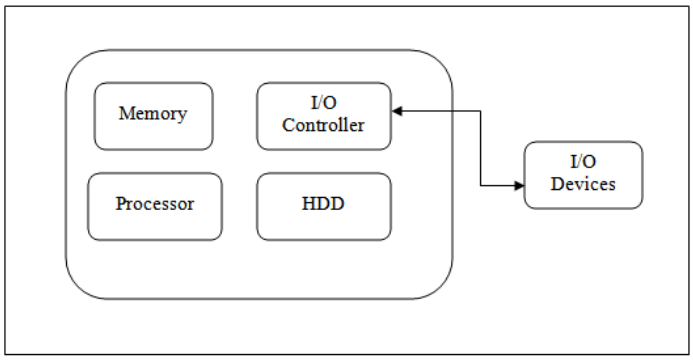
Operating System:

- An operating system is a program which manages all the computer hardwares.
- It provides the base for application program and acts as an intermediary between a user and the computer hardware.
- The operating system has two objectives such as:
  - Firstly, an operating system controls the computer's hardware.
  - The second objective is to provide an interactive interface to the user and interpret commands so that it can communicate with the hardware.
- The operating system is very important part of almost every computer system.

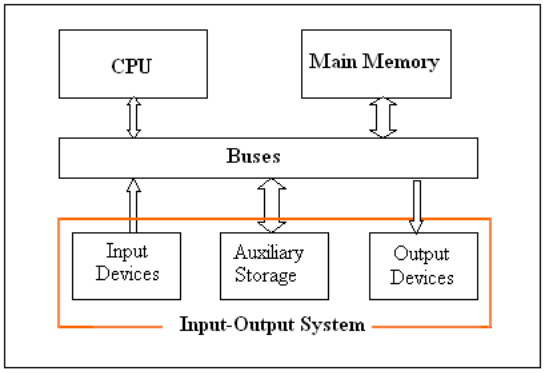
Managing Hardware



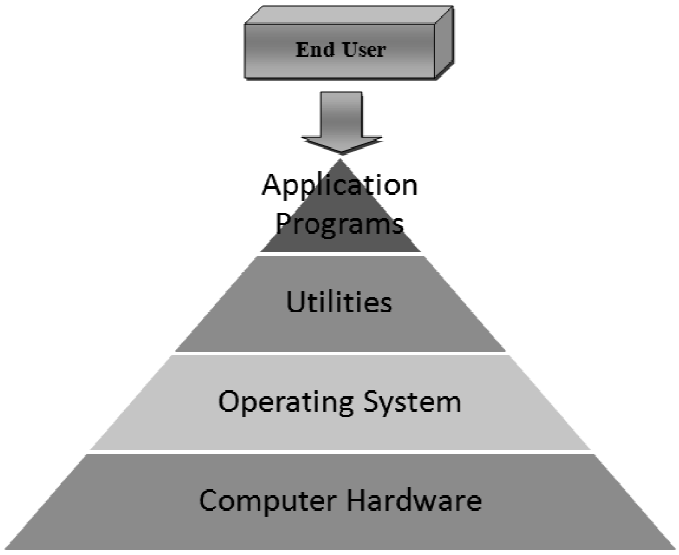
- The prime objective of operating system is to manage & control the various hardware resources of a computer system.
- These hardware resources include processer, memory, and disk space and so on.
- The output result was display in monitor. In addition to communicating with the hardware theoperating system provides on error handling procedure and display an error notification.
- If a device not functioning properly, the operating system cannot be communicate with the device.

Providing an Interface

- The operating system organizes application so that users can easily access, use and store them.



- It provides a stable and consistent way for applications to deal with the hardware without the user having known details of the hardware.
- If the program is not functioning properly, the operating system again takes control, stops the application and displays the appropriate error message.
- Computer system components are divided into 5 parts
  - Computer hardware
  - operating system
  - utilities
  - Application programs
  - End user



- The operating system controls and coordinate a user of hardware and various application programs for various users.
- It is a program that directly interacts with the hardware.
- The operating system is the first encoded with the Computer and it remains on the memory all time thereafter.

System goals

- The purpose of an operating system is to be provided an environment in which an user can execute programs.
- Its primary goals are to make the computer system convenience for the user.
- Its secondary goals are to use the computer hardware in efficient manner.

View of operating system

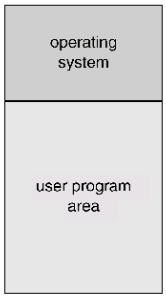
- **User view:**The user view of the computer varies by the interface being used. The examples are -windows XP, vista, windows 7 etc. Most computer user sit in the in front of personal computer (pc) in this case the operating system is designed mostly for easy use with some attention paid to resource utilization. Some user sit at a terminal connected to a mainframe/minicomputer. In this case other users are accessing the same computer through the other terminals. There user are share resources and may exchange the information. The operating system in this case is designed to maximize resources utilization to assume that all available CPU time, memory and I/O are used efficiently and no individual user takes more than his/her fair and share.The other users sit at workstations connected to network of other workstations and servers. These users have dedicated resources but they share resources such as networking and servers like file, compute and print server. Here the operating system is designed to compromise between individual usability and resource utilization.
- **System view:** From the computer point of view the operating system is the program which is most intermediate with the hardware. An operating system has resources as hardware and software which may be required to solve a problem like CPU time, memory space, file storage space and I/O devices and so on. That's why the operating system acts as manager of these resources. Another view of the operating system is it is a control program. A control program manages the execution of user programs to present the errors in proper use of the computer. It is especially concerned of the user the operation and controls the I/O devices.

Types of Operating System

1. **Mainframe System:** It is the system where the first computer used to handle many commercial scientific applications. The growth of mainframe systems traced from simple batch system where the computer runs one and only one application to time shared systems which allowed for user interaction with the computer system
    a. **Batch /Early System:** Early computers were physically large machine. The common input devices were card readers, tape drivers. The common output devices were line printers, tape drivers and card punches. In these systems the user did not interact directly with the computer system. Instead the user preparing a job which consists of programming data and some control information and then submitted it to the computer

operator after some time the output is appeared. The output in these early computer was fairly simple is main task was to transfer control automatically from one job to next. The operating system always resides in the memory. To speed up processing operators batched the jobs with similar needs and ran then together as a group. The disadvantages of batch system are that in this execution environment the CPU is often idle because the speed up of I/O devices is much slower than the CPU.

Memory Layout for a Simple Batch System



b. **Multiprogrammed System:** Multiprogramming concept increases CPU utilization by organization jobs so that the CPU always has one job to execute the idea behind multiprogramming concept. The operating system keeps several jobs in memory simultaneously as shown in below figure.

Operating System
Job 1
Job 2
Job 3
Job 4

This set of job is subset of the jobs kept in the job pool. The operating system picks and beginning to execute one of the jobs in the memory. In this environment the operating system simply switches and executes another job. When a job needs to wait the CPU is simply switched to another job and so on. The multiprogramming operating system is sophisticated because the operating system makes decisions for the user. This is known as scheduling. If several jobs are ready to run at the same time the system choose one among

them. This is known as CPU scheduling. The disadvantages of the multiprogrammed system are

- It does not provide user interaction with the computer system during the program execution.
- The introduction of disk technology solved these problems rather than reading the cards from card reader into disk. This form of processing is known as spooling.

SPOOL stands for simultaneous peripheral operations online. It uses the disk as a huge buffer for reading from input devices and for storing output data until the output devices accept them. It is also use for processing data at remote sides. The remote processing is done and its own speed with no CPU intervention. Spooling overlaps the input, output one job with computation of other jobs. Spooling has a beneficial effect on the performance of the systems by keeping both CPU and I/O devices working at much higher time.

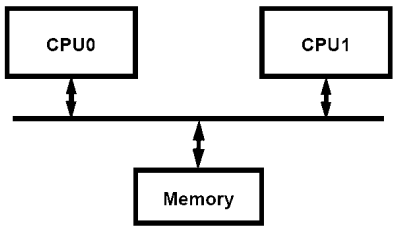
c. **Time Sharing System:**The time sharing system is also known as multi user systems. The CPU executes multiple jobs by switching among them but the switches occurs so frequently that the user can interact with each program while it is running. An interactive computer system provides direct communication between a user and system. The user gives instruction to the operating systems or to a program directly using keyboard or mouse and wait for immediate results. So the response time will be short. The time sharing system allows many users to share the computer simultaneously. Since each action in this system is short, only a little CPU time is needed for each user. The system switches rapidly from one user to the next so each user feels as if the entire computer system is dedicated to his use, even though it is being shared by many users. The disadvantages of time sharing system are:

- It is more complex than multiprogrammed operating system
- The system must have memory management & protection, since several jobs are kept in memory at the same time.
- Time sharing system must also provide a file system, so disk management is required.
- It provides mechanism for concurrent execution which requires complex CPU scheduling schemes.

2. **Personal Computer System/Desktop System:** Personal computer appeared in 1970's. They are microcomputers that are smaller & less expensive than mainframe systems. Instead of maximizing CPU & peripheral utilization, the systems opt for maximizing user convenience & responsiveness. At first file protection was not necessary on a personal machine. But when other computers 2nd other users can access the files on a pc file protection becomes necessary. The lack of protection made if easy for malicious programs to destroy data on such systems. These programs may be self replicating& they spread via worm or virus mechanisms. They can disrupt entire companies or even world wide networks. E.g : windows 98, windows 2000, Linux.
3. **Microprocessor Systems/ Parallel Systems/ Tightly coupled Systems:** These Systems have more than one processor in close communications which share the computer bus, clock, memory & peripheral devices. Ex: UNIX, LINUX. Multiprocessor Systems have 3 main advantages.
    a. **Increased throughput:** No. of processes computed per unit time. By increasing the no. of processors move work can be done in less time. The speed up ratio with N processors is not N, but it is less than N. Because a certain amount of overhead is incurred in keeping all the parts working correctly.
    b. **Increased Reliability:** If functions can be properly distributed among several processors, then the failure of one processor will not halt the system, but slow it down. This ability to continue to operate in spite of failure makes the system fault tolerant.
    c. **Economic scale:** Multiprocessor systems can save money as they can share peripherals, storage & power supplies.

    The various types of multiprocessing systems are:

    - **Symmetric Multiprocessing (SMP):** Each processor runs an identical copy of the operating system & these copies communicate with one another as required. Ex: Encore's version of UNIX for multi max computer. Virtually, all modern operating system including Windows NT, Solaris, Digital UNIX, OS/2 & LINUX now provide support for SMP.



- **Asymmetric Multiprocessing (Master – Slave Processors):** Each processor is designed for a specific task. A master processor controls the system & schedules & allocates the work to the slave processors. Ex- Sun's Operating system SUNOS version 4 provides asymmetric multiprocessing.

4. **Distributed System/Loosely Coupled Systems:** In contrast to tightly coupled systems, the processors do not share memory or a clock. Instead, each processor has its own local memory. The processors communicate with each other by various communication lines such as high speed buses or telephone lines. Distributed systems depend on networking for their functionalities. By being able to communicate distributed systems are able to share computational tasks and provide a rich set of features to the users. Networks vary by the protocols used, the distances between the nodes and transport media. TCP/IP is the most common network protocol. The processor is a distributed system varies in size and function. It may microprocessors, work stations, minicomputer, and large general purpose computers. Network types are based on the distance between the nodes such as LAN (within a room, floor or building) and WAN (between buildings, cities or countries). The advantages of distributed system are resource sharing, computation speed up, reliability, communication.
5. **Real time Systems:** Real time system is used when there are rigid time requirements on the operation of a processor or flow of data. Sensors bring data to the computers. The computer analyzes data and adjusts controls to modify the sensors inputs. System that controls scientific experiments, medical imaging systems and some display systems are real time systems. The disadvantages of real time system are:
   a. A real time system is considered to function correctly only if it returns the correct result within the time constraints.
   b. Secondary storage is limited or missing instead data is usually stored in short term memory or ROM.
   c. Advanced OS features are absent.

   Real time system is of two types such as:

- **Hard real time systems:** It guarantees that the critical task has been completed on time. The sudden task is takes place at a sudden instant of time.
- **Soft real time systems:** It is a less restrictive type of real time system where a critical task gets priority over other tasks and retains that priority until it computes. These have more limited utility than hard real time systems. Missing an occasional deadline is acceptable e.g. QNX, VX works. Digital audio or multimedia is included in this category.

It is a special purpose OS in which there are rigid time requirements on the operation of a processor. A real time OS has well defined fixed time constraints. Processing must be done within the time constraint or the system will fail. A real time system is said to function correctly only if it returns the correct result within the time constraint. These systems are characterized by having time as a key parameter.

Basic Functions of Operation System:

The various functions of operating system are as follows:

1. **Process Management:**

- A program does nothing unless their instructions are executed by a CPU.A process is a program in execution. A time shared user program such as a complier is a process. A word processing program being run by an individual user on a pc is a process.
- A system task such as sending output to a printer is also a process. A process needs certain resources including CPU time, memory files & I/O devices to accomplish its task.
- These resources are either given to the process when it is created or allocated to it while it is running. The OS is responsible for the following activities of process management.
- Creating & deleting both user & system processes.
- Suspending & resuming processes.
- Providing mechanism for process synchronization.
- Providing mechanism for process communication.
- Providing mechanism for deadlock handling.

2. **Main Memory Management:**

The main memory is central to the operation of a modern computer system. Main memory is a large array of words or bytes ranging in size from hundreds of thousand to billions. Main memory stores the quickly accessible data shared by the CPU & I/O device. The central processor reads instruction from main memory during instruction fetch cycle & it both reads

&writes data from main memory during the data fetch cycle. The main memory is generally the only large storage device that the CPU is able to address & access directly. For example, for the CPU to process data from disk. Those data must first be transferred to main memory by CPU generated E/O calls. Instruction must be in memory for the CPU to execute them. The OS is responsible for the following activities in connection with memory management.

- Keeping track of which parts of memory are currently being used & by whom.
- Deciding which processes are to be loaded into memory when memory space becomes available.
- Allocating &deallocating memory space as needed.

3. File Management:

File management is one of the most important components of an OS computer can store information on several different types of physical media magnetic tape, magnetic disk & optical disk are the most common media. Each medium is controlled by a device such as disk drive or tape drive those has unique characteristics. These characteristics include access speed, capacity, data transfer rate & access method (sequential or random).For convenient use of computer system the OS provides a uniform logical view of information storage. The OS abstracts from the physical properties of its storage devices to define a logical storage unit the file. A file is collection of related information defined by its creator. The OS is responsible for the following activities of file management.

- Creating & deleting files.
- Creating & deleting directories.
- Supporting primitives for manipulating files & directories.
- Mapping files into secondary storage.
- Backing up files on non-volatile media.

4. I/O System Management:

One of the purposes of an OS is to hide the peculiarities of specific hardware devices from the user. For example, in UNIX the peculiarities of I/O devices are hidden from the bulk of the OS itself by the I/O subsystem. The I/O subsystem consists of:

- A memory management component that includes buffering, catching & spooling.
- A general device- driver interfaces drivers for specific hardware devices. Only the device driver knows the peculiarities of the specific device to which it is assigned.

5. **Secondary Storage Management:**

   The main purpose of computer system is to execute programs. These programs with the data they access must be in main memory during execution. As the main memory is too small to accommodate all data & programs & because the data that it holds are lost when power is lost. The computer system must provide secondary storage to back-up main memory. Most modern computer systems are disks as the storage medium to store data & program. The operating system is responsible for the following activities of disk management.

   - Free space management.
   - Storage allocation.
   - Disk scheduling

   Because secondary storage is used frequently it must be used efficiently.

Networking:

A distributed system is a collection of processors that don't share memory peripheral devices or a clock. Each processor has its own local memory & clock and the processor communicate with one another through various communication lines such as high speed buses or networks. The processors in the system are connected through communication networks which are configured in a number of different ways. The communication network design must consider message routing & connection strategies are the problems of connection & security.

Protection or security:

If a computer system has multi users & allow the concurrent execution of multiple processes then the various processes must be protected from one another's activities. For that purpose, mechanisms ensure that files, memory segments, CPU & other resources can be operated on by only those processes that have gained proper authorization from the OS.

Command interpretation:

One of the most important functions of the OS is connected interpretation where it acts as the interface between the user & the OS.

System Calls:

System calls provide the interface between a process & the OS. These are usually available in the form of assembly language instruction. Some systems allow system calls to be made directly from a high level language program like C, BCPL and PERL etc. systems calls occur in different ways depending on the computer in use. System calls can be roughly grouped into 5 major categories.

1. **Process Control:**
    - **End, abort:** A running program needs to be able to has its execution either normally (end) or abnormally (abort).
    - **Load, execute:**A process or job executing one program may want to load and executes another program.
    - **Create Process, terminate process:** There is a system call specifying for the purpose of creating a new process or job (create process or submit job). We may want to terminate a job or process that we created (terminates process, if we find that it is incorrect or no longer needed).
    - **Get process attributes, set process attributes:** If we create a new job or process we should able to control its execution. This control requires the ability to determine & reset the attributes of a job or processes (get process attributes, set process attributes).
    - **Wait time:** After creating new jobs or processes, we may need to wait for them to finish their execution (wait time).
    - **Wait event, signal event:** We may wait for a specific event to occur (wait event). The jobs or processes then signal when that event has occurred (signal event).
2. **File Manipulation:**
    - **Create file, delete file:** We first need to be able to create & delete files. Both the system calls require the name of the file & some of its attributes.
    - **Open file, close file:** Once the file is created, we need to open it & use it. We close the file when we are no longer using it.
    - **Read, write, reposition file:** After opening, we may also read, write or reposition the file (rewind or skip to the end of the file).
    - **Get file attributes, set file attributes:** For either files or directories, we need to be able to determine the values of various attributes & reset them if necessary. Two system calls get file attribute & set file attributes are required for their purpose.
3. **Device Management:**
    - **Request device, release device:** If there are multiple users of the system, we first request the device. After we finished with the device, we must release it.
    - **Read, write, reposition:** Once the device has been requested & allocated to us, we can read, write & reposition the device.

4. **Information maintenance:**
   - **Get time or date, set time or date:**Most systems have a system call to return the current date & time or set the current date & time.
   - **Get system data, set system data:** Other system calls may return information about the system like number of current users, version number of OS, amount of free memory etc.
   - **Get process attributes, set process attributes:** The OS keeps information about all its processes & there are system calls to access this information.
5. **Communication:** There are two modes of communication such as:
   - **Message passing model:** Information is exchanged through an inter process communication facility provided by operating system. Each computer in a network has a name by which it is known. Similarly, each process has a process name which is translated to an equivalent identifier by which the OS can refer to it. The get hostid and get processed systems calls to do this translation. These identifiers are then passed to the general purpose open & close calls provided by the file system or to specific open connection system call. The recipient process must give its permission for communication to take place with an accept connection call. The source of the communication known as client & receiver known as server exchange messages by read message & write message system calls. The close connection call terminates the connection.
   - **Shared memory model:** processes use map memory system calls to access regions of memory owned by other processes. They exchange information by reading & writing data in the shared areas. The processes ensure that they are not writing to the same location simultaneously.

SYSTEM PROGRAMS:

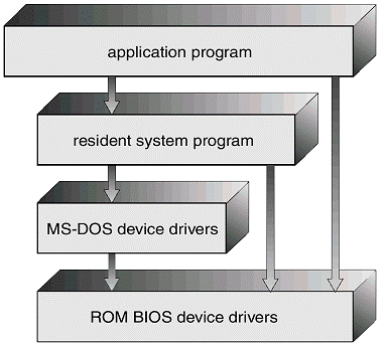
System programs provide a convenient environment for program development & execution. They are divided into the following categories.

- **File manipulation:** These programs create, delete, copy, rename, print & manipulate files and directories.
- **Status information:** Some programs ask the system for date, time & amount of available memory or disk space, no. of users or similar status information.
- **File modification:**Several text editors are available to create and modify the contents of file stored on disk.

- **Programming language support:** compliers, assemblers & interpreters are provided to the user with the OS.
- **Programming loading and execution:** Once a program is assembled or compiled, it must be loaded into memory to be executed.
- **Communications:** These programs provide the mechanism for creating virtual connections among processes users 2nd different computer systems.
- **Application programs:** Most OS are supplied with programs that are useful to solve common problems or perform common operations. Ex: web browsers, word processors & text formatters etc.

System structure:

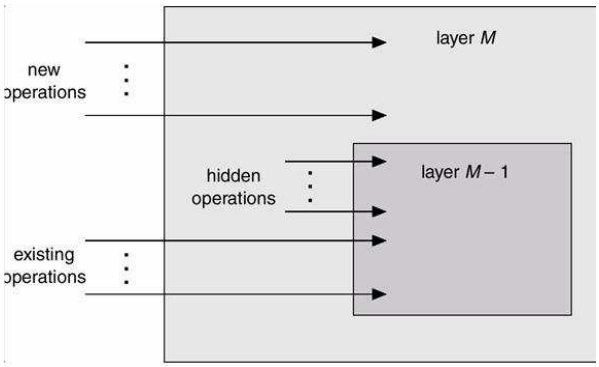
1. **Simple structure:** There are several commercial system that don't have a well- defined structure such operating systems begins as small, simple & limited systems and then grow beyond their original scope. MS-DOS is an example of such system. It was not divided into modules carefully. Another example of limited structuring is the UNIX operating system.



(MS DOS Structure)

2. **Layered approach:** In the layered approach, the OS is broken into a number of layers (levels) each built on top of lower layers. The bottom layer (layer o) is the hardware & top most layer (layer N) is the user interface. The main advantage of the layered approach is modularity.

- The layers are selected such that each users functions (or operations) & services of only lower layer.



- This approach simplifies debugging & system verification, i.e. the first layer can be debugged without concerning the rest of the system. Once the first layer is debugged, its correct functioning is assumed while the 2nd layer is debugged & so on.
- If an error is found during the debugging of a particular layer, the error must be on that layer because the layers below it are already debugged. Thus the design & implementation of the system are simplified when the system is broken down into layers.
- Each layer is implemented using only operations provided by lower layers. A layer doesn't need to know how these operations are implemented; it only needs to know what these operations do.
- The layer approach was first used in the operating system. It was defined in six layers.

Layers	Functions
5	User Program
4	I/O Management
3	Operator Process Communication
2	Memory Management
1	CPU Scheduling
0	Hardware

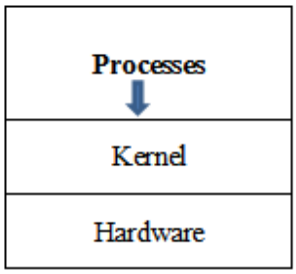
The main disadvantage of the layered approach is:

- The main difficulty with this approach involves the careful definition of the layers, because a layer can use only those layers below it. For example, the device driver for the disk space used by virtual memory algorithm must be at a level lower than that of the memory management routines, because memory management requires the ability to use the disk space.
- It is less efficient than a non layered system (Each layer adds overhead to the system call & the net result is a system call that take longer time than on a non layered system).

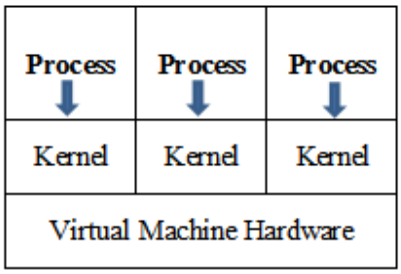
Virtual Machines:

By using CPU scheduling & virtual memory techniques an operating system can create the illusion of multiple processes, each executing on its own processors & own virtual memory. Each processor is provided a virtual copy of the underlying computer. The resources of the computer are shared to

create the virtual machines. CPU scheduling can be used to create the appearance that users have their own processor.



(Non virtual Machine)



(Virtual Machine)

Implementation: Although the virtual machine concept is useful, it is difficult to implement since much effort is required to provide an exact duplicate of the underlying machine. The CPU is being multiprogrammed among several virtual machines, which slows down the virtual machines in various ways.

Difficulty: A major difficulty with this approach is regarding the disk system. The solution is to provide virtual disks, which are identical in all respects except size. These are known as mini disks in IBM's VM OS. The sum of sizes of all mini disks should be less than the actual amount of physical disk space available.

I/O Structure

A general purpose computer system consists of a CPU and multiple device controller which is connected through a common bus. Each device controller is in charge of a specific type of device. A device controller maintains some buffer storage and a set of special purpose register. The device controller is responsible for moving the data between peripheral devices and buffer storage.

I/O Interrupt: To start an I/O operation the CPU loads the appropriate register within the device controller. In turn the device controller examines the content of the register to determine the actions which will be taken. For example, suppose the device controller finds the read request then, the controller will start the transfer of data from the device to the buffer. Once the transfer of data is complete the device controller informs the CPU that the operation has been finished. Once the I/O is started, two actions are possible such as

- In the simplest case the I/O is started then at I/O completion control is return to the user process. This is known as synchronous I/O.

- The other possibility is asynchronous I/O in which the control is return to the user program without waiting for the I/O completion. The I/O then continues with other operations.

When an interrupt occurs first determine which I/O device is responsible for interrupting. After searching the I/O device table the signal goes to the each I/O request. If there are additional request waiting in the queue for one device the operating system starts processing the next request. Finally control is return from the I/O interrupt.

DMA controller: DMA is used for high speed I/O devices. In DMA access the device controller transfers on entire block of data to of from its own buffer storage to memory. In this access the interrupt is generated per block rather than one interrupt per byte. The operating system finds a buffer from the pool of buffers for the transfer. Then a portion of the operating system called a device driver sets the DMA controller registers to use appropriate source and destination addresses and transfer length. The DMA controller is then instructed to start the I/O operation. While the DMA controller is performing the data transfer, the CPU is free to perform other tasks. Since the memory generally can transfer only one word at a time, the DMA controller steals memory cycles from the CPU. This cycle stealing can slow down the CPU execution while a DMA transfer is in progress. The DMA controller interrupts the CPU when the transfer has been completed.

Storage Structure

Thestorage structure of a computer system consists of two types of memory such as

- Main memory
- Secondary memory

Basically the programs & data are resided in main memory during the execution. The programs and data are not stored permanently due to following two reasons.

- Main memory is too small to store all needed programs and data permanently.
- Main memory is a volatile storage device which lost its contents when power is turned off.

Main Memory: The main memory and the registers are the only storage area that the CPU can access the data directly without any help of other device. The machine instruction which take memory address as arguments do not take disk address. Therefore in execution any instructions and any data must be resided in any one of direct access storage device. If the data are not in memory they must be moved before the CPU can operate on them. There are two types of main memory such as:

- **RAM (Random Access Memory):** The RAM is implemented in a semiconductor technology is called D-RAM (Dynamic RAM) which forms an array of memory words/cells. Each & every word should have its own address/locator. Instruction is performed through a sequence of load and store instruction to specific memory address. Each I/O controller includes register to hold commands of the data being transferred. To allow more convenient access to I/O device many computer architecture provide memory mapped I/O. In the case of memory mapped I/O ranges of memory address are mapped to the device register. Read and write to this memory addressbecause the data to be transferred to and from the device register.

Secondary Storage: The most common secondary storage devices are magnetic disk and magnetic tape which provide permanent storage of programs and data.

Magnetic Disk: It provides the bulk of secondary storage for modern computer systems. Each disk platter has flat circular shape like a CD. The diameter of a platter range starts from 1.8 to 5.25 inches. The two surfaces of a platter are covered with a magnetic material which records the information/data is given by the user. The read, write head are attached to a disk arm, which moves all the heads as a unit. The surface of a platter is logically divided into circular tracks which are sub divided into sectors. The set of tracks which are at one arm position forms a cylinder. There are may be thousands of cylinders in a disk drive & each track contains 100 of sectors. The storage capacity of a common disk drive is measured in GB. When the disk is in use a drive motor spins it at high speed. Most drives rotated 62 to 200 time/sec. The disk speed has two parts such as transfer rate & positioning time. The transfer rate is the rate at which data flow between the drive & the computer. The positioning time otherwise called as random access time. It consists of two parts such as seek time & rotational latency. The seek time is the time taken to move the disk arc to the desired cylinder. The rotational latency is the time taken to rotate the disk head.

Magnetic Tape: It was used as early secondary storage medium. It is also permanent and can hold large quantity of data. Its access time is slower, comparison to main memory devices. Magnetic tapes are sequential in nature. That's why random access to magnetic tape is thousand times slower than the random access to magnetic disk. The magnetic tapes are used mainly for backup the data. The magnetic tape must be kept in a non dusty environment and temperature controlled area. But the main advantage of the secondary storage device is that it can hold 2 to 3 times more data than a large disk drive. There are 4 types of magnetic tapes such as:

- ½ Inch

- ¼ Inch
- 4 mm
- 8 mm

Operating System Services

An operating system provides an environment for the execution of the program. It provides some services to the programs. The various services provided by an operating system are as follows:

- **Program Execution:** The system must be able to load a program into memory and to run that program. The program must be able to terminate this execution either normally or abnormally.
- **I/O Operation:** A running program may require I/O. This I/O may involve a file or a I/O device for specific device. Some special function can be desired. Therefore the operating system must provide a means to do I/O.
- **File System Manipulation:** The programs need to create and delete files by name and read and write files. Therefore the operating system must maintain each and every files correctly.
- **Communication:** The communication is implemented via shared memory or by the technique of message passing in which packets of information are moved between the processes by the operating system.
- **Error detection:** The operating system should take the appropriate actions for the occurrences of any type like arithmetic overflow, access to the illegal memory location and too large user CPU time.
- **Research Allocation:** When multiple users are logged on to the system the resources must be allocated to each of them. For current distribution of the resource among the various processes the operating system uses the CPU scheduling run times which determine which process will be allocated with the resource.
- **Accounting:** The operating system keep track of which users use how many and which kind of computer resources.
- **Protection:** The operating system is responsible for both hardware as well as software protection. The operating system protects the information stored in a multiuser computer system.

Process Management:

Process: A process or task is an instance of a program in execution. The execution of a process must programs in a sequential manner. At any time at most one instruction is executed. The process includes the current activity as represented by the value of the program counter and the content of the processors registers. Also it includes the process stack which contain temporary data (such as method parameters return address and local variables) & a data section which contain global variables.

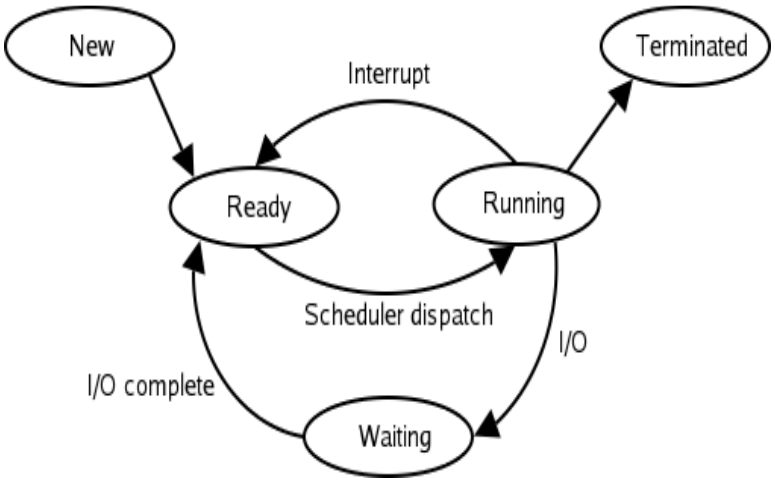
Difference between process & program:

A program by itself is not a process. A program in execution is known as a process. A program is a passive entity, such as the contents of a file stored on disk where as process is an active entity with a program counter specifying the next instruction to execute and a set of associated resources may be shared among several process with some scheduling algorithm being used to determinate when the stop work on one process and service a different one.

Process state: As a process executes, it changes state. The state of a process is defined by the correct activity of that process. Each process may be in one of the following states.

- **New:** The process is being created.
- **Ready:** The process is waiting to be assigned to a processor.
- **Running:** Instructions are being executed.
- **Waiting:** The process is waiting for some event to occur.
- **Terminated:** The process has finished execution.

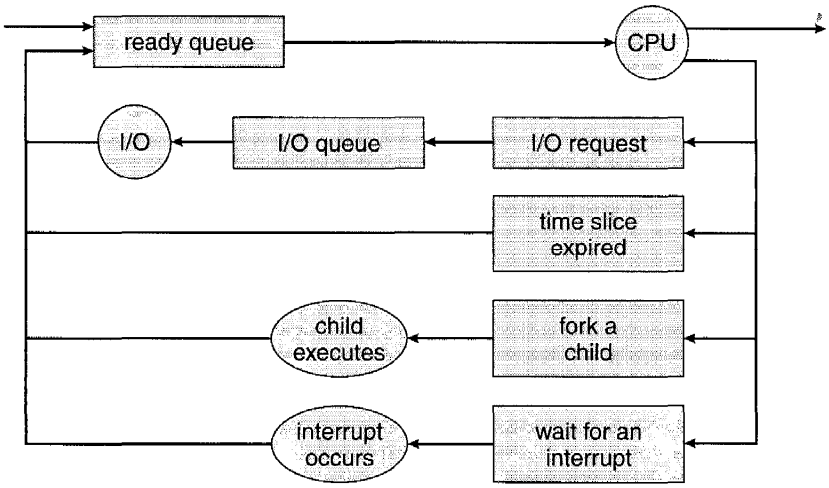
Many processes may be in ready and waiting state at the same time. But only one process can be running on any processor at any instant.



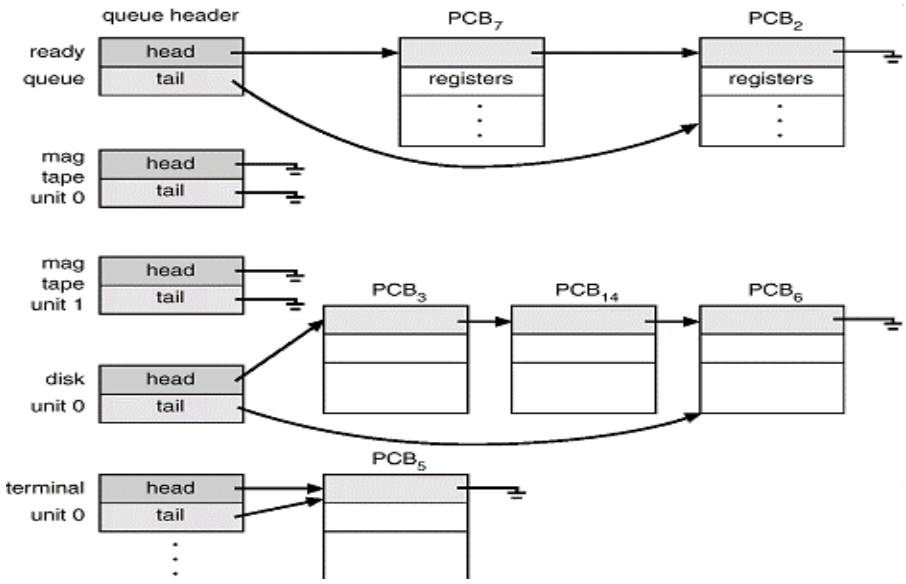
Process scheduling:

Scheduling is a fundamental function of OS. When a computer is multiprogrammed, it has multiple processes completing for the CPU at the same time. If only one CPU is available, then a choice has to be made regarding which process to execute next. This decision making process is known as scheduling and the part of the OS that makes this choice is called a scheduler. The algorithm it uses in making this choice is called scheduling algorithm.

Scheduling queues: As processes enter the system, they are put into a job queue. This queue consists of all process in the system. The process that are residing in main memory and are ready & waiting to execute or kept on a list called ready queue.



This queue is generally stored as a linked list. A ready queue header contains pointers to the first & final PCB in the list. The PCB includes a pointer field that points to the next PCB in the ready queue. The lists of processes waiting for a particular I/O device are kept on a list called device queue. Each device has its own device queue. A new process is initially put in the ready queue. It waits in the ready queue until it is selected for execution & is given the CPU.



SCHEDULERS:

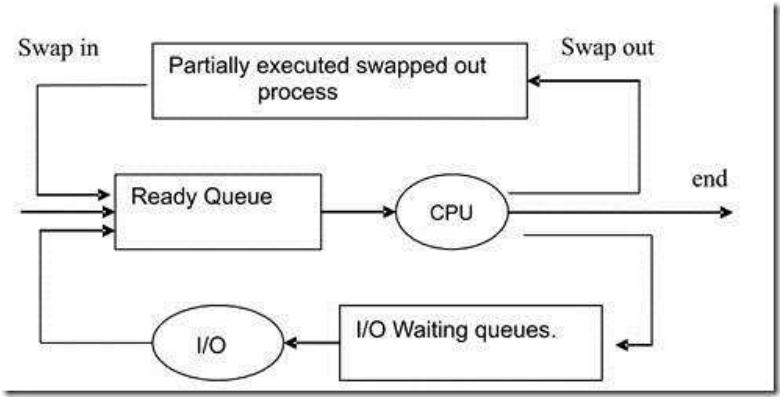
A process migrates between the various scheduling queues throughout its life-time purposes. The OS must select for scheduling processes from these queues in some fashion. This selection process is carried out by the appropriate scheduler. In a batch system, more processes are submittedand then executed immediately. So these processes are spooled to a mass storage device like disk, where they are kept for later execution.

Types of schedulers:

There are 3 types of schedulers mainly used:

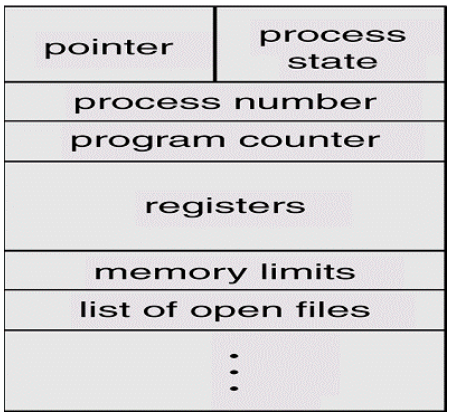
1. **Long term scheduler:** Long term scheduler selects process from the disk & loads them into memory for execution. It controls the degreeof multi-programming i.e. no. of processes in memory. It executes less frequently than other schedulers. If the degree of multiprogramming is stable than the average rate of process creation is equal to the average departure rate of processes leaving the system. So, the long term scheduler is needed to be invoked only when a process leaves the system. Due to longer intervals between executions it can afford to take more time to decide which process should be selected for execution. Most processes in the CPU are either I/O bound or CPU bound. An I/O bound process (an interactive 'C' program is one that spends most of its time in I/O operation than it spends in doing I/O operation. A CPU bound process is one that spends more of its time in doing computations than I/O operations (complex sorting program). It is important that the long term scheduler should select a good mix of I/O bound & CPU bound processes.

2. **Short - term scheduler:** The short term scheduler selects among the process that are ready to execute & allocates the CPU to one of them. The primary distinction between these two schedulers is the frequency of their execution. The short-term scheduler must select a new process for the CPU quite frequently. It must execute at least one in 100ms. Due to the short duration of time between executions, it must be very fast.
3. **Medium - term scheduler:** some operating systems introduce an additional intermediate level of scheduling known as medium - term scheduler. The main idea behind this scheduler is that sometimes it is advantageous to remove processes from memory & thus reduce the degree of multiprogramming. At some later time, the process can be reintroduced into memory & its execution can be continued from where it had left off. This is called as swapping. The process is swapped out & swapped in later by medium term scheduler. Swapping is necessary to improve theprocess miss or due to some change in memory requirements, the available memory limit is exceeded which requires some memory to be freed up.



Process control block:

Each process is represented in the OS by a process control block. It is also by a process control block. It is also known as task control block.



A process control block contains many pieces of information associated with a specific process. It includes the following informations.

- **Process state:** The state may be new, ready, running, waiting or terminated state.
- **Program counter:**it indicates the address of the next instruction to be executed for this purpose.
- **CPU registers:** The registers vary in number & type depending on the computer architecture. It includes accumulators, index registers, stack pointer & general purpose registers, plus any condition- code information must be saved when an interrupt occurs to allow the process to be continued correctly after- ward.
- **CPU scheduling information:**This information includes process priority pointers to scheduling queues & any other scheduling parameters.
- **Memory management information:** This information may include such information as the value of the bar & limit registers, the page tables or the segment tables, depending upon the memory system used by the operating system.
- **Accounting information:** This information includes the amount of CPU and real time used, time limits, account number, job or process numbers and so on.
- **I/O Status Information:** This information includes the list of I/O devices allocated to this process, a list of open files and so on. The PCB simply serves as the repository for any information that may vary from process to process.

~~CPU Scheduling Algorithm:~~

~~CPU Scheduling deals with the problem of deciding which of the processes in the ready queue is to be allocated first to the CPU. There are four types of CPU scheduling that exist.~~

A co-operation process is one that can affect or be affected by other processes executing in the system. Co-operating process may either directly share a logical address space or be allotted to the shared data only through files. This concurrent access is known as Process synchronization.

Critical Section Problem:

Consider a system consisting of n processes (P_0, P_1, ……P_{n-1}) each process has a segment of code which is known as critical section in which the process may be changing common variable, updating a table, writing a file and so on. The important feature of the system is that when the process is executing in its critical section no other process is to be allowed to execute in its critical section. The execution of critical sections by the processes is a mutually exclusive. The critical section problem is to design a protocol that the process can use to cooperate each process must request permission to enter its critical section. The section of code implementing this request is the entry section. The critical section is followed on exit section. The remaining code is the remainder section.

Example:

```
While (1)
{
Entry Section;
        Critical Section;
Exit Section;
        Remainder Section;
}
```

A solution to the critical section problem must satisfy the following three conditions.

1. **Mutual Exclusion:** If process P_i is executing in its critical section then no any other process can be executing in their critical section.
2. **Progress:** If no process is executing in its critical section and some process wish to enter their critical sections then only those process that are not executing in their remainder section can enter its critical section next.
3. **Bounded waiting:** There exists a bound on the number of times that other processes are allowed to enter their critical sections after a process has made a request.

Semaphores:

For the solution to the critical section problem one synchronization tool is used which is known as semaphores. A semaphore 'S' is an integer variable which is accessed through two standard

operations such as wait and signal. These operations were originally termed 'P' (for wait means to test) and 'V' (for single means to increment). The classical definition of wait is

```
Wait (S)
{
        While (S <= 0)
        {
                Test;
        }
        S--;
}
```

The classical definition of the signal is

```
Signal (S)
{
        S++;
}
```

In case of wait the test condition is executed with interruption and the decrement is executed without interruption.

Binary Semaphore:

A binary semaphore is a semaphore with an integer value which can range between 0 and 1.

Let 'S' be a counting semaphore. To implement the binary semaphore we need following the structure of data.

Binary Semaphores S_1, S_2;

int C;

Initially $S_1 = 1$, S2 = 0 and the value of C is set to the initial value of the counting semaphore 'S'.

Then the wait operation of the binary semaphore can be implemented as follows.

```
Wait (S1)
C--;
if (C < 0)
{
        Signal (S1);
        Wait (S2);
}
```

Signal (S_1);

The signal operation of the binary semaphore can be implemented as follows:

Wait (S_1);

C++;

if (C <=0)

Signal (S_2);

Else

Signal (S_1);

Classical Problem on Synchronization:

There are various types of problem which are proposed for synchronization scheme such as

- **Bounded Buffer Problem:** This problem was commonly used to illustrate the power of synchronization primitives. In this scheme we assumed that the pool consists of 'N' buffer and each capable of holding one item. The 'mutex' semaphore provides mutual exclusion for access to the buffer pool and is initialized to the value one. The empty and full semaphores count the number of empty and full buffer respectively. The semaphore empty is initialized to 'N' and the semaphore full is initialized to zero. This problem is known as procedure and consumer problem. The code of the producer is producing full buffer and the code of consumer is producing empty buffer. The structure of producer process is as follows:

  do {

  produce an item in nextp

  . . . . . .

  Wait (empty);

  Wait (mutex);

  . . . . . .

  add nextp to buffer

  . . . . . .

  Signal (mutex);

  Signal (full);

  } While (1);

  The structure of consumer process is as follows:

do {

Wait (full);

Wait (mutex);

. . . . . .

Remove an item from buffer to nextc

. . . . . .

Signal (mutex);

Signal (empty);

. . . . . .

Consume the item in nextc;

. . . .

} While (1);

- **Reader Writer Problem:** In this type of problem there are two types of process are used such as Reader process and Writer process. The reader process is responsible for only reading and the writer process is responsible for writing. This is an important problem of synchronization which has several variations like
    - The simplest one is referred as first reader writer problem which requires that no reader will be kept waiting unless a writer has obtained permission to use the shared object. In other words no reader should wait for other reader to finish because a writer is waiting.
    - The second reader writer problem requires that once a writer is ready then the writer performs its write operation as soon as possible.

  The structure of a reader process is as follows:

  Wait (mutex);

  Read count++;

  if (read count == 1)

  Wait (wrt);

  Signal (mutex);

  . . . . . .

  Reading is performed

  . . . . . .

  Wait (mutex);

Read count --;

if (read count == 0)

Signal (wrt);

Signal (mutex);

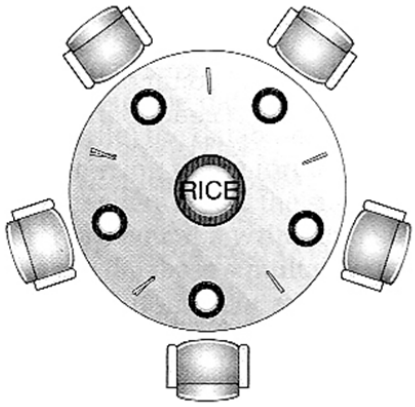
The structure of the writer process is as follows:

Wait (wrt);

Writing is performed;

Signal (wrt);

- **Dining Philosopher Problem:** Consider 5 philosophers to spend their lives in thinking & eating. A philosopher shares common circular table surrounded by 5 chairs each occupies by one philosopher. In the center of the table there is a bowl of rice and the table is laid with 6 chopsticks as shown in below figure.



When a philosopher thinks she does not interact with her colleagues. From time to time a philosopher gets hungry and tries to pickup two chopsticks that are closest to her. A philosopher may pickup one chopstick or two chopsticks at a time but she cannot pickup a chopstick that is already in hand of the neighbor. When a hungry philosopher has both her chopsticks at the same time, she eats without releasing her chopsticks. When she finished eating, she puts down both of her chopsticks and starts thinking again. This problem is considered as classic synchronization problem. According to this problem each chopstick is represented by a semaphore. A philosopher grabs the chopsticks by executing the wait operation on that semaphore. She releases the chopsticks by executing the signal operation on the appropriate semaphore. The structure of dining philosopher is as follows:

do{

```
Wait ( chopstick [i]);
Wait (chopstick [(i+1)%5]);
. . . . . .
Eat
. . . . . .
Signal (chopstick [i]);
Signal (chopstick [(i+1)%5]);
. . . . . .
Think
. . . . . .
} While (1);
```

Critical Region:

According to the critical section problem using semaphore all processes must share a semaphore variablemutex which is initialized to one. Each process must execute wait (mutex) before entering the critical section and execute the signal (mutex) after completing the execution but there are various difficulties may arise with this approach like:

Case 1: Suppose that a process interchanges the order in which the wait and signal operations on the semaphore mutex are executed, resulting in the following execution:

```
Signal (mutex);
. . . . . .
Critical Section
. . . . . .
Wait (mutex);
```

In this situation several processes may be executing in their critical sections simultaneously, which is violating mutual exclusion requirement.

Case 2: Suppose that a process replaces the signal (mutex) with wait (mutex). The execution is as follows:

```
Wait (mutex);
. . . . . .
Critical Section
. . . . . .
Wait (mutex);
```

In this situation a deadlock will occur

Case 3: Suppose that a process omits the wait (mutex) and the signal (mutex). In this case the mutual exclusion is violated or a deadlock will occur.

To illustrate the various types or error generated by using semaphore there are some high level language constructs have been introduced such as critical region and monitor.

Critical region is also known as conditional critical regions. It constructs guards against certain simple errors associated with semaphore. This high level language synchronization construct requires a variable V of type T which is to be shared among many processes. It is declared as

V: shared T;

The variable V can be accessed only inside a region statement as like below:

```
Wait (mutex);
While (! B) {
First_count++;
if (second_count> 0)
        Signal (second_delay);
Else
        Signal (mutex);
Wait (first_delay);
First_count--;
Second_count++;
if (first_count> 0)
        Signal (first_delay);
Else
        Signal (second_delay);
Wait (second_delay);
Second_count --;
}
S;
if (first_count> 0)
        Signal (first_delay);
Else if (second_count> 0)
        Signal (second_delay);
```

Else

Signal (mutex);

(Implementation of the conditional region constructs)

Where B is a Boolean variable which governs the access to the critical regions which is initialized to false.Mutex, First_delay and Second_delay are the semaphores which are initialized to 1, 0, and 0 respectively. First_count and Second_count are the integer variables which are initialized to zero.

Monitor:

It is characterized as a set of programmer defined operators. Its representation consists of declaring of variables, whose value defines the state of an instance. The syntax of monitor is as follows.

```
Monitor monitor_name
{
        Shared variable declarations
        Procedure body P1 (……) {
        . . . . . .
        }
        Procedure body P2 (……) {
        . . . . . .
        }
                .
                .
                .
        Procedure body Pn (……) {
        . . . . . .
        }
        {
                Initialization Code
        }
}
```

Atomic Transaction:

This section is related to the field of database system. Atomic transaction describes the various techniques of database and how they are can be used by the operating system. It ensures that the critical sections are executed automatically. To determine how the system should ensure atomicity

we need first to identify the properties of the devices used to for storing the data accessed by the transactions. The various types storing devices are as follows:

- **Volatile Storage:** Information residing in volatile storage does not survive in case of system crash. Example of volatile storage is main memory and cache memory.
- **Non volatile Storage:** Information residing in this type of storage usually survives in case of system crash. Examples are Magnetic Disk, Magnetic Tape and Hard Disk.
- **Stable Storage:** Information residing in stable storage is never lost. Example is non volatile cache memory.

The various techniques used for ensuring the atomicity are as follows:

1. **Log based Recovery:** This technique is used for achieving the atomicity by using data structure called log. A log has the following fields:
    a. **Transaction Name:** This is the unique name of the transaction that performed the write operation.
    b. **Data Item Name:** This is the unique name given to the data.
    c. **Old Value:** This is the value of the data before to the write operation.
    d. **New value:** This is the value of the data after the write operation.

    This recovery technique uses two processes such as Undo and Redo. Undo restores the value of old data updated by a transaction to the old values. Redo sets the value of the data updated by a transaction to the new values.
2. **Checkpoint:** In this principle system maintains the log. The checkpoint requires the following sequences of action.
    a. Output all the log records from volatile storage into stable storage.
    b. Output all modified data residing in volatile to the stable storage.
    c. Output a checkpoint onto the stable storage.

T_0	T_1
Read (A)	
Write (A)	
Read (B)	

Write (B)	
transaction	Read (A)
system	Write (A)
read and	Read (B)
their	Write (B)

3. **Serializibility:** In this technique the executed serially in some arbitrary order. Consider a consisting two data items A and B which are both written by two transactions T_0 and T_1. Suppose that transactions are executed automatically in the order T_0 followed by T_1. This execution sequence is known as schedule which is represented as below.

If transactions are overlapped then their execution resulting schedule is known as non-serial scheduling or concurrent schedule as like below:

T_0	T_1
Read (A)	
Write (A)	
	Read (A)
	Write (A)
Read (B)	
Write (B)	
	Read (B)
	Write (B)

4. **Locking:** This technique governs how the locks are acquired and released. There are two types of lock such as shared lock and exclusive lock. If a transaction T has obtained a shared lock (S) on data item Q then T can read this item but cannot write. If a transaction T has obtained an exclusive lock (S) on data item Q then T can both read and write in the data item Q.
5. **Timestamp:** In this technique each transaction in the system is associated with unique fixed timestamp denoted by TS. This timestamp is assigned by the system before the transaction starts. If a transaction T_i has been assigned with a timestamp TS (T_i) and later a new transaction T_j enters the system then TS (T_i) < TS (T_j). There are two types of timestamp such as W-timestamp and R-timestamp. W-timestamp denotes the largest timestamp of any transaction that performed write operation successfully. R-timestamp denotes the largest timestamp of any transaction that executed read operation successfully.

~~Deadlock:~~