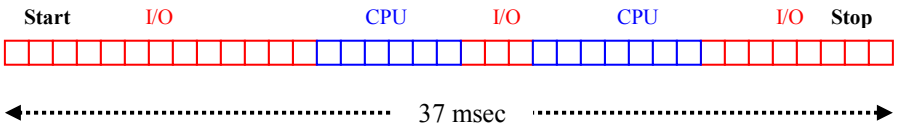
CPU Wastage = [(I/O Time / Total time) x 100]
CPU Wastage = [(40 / 50) x 100]
CPU Wastage = 80 %

Calculate CPU Utilization and CPU Wastage of the following time line of processes.

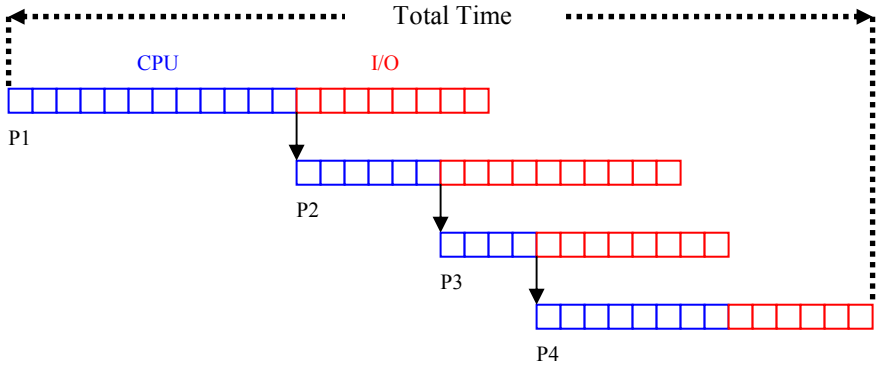
1.



2.



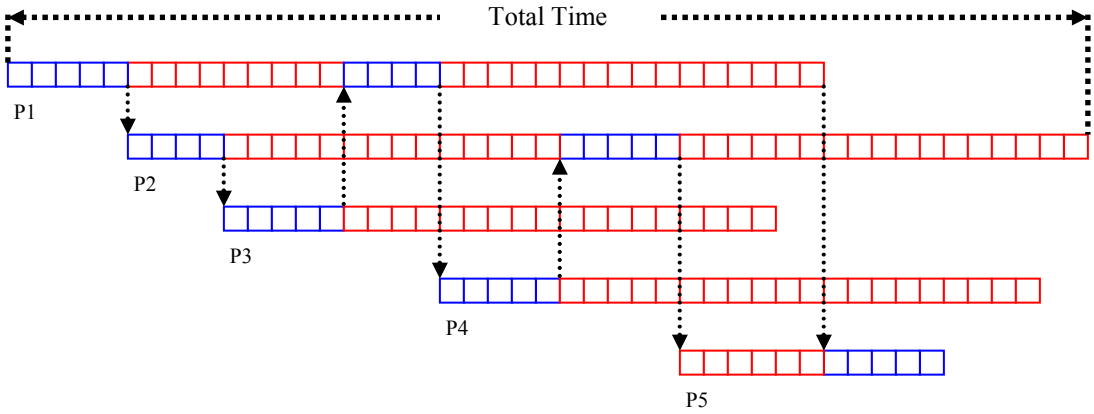
In multiprogramming, CPU will not sit idle, as the Operating System will switch it to the next job. When that job needs to wait, the CPU will again be switched to another job, and so on.



Process / Job execution in Multi-Programmed Operating System

CPU Utilization = {[CPU Times of P1 + P2 + P3 + P4 / Total Time (CPU + I/O)] x 100}
CPU Utilization = [(12 + 6 + 4 + 8 / 36) x 100]
CPU Utilization = 83.33 %

Calculate CPU Utilization and CPU Wastage of the following time line of processes.



In multiprogramming CPU sits idle only when there is not job to execute. Multiprogramming is done in two ways.

- **Multiprogramming with Fixed partition.**
- **Multiprogramming with Variable partition.**

Time Sharing (Multi-Tasking)

Time sharing also called Multi-tasking is a logical extension of multi-programming. Multiple jobs are executed by the CPU switching between them and the switches occur so frequently that the user may interact with each program while it is running.

The rapid switching back and forth of the CPU between programs is called "pseudoparellelism". As the system switches rapidly from one user to next, each user assumes that only he is using the system, but in actuality CPU is sharing its time with all the users.

Time sharing systems are sophisticated. A time-shared operating system uses CPU scheduling and Multi-programming to provide each user with a small portion of a time-shared computer. Each user has a program in memory and when it executes then only a small portion of time is given to that. In that time either it finishes or needs to perform I/O operations. As I/O operations are slower, so operating system rapidly switches the CPU to other program and like this instead of sitting idle, the CPU time is given to another process.

Multi Processing

Systems that have more than one CPU are called "Multi-processing System". In multiprocessing systems, CPU shares the computer bus and sometimes memory and I/O devices etc.

Multi-processor systems are more reliable and fast as compared to the systems that can not do multiprocessing. The speed-up ratio with N processors is not N, but less than N. When multiple processors process the tasks, then many overheads are to be catered in order to keep the process working correctly, so this catering of overheads lowers the expected results.

Multiprocessor systems are reliable in the sense that system will not halt or crash on failing a processor, but will only slow down the speed. Suppose a multiprocessor system has ten processors and one out of ten fails, then remaining nine processors will pick up the share of the work of the failed processor, so the entire system will run only 10 percent slow, instead of failing altogether.

Graceful Degradation

The ability to keep providing services even when hardware failure occurs is called Graceful Degradation.

Fail-Soft

System that are designed for graceful degradation are also called Fail-Soft.
There are two techniques that are mostly used in multiprocessing systems.

- **Symmetric Multiprocessing.**
- **Asymmetric Multiprocessing.**

In **Symmetric Multiprocessing**, each processor runs a copy of the Operating System and these copies communicate with each other when needed. Encore's version of Unix for Multimax Computers uses Symmetric Multiprocessing. This computer has many processors and each processor runs a copy of Unix. Benefit here is that all processes can run at once (N processes can run at the same time when there are N CPU's) without causing a deterioration in performance. In such systems I/O's need to be controlled carefully to ensure that data reaches the appropriate processor.

Inefficiencies may occur, as CPU's are separate, one may be sitting idle while another is overloaded. In order to avoid such kind of inefficiencies, Multiprocessing system should allow the jobs are resources etc to be shared among various processors. These kind of systems are very delicate and should be written carefully.

In **Asymmetric Multiprocessing**, each processor is assigned a specific task. Processor that controls the system is called the master processor and it gives instructions to the slave processors. So, in Asymmetric multiprocessing we have master-slave relationship. Master processor schedules and allocates work to the slave processors.

Asymmetric multiprocessing is normally used in very large systems where most time consuming activities are processing of I/O's.

Distributed Systems

Distributed Systems distribute jobs or computation among several physical processors. So, several processors interconnected with each other by a communication network forms a Distributed System.

The processors in a Distributed System may vary in Size and Function. They may include small Microprocessors, Workstations, Mini Computers and even Large general purpose computer systems. These processors can be referred by a number of different names, i.e. Sites, Nodes, Computers, Machines etc. Generally, one site, the Server, has a resource that another site, the client (user) would like to use. It is the purpose of the Distributed System to provide an efficient and convenient environment for this type of Sharing of Resources.

Two schemes are mostly used for building Distributed Systems.

i) Tightly Coupled System

In a Tightly Coupled System, the processors share Memory and a Clock. In such kind of Multiprocessor Systems, communication takes place through the shared Memory.

ii) Loosely Coupled System

In a Loosely Coupled System, the processors do not share Memory or Clock, and each processor has its own Local Memory, and the processors communicate with each other through various communication lines i.e. High Speed Buses or Telephone Lines.

Reasons for building Distributed Systems

A Distributed System is designed due to the following reasons

1) Resource Sharing

In a Distributed System, users can share or use the resources available at other sites or machines. A user working in a Distributed Environment can use the Printer (resource) connected with some other computer. Similarly, a user can Access files present on some other computer in a Distributed Environment. So, Processing information in a Distributed Database, Printing files at remote sites and using remote specialized hardware are all examples of resource sharing in a Distributed System.

2) Computation Speed Up

Computation speed can be increased in a Distributed Environment if a computation can be partitioned into a number of sub computations that can run concurrently.

Load Sharing If a particular site is overloaded with jobs, then some of them can be moved or transferred to other sites in the Distributed System that are not overloaded or are lightly loaded. This movement of jobs is called Load Sharing.

3) Reliability

High Reliability is achieved in Distributed Systems that if one site fails due to any reason, then the remaining sites can continue operating.

If a system is composed of a number of small machines and each of which is responsible for some important system function than a single failure (hardware or software) in that environment may cause the halt of the whole system.

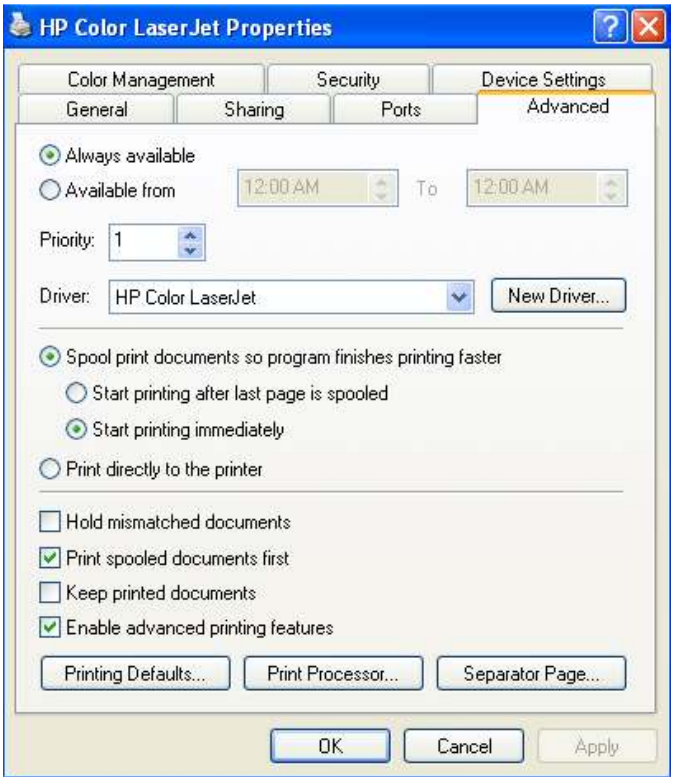
On the other hand, Distributed Systems are reliable in a way that the system can continue its operation even if some of the sites have failed due to any reason or have stopped working.

4) Communication

Distributed Systems provides a way or mechanism that programs can exchange data with one another. A user can transfer files to a system that is geographically located at a distant place. Similarly electronic mail provides a way of communication with other users working at remote sites.

SPOOLing

SPOOL stands for "Simultaneous Peripheral Operations On Line". A computer document or task list (or "job") is to read it in and store it, usually on a hard disk or larger storage medium so that it can be printed or otherwise processed at a more convenient time (for example, when a printer is finished printing its current document). One can envision SPOOLing as reeling a document or task list onto a spool of thread so that it can be unreeled at a more convenient time.



The idea of spooling originated in early computer days when input was read in on punched cards for immediate printing (or processing and then immediately printing of the results). Since the computer operates at a much faster rate than input/output devices such as printers, it was more effective to store the read-in lines on a magnetic disk until they could be conveniently printed when the printer was free and the computer was less busy working on other tasks. Actually, a printer has a buffer but frequently the buffer isn't large enough to hold the entire document, requiring multiple I/O operations with the printer.

The spooling of documents for printing and batch job requests still goes on in mainframe computers where many users share a pool of resources. On personal computers, your print jobs (for example, a Web page you want to print) are spooled to an output file on hard disk if your printer is already printing another file.

Buffering

A buffer is a data area shared by hardware devices or program processes that operate at different speeds or with different sets of priorities. The buffer allows each device or process to operate without being held up by the other. In order for a buffer to be effective, the size of the buffer and the algorithms for moving data into and out of the buffer need to be considered by the buffer designer. Like a cache, a buffer is a "midpoint holding place" but exists not so much to accelerate the speed of an activity as to support the coordination of separate activities.

This term is used both in programming and in hardware. In programming, buffering sometimes implies the need to screen data from its final intended place so that it can be edited or otherwise processed before being moved to a regular file or database.

Real Time System

Real time systems monitor/analyze and control real world events as they occur. A real time system is used as a control device. Elements of real time system include data gathering from an external environment, analysis as required by the application and the control/output that responds to the external environment. Normally a monitoring component coordinates all the elements so that a real time response can be maintained. (Typically real time response ranges from 1 msec to 1 minute).

Real-time systems are scientific experiments, medical-imaging systems and weapons systems etc. A real time system has fixed time constraints. Processing must be done within the defined period otherwise system will fail. In real-time systems, secondary storage is almost missing and the data stored is present in ROM (Read-only-memory).

Booting Mechanism

This procedure may differ slightly for Mac, UNIX, OS/2, or other operating systems. When you turn on your computer, chances are that the operating system has been set up to boot (load into RAM) automatically in this sequence:

1. As soon as the computer is turned on, the Basic Input-Output System (basic input/output system) on your system's read-only memory (read-only memory) chip is "woken up" and takes charge. BIOS is already loaded because it's built-in to the ROM chip and, unlike random access memory, ROM contents don't get erased when the computer is turned off.
2. BIOS first does a "power-on self test" (POST) to make sure all the computer's components are operational. Then the BIOS's boot program looks for the special boot programs that will actually load the operating system onto the hard disk.
3. First, it looks on drive A (unless you've set it up some other way or there is no diskette drive) at a specific place where operating system boot files are located. (If the operating system is MS-DOS, for example, it will find two files named IO.SYS and MSDOS.SYS.) If there is a diskette in drive A but it's not a system disk, BIOS will send you a message that drive A doesn't contain a system disk. If

there is no diskette in drive A (which is the most common case), BIOS looks for the system files at a specific place on your hard drive.

4. Having identified the drive where boot files are located, BIOS next looks at the first *sector* (a 512-byte area) and copies information from it into specific locations in RAM. This information is known as the *boot record* or Master Boot Record (MBR)
5. It then loads the boot record into a specific place (hexadecimal address 7C00) in RAM.
6. The boot record contains a program that BIOS now branches to, giving the boot record control of the computer.
7. The boot record loads the initial system file (for example, for DOS systems, IO.SYS) into RAM from the diskette or hard disk.
8. The initial file (for example, IO.SYS, which includes a program called SYSINIT) then loads the rest of the operating system into RAM. (At this point, the boot record is no longer needed and can be overlaid by other data.)
9. The initial file (for example, SYSINIT) loads a system file (for example, MSDOS.SYS) that knows how to work with the BIOS.
10. One of the first operating system files that is loaded is a system configuration file (for DOS, it's called CONFIG.SYS). Information in the configuration file tells the loading program which specific operating system files need to be loaded (for example, specific device driver.
11. Another special file that is loaded is one that tells which specific applications or commands the user wants to have included or performed as part of the boot process. In DOS, this file is named AUTOEXEC.BAT. In Windows, it's called WIN.INI.
12. After all operating system files have been loaded, the operating system is given control of the computer and performs requested initial commands and then waits for the first interactive user input.