



आईएफटीएम विश्वविद्यालय, मुरादाबाद, उत्तर प्रदेश  
**IFTM University, Moradabad, Uttar Pradesh**  
**NAAC ACCREDITED**

**E-Content**

**IFTM University, Moradabad**

# BCA-315

## UNIT-1

### OPERATING SYSTEM

The operating system is the most important program that runs on a computer. Every general purpose computer must have an operating system to run other programs and applications. Operating systems perform basic tasks, such as recognizing input from the keyboard, sending output to the display screen, keeping track of files and directories on the disk, and controlling peripheral devices such as disk drives and printers.

**Definition:-** An operating system is a system software which act as an interface between a user and computer hardware. It provides an environment in which a user may execute programs. A computer system can be divided in to four component.

Operating system as User Interface –

1. User
2. System and application programs
3. Operating system
4. Hardware

Every general-purpose computer consists of the hardware, operating system, system programs, and application programs. The hardware consists of memory, CPU, ALU, and I/O devices, peripheral device, and storage device. System program consists of compilers, loaders, editors, OS, etc. The application program consists of business programs, database programs.

{

“Hardware Provides basic computing resources (CPU, memory, I/O devices).

Operating System- Controls and coordinates the use of hardware among application programs.

Application Programs- Solve computing problems of users (compilers, database systems, video games, business programs such as banking software).

Users- People, machines, other computers”

}

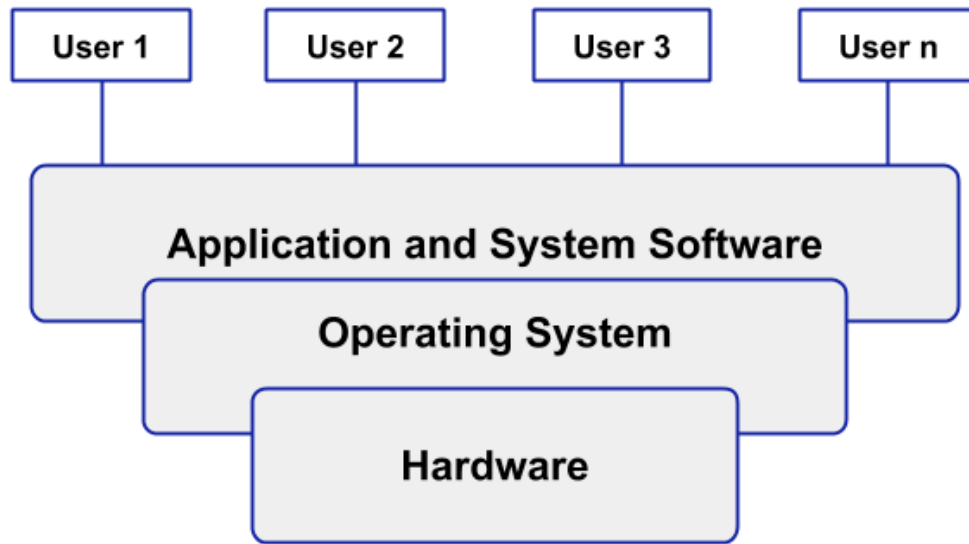


Fig1: Conceptual view of a computer system

Every computer must have an operating system to run other programs. The operating system coordinates the use of the hardware among the various system programs and application programs for various users. It simply provides an environment within which other programs can do useful work.

### **Goals of the Operating System:-**

There are two types of goals of an Operating System i.e. Primary Goals and Secondary Goal.

- **Primary Goal:** The primary goal of an Operating System is to provide a user-friendly and convenient environment. We know that it is not compulsory to use the Operating System, but things become harder when the user has to perform all the process scheduling and converting the user code into machine code is also very difficult. So, we make the use of an Operating System to act as an intermediate between us and the hardware. All you need to do is give commands to the Operating System and the Operating System will do the rest for you. So, the Operating System should be convenient to use.
- **Secondary Goal:** The secondary goal of an Operating System is efficiency. The Operating System should perform all the management of resources in such a way that the resources are fully utilized and no resource should be held idle if some request to that resource is there at that instant of time.

So, in order to achieve the above primary and secondary goals, the Operating System performs a number of functions.

### **Evolution of Operating System:-**

**Evolution** -Evolution mean the gradual development of something.

Evolution of operating system is divided into 5 phases

#### **Phase 0: (1940-1955):**

Phase 0: No operating system

- Computers are exotic experimental equipment.
- Program in machine language.
- Use plug boards to direct computer.
- No overlap between computation, I/O, think time, and response time.
- Programs manually loaded via card decks.

#### **Modifications:**

- More efficient use of hardware.
- Efficiency increases because it processes the jobs as a batch collectively rather than individually.

#### **Limitations:**

- No protection
- Difficult to debug!

#### **Phase 1 (1955-1970):**

- Make more efficient use of the computer
- computer: move the person away from the machine.
- User at console: one user at a time
- Batch monitor: load program, run, print
- OS becomes a batch monitor: a program that loads a user's
- If program failed, the OS record the contents of memory and saves it somewhere.
- Os/360 was introduced in 1963; worked in 1968. This Systems were enormously complicated. They were written in assembly code. No structured programming.

- OS were written in Assembly language.
- No structured programming.
- More efficient use of hardware.
- No protection
- difficult to debug!

#### **Phase 2 (1970-1980):**

- Interactive timesharing: CTSS:
- Developed at MIT. One of the first timesharing systems. to let multiple users interact with the system at the same time
- Sacrifice CPU time to get better response time
- Interactive timesharing.
- One of the first timesharing systems.
- To let multiple users interact with the system at the same time.
- Users do debugging, editing, and email online.
- More than one user executes their tasks simultaneously.

#### **Modifications:**

- Better utilization of resources.
- More than one user executes their tasks simultaneously.

#### **Limitations:**

- Thrashing- Thrashing caused by many Factors including
- Swapping
- Inefficient queuing
- Performance very non-linear response with load

#### **Phase 3 (1980-1990):**

- Created MS- DOS.
- GUI operating systems was developed first time.
- Microsoft Windows: Win 1.0 (1985) .

#### **Phase 4 (1990-2000):**

- Networked Systems: (LAN).
- Different machines share resources, printers, File Servers, Web Servers.
- Internet service providers (service between OS and apps).

**Modifications:**

- Internet service providers (service between OS and apps)
- Information becomes a commodity.
- Advertising becomes a computer marketplace.

**Limitations:**

- complicated as compare to uniprogramming been developed in different ways and for different uses. Computer OS products are older and more familiar to larger groups of users. Through the last 20 years, the simple idea of a computer operating system has been continually built on and improved. Through this time, Microsoft Windows, Android and Apple's Mac OS have emerged as the two dominant operating system designs.

**Phase 5 (2000-present):**

- Mobile and computer operating systems have been developed in different ways and for different uses.
- Computer OS products are older and more familiar to larger groups of users.
- Through this time, Microsoft Windows and Apple's Mac OS have emerged as the two dominant operating system designs.
- So many types of GUI operating systems are develop in phase 5 major types are: OS system of mobiles. window 95, window 98, window XP, window crystal vista window 8, window 10.

**Modifications:**

- OS becomes a subroutine library and command executive.
- finish quickly and run existing programs.

**Limitations:-**

- Eventually PCs become powerful: OS regains all the complexity of a “big” OS
- memory protection because of multiprogramming.

So many types of GUI operating systems are develop in phase 5 major types are:

- OS system of mobiles.
- window 95,

- window 98,
- window XP,
- window crystal vista window 8,
- window 10,
- Android all Versions.

### **Functions of an Operating System:-**

To achieve the goals of an Operating system, the Operating System performs a number of functionalities. They are:

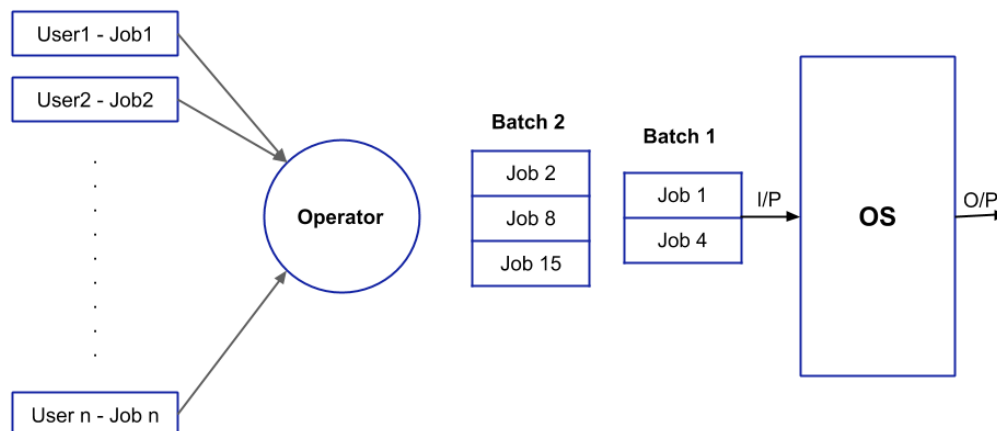
- **Process Management:** At a particular instant of time, the CPU may have a number of processes that are in the ready state. But at a time, only one process can be processed by a processor. So, the CPU should apply some kind of algorithm that can be used to provide uniform and efficient access to resources by the processes. The CPU should not give priority to only one process and it should make sure that every process which is in the ready state will be executed. Some of the CPU scheduling algorithms are First Come First Serve, Round Robin, Shortest Job First, Priority Scheduling, etc.
- **Memory Management:** For the execution of a process, the whole process is put into the main memory and the process is executed and after the execution of the process, the memory is freed and that memory can be used for other processes. So, it is the duty of the Operating System to manage the memory by allocating and deallocating the memory for the process.
- **I/O Device Management:** There are various I/O devices that are present in a system. Various processes require access to these resources and the process should not directly access these devices. So, it is the duty of the Operating System to allow the use of I/O devices by the various process that are requiring these resources.
- **File Management:** There are various files, folders and directory system in a particular computer. All these are maintained and managed by the Operating System of the computer. All these files related information are maintained by using a File Allocation Table or FAT. So, every detail related to the file i.e. filename, file size, file type, etc is stored in the File Allocation Table. Also, it is the duty of the Operating System to make sure that the files should not be opened by some unauthorized access.

- **Virtual Memory:** When the size of the program is larger than the main memory then it is the duty of the Operating System to load only frequently used pages in the main memory. This is called Virtual Memory.

## Types of an Operating System:-

### 1. Batch Operating System

In a Batch Operating System, the similar jobs are grouped together into batches with the help of some operator and these batches are executed one by one. For example, let us assume that we have 10 programs that need to be executed. Some programs are written in C++, some in C and rest in Java. Now, every time when we run these programmes individually then we will have to load the compiler of that particular language and then execute the code. But what if we make a batch of these 10 programmes. The benefit with this approach is that, for the C++ batch, you need to load the compiler only once. Similarly, for Java and C, the compiler needs to be loaded only once and the whole batch gets executed. The following image describes the working of a Batch Operating System.



### **Advantages:**

1. The overall time taken by the system to execute all the programmes will be reduced.
2. The Batch Operating System can be shared between multiple users.

### **Disadvantages:**

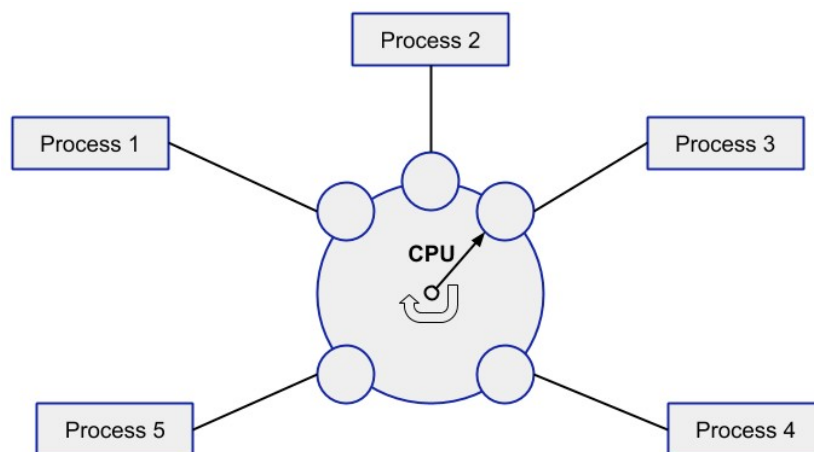


1. Manual interventions are required between two batches.
2. The CPU utilization is low because the time taken in loading and unloading of batches is very high as compared to execution time.

**Examples of the batch operating system: transactions, payroll system, bank statements, reporting, integration, etc.**

## 2. Time-Sharing or Multi-tasking Operating System

In a Multi-tasking Operating System, more than one processes are being executed at a particular time with the help of the time-sharing concept. So, in the time-sharing environment, we decide a time that is called time quantum and when the process starts its execution then the execution continues for only that amount of time and after that, other processes will be given chance for that amount of time only. In the next cycle, the first process will again come for its execution and it will be executed for that time quantum only and again next process will come. This process will continue. The following image describes the working of a Time-Sharing Operating System.



### Advantages:

1. Since equal time quantum is given to each process, so each process gets equal opportunity to execute.
2. The CPU will be busy in most of the cases and this is good to have case.

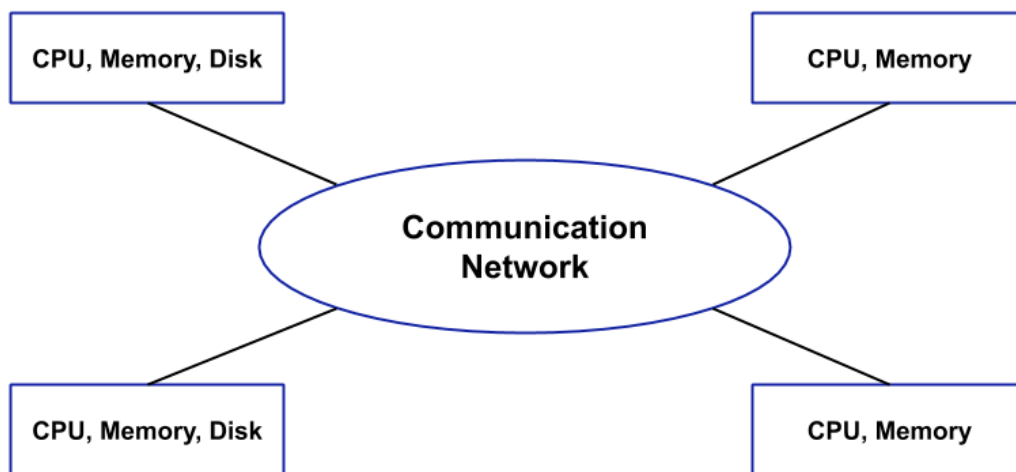
### Disadvantages:

1. Process having higher priority will not get the chance to be executed first because the equal opportunity is given to each process.

**Examples of multitasking: eating and watching TV simultaneously, chatting during classes, eating chocolates while walking, talking on a phone while walking, etc.**

### **3. Distributed Operating System**

In a Distributed Operating System, we have various systems and all these systems have their own CPU, main memory, secondary memory, and resources. These systems are connected to each other using a shared communication network. Here, each system can perform its task individually. The best part about these Distributed Operating System is remote access i.e. one user can access the data of the other system and can work accordingly. So, remote access is possible in these distributed Operating Systems. The following image shows the working of a Distributed Operating System.



#### **Advantages:**

1. Since the systems are connected with each other so, the failure of one system can't stop the execution of processes because other systems can do the execution.
2. Resources are shared between each other.
3. The load on the host computer gets distributed and this, in turn, increases the efficiency.

#### **Disadvantages:**

1. Since the data is shared among all the computers, so to make the data secure and accessible to few computers, you need to put some extra efforts.

2. If there is a problem in the communication network then the whole communication will be broken.

**Examples of distributed OS: intranets, the internet, sensors networks, etc.**

#### 4. Embedded Operating System

An Embedded Operating System is designed to perform a specific task for a particular device which is not a computer. For example, the software used in elevators is dedicated to the working of elevators only and nothing else. So, this can be an example of Embedded Operating System. The Embedded Operating System allows the access of device hardware to the software that is running on the top of the Operating System.

The block diagram of an embedded system consists of input devices, output devices, and memory.

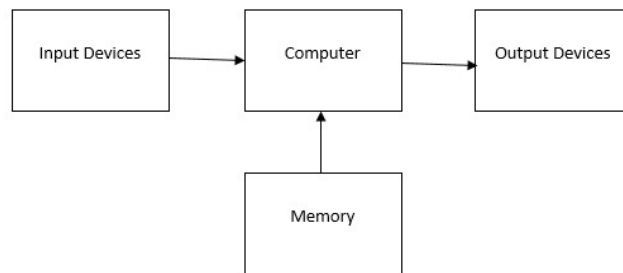


Fig The block diagram of an embedded

**Input Devices:** Input devices are used to send the data from the user to the system, here the user is the input. Some of the input devices are Keyboard, mouse, microphone, hard disk, sensors, switches, etc.

**Output Devices:** Out devices show the result to the humans in the form of text, image or sounds. Some of the output devices are printers, monitors, LCD, LED, motors, relays, buzzers, etc.

**Memory:** The memory is used to store the data. Some of the memory devices are SD card, EEPROM (Electrically Erasable Programmable Read-Only Memory), Flash memory. The memory devices used in the embedded system are Non-volatile RAM, volatile RAM, Dynamic Random Access Memory), etc.

**Advantages:**

1. Since it is dedicated to a particular job, so it is fast.
2. Low cost.
3. These consume less memory and other resources.

**Disadvantages:**

1. Only one job can be performed.
2. It is difficult to upgrade or is nearly scalable.

**Some applications of the embedded operating system are shown in the below**

- **Mobiles, Washing machines, Televisions, Microwave Ovens, Televisions, Computers, Laptops, Dishwashers, ATM's, Satellites, Vehicles**

**5. Real-time Operating System**

The Real-time Operating Systems are used in the situation where we are dealing with some real-time data. So, as soon as the data comes, the execution of the process should be done and there should be no delay i.e. no buffer delays should be there. Real-time OS is a time-sharing system that is based on the concept of clock interrupt. So, whenever you want to process a large number of request in a very short period of time, then you should use Real-time Operating System. For example, the details of the temperature of the petroleum industry are very crucial and this should be done in real-time and in a very short period of time. A small delay can result in a life-death situation. So, this is done with the help of Real-time Operating System. There are two types of Real-time Operating System:

1. **Hard Real-time:** In this type, a small delay can lead to drastic change. So, when the time constraint is very important then we use the Hard Real-time.
2. **Soft Real-time:** Here, the time constraint is not that important but here also we are dealing with some real-time data.

**Advantages:**

1. There is maximum utilization of devices and resources.
2. These systems are almost error-free.

**Disadvantages:**

1. The algorithms used in Real-time Operating System is very complex.
2. Specific device drivers are used for responding to the interrupts as soon as possible.

**Example of real time OS:-**

- **Radar systems, network switching control systems, satellite monitoring systems, satellite launch-control and maneuvering mechanisms, global positioning systems all have their roots in RTOS.**
- **Now a days RTOS are increasingly finding use in strategic and military operations. These are used in guided missile launching units, track-and-trace spy satellites, etc.**

## **6.Network operating System**

A Network Operating System runs on a server and provides the server the capability to manage data, users, groups, security, applications, and other networking functions. The primary purpose of the network operating system is to allow shared file and printer access among multiple computers in a network, typically a local area network (LAN), a private network or to other networks.

Examples of network operating systems include Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD.

The advantages of network operating systems are as follows –

- Centralized servers are highly stable.
- Security is server managed.
- Upgrades to new technologies and hardware can be easily integrated into the system.
- Remote access to servers is possible from different locations and types of systems.

The disadvantages of network operating systems are as follows –

- High cost of buying and running a server.
- Dependency on a central location for most operations.
- Regular maintenance and updates are required.

**Examples of network OS: Windows 2000, Linux, Microsoft windows, etc.**

## **Operating System Services:-**

An Operating System provides services to both the users and to the programs.

- It provides programs an environment to execute.
- It provides users the services to execute the programs in a convenient manner.

Following are a few common services provided by an operating system –

- Program execution
- I/O operations
- File System manipulation
- Communication
- Error Detection
- Resource Allocation
- Protection

### **Program execution:-**

Operating systems handle many kinds of activities from user programs to system programs like printer spooler, name servers, file server, etc. Each of these activities is encapsulated as a process.

A process includes the complete execution context (code to execute, data to manipulate, registers, OS resources in use). Following are the major activities of an operating system with respect to program management –

- Loads a program into memory.
- Executes the program.
- Handles program's execution.
- Provides a mechanism for process synchronization.
- Provides a mechanism for process communication.
- Provides a mechanism for deadlock handling.

### **I/O Operation:-**

An I/O subsystem comprises of I/O devices and their corresponding driver software. Drivers hide the peculiarities of specific hardware devices from the users.

An Operating System manages the communication between user and device drivers.

- I/O operation means read or write operation with any file or any specific I/O device.
- Operating system provides the access to the required I/O device when required.

## **File system manipulation:-**

A file represents a collection of related information. Computers can store files on the disk (secondary storage), for long-term storage purpose. Examples of storage media include magnetic tape, magnetic disk and optical disk drives like CD, DVD. Each of these media has its own properties like speed, capacity, data transfer rate and data access methods.

A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions. Following are the major activities of an operating system with respect to file management –

- Program needs to read a file or write a file.
- The operating system gives the permission to the program for operation on file.
- Permission varies from read-only, read-write, denied and so on.
- Operating System provides an interface to the user to create/delete files.
- Operating System provides an interface to the user to create/delete directories.
- Operating System provides an interface to create the backup of file system.

## **Communication:-**

In case of distributed systems which are a collection of processors that do not share memory, peripheral devices, or a clock, the operating system manages communications between all the processes. Multiple processes communicate with one another through communication lines in the network.

The OS handles routing and connection strategies, and the problems of contention and security. Following are the major activities of an operating system with respect to communication –

- Two processes often require data to be transferred between them
- Both the processes can be on one computer or on different computers, but are connected through a computer network.
- Communication may be implemented by two methods, either by Shared Memory or by Message Passing.

## **Error handling:-**

Errors can occur anytime and anywhere. An error may occur in CPU, in I/O devices or in the memory hardware. Following are the major activities of an operating system with respect to error handling –

- The OS constantly checks for possible errors.

- The OS takes an appropriate action to ensure correct and consistent computing.

### **Resource Management:-**

In case of multi-user or multi-tasking environment, resources such as main memory, CPU cycles and files storage are to be allocated to each user or job. Following are the major activities of an operating system with respect to resource management –

- The OS manages all kinds of resources using schedulers.
- CPU scheduling algorithms are used for better utilization of CPU.

### **Protection:-**

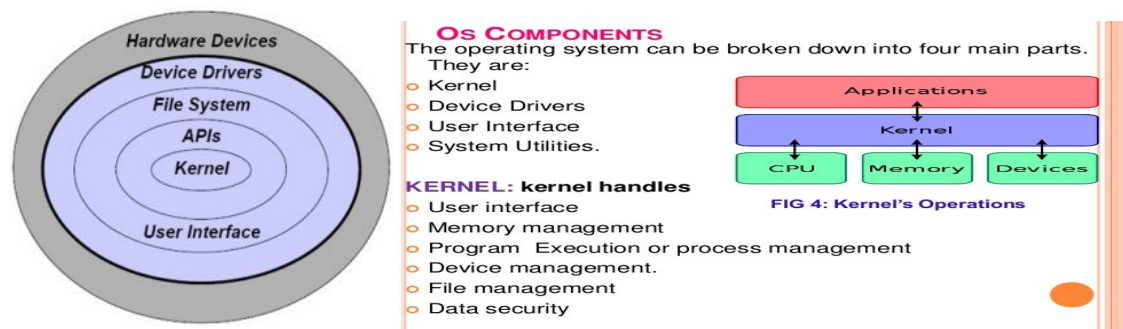
Considering a computer system having multiple users and concurrent execution of multiple processes, the various processes must be protected from each other's activities.

Protection refers to a mechanism or a way to control the access of programs, processes, or users to the resources defined by a computer system. Following are the major activities of an operating system with respect to protection –

- The OS ensures that all access to system resources is controlled.
- The OS ensures that external I/O devices are protected from invalid access attempts.
- The OS provides authentication features for each user by means of passwords.

### **Components of Operating System:-**

The components of an operating system play a key role to make a variety of computer system parts work together. The operating components are discussed below.



### **Kernel**



The kernel in the OS provides the basic level of control on all the computer peripherals. In the operating system, the kernel is an essential component that loads firstly and remains within the main memory. So that memory accessibility can be managed for the programs within the RAM, it creates the programs to get access from the hardware resources. It resets the operating states of the CPU for the best operation at all times.

### **Process Execution**

The OS gives an interface between the hardware as well as an application program so that the program can connect through the hardware device by simply following procedures & principles configured into the OS. The program execution mainly includes a process created through an OS kernel that uses memory space as well as different types of other resources.

### **Interrupt**

In the operating system, interrupts are essential because they give a reliable technique for the OS to communicate & react to their surroundings. An interrupt is nothing but one kind of signal between a device as well as a computer system otherwise from a program in the computer that requires the OS to leave and decide accurately what to do subsequently. Whenever an interrupt signal is received, then the hardware of the computer puts on hold automatically whatever computer program is running presently, keeps its status & runs a computer program which is connected previously with the interrupt.

### **Memory Management**

The functionality of an OS is nothing but memory management which manages main memory & moves processes backward and forward between disk & main memory during implementation. This tracks each & every memory position; until it is assigned to some process otherwise it is open. It verifies how much memory can be allocated to processes and also makes a decision to know which process will obtain memory at what time. Whenever memory is unallocated, then it tracks correspondingly to update the status. Memory management work can be divided into three important groups like memory management of hardware, OS and application memory management.

### **Multitasking**

It describes the working of several independent computer programs on a similar computer system. Multitasking in an OS allows an operator to execute one or more computer tasks at a

time. Since many computers can perform one or two tasks at a time, usually this can be done with the help of time-sharing, where each program uses the time of a computer to execute.

### **Networking**

Networking can be defined as when the processor interacts with each other through communication lines. The design of communication-network must consider routing, connection methods, safety, the problems of opinion & security.

Presently most of the operating systems maintain different networking techniques, hardware, & applications. This involves that computers that run on different operating systems could be included in a general network to share resources like data, computing, scanners, printers, which uses the connections of either wired otherwise wireless.

### **Security**

If a computer has numerous individuals to allow the immediate process of various processes, then the many processes have to be protected from other activities. This system security mainly depends upon a variety of technologies that work effectively. Current operating systems give an entry to a number of resources, which are obtainable to work the software on the system, and to external devices like networks by means of the kernel. The operating system should be capable of distinguishing between demands which have to be allowed for progressing & others that don't need to be processed. Additionally, to permit or prohibit a security version, a computer system with a high level of protection also provides auditing options. So this will allow monitoring the requests from accessibility to resources

### **User Interface**

A GUI or user interface (UI) is the part of an OS that permits an operator to get the information. A user interface based on text displays the text as well as its commands which are typed over a command line with the help of a keyboard.

The OS-based applications mainly provide a specific user interface for efficient communication. The main function of a user interface of an application is to get the inputs from the operator & to provide o/ps to the operator. But, the sorts of inputs received from the user interface as well as the o/p types offered by the user interface may change from application to application. The UI of any application can be classified into two types namely GUI (graphical UI) & CLI (command line user interface).

Thus, this is all about an overview of an operating system. The main components of an OS mainly include kernel, API or application program interface, user interface & file system, hardware devices and device drivers.

.

# OPERATING SYSTEM

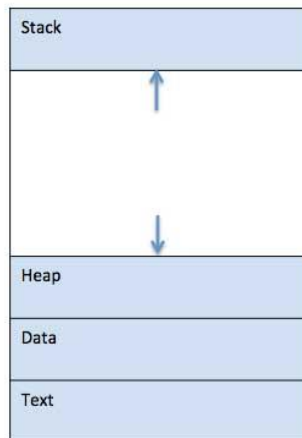
## UNIT-2

### CPU SCHEDULING

#### Process:-

A program in execution is called process. The processor (A processor, is a small chip that resides in computers and other electronic devices. Its basic job is to receive input and provide the appropriate output.) has to manage several activities at one time. Each activity is correspond to one process. Process execution must progress in sequential fashion. The OS must maintain a data structure for each process, which describes the state and resource ownership of that process, and which enables the OS to exert control over each process.

When a program is loaded into the memory and it becomes a process, it can be divided into four sections — stack, heap, text and data. The following image shows a simplified layout of a process inside main memory –



1	<b>Stack</b> The process Stack contains the temporary data such as method/function parameters, return address and local variables.
2	<b>Heap</b> This is dynamically allocated memory to a process during its run time.
3	<b>Text</b> This includes the current activity represented by the value of Program Counter and the contents of the processor's registers.
4	<b>Data</b> This section contains the global and static variables.

## Difference between Process and a Program:-

**Main()**

```
{  
  int i, prod=1;  
  for(i=0;i<100;i++)  
    prod=prod*i;  
}
```

- This is a program that contain one multiplication statement(prod=prod\*i) but the process will execute 100 multiplication.
- Process is active entity.
- Program is passive entity.

SR.NO.	PROGRAM	PROCESS
1.	Program contains a set of instructions designed to complete a specific task.	Process is an instance of an executing program.
2.	Program is a passive entity as it resides in the secondary memory.	Process is a active entity as it is created during execution and loaded into the main memory.
3.	Program exists at a single place and continues to exist until it is deleted.	Process exists for a limited span of time as it gets terminated after the completion of task.
4.	Program is a static entity.	Process is a dynamic entity.
5.	Program does not have any resource requirement, it only requires memory space for storing the instructions.	Process has a high resource requirement, it needs resources like CPU, memory address, I/O during its lifetime.
6.	Program does not have any control block.	Process has its own control block called Process Control Block.

## Process States or Process Life Cycle:-

When a process comes in execution it change its states. The state of a process is defined in part by the current activity of that process. Each process may be one of the following state during the time of execution.

**New:** In this state the process is being created.

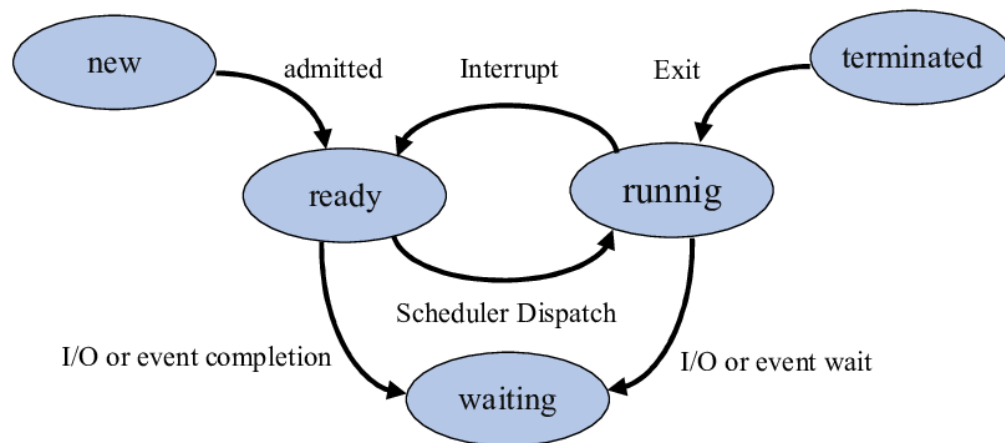
**Ready:** In this state the process is to be assign a duty by the processor.

**Waiting:** In this state the process is waiting or blocked for some event to occur such as input/output completion.

**Running:** In this state Instructions are executed.

**Terminated:** In this state the process has finished the execution

## State Transition Diagram:-



## Process Control Block (PCB):-

In OS each process is represented by a process control block or task control block. It is a data structure that physically represent a process in the memory of a computer system. It contains pieces of information associate with a specific process that include following:-

## PCB

pointer	process state
process number	
program counter	
registers	
memory limits	
list of open files	
⋮	

**Pointer:-** A pointer to parent process.

**Process State:-** It indicates the information about the state of process such as blocked ready running etc.

**Process ID:-** Each process is assigned a unique identification number, when it is entered into the system.

**Program Counter:-** It indicates the address of the next instruction to be executed.

**CPU Registers:-** It indicates the information about the contents of the CPU registers. The information of CPU registers must be saved when an interrupt occurs, so that the process can be continued correctly afterward. Registers hold the processed the result of calculations or addresses pointing to the memory locations of desired data.

**CPU Scheduling Information:-** It indicates the information needed for CPU scheduling such as process priority, pointers to scheduling queues and other scheduling parameters.

**Memory Management Information:-** It indicates the information needed for memory management such as value of the base and limit registers, page tables or segment tables, amount

of memory units allocated to the process etc.

**Accounting Information:-** It indicates the information about process number, CPU used by the process time limits etc.

**I/O Status Information:-** It indicates the information about I/O devices allocated to the process a list of open files access rights of files opened and so on,

**Link to Parent Process:-** A new process can be created from existing process; the existing process is called the parent process of the newly created process. The address of the PCB of parent process is stored.

**Link to Child Process:-** The addresses of the PCBs of the child processes in the main memory are stored.

## **CPU Scheduling:-**

CPU scheduling is a process which allows one process to use the CPU while the execution of another process is on hold(in waiting state) due to unavailability of any resource like I/O etc, thereby making full use of CPU. The aim of CPU scheduling is to make the system efficient, fast and fair.

Whenever the CPU becomes idle, the operating system must select one of the processes in the **ready queue** to be executed. The selection process is carried out by the short-term scheduler (or CPU scheduler). The scheduler selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them.

## **Schedulers:-**

A scheduler is a type of system software that allows you to handle process scheduling.

There are mainly three types of Process Schedulers:

1. Long Term
2. Short Term
3. Medium Term

## **Long Term Scheduler or Job Scheduler :-**

Long term scheduler is also known as a **job scheduler**. This scheduler regulates the program and select process from the queue and loads them into memory for execution. It also regulates the degree of multi-programming.

However, the main goal of this type of scheduler is to offer a balanced mix of jobs, like Processor, I/O jobs., that allows managing multiprogramming.



## Medium Term Scheduler:-

Medium-term scheduling is an important part of **swapping**. It enables you to handle the swapped out-processes. In this scheduler, a running process can become suspended, which makes an I/O request.

A running process can become suspended if it makes an I/O request. A suspended processes can't make any progress towards completion. In order to remove the process from memory and make space for other processes, the suspended process should be moved to secondary storage.

## Short Term Scheduler:-

Short term scheduling is also known as **CPU scheduler**. The main goal of this scheduler is to boost the system performance according to set criteria. This helps you to select from a group of processes that are ready to execute and allocates CPU to one of them. The dispatcher gives control of the CPU to the process selected by the short term scheduler.

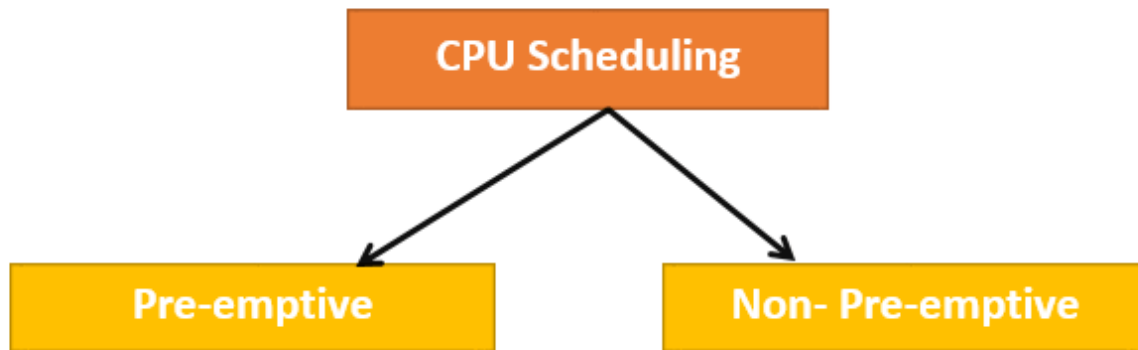
## Difference between Schedulers:-

Long-Term Vs. Short Term Vs. Medium-Term

Long-Term	Short-Term	Medium-Term
Long term is also known as a job scheduler	Short term is also known as CPU scheduler	Medium-term is also called swapping scheduler.
It is either absent or minimal in a time-sharing system.	It is insignificant in the time-sharing order.	This scheduler is an element of Time-sharing systems.
Speed is less compared to the short term scheduler.	Speed is the fastest compared to the short-term and medium-term scheduler.	It offers medium speed.
Allow you to select processes from the loads and pool back into the memory	It only selects processes that is in a ready state of the execution.	It helps you to send process back to memory.
Offers full control	Offers less control	Reduce the level of multiprogramming.

## Types of CPU Scheduling

Here are two kinds of Scheduling methods:



### Preemptive Scheduling

In Preemptive Scheduling, the tasks are mostly assigned with their priorities. Sometimes it is important to run a task with a higher priority before another lower priority task, even if the lower priority task is still running. The lower priority task holds for some time and resumes when the higher priority task finishes its execution.

### Non-Preemptive Scheduling

In this type of scheduling method, the CPU has been allocated to a specific process. The process that keeps the CPU busy will release the CPU either by switching context or terminating. It is the only method that can be used for various hardware platforms. That's because it doesn't need special hardware (for example, a timer) like preemptive scheduling.

### When scheduling is Preemptive or Non-Preemptive?

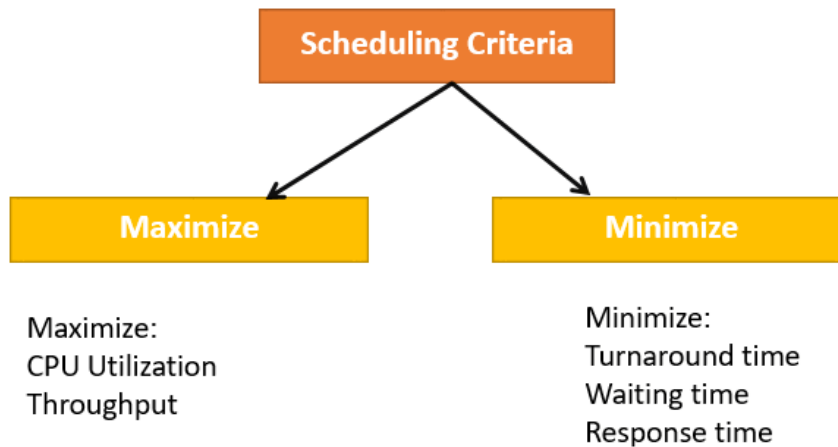
To determine if scheduling is preemptive or non-preemptive, consider these four parameters:

1. A process switches from the running to the waiting state.
2. Specific process switches from the running state to the ready state.
3. Specific process switches from the waiting state to the ready state.
4. Process finished its execution and terminated.

**Only conditions 1 and 4 apply, the scheduling is called non- preemptive.**  
**All other scheduling are preemptive.**

## CPU Scheduling Criteria

A CPU scheduling algorithm tries to maximize and minimize the following:



### Maximize:

**CPU utilization:** CPU utilization is the main task in which the operating system needs to make sure that CPU remains as busy as possible. It can range from 0 to 100 percent. However, for the RTOS, it can be range from 40 percent for low-level and 90 percent for the high-level system.

**Throughput:** The number of processes that finish their execution per unit time is known Throughput. So, when the CPU is busy executing the process, at that time, work is being done, and the work completed per unit time is called Throughput.

### Minimize:

**Waiting time:** Waiting time is an amount that specific process needs to wait in the ready queue.

**Response time:** It is an amount to time in which the request was submitted until the first response is produced.

**Turnaround Time:** Turnaround time is an amount of time to execute a specific process. It is the calculation of the total time spent waiting to get into the memory, waiting in the queue and, executing on the CPU. The period between the time of process submission to the completion time is the turnaround time.

## Interval Timer

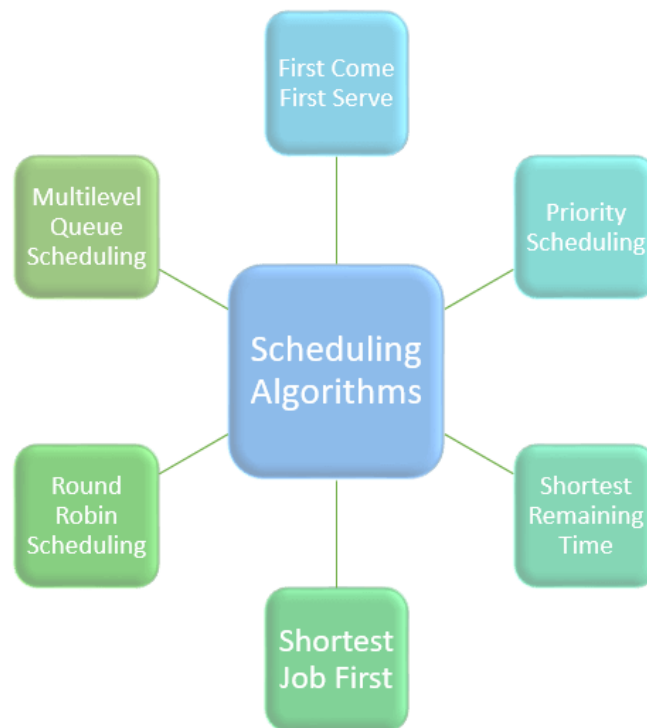
Timer interruption is a method that is closely related to preemption. When a certain process gets the CPU allocation, a timer may be set to a specified interval. Both timer interruption and preemption force a process to return the CPU before its CPU burst is complete.

Most of the multi-programmed operating system uses some form of a timer to prevent a process from tying up the system forever.

## Types of CPU scheduling Algorithm

There are mainly six types of process scheduling algorithms

1. First Come First Serve (FCFS)
2. Shortest-Job-First (SJF) Scheduling
3. Shortest Remaining Time
4. Priority Scheduling
5. Round Robin Scheduling
6. Multilevel Queue Scheduling



## Scheduling Algorithms

## **First Come First Serve**

First Come First Serve is the full form of FCFS. It is the easiest and most simple CPU scheduling algorithm. In this type of algorithm, the process which requests the CPU gets the CPU allocation first. This scheduling method can be managed with a FIFO queue.

As the process enters the ready queue, its PCB (Process Control Block) is linked with the tail of the queue. So, when CPU becomes free, it should be assigned to the process at the beginning of the queue.

### **Characteristics of FCFS method:**

- It offers non-preemptive and pre-emptive scheduling algorithm.
- Jobs are always executed on a first-come, first-serve basis.
- It is easy to implement and use.
- However, this method is poor in performance, and the general wait time is quite high.

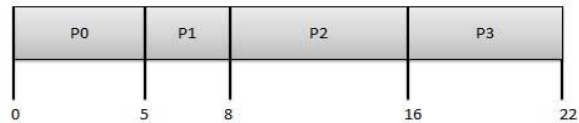


## **First Come First Serve (FCFS)**

- In the "First come first serve" scheduling algorithm, as the name suggests, the process which arrives first, gets executed first, or we can say that the process which requests the CPU first, gets the CPU allocated first.
- It is non-preemptive and preemptive scheduling algorithm.
- Its implementation is based on FIFO queue.
- When the CPU is free it is allocated to the process at the head of the queue.
- Once the CPU has been given to a process keeps the CPU busy.

# First Come First Serve (FCFS)

Process	Arrival Time	Execute Time	Service Time
P0	0	5	0
P1	1	3	5
P2	2	8	8
P3	3	6	16



Wait time of each process is as follows –

Process	Wait Time : Service Time - Arrival Time
P <sub>0</sub>	0 - 0 = 0
P <sub>1</sub>	5 - 1 = 4
P <sub>2</sub>	8 - 2 = 6
P <sub>3</sub>	16 - 3 = 13

Lets take an example:-

Q:-Draw the Gantt chart for FCFS at arrival time zero and burst-time is given in micro second calculate average waiting time and also calculate turn around time for the same process.

Process	Burst Time
P1	13
P2	08
P3	83

**Solution:-**

**Gantt Chart:-**



**Waiting time for each process is-**

Process	Waiting Time
P1	00
P2	13
P3	21

- **Average Waiting Time** =  $\frac{0 + 13 + 21}{3} = \frac{34}{3} = 11.33$  Micro-second

- **Turn Around Time:-**

Process	Turn Around Time
P1	13-0=13
P2	21-13=08
P3	104-21=83

- **Average Turn Around Time** =  $\frac{13 + 8 + 83}{3} = \frac{104}{3} = 3.46$  Micro-second

## Problems with FCFS Scheduling:-

Below we have a few shortcomings or problems with the FCFS scheduling algorithm:

- It is **Non Pre-emptive** algorithm, which means the **process priority** doesn't matter. If a process with very least priority is being executed, more like **daily routine backup** process, which takes more time, and all of a sudden some other high priority process arrives, like **interrupt to avoid system crash**, the high priority process will have to wait, and hence in this case, the system will crash, just because of improper process scheduling.
- Not optimal Average Waiting Time.
- Resources utilization in parallel is not possible, which leads to **Convoy Effect**, and hence poor resource(CPU, I/O etc) utilization.

## Shortest Remaining Time

The full form of SRT is Shortest remaining time. It is also known as SJF preemptive scheduling. In this method, the process will be allocated to the task, which is closest to its completion. This method prevents a newer ready state process from holding the completion of an older process.

### Characteristics of SRT scheduling method:

- This method is mostly applied in batch environments where short jobs are required to be given preference.
- This is not an ideal method to implement it in a shared system where the required CPU time is unknown.
- Associate with each process as the length of its next CPU burst. So that operating system uses these lengths, which helps to schedule the process with the shortest possible time.

## Shortest Job First

SJF is a full form of (Shortest job first) is a scheduling algorithm in which the process with the shortest execution time should be selected for execution next. This scheduling method can be preemptive or non-preemptive. It significantly reduces the average waiting time for other processes awaiting execution.

Shortest Job First scheduling works on the process with the shortest **burst time** or **duration** first.

- This is the best approach to minimize waiting time.
- This is used in Batch Systems.
- It is of two types:
  1. Non Pre-emptive



## 2. Pre-emptive

- To successfully implement it, the burst time/duration time of the processes should be known to the processor in advance, which is practically not feasible all the time.
- This scheduling algorithm is optimal if all the jobs/processes are available at the same time. (either Arrival time is 0 for all, or Arrival time is same for all).

## Characteristics of SJF Scheduling

- It is associated with each job as a unit of time to complete.
- In this method, when the CPU is available, the next process or job with the shortest completion time will be executed first.
- It is Implemented with non-preemptive policy.
- This algorithm method is useful for batch-type processing, where waiting for jobs to complete is not critical.
- It improves job output by offering shorter jobs, which should be executed first, which mostly have a shorter turnaround time.

## Non Pre-emptive Shortest Job First

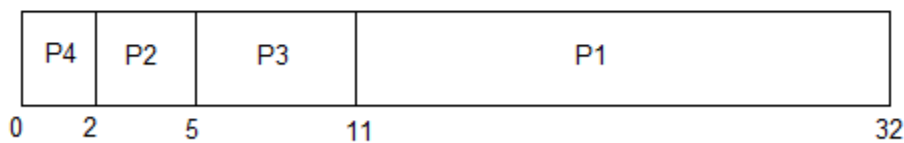
Consider the below processes available in the ready queue for execution, with **arrival time** as 0 for all and given **burst times**.

PROCESS	BURST TIME
P1	21
P2	3
P3	6
P4	2



In Shortest Job First Scheduling, the shortest Process is executed first.

Hence the GANTT chart will be following :



Now, the average waiting time will be =  $(0 + 2 + 5 + 11)/4 = 4.5$  ms

As you can see in the **GANTT chart** above, the process **P4** will be picked up first as it has the shortest burst time, then **P2**, followed by **P3** and at last **P1**.

We scheduled the same set of processes using the First come first serve algorithm in the previous tutorial, and got average waiting time to be 18.75 ms, whereas with SJF, the average waiting time comes out 4.5 ms.

### **Problem with Non Pre-emptive SJF**

If the **arrival time** for processes are different, which means all the processes are not available in the ready queue at time 0, and some jobs arrive after some time, in such situation, sometimes process with short burst time have to wait for the current process's execution to finish, because in Non Pre-emptive SJF, on arrival of a process with short duration, the existing job/process's execution is not halted/stopped to execute the short job first.

This leads to the problem of **Starvation**, where a shorter process has to wait for a long time until the current longer process gets executed. This happens if shorter jobs keep coming, but this can be solved using the concept of **aging**.

### **Pre-emptive Shortest Job First**

In Preemptive Shortest Job First Scheduling, jobs are put into ready queue as they arrive, but as a process with **short burst time** arrives, the existing process is preempted or removed from execution, and the shorter job is executed first.

PROCESS	BURST TIME	ARRIVAL TIME
P1	21	0
P2	3	1
P3	6	2
P4	2	3

The GANTT chart for Preemptive Shortest Job First Scheduling will be,

P1	P2	P4	P2	P3	P1	
0	1	3	5	6	12	32

The average waiting time will be,  $((5-3) + (6-2) + (12-1))/4 = 4.25$  ms

The average waiting time for preemptive shortest job first scheduling is less than both, non-preemptive SJF scheduling and FCFS scheduling.

As you can see in the **GANTT chart** above, as **P1** arrives first, hence its execution starts immediately, but just after 1 ms, process **P2** arrives with a **burst time** of 3 ms which is less than the burst time of **P1**, hence the process **P1** (1 ms done, 20 ms left) is preempted and process **P2** is executed.

As **P2** is getting executed, after 1 ms, **P3** arrives, but it has a burst time greater than that of **P2**, hence execution of **P2** continues. But after another millisecond, **P4** arrives with a burst time of 2 ms, as a result **P2** (2 ms done, 1 ms left) is preempted and **P4** is executed.

After the completion of **P4**, process **P2** is picked up and finishes, then **P2** will get executed and at last **P1**.

The Pre-emptive SJF is also known as **Shortest Remaining Time First**, because at any given point of time, the job with the shortest remaining time is executed first.

### **Priority Based Scheduling**

Priority scheduling is a method of scheduling processes based on priority. In this method, the scheduler selects the tasks to work as per the priority.

Priority scheduling also helps OS to involve priority assignments. The processes with higher priority should be carried out first, whereas jobs with equal priorities are carried out on a round-robin or FCFS basis. Priority can be decided based on memory requirements, time requirements, etc.

In case of priority scheduling the priority is not always set as the inverse of the CPU burst time, rather it can be internally or externally set, but yes the scheduling is done on the basis of priority of the process where the process which is most urgent is processed first, followed by the ones with lesser priority in order.

Processes with same priority are executed in FCFS manner.

The priority of process, when internally defined, can be decided based on **memory requirements, time limits, number of open files, ratio of I/O burst to CPU burst** etc.

Whereas, external priorities are set based on criteria outside the operating system, like the importance of the process, funds paid for the computer resource use, market factor etc.

### **Types of Priority Scheduling Algorithm**

Priority scheduling can be of two types:



1. **Preemptive Priority Scheduling:** If the new process arrived at the ready queue has a higher priority than the currently running process, the CPU is preempted, which means the processing of the current process is stopped and the incoming new process with higher priority gets the CPU for its execution.
2. **Non-Preemptive Priority Scheduling:** In case of non-preemptive priority scheduling algorithm if a new process arrives with a higher priority than the current running process, the incoming process is put at the head of the ready queue, which means after the execution of the current process it will be processed.

**Advantages-**

- It considers the priority of the processes and allows the important processes to run first.
- Priority scheduling in preemptive mode is best suited for real time operating system.

**Disadvantages-**

- Processes with lesser priority may starve for CPU.
- There is no idea of response time and waiting time.

**Important Notes-**

**Note-01:**

- The waiting time for the process having the highest priority will always be zero in preemptive mode.
- The waiting time for the process having the highest priority may not be zero in non-preemptive mode.

**Note-02:**

Priority scheduling in preemptive and non-preemptive mode behaves exactly same under following conditions-

- The arrival time of all the processes is same
- All the processes become available

## PRACTICE PROBLEMS BASED ON PRIORITY SCHEDULING-

### Problem-01:

Consider the set of 5 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time	Priority
P1	0	4	2
P2	1	3	3
P3	2	1	4
P4	3	5	5
P5	4	2	5

If the CPU scheduling policy is priority non-preemptive, calculate the average waiting time and average turn around time. (Higher number represents higher priority)

**Solution- Gantt Chart-**



**Gantt Chart**

Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Process Id	Exit time	Turn Around time	Waiting time
P1	4	$4 - 0 = 4$	$4 - 4 = 0$
P2	15	$15 - 1 = 14$	$14 - 3 = 11$
P3	12	$12 - 2 = 10$	$10 - 1 = 9$

P4	9	$9 - 3 = 6$	$6 - 5 = 1$
P5	11	$11 - 4 = 7$	$7 - 2 = 5$

- Average Turn Around time =  $(4 + 14 + 10 + 6 + 7) / 5 = 41 / 5 = 8.2$  unit
- Average waiting time =  $(0 + 11 + 9 + 1 + 5) / 5 = 26 / 5 = 5.2$  unit

### Problem-02:

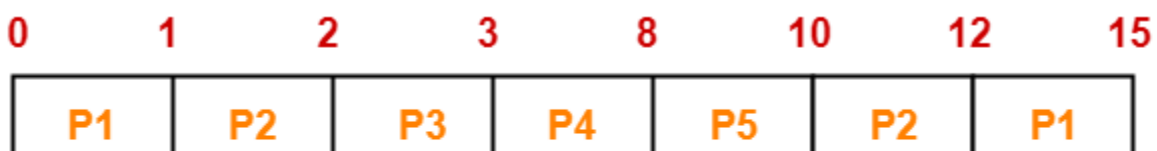
Consider the set of 5 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time	Priority
P1	0	4	2
P2	1	3	3
P3	2	1	4
P4	3	5	5
P5	4	2	5

If the CPU scheduling policy is priority preemptive, calculate the average waiting time and average turn around time. (Higher number represents higher priority)

**Solution-**

**Gantt Chart-**



**Gantt Chart**

Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Process Id	Exit time	Turn Around time	Waiting time
P1	15	$15 - 0 = 15$	$15 - 4 = 11$
P2	12	$12 - 1 = 11$	$11 - 3 = 8$
P3	3	$3 - 2 = 1$	$1 - 1 = 0$
P4	8	$8 - 3 = 5$	$5 - 5 = 0$
P5	10	$10 - 4 = 6$	$6 - 2 = 4$

Now,

- Average Turn Around time =  $(15 + 11 + 1 + 5 + 6) / 5 = 38 / 5 = 7.6$  unit
- Average waiting time =  $(11 + 8 + 0 + 0 + 4) / 5 = 23 / 5 = 4.6$  unit

### Problem:3

Given: Table of processes, and their Arrival time, Execution time, and priority. Here we are considering 1 is the lowest priority.

Process	Arrival Time	Execution Time	Priority	Service Time
P0	0	5	1	0
P1	1	3	2	11
P2	2	8	1	14
P3	3	6	3	5

**Waiting time** of each process is as follows –

Process	Waiting Time
P0	$0 - 0 = 0$
P1	$11 - 1 = 10$
P2	$14 - 2 = 12$
P3	$5 - 3 = 2$

Average Wait Time:  $(0 + 10 + 12 + 2)/4 = 24 / 4 = 6$

### **Problem with Priority Scheduling Algorithm**

In priority scheduling algorithm, the chances of **indefinite blocking** or **starvation**.

A process is considered blocked when it is ready to run but has to wait for the CPU as some other process is running currently.

But in case of priority scheduling if new higher priority processes keeps coming in the ready queue then the processes waiting in the ready queue with lower priority may have to wait for long durations before getting the CPU for execution.

In 1973, when the IBM 7904 machine was shut down at MIT, a low-priority process was found which was submitted in 1967 and had not yet been run.\

### **Round-Robin Scheduling**

Round robin is the oldest, simplest scheduling algorithm. The name of this algorithm comes from the round-robin principle, where each person gets an equal share of something in turn. It is mostly used for scheduling algorithms in multitasking. This algorithm method helps for starvation free execution of processes.

### **Characteristics of Round-Robin Scheduling**

- Round robin is a hybrid model which is clock-driven
- Time slice should be minimum, which is assigned for a specific task to be processed. However, it may vary for different processes.
- It is a real time system which responds to the event within a specific time limit.



### Key Points:-

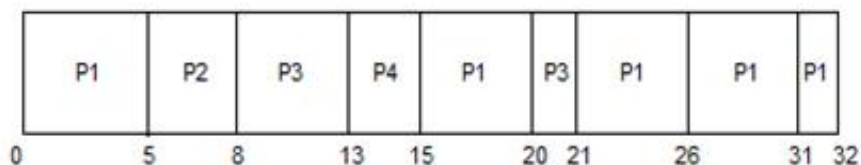
- A fixed time is allotted to each process, called **quantum**, for execution.
- Once a process is executed for given time period that process is preempted and other process executes for given time period.
- Context switching is used to save states of preempted processes.

Example:- In the given table Process(P1, P2, P3, P4) and burst time(21, 3, 6, 2) is given. The time quantum is 5 make a gantt chart and also calculate average waiting time.

PROCESS	BURST TIME
P1	21
P2	3
P3	6
P4	2



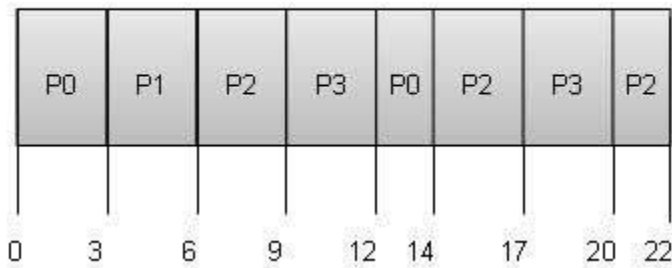
The GANTT chart for round robin scheduling will be,



Process	Waiting Time
P1	$(0-0) + (15-5) + (21-20) + (26-21) + (31-26) + (32-31) = 22$
P2	$(5-1) = 4$
P3	$(8-2) + (20-13) = 13$
p4	$(13-3) = 10$

$$\text{Average Waiting Time} = \frac{22+4+13+10}{4} = 12.25$$

Quantum = 3



**Q2.**

**Wait time** of each process is as follows –

Process	Wait Time : Service Time - Arrival Time
P0	$(0 - 0) + (12 - 3) = 9$
P1	$(3 - 1) = 2$
P2	$(6 - 2) + (14 - 9) + (20 - 17) = 12$
P3	$(9 - 3) + (17 - 12) = 11$

Average Wait Time:  $(9+2+12+11) / 4 = 8.5$

## Multiple-Level Queues Scheduling

This algorithm separates the ready queue into various separate queues. In this method, processes are assigned to a queue based on a specific property of the process, like the process priority, size of the memory, etc.

However, this is not an independent scheduling OS algorithm as it needs to use other types of algorithms in order to schedule the jobs.

### Characteristic of Multiple-Level Queues Scheduling:

- Multiple queues should be maintained for processes with some characteristics.
- Every queue may have its separate scheduling algorithms.
- Priorities are given for each queue.

### The Purpose of a Scheduling algorithm

Here are the reasons for using a scheduling algorithm:

- The CPU uses scheduling to improve its efficiency.

- It helps you to allocate resources among competing processes.
- The maximum utilization of CPU can be obtained with multi-programming.
- The processes which are to be executed are in ready queue.
- 

## **Multiple-Processor Scheduling in Operating System**

In multiple-processor scheduling **multiple** CPU's are available and hence **Load Sharing** becomes possible. However multiple processor scheduling is more **complex** as compared to single processor scheduling. In multiple processor scheduling there are cases when the processors are identical i.e. HOMOGENEOUS, in terms of their functionality, we can use any processor available to run any process in the queue.

### **Approaches to Multiple-Processor Scheduling –**

One approach is when all the scheduling decisions and I/O processing are handled by a single processor which is called the **Master Server** and the other processors executes only the **user code**. This is simple and reduces the need of data sharing. This entire scenario is called **Asymmetric Multiprocessing**.

A second approach uses **Symmetric Multiprocessing** where each processor is **self scheduling**. All processes may be in a common ready queue or each processor may have its own private queue for ready processes. The scheduling proceeds further by having the scheduler for each processor examine the ready queue and select a process to execute.

# OPERATING SYSTEM

## UNIT-3

### Deadlock

Every process needs some resources to complete its execution. However, the resource is granted in a sequential order.

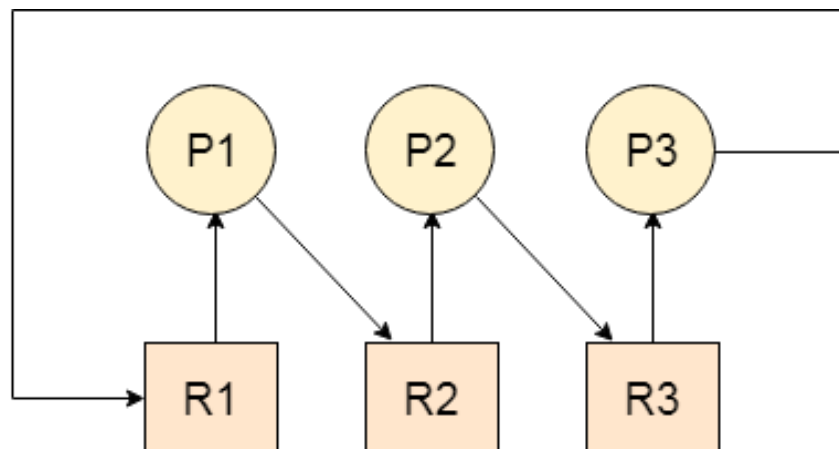
1. The process requests for some resource.
2. OS grant the resource if it is available otherwise let the process waits.
3. The process uses it and release on the completion.

A Deadlock is a situation where each of the computer process waits for a resource which is being assigned to some another process. In this situation, none of the process gets executed since the resource it needs, is held by some other process which is also waiting for some other resource to be released.

Let us assume that there are three processes P1, P2 and P3. There are three different resources R1, R2 and R3. R1 is assigned to P1, R2 is assigned to P2 and R3 is assigned to P3.

After some time, P1 demands for R2 which is being used by P2. P1 halts its execution since it can't complete without R2. P2 also demands for R3 which is being used by P3. P2 also stops its execution because it can't continue without R3. P3 also demands for R1 which is being used by P1 therefore P3 also stops its execution.

In this scenario, a cycle is being formed among the three processes. None of the process is progressing and they are all waiting. The computer becomes unresponsive since all the processes got blocked.



## System Model:-

- For the purposes of deadlock discussion, a system can be modeled as a collection of limited resources, which can be partitioned into different categories, to be allocated to a number of processes, each having different needs.
- Resource categories may include memory, printers, CPUs, open files, tape drives, CD-ROMS, etc.
- By definition, all the resources within a category are equivalent, and a request of this category can be equally satisfied by any one of the resources in that category. If this is not the case ( i.e. if there is some difference between the resources within a category ), then that category needs to be further divided into separate categories. For example, "printers" may need to be separated into "laser printers" and "color inkjet printers".
- Some categories may have a single resource.
- In normal operation a process must request a resource before using it, and release it when it is done, in the following sequence:
  1. **Request** - If the request cannot be immediately granted, then the process must wait until the resource(s) it needs become available. For example the system calls `open( )`, `malloc( )`, `new( )`, and `request( )`.
  2. **Use** - The process uses the resource, e.g. prints to the printer or reads from the file.
  3. **Release** - The process relinquishes the resource. so that it becomes available for other processes. For example, `close( )`, `free( )`, `delete( )`, and `release( )`.
- For all kernel-managed resources, the kernel keeps track of what resources are free and which are allocated, to which process they are allocated, and a queue of processes waiting for this resource to become available. Application-managed resources can be controlled using mutexes or `wait( )` and `signal( )` calls, ( i.e. binary or counting semaphores. )
- A set of processes is deadlocked when every process in the set is waiting for a resource that is currently allocated to another process in the set ( and which can only be released when that other waiting process makes progress.)

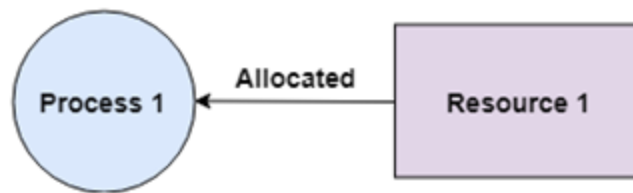
## Deadlock Characterization:-

### Necessary conditions for Deadlocks

#### 1. Mutual Exclusion

A resource can only be shared in mutually exclusive manner. It implies, if two process cannot use the same resource at the same time.

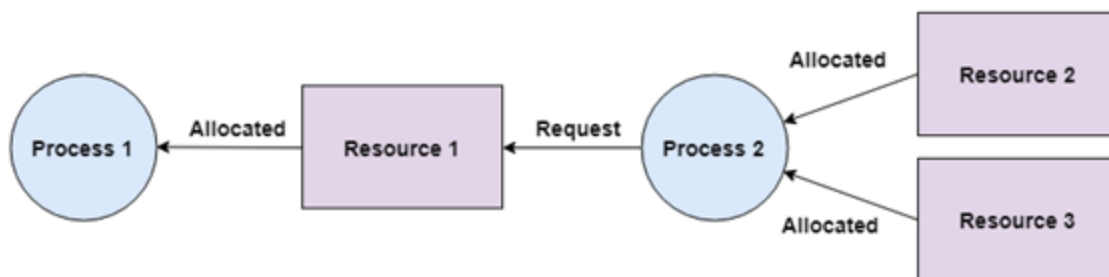
There should be a resource that can only be held by one process at a time. In the diagram below, there is a single instance of Resource 1 and it is held by Process 1 only.



## 2. Hold and Wait

A process waits for some resources while holding another resource at the same time.

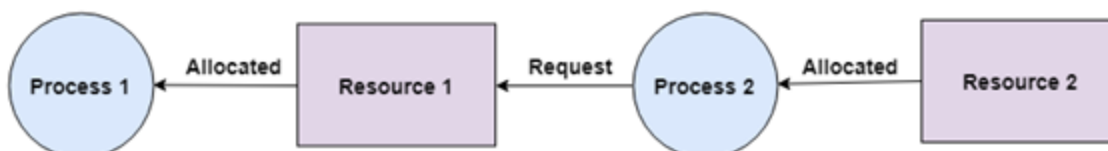
A process can hold multiple resources and still request more resources from other processes which are holding them. In the diagram given below, Process 2 holds Resource 2 and Resource 3 and is requesting the Resource 1 which is held by Process 1.



## 3. No preemption

The process which once scheduled will be executed till the completion. No other process can be scheduled by the scheduler meanwhile.

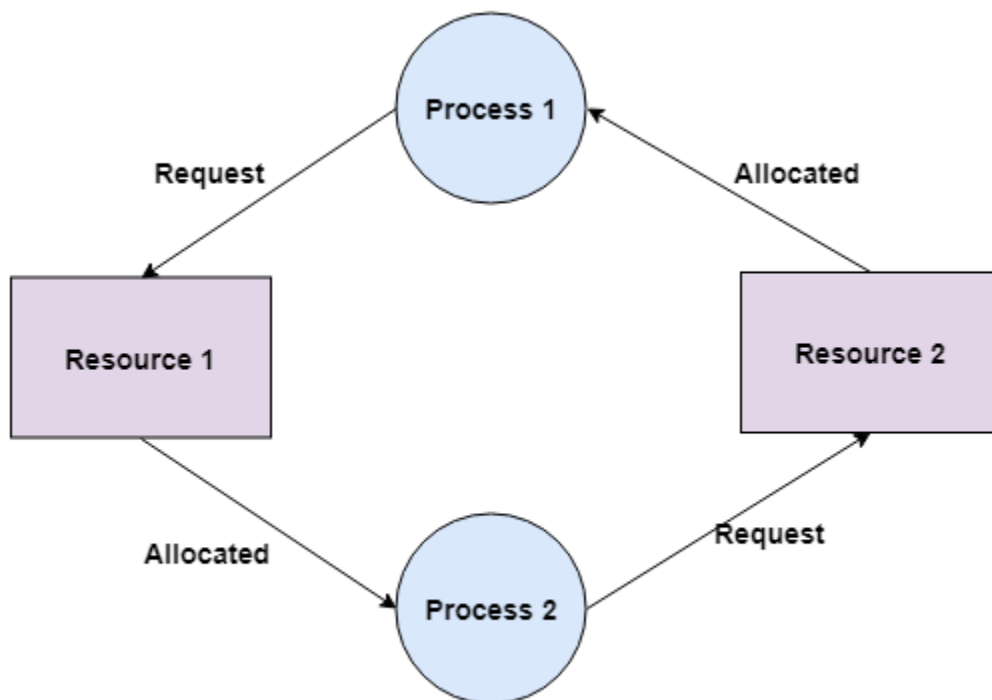
A resource cannot be preempted from a process by force. A process can only release a resource voluntarily. In the diagram below, Process 2 cannot preempt Resource 1 from Process 1. It will only be released when Process 1 relinquishes it voluntarily after its execution is complete.



#### 4. Circular Wait

All the processes must be waiting for the resources in a cyclic manner so that the last process is waiting for the resource which is being held by the first process.

A process is waiting for the resource held by the second process, which is waiting for the resource held by the third process and so on, till the last process is waiting for a resource held by the first process. This forms a circular chain. For example: Process 1 is allocated Resource2 and it is requesting Resource 1. Similarly, Process 2 is allocated Resource 1 and it is requesting Resource 2. This forms a circular wait loop.



#### Methods for Handling Deadlocks

- Generally speaking there are three ways of handling deadlocks:
1. **Deadlock prevention or avoidance** - Do not allow the system to get into a deadlocked state. In order to avoid deadlocks, the system must have additional information about all processes. In particular, the system must know what resources a process will or may request in the future. (Ranging from a simple worst-case maximum to a complete resource request and release plan for each process, depending on the particular algorithm.)

2. **Deadlock detection and recovery** - Abort a process or preempt some resources when deadlocks are detected. Deadlock detection is fairly straightforward, but deadlock recovery requires either aborting processes or preempting resources, neither of which is an attractive alternative.
3. **Ignore the problem all together** - If deadlocks only occur once a year or so, it may be better to simply let them happen and reboot as necessary than to incur the constant overhead and system performance penalties associated with deadlock prevention or detection. This is the approach that both Windows and UNIX take. If deadlocks are neither prevented nor detected, then when a deadlock occurs the system will gradually slow down, as more and more processes become stuck waiting for resources currently held by the deadlock and by other waiting processes. Unfortunately this slowdown can be indistinguishable from a general system slowdown when a real-time process has heavy computing needs.

## 1. Deadlock Prevention

If we simulate deadlock with a table which is standing on its four legs then we can also simulate four legs with the four conditions which when occurs simultaneously, cause the deadlock.

However, if we break one of the legs of the table then the table will fall definitely. The same happens with deadlock, if we can be able to violate one of the four necessary conditions and don't let them occur together then we can prevent the deadlock.

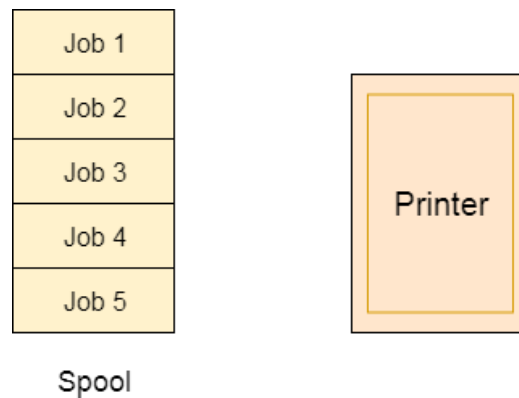
Let's see how we can prevent each of the conditions.

## Mutual Exclusion

### Spooling

For a device like printer, spooling can work. There is a memory associated with the printer which stores jobs from each of the process into it. Later, Printer collects all the jobs and print each one of them according to FCFS. By using this mechanism, the process doesn't have to wait for the printer and it can continue whatever it was doing. Later, it collects the output when it is produced.





Although, Spooling can be an effective approach to violate mutual exclusion but it suffers from two kinds of problems.

1. This cannot be applied to every resource.
2. After some point of time, there may arise a race condition between the processes to get space in that spool.

We cannot force a resource to be used by more than one process at the same time since it will not be fair enough and some serious problems may arise in the performance. Therefore, we cannot violate mutual exclusion for a process practically.

## 2. Hold and Wait

Hold and wait condition lies when a process holds a resource and waiting for some other resource to complete its task. Deadlock occurs because there can be more than one process which are holding one resource and waiting for other in the cyclic order.

However, we have to find out some mechanism by which a process either doesn't hold any resource or doesn't wait. That means, a process must be assigned all the necessary resources before the execution starts. A process must not wait for any resource once the execution has been started.

**!(Hold and wait) = !hold or !wait (negation of hold and wait is, either you don't hold or you don't wait)**

This can be implemented practically if a process declares all the resources initially. However, this sounds very practical but can't be done in the computer system because a process can't determine necessary resources initially.

Process is the set of instructions which are executed by the CPU. Each of the instruction may demand multiple resources at the multiple times. The need cannot be fixed by the OS.

The problem with the approach is:

1. Practically not possible.
2. Possibility of getting starved will be increases due to the fact that some process may hold a resource for a very long time.

### **3. No Preemption**

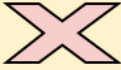
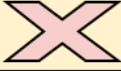


Deadlock arises due to the fact that a process can't be stopped once it starts. However, if we take the resource away from the process which is causing deadlock then we can prevent deadlock.

This is not a good approach at all since if we take a resource away which is being used by the process then all the work which it has done till now can become inconsistent.

Consider a printer is being used by any process. If we take the printer away from that process and assign it to some other process then all the data which has been printed can become inconsistent and ineffective and also the fact that the process can't start printing again from where it has left which causes performance inefficiency.

### **4. Circular Wait**

To violate circular wait, we can assign a priority number to each of the resource. A process can't request for a lesser priority resource. This ensures that not a single process can request a resource which is being utilized by some other process and no cycle will be formed.

Condition	Approach	Is Practically Possible?
Mutual Exclusion	Spooling	
Hold and Wait	Request for all the resources initially	
No Preemption	Snatch all the resources	
Circular Wait	Assign priority to each resources and order resources numerically	

Among all the methods, violating Circular wait is the only approach that can be implemented practically.

### **Deadlock Avoidance:-**

In deadlock avoidance, the request for any resource will be granted if the resulting state of the system doesn't cause deadlock in the system. The state of the system will continuously be checked for safe and unsafe states.

In order to avoid deadlocks, the process must tell OS, the maximum number of resources a process can request to complete its execution.

The simplest and most useful approach states that the process should declare the maximum number of resources of each type it may ever need. The Deadlock avoidance algorithm examines the resource allocations so that there can never be a circular wait condition.

### **Safe and Unsafe States**

The resource allocation state of a system can be defined by the instances of available and allocated resources, and the maximum instance of the resources demanded by the processes.

A state of a system recorded at some random time is shown below.

## Resources Assigned

Process	Type 1	Type 2	Type 3	Type 4
A	3	0	2	2
B	0	0	1	1
C	1	1	1	0
D	2	1	4	0

## Resources still needed

Process	Type 1	Type 2	Type 3	Type 4
A	1	1	0	0
B	0	1	1	2
C	1	2	1	0
D	2	1	1	2

$$E = (7 \ 6 \ 8 \ 4)$$

$$P = (6 \ 2 \ 8 \ 3)$$

$$A = (1 \ 4 \ 0 \ 1)$$

Above tables and vector E, P and A describes the resource allocation state of a system. There are 4 processes and 4 types of the resources in a system. Table 1 shows the instances of each resource assigned to each process.

Table 2 shows the instances of the resources, each process still needs. Vector E is the representation of total instances of each resource in the system.

Vector P represents the instances of resources that have been assigned to processes. Vector A represents the number of resources that are not in use.

A state of the system is called safe if the system can allocate all the resources requested by all the processes without entering into deadlock.

If the system cannot fulfill the request of all processes then the state of the system is called unsafe.

The key of Deadlock avoidance approach is when the request is made for resources then the request must only be approved in the case if the resulting state is also a safe state.

### **Banker's Algorithm:-**

Banker's algorithm is a **deadlock avoidance algorithm**. It is named so because this algorithm is used in banking systems to determine whether a loan can be granted or not.

Consider there are  $n$  account holders in a bank and the sum of the money in all of their accounts is  $S$ . Every time a loan has to be granted by the bank, it subtracts the **loan amount** from the **total money** the bank has. Then it checks if that difference is greater than  $S$ . It is done because, only then, the bank would have enough money even if all the  $n$  account holders draw all their money at once.

Banker's algorithm works in a similar way in computers.

Whenever a new process is created, it must specify the maximum instances of each resource type that it needs, exactly.

Let us assume that there are  $n$  processes and  $m$  resource types. Some data structures that are used to implement the banker's algorithm are:

### 1. Available

It is an **array** of length  $m$ . It represents the number of available resources of each type. If  $Available[j] = k$ , then there are  $k$  instances available, of resource type  $R(j)$ .

### 2. Max

It is an  $n \times m$  matrix which represents the maximum number of instances of each resource that a process can request. If  $Max[i][j] = k$ , then the process  $P(i)$  can request atmost  $k$  instances of resource type  $R(j)$ .

### 3. Allocation

It is an  $n \times m$  matrix which represents the number of resources of each type currently allocated to each process. If  $Allocation[i][j] = k$ , then process  $P(i)$  is currently allocated  $k$  instances of resource type  $R(j)$ .

### 4. Need

It is an  $n \times m$  matrix which indicates the remaining resource needs of each process. If  $Need[i][j] = k$ , then process  $P(i)$  may need  $k$  more instances of resource type  $R(j)$  to complete its task.

$$Need[i][j] = Max[i][j] - Allocation[i][j]$$

### Safety Algorithm:-

**This algo used for finding out whether a system in a safe state or not.**

1) Let *Work* and *Finish* be vectors of length ' $m$ ' and ' $n$ ' respectively.

Initialize:  $Work = Available$

$Finish[i] = false$ ; for  $i=1, 2, 3, 4, \dots, n$

2) Find an  $i$  such that both

a)  $Finish[i] = false$

b)  $Need_i \leq Work$

if no such  $i$  exists goto step (4)

3)  $Work = Work + Allocation[i]$

$Finish[i] = true$

goto step (2)

4) if  $Finish[i] = true$  for all  $i$

then the system is in a safe state

## Resource-Request Algorithm

Let  $Request_i$  be the request array for process  $P_i$ .  $Request_i[j] = k$  means process  $P_i$  wants  $k$  instances of resource type  $R_j$ . When a request for resources is made by process  $P_i$ , the following actions are taken:

1) If  $Request_i \leq Need_i$

Goto step (2) ; otherwise, raise an error condition, since the process has exceeded its maximum claim.

2) If  $Request_i \leq Available$

Goto step (3); otherwise,  $P_i$  must wait, since the resources are not available.

3) Have the system pretend to have allocated the requested resources to process  $P_i$  by modifying the state as follows:

$Available = Available - Request_i$

$Allocation_i = Allocation_i + Request_i$

$Need_i = Need_i - Request_i$

### Example:

Considering a system with five processes  $P_0$  through  $P_4$  and three resources of type A, B, C. Resource type A has 10 instances, B has 5 instances and type C has 7 instances. Suppose at time  $t_0$  following snapshot of the system has been taken:

Process	Allocation	Max	Available
	A B C	A B C	A B C
$P_0$	0 1 0	7 5 3	3 3 2
$P_1$	2 0 0	3 2 2	
$P_2$	3 0 2	9 0 2	
$P_3$	2 1 1	2 2 2	
$P_4$	0 0 2	4 3 3	

### Question1. What will be the content of the Need matrix?

$Need[i, j] = Max[i, j] - Allocation[i, j]$

For  $P_0 = (7, 5, 3) - (0, 1, 0)$

Need for  $P_0 = (7, 4, 3)$

For  $P_1 = (3, 2, 2) - (2, 0, 0)$

Need for  $P_1 = (1, 2, 2)$

For  $P_2 = (9,0,2) - (3,0,2)$

Need for  $P_2 = (6,0,0)$

For  $P_3 = (2,2,2) - (2,1,1)$

Need for  $P_3 = (0,1,1)$

For  $P_4 = (4,3,3) - (0,0,2)$

Need for  $P_4 = (4,3,1)$

So, the content of Need Matrix is:

Process	Need		
	A	B	C
$P_0$	7	4	3
$P_1$	1	2	2
$P_2$	6	0	0
$P_3$	0	1	1
$P_4$	4	3	1



## Question2. Is the system in a safe state? If Yes, then what is the safe sequence?

Applying the Safety algorithm on the given system,

$m=3, n=5$  Step 1 of Safety Algo

Work = Available

Work = 

3	3	2
---	---	---

0      1      2      3      4

Finish = 

false	false	false	false	false
-------	-------	-------	-------	-------

For  $i=0$  Step 2:

Need<sub>0</sub> = 7, 4, 3 ✗

Finish [0] is false and Need<sub>0</sub> > Work ✗

So P<sub>0</sub> must wait But Need ≤ Work

For  $i=1$  Step 2:

Need<sub>1</sub> = 1, 2, 2 ✓

Finish [1] is false and Need<sub>1</sub> < Work ✓

So P<sub>1</sub> must be kept in safe sequence

Step 3

Work = Work + Allocation<sub>1</sub>

Work = 

5	3	2
---	---	---

0      1      2      3      4

Finish = 

false	true	false	false	false
-------	------	-------	-------	-------

For  $i=2$  Step 2:

Need<sub>2</sub> = 6, 0, 0 ✗

Finish [2] is false and Need<sub>2</sub> > Work ✗

So P<sub>2</sub> must wait

For  $i=3$  Step 2:

Need<sub>3</sub> = 0, 1, 1 ✓

Finish [3] = false and Need<sub>3</sub> < Work ✓

So P<sub>3</sub> must be kept in safe sequence

Step 3

Work = Work + Allocation<sub>3</sub>

Work = 

7	4	3
---	---	---

0      1      2      3      4

Finish = 

false	true	false	true	false
-------	------	-------	------	-------

For  $i=4$  Step 2:

Need<sub>4</sub> = 4, 3, 1 ✓

Finish [4] = false and Need<sub>4</sub> < Work ✓

So P<sub>4</sub> must be kept in safe sequence

Step 3

Work = Work + Allocation<sub>4</sub>

Work = 

7	4	5
---	---	---

0      1      2      3      4

Finish = 

false	true	false	true	true
-------	------	-------	------	------

For  $i=0$  Step 2:

Need<sub>0</sub> = 7, 4, 3 ✓

Finish [0] is false and Need < Work ✓

So P<sub>0</sub> must be kept in safe sequence

Step 3

Work = Work + Allocation<sub>0</sub>

Work = 

7	5	5
---	---	---

0      1      2      3      4

Finish = 

true	true	false	true	true
------	------	-------	------	------

For  $i=2$  Step 2:

Need<sub>2</sub> = 6, 0, 0 ✓

Finish [2] is false and Need<sub>2</sub> < Work ✓

So P<sub>2</sub> must be kept in safe sequence

Step 3

Work = Work + Allocation<sub>2</sub>

Work = 

10	5	7
----	---	---

0      1      2      3      4

Finish = 

true	true	true	true	true
------	------	------	------	------

Step 4

Finish [i] = true for  $0 \leq i \leq n$

Hence the system is in Safe state

The safe sequence is P<sub>1</sub>, P<sub>3</sub>, P<sub>4</sub>, P<sub>0</sub>, P<sub>2</sub>

**Question3.** What will happen if process  $P_1$  requests one additional instance of resource type A and two instances of resource type C?

A B C  
Request<sub>1</sub> = 1, 0, 2

To decide whether the request is granted we use Resource Request algorithm

Step 1  
Request<sub>1</sub> < Need<sub>1</sub> ✓

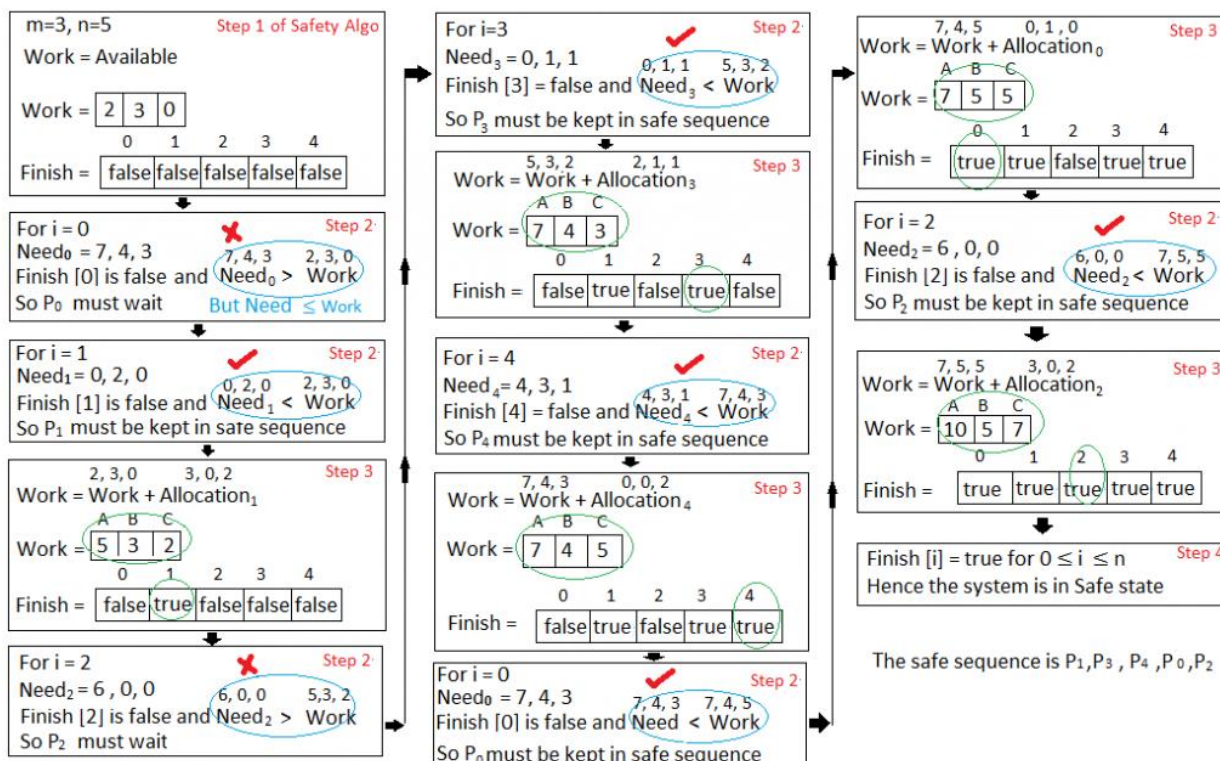
Step 2  
Request<sub>1</sub> < Available ✓

Step 3

Available = Available - Request<sub>1</sub>  
Allocation<sub>1</sub> = Allocation<sub>1</sub> + Request<sub>1</sub>  
Need<sub>1</sub> = Need<sub>1</sub> - Request<sub>1</sub>

Process	Allocation	Need	Available
	A B C	A B C	A B C
P <sub>0</sub>	0 1 0	7 4 3	2 3 0
P <sub>1</sub>	3 0 2	0 2 0	
P <sub>2</sub>	3 0 2	6 0 0	
P <sub>3</sub>	2 1 1	0 1 1	
P <sub>4</sub>	0 0 2	4 3 1	

We must determine whether this new system state is safe. To do so, we again execute Safety algorithm on the above data structures.



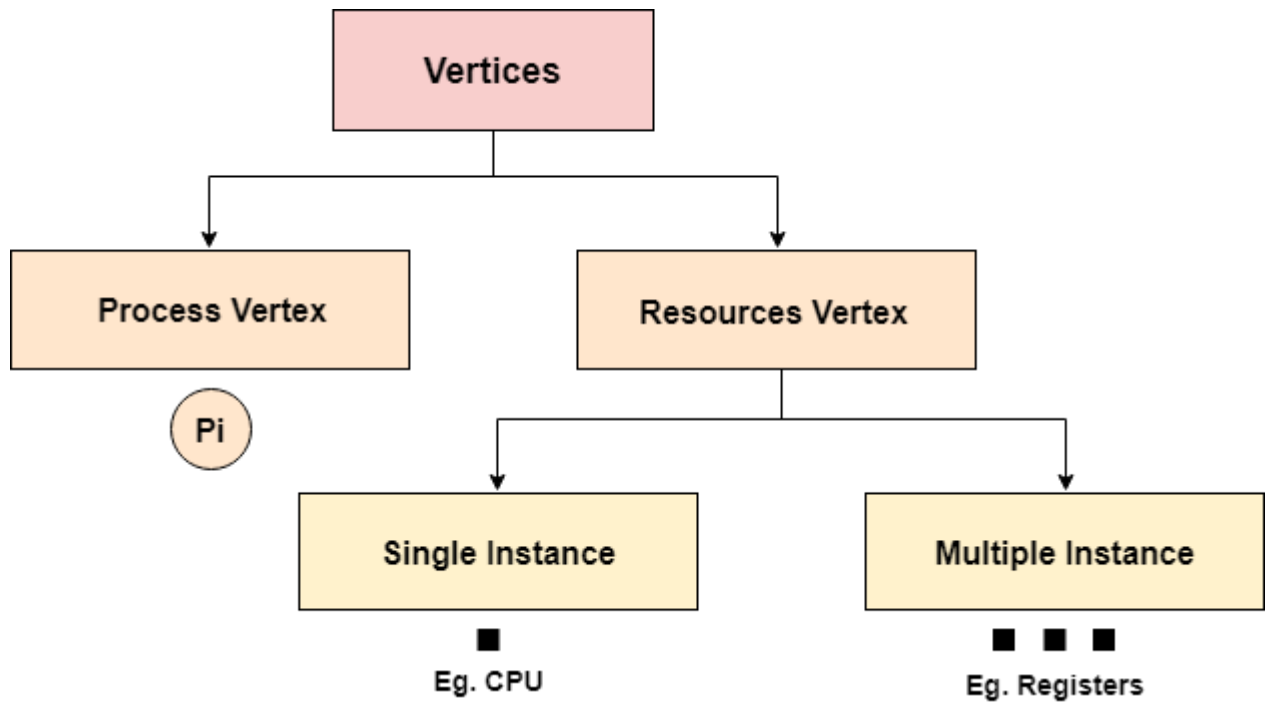
Hence the new system state is safe, so we can immediately grant the request for process  $P_1$ .

## Resource Allocation Graph:-

The resource allocation graph is the pictorial representation of the state of a system. As its name suggests, the resource allocation graph is the complete information about all the processes which are holding some resources or waiting for some resources.

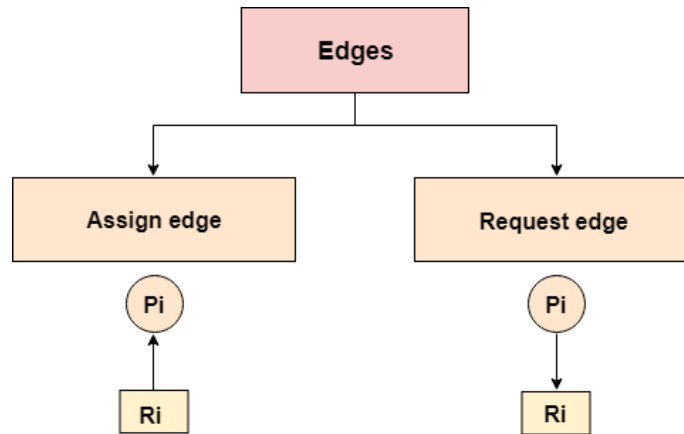
It also contains the information about all the instances of all the resources whether they are available or being used by the processes.

In Resource allocation graph, the process is represented by a Circle while the Resource is represented by a rectangle. Let's see the types of vertices and edges in detail.



Vertices are mainly of two types, Resource and process. Each of them will be represented by a different shape. Circle represents process while rectangle represents resource.

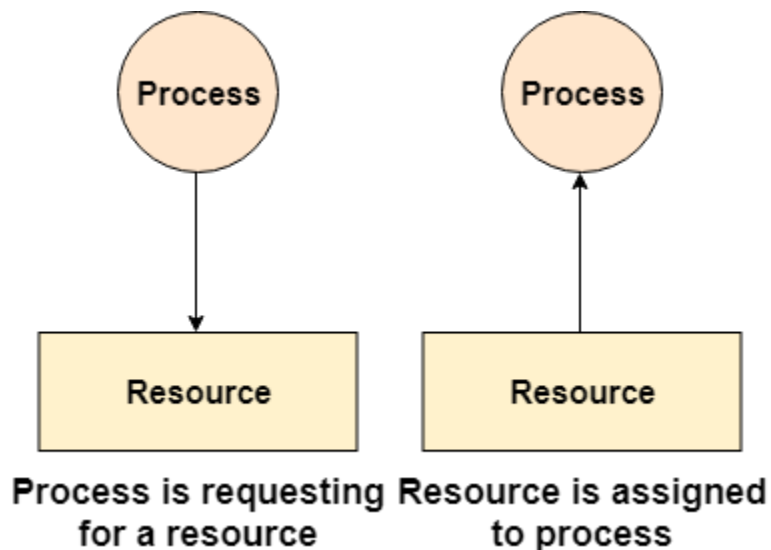
A resource can have more than one instance. Each instance will be represented by a dot inside the rectangle.



Edges in RAG are also of two types, one represents assignment and other represents the wait of a process for a resource. The above image shows each of them.

A resource is shown as assigned to a process if the tail of the arrow is attached to an instance to the resource and the head is attached to a process.

A process is shown as waiting for a resource if the tail of an arrow is attached to the process while the head is pointing towards the resource.

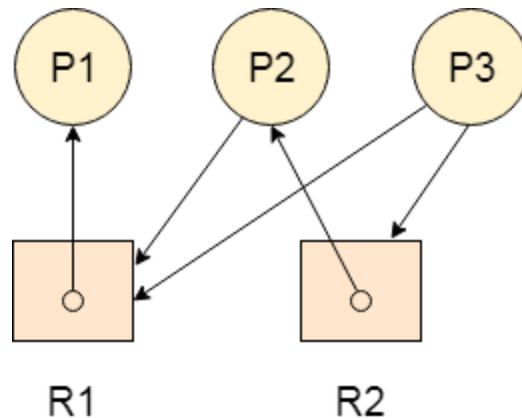


## Example

Let's consider 3 processes P1, P2 and P3, and two types of resources R1 and R2. The resources are having 1 instance each.

According to the graph, R1 is being used by P1, P2 is holding R2 and waiting for R1, P3 is waiting for R1 as well as R2.

The graph is deadlock free since no cycle is being formed in the graph.

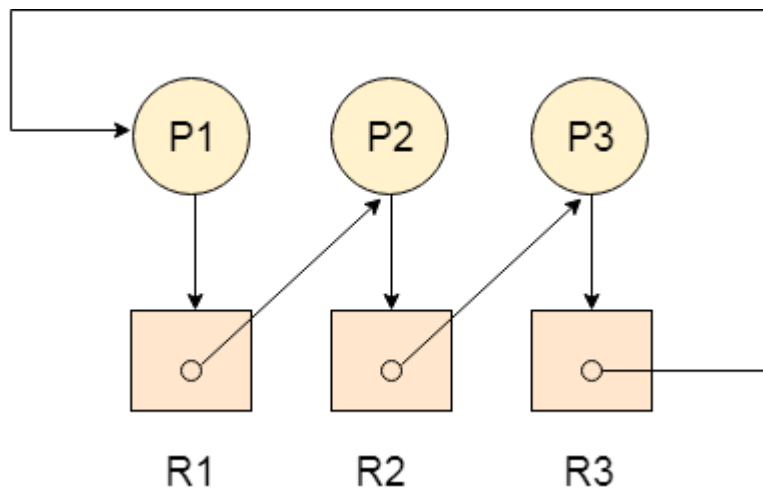


### Deadlock Detection using RAG

If a cycle is being formed in a Resource allocation graph where all the resources have the single instance then the system is deadlocked.

In Case of Resource allocation graph with multi-instanced resource types, Cycle is a necessary condition of deadlock but not the sufficient condition.

The following example contains three processes P1, P2, P3 and three resources R1, R2, R3. All the resources are having single instances each.



If we analyze the graph then we can find out that there is a cycle formed in the graph since the system is satisfying all the four conditions of deadlock.

### Allocation Matrix

Allocation matrix can be formed by using the Resource allocation graph of a system. In Allocation matrix, an entry will be made for each of the resource assigned. For Example, in the following matrix, an entry is being made in front of P1 and below R3 since R3 is assigned to P1.

Process	R1	R2	R3
P1	0	0	1
P2	1	0	0
P3	0	1	0

### Request Matrix

In request matrix, an entry will be made for each of the resource requested. As in the following example, P1 needs R1 therefore an entry is being made in front of P1 and below R1.

Process	R1	R2	R3
P1	1	0	0
P2	0	1	0
P3	0	0	1

**Avial = (0,0,0)**

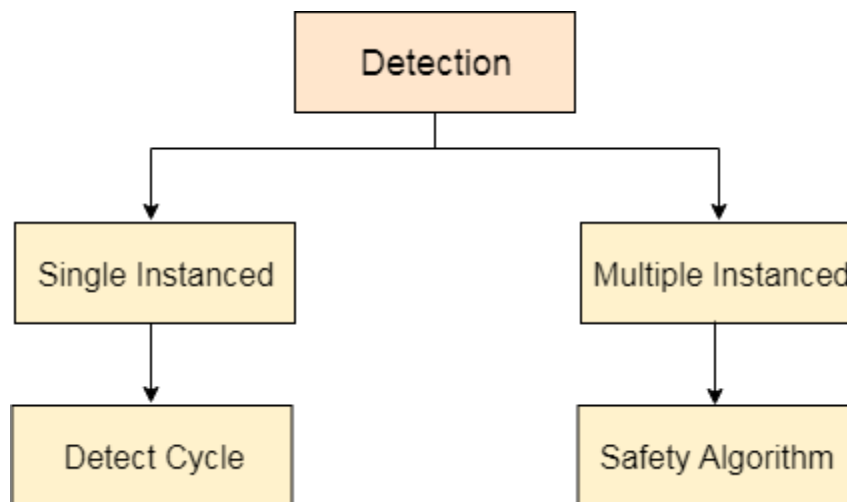
Neither we are having any resource available in the system nor a process going to release. Each of the process needs at least single resource to complete therefore they will continuously be holding each one of them.

We cannot fulfill the demand of at least one process using the available resources therefore the system is deadlocked as determined earlier when we detected a cycle in the graph.

## Deadlock Detection and Recovery

In this approach, The OS doesn't apply any mechanism to avoid or prevent the deadlocks. Therefore the system considers that the deadlock will definitely occur. In order to get rid of deadlocks, The OS periodically checks the system for any deadlock. In case, it finds any of the deadlock then the OS will recover the system using some recovery techniques.

The main task of the OS is detecting the deadlocks. The OS can detect the deadlocks with the help of Resource allocation graph.



In single instanced resource types, if a cycle is being formed in the system then there will definitely be a deadlock. On the other hand, in multiple instanced resource type graph, detecting a cycle is not just enough. We have to apply the safety algorithm on the system by converting the resource allocation graph into the allocation matrix and request matrix.

In order to recover the system from deadlocks, either OS considers resources or processes

## For Resource

### Preempt the resource

We can snatch one of the resources from the owner of the resource (process) and give it to the other process with the expectation that it will complete the execution and will release this resource sooner. Well, choosing a resource which will be snatched is going to be a bit difficult.

### Rollback to a safe state

System passes through various states to get into the deadlock state. The operating system can rollback the system to the previous safe state. For this purpose, OS needs to implement check pointing at every state.

The moment, we get into deadlock, we will rollback all the allocations to get into the previous safe state.

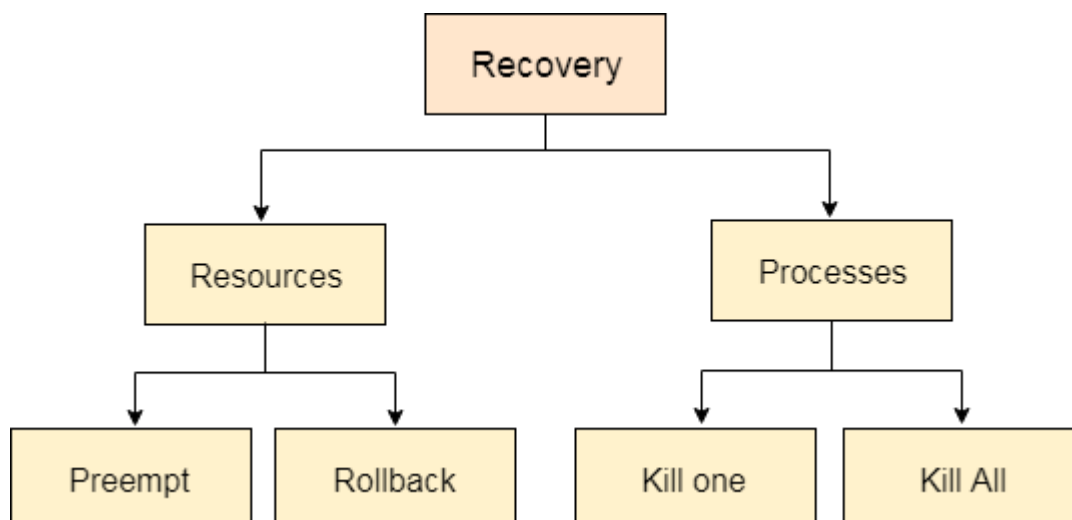
## For Process

### Kill a process

Killing a process can solve our problem but the bigger concern is to decide which process to kill. Generally, Operating system kills a process which has done least amount of work until now.

### Kill all process

This is not a suggestible approach but can be implemented if the problem becomes very serious. Killing all process will lead to inefficiency in the system because all the processes will execute again from starting.





## **Computer Memory:-**

Computer memory can be defined as a collection of some data represented in the binary format. A computer device that is capable to store any information or data temporally or permanently is called storage device.

## **Memory Management:-**

Memory management is the functionality of an operating system which handles or manages primary memory and moves processes back and forth between main memory and disk during execution. Memory management keeps track of each and every memory location, regardless of either it is allocated to some process or it is free. It checks how much memory is to be allocated to processes. It decides which process will get memory at what time. It tracks whenever some memory gets freed or unallocated and correspondingly it updates the status.

The memory management method deals with allocation of finite amount of memory to the requesting process. In ready state it is necessary that the process should have access to the certain amount of memory. Memory is a long array of bytes. Each with its own address using read and write statement the CPU and input/output device interact with the memory.



---

One of the main tasks of an operating system is to manage the computer's memory. This includes many responsibilities, including

- Being aware of what parts of the memory are in use and which parts are not.
- Allocating memory to processes when they request it and de-allocating memory when a process releases its memory.
- Moving data from memory to disc, when the physical capacity becomes full, and vice versa.

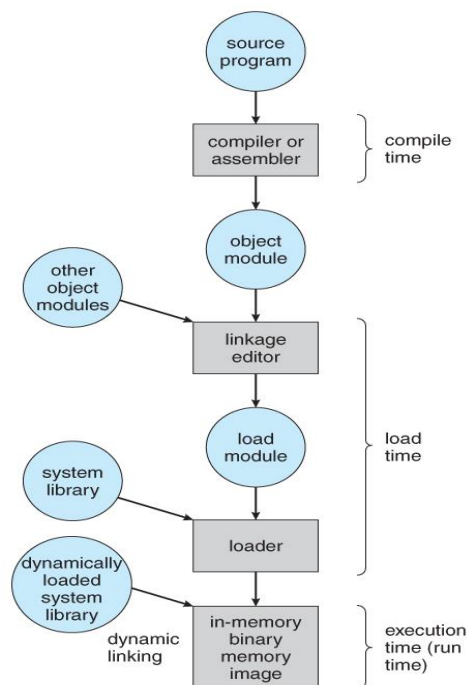
**Address:-** It is two types

1. **Logical address:-** Logical address are generated by the CPU. These address are defined by CPU while writing any program generated by CPU.
2. **Physical address:-** The address of memory area where the user program actually resides is called Physical address.

**Address Binding Mechanism:-** Memory consist of large array of words or bytes. Each with its own address. The processor is to select one of the process from the queue and load into memory as the process is executed. It access data and program from the memory.

User programs typically refer to memory addresses with symbolic names such as "i", "count", and "average Temperature". These symbolic names must be mapped or bound to physical memory addresses, which typically occurs in several stages:

Diagram shows the various stages of the binding processes and the units involved in each stage:



### Multistep processing of a user program

**Compile Time** - If it is known at compile time where a program will reside in physical memory, then absolute code can be generated by the compiler, containing actual physical addresses. However if the load address changes at some later time, then the program will have to be recompiled. DOS .COM programs use compile time binding.

**Load Time** - If the location at which a program will be loaded is not known at compile time, then the compiler must generate relocatable code, which references addresses relative to the start of the program. If that starting address changes, then the program must be reloaded but not recompiled.

**Execution Time** - If a program can be moved around in memory during the course of its execution, then binding must be delayed until execution time. This requires special hardware, and is the method implemented by most modern Operating System .

**Dynamic loading:-** Size of the process depend on the size of physical memory. Hence to obtain the better utilization of the memory space dynamic loading is performed. The major advantage of dynamic loading is that it never load any unused process. This method is useful while handling the large amount of code.

## **Swapping:-**

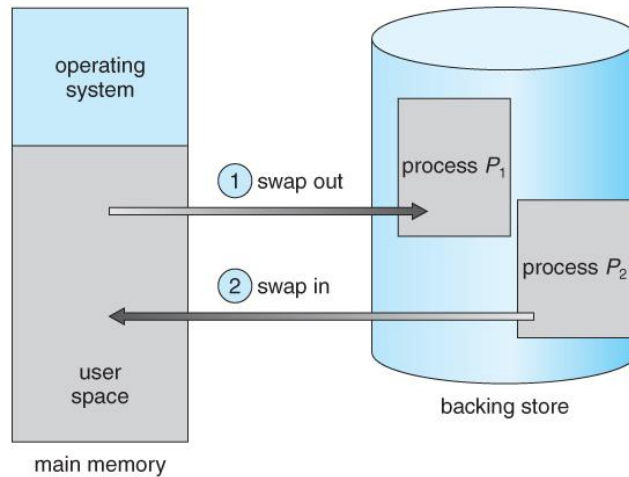
Swapping is a technique for making memory compact. It is a mechanism that is used to temporarily swap processes out of the main memory to secondary memory, and this makes more memory available for some other processes. At some later time, the system can swap back the process from the secondary memory to the main memory.

Swapping does affect the performance of the system, but it helps in running multiple processes parallelly. The total time taken by the swapping of a process includes the time it takes to move the entire process to the secondary memory and then again to the main memory.

It is a method of taking out the current content of memory to backstore(disk) and bring the content of backstore to main memory.

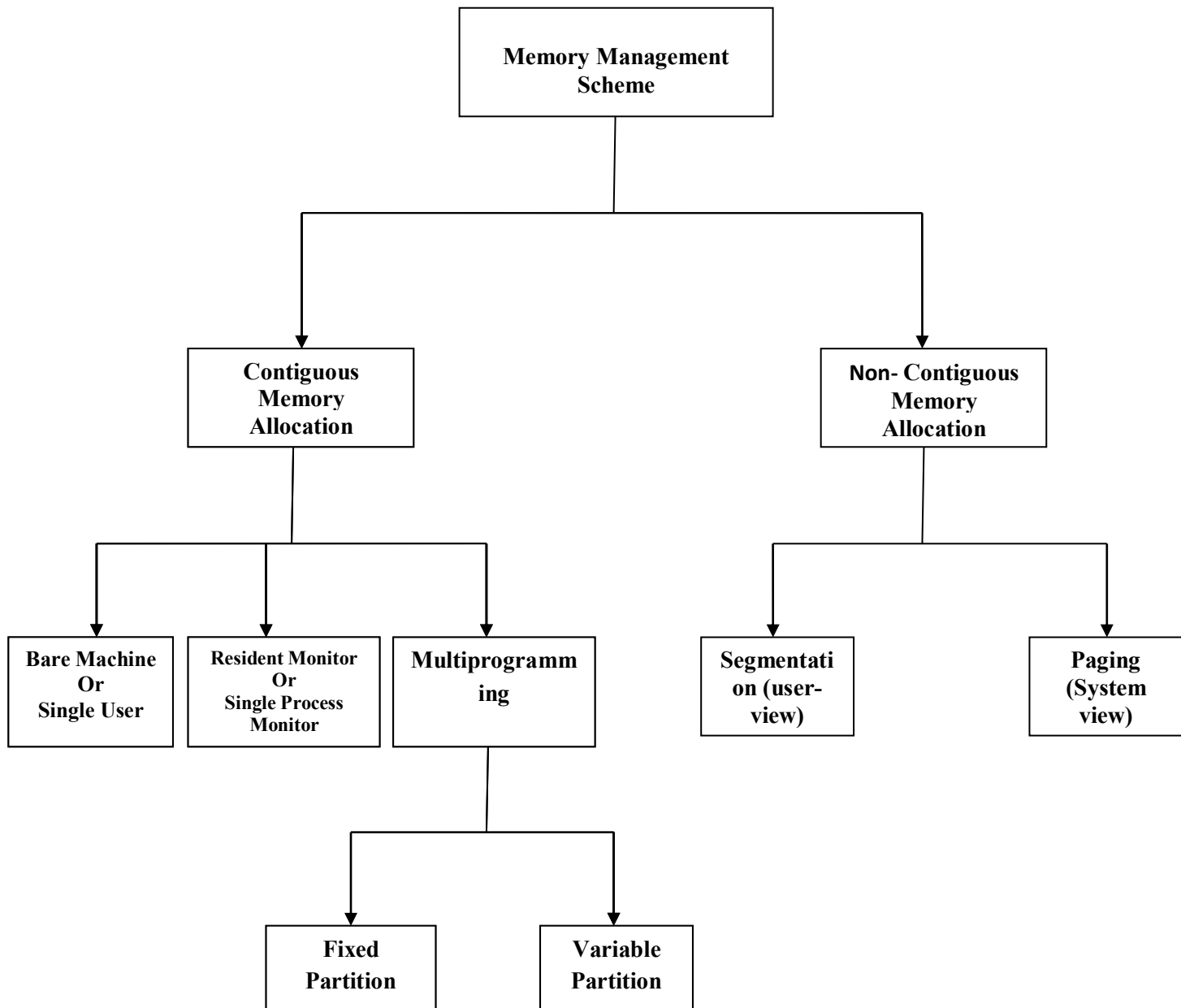
There are two operations in swapping method:-

1. Swap out(Read out):- take out to the current data from the main memory.
2. Swap in(Read in):- bring the data of new user into main memory.



Swapping of two processes using a disk as a backing store

### Memory Management Scheme:-



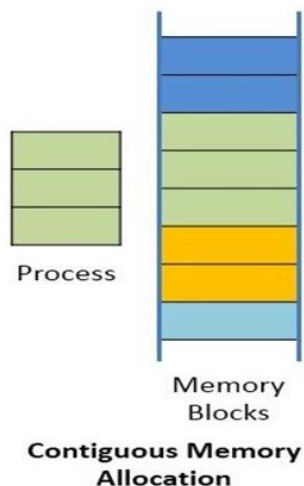
## Contiguous Memory Allocation:-

In **contiguous memory allocation**, all the available memory space remain together in one place. It means freely available memory partitions are not scattered here and there across the whole memory space.

In the **contiguous memory allocation**, both the operating system and the user must reside in the main memory. The main memory is divided into two portions one portion is for the operating and other is for the user program.

In the **contiguous memory allocation** when any user process request for the memory a single section of the contiguous memory block is given to that process according to its need. We can achieve contiguous memory allocation by dividing memory into the fixed-sized partition.

A single process is allocated in that fixed sized single partition. But this will increase the degree of multiprogramming means more than one process in the main memory that bounds the number of fixed partition done in memory. Internal fragmentation increases because of the contiguous memory allocation.

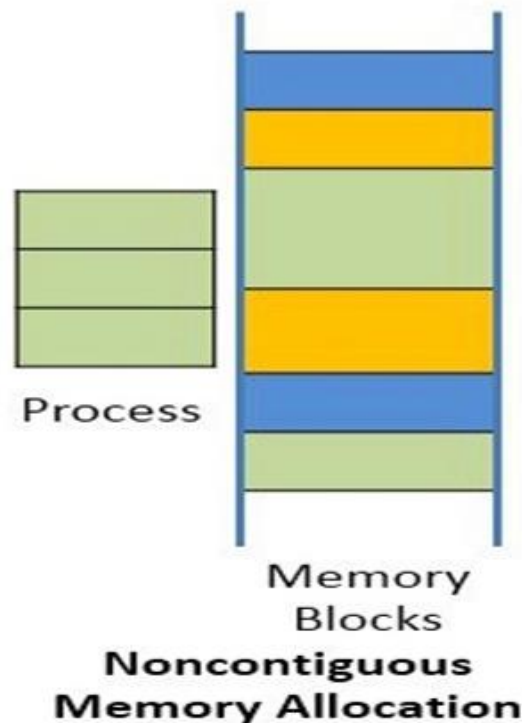


## Non-contiguous memory allocation:-

In the **non-contiguous memory allocation** the available free memory space are scattered here and there and all the free memory space is not at one place. So this is time-consuming.

In the **non-contiguous memory allocation**, a process will acquire the memory space but it is not at one place it is at the different locations according to the process requirement.

This technique of **non-contiguous memory allocation** reduces the wastage of memory which leads to internal and external fragmentation. This utilizes all the free memory space which is created by a different process.



### **Difference between Contiguous and Non-contiguous Memory Allocation :**

S.NO.	CONTIGUOUS MEMORY ALLOCATION	NON-CONTIGUOUS MEMORY ALLOCATION
1.	Contiguous memory allocation allocates consecutive blocks of memory to a file/process.	Non-Contiguous memory allocation allocates separate blocks of memory to a file/process.
2.	Faster in Execution.	Slower in Execution.
3.	It is easier for the OS to control.	It is difficult for the OS to control.
4.	Overhead is minimum as not much address translations are there while executing a process.	More Overheads are there as there are more address translations.
5.	Internal fragmentation occurs in Contiguous memory allocation method.	External fragmentation occurs in Non-Contiguous memory allocation method.

S.NO.	CONTIGUOUS MEMORY ALLOCATION	NON-CONTIGUOUS MEMORY ALLOCATION
6.	It includes single partition allocation and multi-partition allocation.	It includes paging and segmentation.
7.	Wastage of memory is there.	No memory wastage is there.
8.	In contiguous memory allocation, swapped-in processes are arranged in the originally allocated space.	In non-contiguous memory allocation, swapped-in processes can be arranged in any place in the memory.

### **Bare Machine:-**

Bare Machine is logic hardware in the computer system which can execute the programs in the processor without using the Operating System. Till now we have studied that we cannot execute any process inside the processor without the Operating System. But, with the Bare Machine, it is possible.

In the early days, before the Operating systems were developed, the instructions were executed directly on the hardware without any interfering software. But the only drawback was that the Bare Machine accepts the program and instructions in Machine Language. Due to this, only the trained people who were qualified in the computer field and were able to understand and instruct the computer in Machine language were able to operate on a computer. Due to this reason, the Bare Machine was termed as inefficient and cumbersome after the development of different Operating Systems.

It is one of the simplest memory method in which user has complete control over the memory.

The main advantage of bare machine is –

1. Flexible
2. No hardware and software support needed

### **Resident Monitor:-**

The Resident Monitor is a code which runs on Bare Machine. Its acts like an operating system which controls everything inside a processor and performs all the functions. The Resident Monitor is thus also known as the Job Sequencer because like the Operating system, it also sequences the jobs and sends it to

the processor for execution. After the jobs are scheduled, the Resident Monitor loads the Programs one by one into the main memory according to their sequence. The advantage of using a Resident Monitor over an Operating System is that there is no gap or lag between the program executions. So, the processing is faster in the Resident Monitors.

The Resident Monitors are divided into 3 parts:

1. **Control Language Interpreter**

The job of the Control Language Interpreter is to read and carry out the instructions line by line to the next level.

2. **Loader**

The Loader is the main part of the Resident Monitor. As the name suggests, it Loads all the required system and application programs into the main memory.

3. **Device Driver**

The Device Driver Takes care of all the Input-Output devices connected with the system. So, all the communication that takes place between the user and the system is handled by the Device Driver. It simply acts as an intermediate between the requests and the response, requests that are made by the user to the system, and they respond that the system produces to fulfill these requests.

## **Multiprogramming:-**

In the multiprogramming, the multiple users can share the memory simultaneously. By multiprogramming we mean there will be more than one process in the main memory and if the running process wants to wait for an event like I/O then instead of sitting idle CPU will make a context switch and will pick another process.

Multiprogramming are two types:-

1. Fixed (Static)
2. Variable (Dynamic)

**Fixed sized partition (Static) :-** In the fixed sized partition the system divides memory into fixed size partition (may or may not be of the same size) here entire partition is allowed to a process and if there is



some wastage inside the partition is allocated to a process and if there is some wastage inside the partition then it is called internal fragmentation.

In this technique, the main memory is divided into partitions of equal or different sizes. The operating system always resides in the first partition while the other partitions can be used to store user processes. The memory is assigned to the processes in contiguous way.

In fixed partitioning,

1. The partitions cannot overlap.
2. A process must be contiguously present in a partition for the execution.

There are various cons of using this technique.

### **1. Internal Fragmentation**

If the size of the process is lesser than the total size of the partition then some size of the partition get wasted and remain unused. This is wastage of the memory and called internal fragmentation.

As shown in the image below, the 4 MB partition is used to load only 3 MB process and the remaining 1 MB got wasted.

### **2. External Fragmentation**

The total unused space of various partitions cannot be used to load the processes even though there is space available but not in the contiguous form.

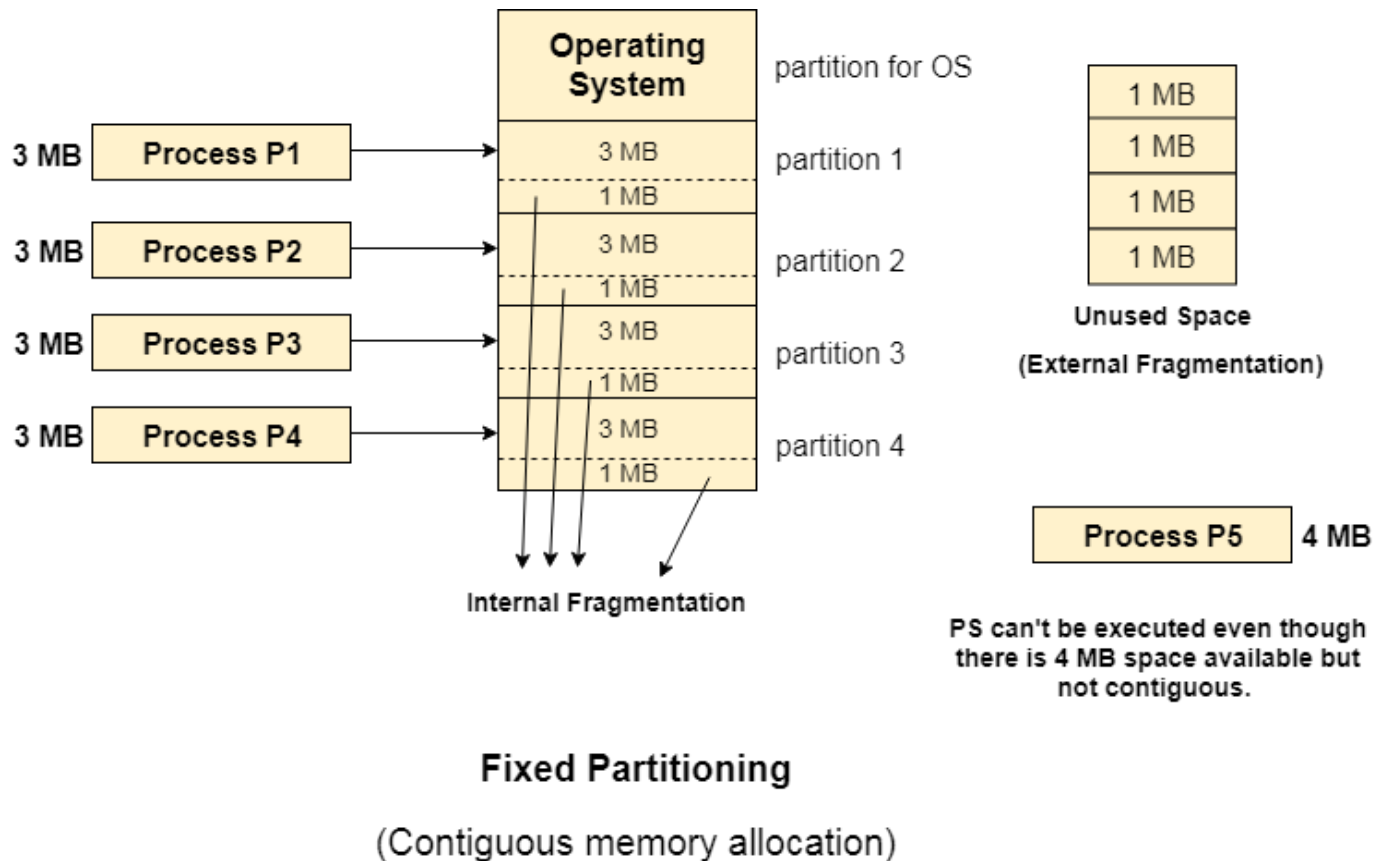
As shown in the image below, the remaining 1 MB space of each partition cannot be used as a unit to store a 4 MB process. Despite of the fact that the sufficient space is available to load the process, process will not be loaded.

### **3. Limitation on the size of the process**

If the process size is larger than the size of maximum sized partition then that process cannot be loaded into the memory. Therefore, a limitation can be imposed on the process size that is it cannot be larger than the size of the largest partition.

#### 4. Degree of multiprogramming is less

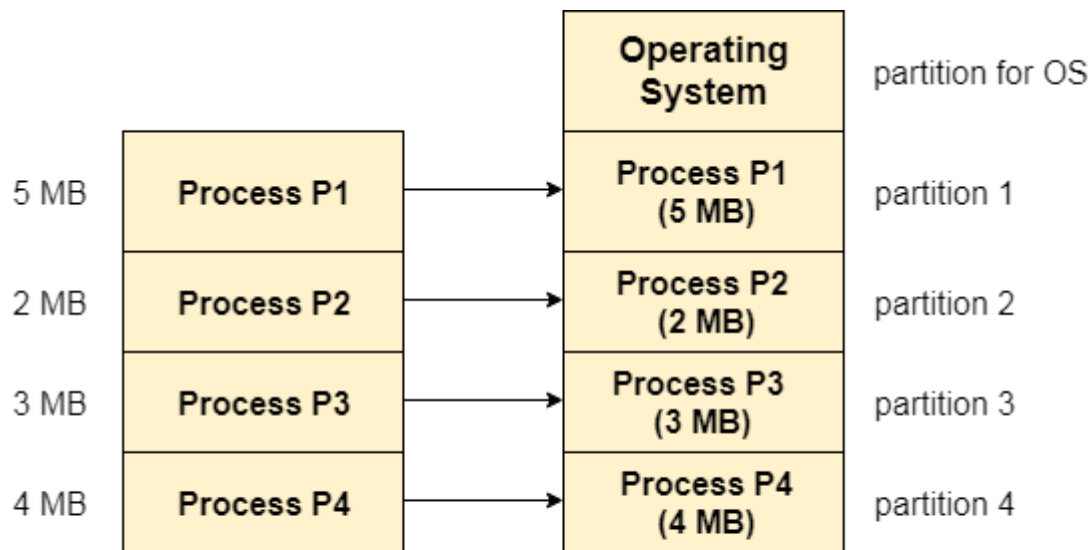
By Degree of multi programming, we simply mean the maximum number of processes that can be loaded into the memory at the same time. In fixed partitioning, the degree of multiprogramming is fixed and very less due to the fact that the size of the partition cannot be varied according to the size of processes.



#### Variable (Dynamic) Partitioning:-

Dynamic partitioning tries to overcome the problems caused by fixed partitioning. In this technique, the partition size is not declared initially. It is declared at the time of process loading.

The first partition is reserved for the operating system. The remaining space is divided into parts. The size of each partition will be equal to the size of the process. The partition size varies according to the need of the process so that the internal fragmentation can be avoided.



## Dynamic Partitioning

(Process Size = Partition Size)

### Advantages of Dynamic Partitioning over fixed partitioning

#### 1. No Internal Fragmentation

Given the fact that the partitions in dynamic partitioning are created according to the need of the process, It is clear that there will not be any internal fragmentation because there will not be any unused remaining space in the partition.

#### 2. No Limitation on the size of the process

In Fixed partitioning, the process with the size greater than the size of the largest partition could not be executed due to the lack of sufficient contiguous memory. Here, In Dynamic partitioning, the process size can't be restricted since the partition size is decided according to the process size.

#### 3. Degree of multiprogramming is dynamic

Due to the absence of internal fragmentation, there will not be any unused space in the partition hence more processes can be loaded in the memory at the same time.

### Disadvantages of dynamic partitioning

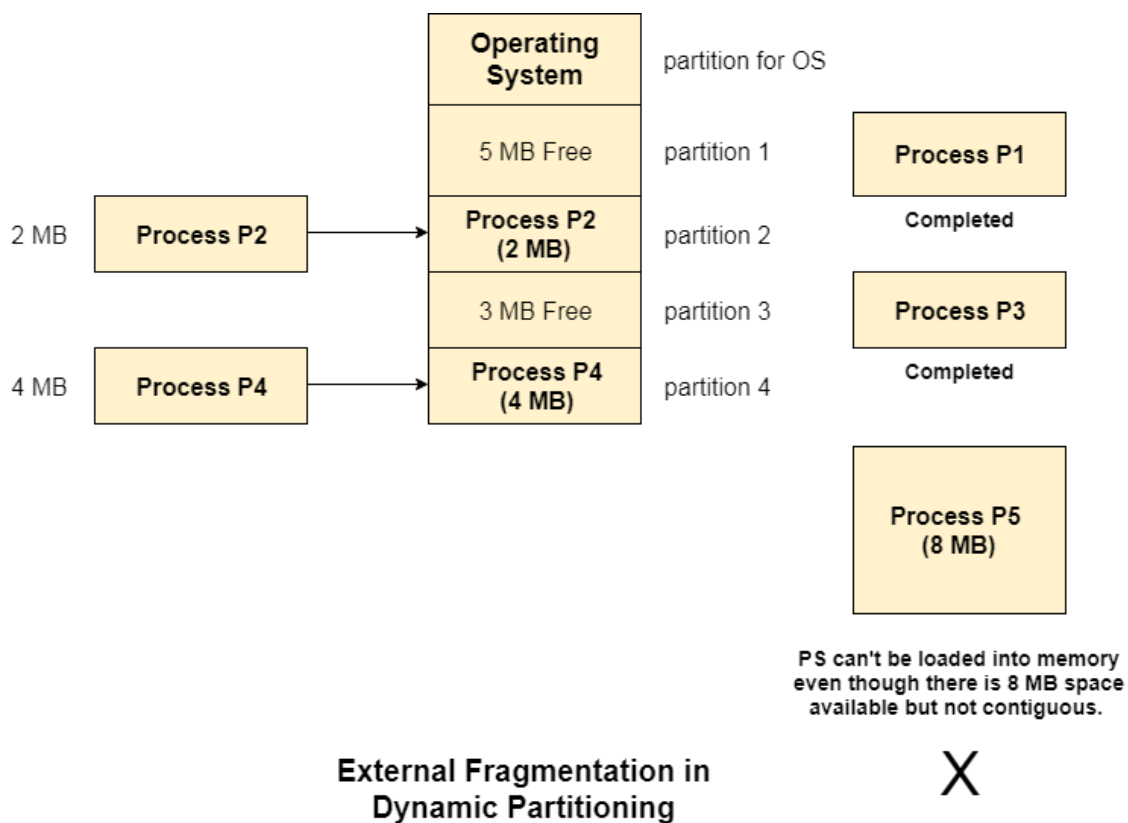
## External Fragmentation

Absence of internal fragmentation doesn't mean that there will not be external fragmentation.

Let's consider three processes P1 (1 MB) and P2 (3 MB) and P3 (1 MB) are being loaded in the respective partitions of the main memory.

After some time P1 and P3 got completed and their assigned space is freed. Now there are two unused partitions (1 MB and 1 MB) available in the main memory but they cannot be used to load a 2 MB process in the memory since they are not contiguously located.

The rule says that the process must be contiguously present in the main memory to get executed. We need to change this rule to avoid external fragmentation.



## Complex Memory Allocation

In Fixed partitioning, the list of partitions is made once and will never change but in dynamic partitioning, the allocation and deallocation is very complex since the partition size will be varied every time when it is assigned to a new process. OS has to keep track of all the partitions.

Due to the fact that the allocation and deallocation are done very frequently in dynamic memory allocation and the partition size will be changed at each time, it is going to be very difficult for OS to manage everything.

### Algorithm use for selection of a Free area of a memory:-

1. **First Fit:**

The first hole that is big enough is allocated to program.

2. **Best Fit:**

The smallest hole that is big enough is allocated to program.

3. **Worst Fit:**

The largest hole that is big enough is allocated to program.

**Q1. For the Partition of 200k, 500k, 300k, 600k (In order) place the process of 212k, 417k, 112k, 426k (In order) now fit the value according to the Best Fit algorithm.**

**Solution:-**

200k	500k	300k	600k
------	------	------	------

P1= 212k,

P2= 417k,

P3= 112k,

P4= 426k

	OS
200k	P3
500k	P2
300k	P1
600k	P4

		Fragment
		Size
P1	→ 212k → 300k	88k
P2	→ 417k → 500k	83k
P3	→ 112k → 200k	88k
P4	→ 426k → 600k	174k
Unused Space =		433k

**Q2. For the Partition of 100k, 500k, 200k, 300k, 600k (In order) place the process of 212k, 417k, 112k, 426k (In order) now fit the value according to the First Fit algorithm.**

**Solution:-**

212k	417k	112k	426k
------	------	------	------

P1      P2      P3      P4

	OS	
100k		
500k	P1 = 212k	P4=
200k	P3 = 112k	426k
300k		Wait
600k	P2 = 417k	

	Allocation Partitions	Fragment Size
P1 = 212k	500k	288k
P2 = 417k	600k	183k
P3 = 112K	200k	88k
P4 = 426k	Wait	Wait
	Unused Space	559k

**Q3. For the Partition of 100k, 500k, 200k, 300k, 600k (In order) place the process of 212k, 417k, 112k, 426k (In order) now fit the value according to the Worst Fit algorithm.**

**Solution:-**

OS	100k	500k	200k	300k	600k
----	------	------	------	------	------

**Worst Fit:-**

	100k	
<b>83k</b>	500k	<b>P2</b>
	200k	<b>P4 Wait</b>
<b>288k</b>	300k	<b>P3</b>
<b>388k</b>	600k	<b>P4</b>

$$\text{Unused Space} = 388 + 288 + 83$$

$$= 759k$$

**Memory Fragmentation:-** It is a drawback of memory management method.

### 1. Internal Fragmentation

If the size of the process is lesser than the total size of the partition then some size of the partition get wasted and remain unused. This is wastage of the memory and called internal fragmentation.

As shown in the image below, the 4 MB partition is used to load only 3 MB process and the remaining 1 MB got wasted.

### 2. External Fragmentation

The total unused space of various partitions cannot be used to load the processes even though there is space available but not in the contiguous form.

As shown in the image below, the remaining 1 MB space of each partition cannot be used as a unit to store a 4 MB process. Despite of the fact that the sufficient space is available to load the process, process will not be loaded.

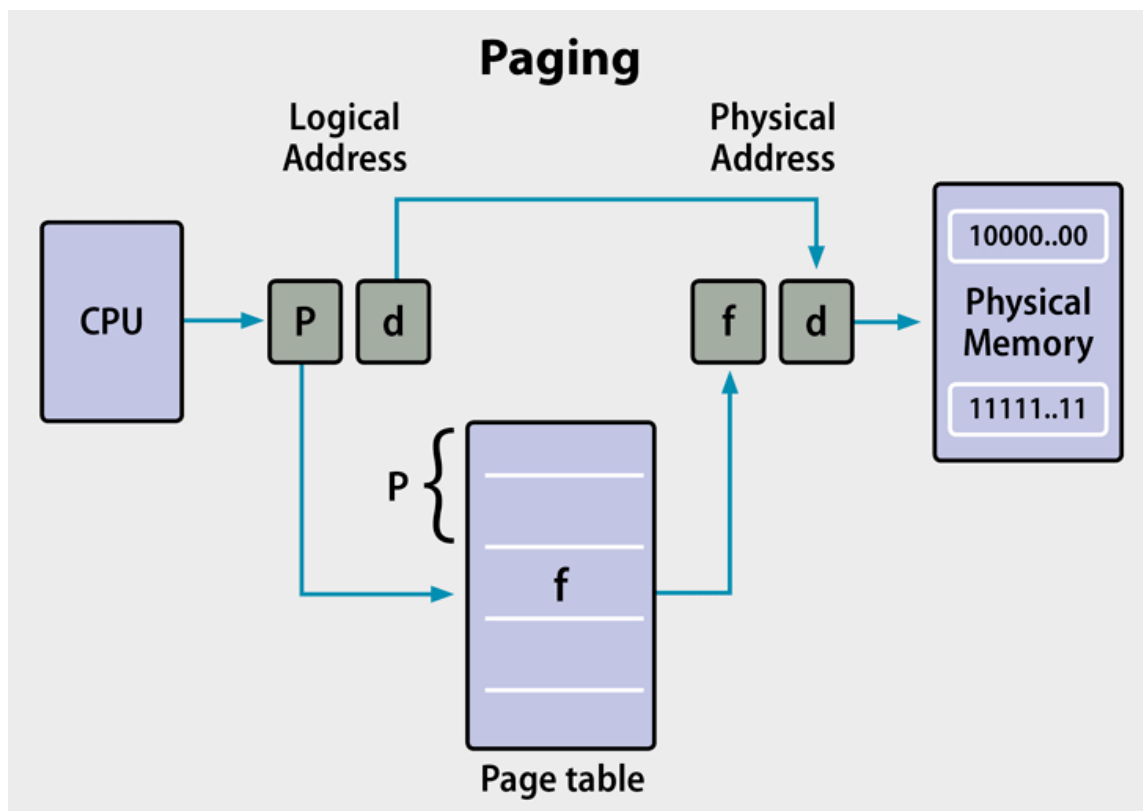
**Paging** :-As mentioned above, the memory management function called *paging* specifies storage locations to the CPU as additional memory, called virtual memory. The CPU cannot directly access storage disk, so the MMU emulates memory by mapping pages to frames that are in RAM.

Before we launch into a more detailed explanation of pages and frames, let's define some technical terms.

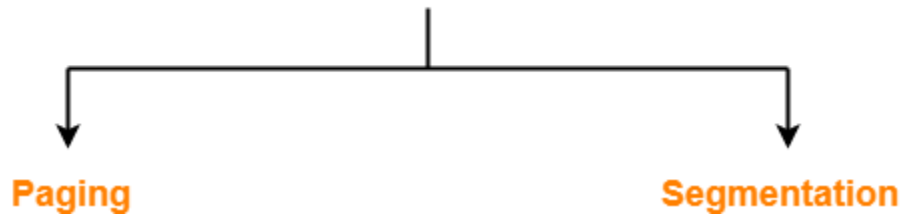
- **Page:** A fixed-length contiguous block of virtual memory residing on disk.



- **Frame:** A fixed-length contiguous block located in RAM; whose sizing is identical to pages.
- **Physical memory:** The computer's random access memory (RAM), typically contained in DIMM cards attached to the computer's motherboard.
- **Virtual memory:** Virtual memory is a portion of an HDD or SSD that is reserved to emulate RAM. The MMU serves up virtual memory from disk to the CPU to reduce the workload on physical memory.
- **Virtual address:** The CPU generates a virtual address for each active process. The MMU maps the virtual address to a physical location in RAM and passes the address to the bus. A virtual address space is the range of virtual addresses under CPU control.
- **Physical address:** The physical address is a location in RAM. The physical address space is the set of all physical addresses corresponding to the CPU's virtual addresses. A physical address space is the range of physical addresses under MMU control.



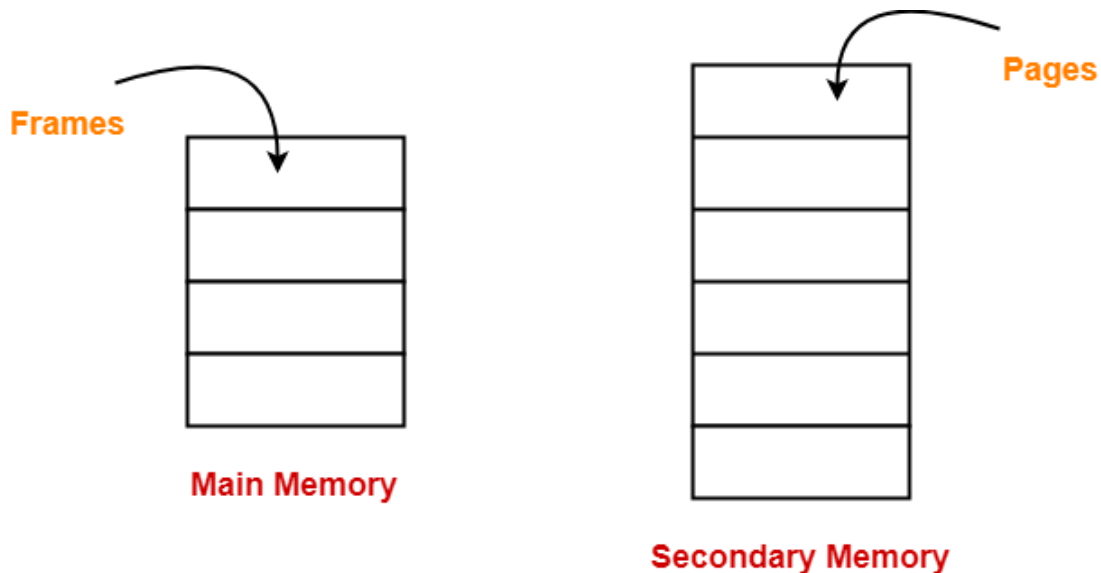
## Non-Contiguous Memory Allocation Techniques



### Paging:-

A solution to fragmentation problem is Paging. Paging is a memory management mechanism that allows the physical address space of a process to be non-contiguous. Here physical memory is divided into blocks of equal size called **Pages**. The pages belonging to a certain process are loaded into available memory frames.

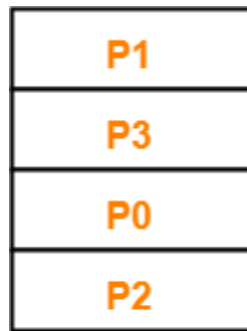
- Paging is a fixed size partitioning scheme.
- In paging, secondary memory and main memory are divided into equal fixed size partitions.
- The partitions of secondary memory are called as **pages**.
- The partitions of main memory are called as **frames**.



- Each process is divided into parts where size of each part is same as page size.
- The size of the last part may be less than the page size.
- The pages of process are stored in the frames of main memory depending upon their availability.

### Example-

- Consider a process is divided into 4 pages  $P_0$ ,  $P_1$ ,  $P_2$  and  $P_3$ .
- Depending upon the availability, these pages may be stored in the main memory frames in a non-contiguous fashion as shown-



**Main Memory**

### Translating Logical Address into Physical Address-

- CPU always generates a logical address.
- A physical address is needed to access the main memory.

Following steps are followed to translate logical address into physical address-

#### Step-01:

CPU generates a logical address consisting of two parts-

1. Page Number
2. Page Offset



- Page Number specifies the specific page of the process from which CPU wants to read the data.
- Page Offset specifies the specific word on the page that CPU wants to read.

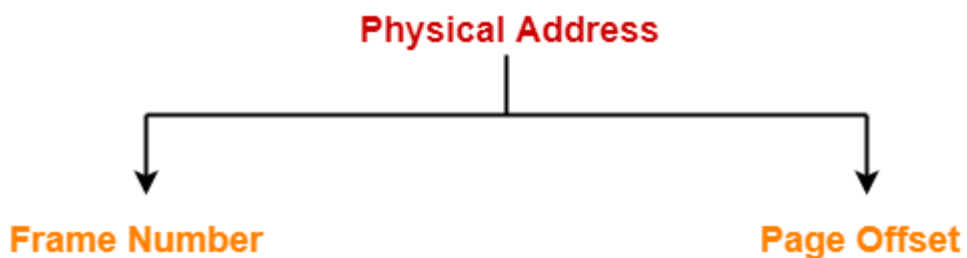
### **Step-02:**

For the page number generated by the CPU,

- **Page Table** provides the corresponding frame number (base address of the frame) where that page is stored in the main memory.

### **Step-03:**

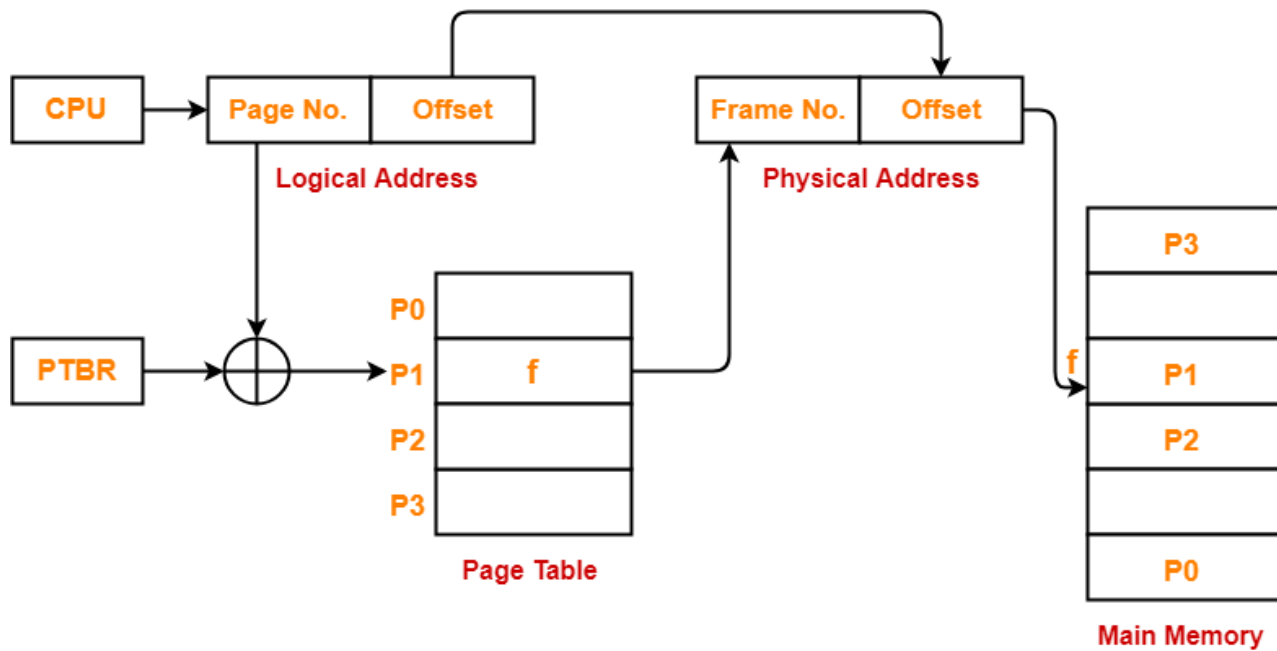
- The frame number combined with the page offset forms the required physical address.



- Frame number specifies the specific frame where the required page is stored.
- Page Offset specifies the specific word that has to be read from that page.

### Diagram-

The following diagram illustrates the above steps of translating logical address into physical address-



**Translating Logical Address into Physical Address**

### Advantages-

The advantages of paging are-

- It allows to store parts of a single process in a non-contiguous fashion.
- It solves the problem of external fragmentation.

### Disadvantages-

The disadvantages of paging are-

- It suffers from internal fragmentation.
- There is an overhead of maintaining a page table for each process.
- The times taken to fetch the instruction increases since now two memory accesses are required.

**Q- Why is page size always power of 2?**

**Solution:-** Recall that paging is implemented by breaking up an address into a page and offset number. It is most efficient to break the address into X page bits and Y offset bits, rather than perform arithmetic on the address to calculate the page number and offset. Because each bit position represents a power of 2, splitting an address between bits results in a page size that is a power of 2.

**Q- Consider a logical address of eight pages of 1024 words each mapped on to physical memory of 32 frames then find out**

**1. How many bits in the logical address?**

**2. How many bits in the physical address?**

**Solution 1:-**

Let No of bits in Physical memory is m

Then size of physical memory is  $2^m$

No of pages = 8 =  $2^3$

No of frames = 32 =  $2^5$

Size of each frame = Size of each page =  $2^{10}$

No of Frame =  $\frac{\text{Physical memory}}{\text{Size of page}}$

Frame(f) =  $\frac{m}{p}$

$2^5 = \frac{2^m}{2^{10}}$

$2^m = 2^5 \times 2^{10}$

m = 15

m is Physical address

**Solution 2:-**

Let No of bits in logical memory bits is n

Then size of logical memory is  $2^n$

No of Pages =  $\frac{\text{Size of logical memory}}{\text{Size of each page}}$

$$2^3 = \frac{2^n}{2^{10}}$$

n = 13

n = logical address

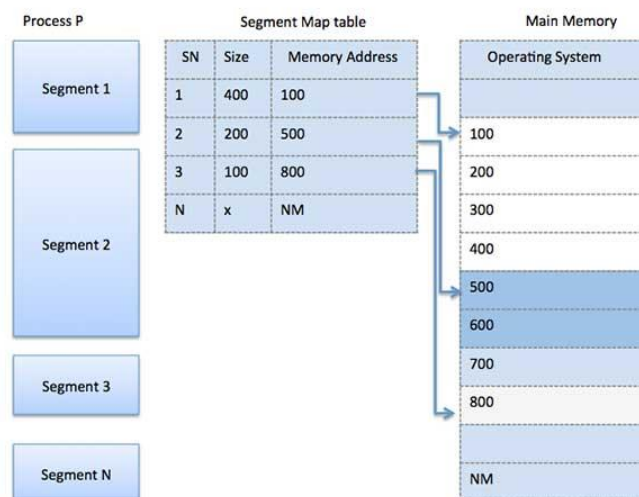
## Segmentation:-

Segmentation is a memory management technique in which each job is divided into several segments of different sizes, one for each module that contains pieces that perform related functions. Each segment is actually a different logical address space of the program.

When a process is to be executed, its corresponding segmentation are loaded into non-contiguous memory though every segment is loaded into a contiguous block of available memory.

Segmentation memory management works very similar to paging but here segments are of variable-length where as in paging pages are of fixed size.

A program segment contains the program's main function, utility functions, data structures, and so on. The operating system maintains a **segment map table** for every process and a list of free memory blocks along with segment numbers, their size and corresponding memory locations in main memory. For each segment, the table stores the starting address of the segment and the length of the segment. A reference to a memory location includes a value that identifies a segment and an offset.



- Like Paging, Segmentation is another non-contiguous memory allocation technique.

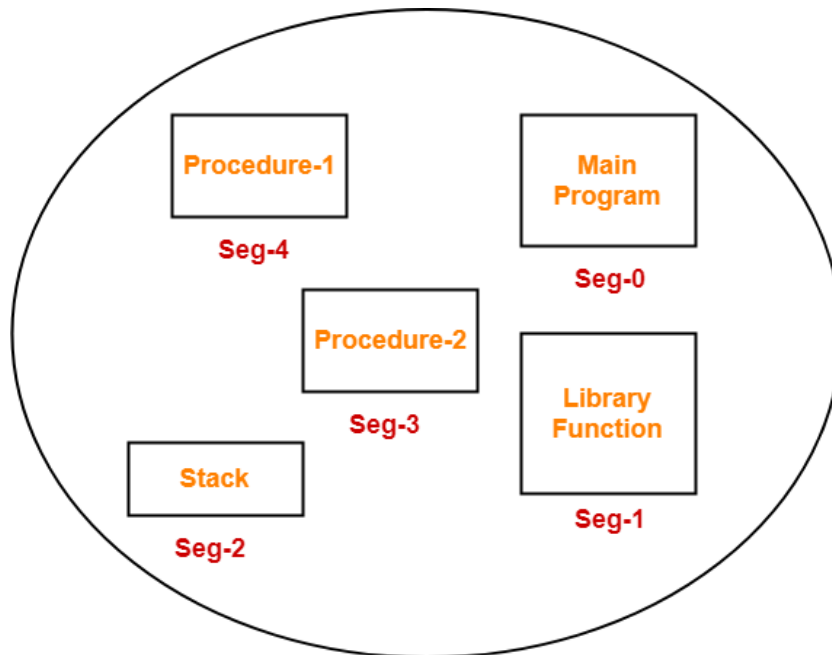
- In segmentation, process is not divided blindly into fixed size pages.
- Rather, the process is divided into modules for better visualization.

### Characteristics-

- Segmentation is a variable size partitioning scheme.
- In segmentation, secondary memory and main memory are divided into partitions of unequal size.
- The size of partitions depend on the length of modules.
- The partitions of secondary memory are called as **segments**.

### Example-

Consider a program is divided into 5 segments as-



### Segment Table-



- Segment table is a table that stores the information about each segment of the process.
- It has two columns.
- First column stores the size or length of the segment.
- Second column stores the base address or starting address of the segment in the main memory.
- Segment table is stored as a separate segment in the main memory.
- Segment table base register (STBR) stores the base address of the segment table.

For the above illustration, consider the segment table is-

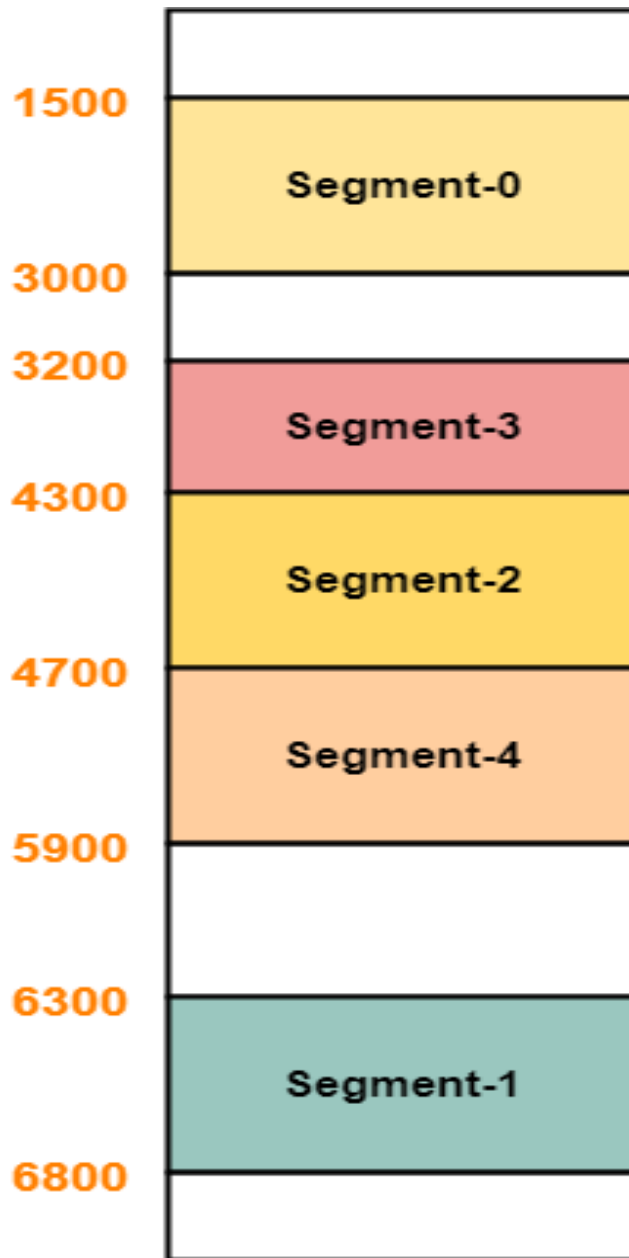
	<b>Limit</b>	<b>Base</b>
<b>Seg-0</b>	<b>1500</b>	<b>1500</b>
<b>Seg-1</b>	<b>500</b>	<b>6300</b>
<b>Seg-2</b>	<b>400</b>	<b>4300</b>
<b>Seg-3</b>	<b>1100</b>	<b>3200</b>
<b>Seg-4</b>	<b>1200</b>	<b>4700</b>

**Segment Table**

Here,

- Limit indicates the length or size of the segment.
- Base indicates the base address or starting address of the segment in the main memory.

In accordance to the above segment table, the segments are stored in the main memory as-



## **Main Memory**

### **Translating Logical Address into Physical Address-**

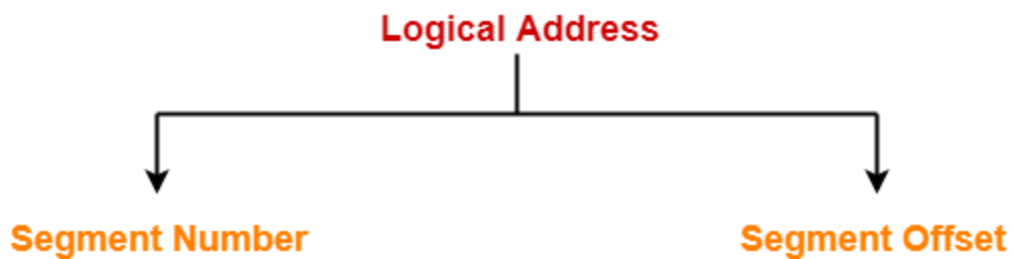
- CPU always generates a logical address.
- A physical address is needed to access the main memory.

Following steps are followed to translate logical address into physical address-

Step-01:

CPU generates a logical address consisting of two parts-

1. Segment Number
2. Segment Offset



- Segment Number specifies the specific segment of the process from which CPU wants to read the data.
- Segment Offset specifies the specific word in the segment that CPU wants to read.

Step-02:

- For the generated segment number, corresponding entry is located in the segment table.
- Then, segment offset is compared with the limit (size) of the segment.

Now, two cases are possible-

Case-01: Segment Offset  $\geq$  Limit

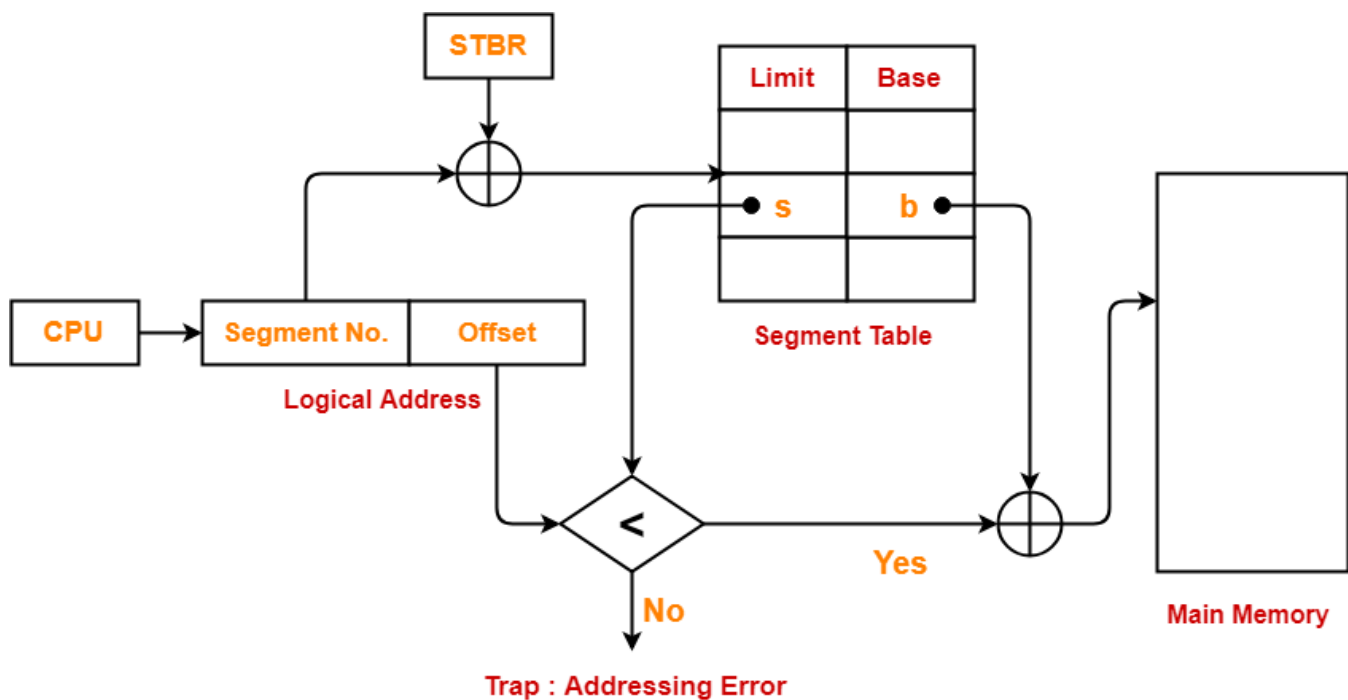
- If segment offset is found to be greater than or equal to the limit, a trap is generated.

### Case-02: Segment Offset < Limit

- If segment offset is found to be smaller than the limit, then request is treated as a valid request.
- The segment offset must always lie in the range  $[0, \text{limit}-1]$ ,
- Then, segment offset is added with the base address of the segment.
- The result obtained after addition is the address of the memory location storing the required word.

### Diagram-

The following diagram illustrates the above steps of translating logical address into physical address-



**Translating Logical Address into Physical Address**

### Advantages-

The advantages of segmentation are-

- It allows to divide the program into modules which provides better visualization.
- Segment table consumes less space as compared to **Page Table** in paging.
- It solves the problem of internal fragmentation.

### Disadvantages-

The Dis-advantages of segmentation are-

- There is an overhead of maintaining a segment table for each process.
- The time taken to fetch the instruction increases since now two memory accesses are required.
- Segments of unequal size are not suited for swapping.
- It suffers from external fragmentation as the free space gets broken down into smaller pieces with the processes being loaded and removed from the main memory.

Paging VS Segmentation

S.r. No.	Paging	Segmentation
1	Non-Contiguous memory allocation	Non-contiguous memory allocation
2	Paging divides program into fixed size pages.	Segmentation divides program into variable size segments.
3	OS is responsible	Compiler is responsible.
4	Paging is faster than segmentation	Segmentation is slower than paging
5	Paging is closer to Operating System	Segmentation is closer to User

6	It suffers from internal fragmentation	It suffers from external fragmentation
7	There is no external fragmentation	There is no external fragmentation
8	Logical address is divided into page number and page offset	Logical address is divided into segment number and segment offset
9	Page table is used to maintain the page information.	Segment Table maintains the segment information
10	Page table entry has the frame number and some flag bits to represent details about pages.	Segment table entry has the base address of the segment and some protection bits for the segments.

**Q1. Consider the following Segment table:-**

Segment	Base	Length
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

**What are the Physical address of following logical address**

1. 0430,
2. 110
3. 2500
4. 3400
5. 4112

**Solution 1 :-** In 0430 first digit 0 refer the segment value and 430 refer the offset value

So, Physical address= Base + Offset

$$= 219 + 430$$

$$= 649$$

**Solution 2 :-** In 110 first digit 1 refer the segment value and 10 refer the offset value

96 < 112

So, Physical address = Base + Offset

$$= 2300 + 10$$

$$= 2310$$

**Solution 3 :-** In 2500 first digit 2 refer the segment value and 500 refer the offset value

So, Physical address = Base + Offset

$$= 90 + 500$$

$$= 590$$

**Solution 4 :-** In 3400 first digit 3 refer the segment value and 400 refer the offset value

So, Physical address = Base + Offset

$$= 1327 + 400$$

$$= 1727$$

**Solution 5 :-** In 4112 first digit 4 refer the segment value and 112 refer the offset value

So, Physical address = Base + Offset

$$= 1952 + 112$$

$$= 2064$$

### **Paged Segmentation:-**

Pure segmentation is not very popular and not being used in many of the operating systems. However, Segmentation can be combined with Paging to get the best features out of both the techniques.

- Segmented Paging is a scheme that implements the combination of **Segmentation** and **Paging**.
- First, segmentation divides the process into segments.
- Then, paging divides each segment into pages.

In Segmented Paging, the main memory is divided into variable size segments which are further divided into fixed size pages.

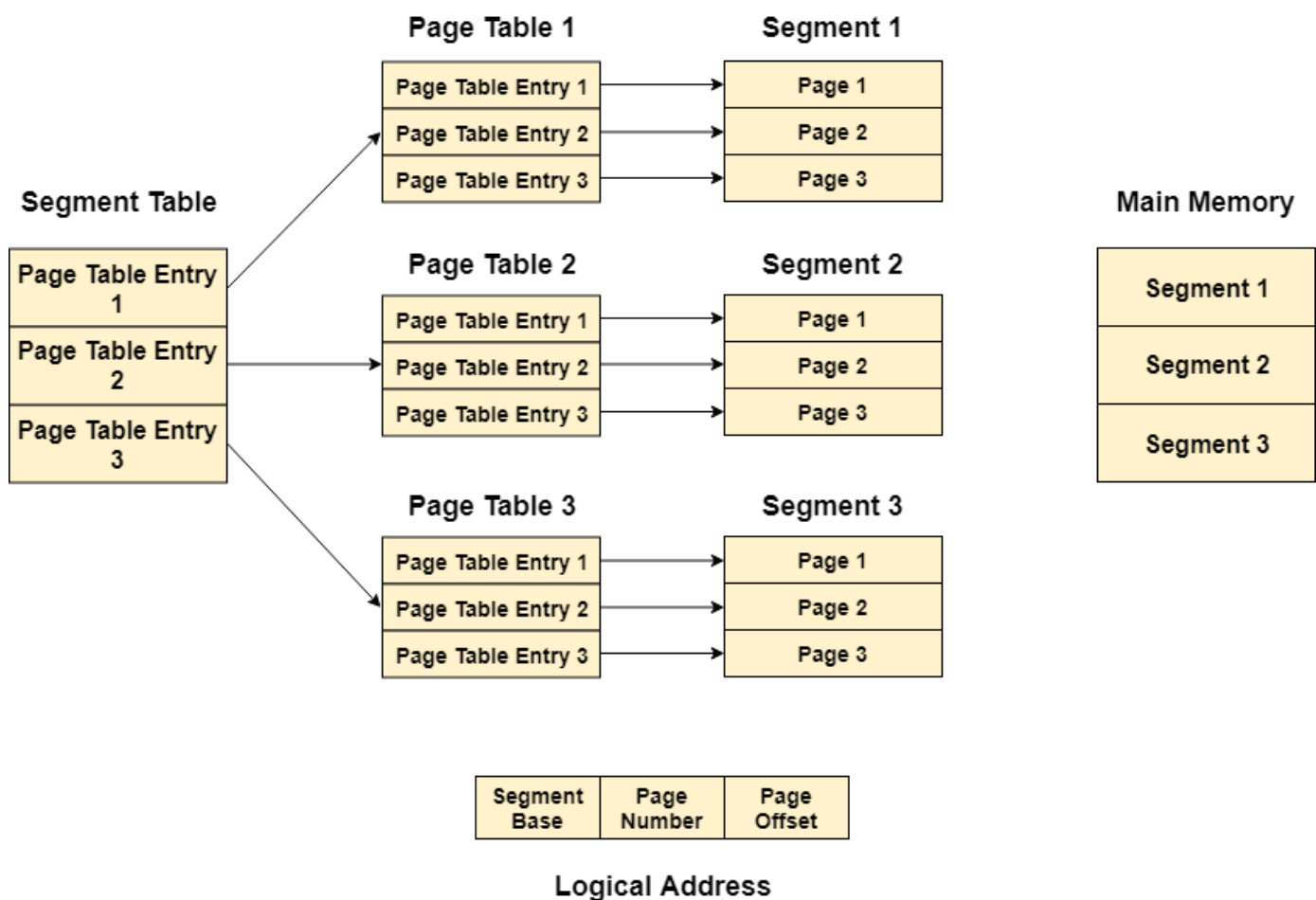
1. Pages are smaller than segments.
2. Each Segment has a page table which means every program has multiple page tables.
3. The logical address is represented as Segment Number (base address), Page number and page offset.

**Segment Number** → It points to the appropriate Segment Number.

**Page Number** → It Points to the exact page within the segment

**Page Offset** → Used as an offset within the page frame

Each Page table contains the various information about every page of the segment. The Segment Table contains the information about every segment. Each segment table entry points to a page table entry and every page table entry is mapped to one of the page within a segment.

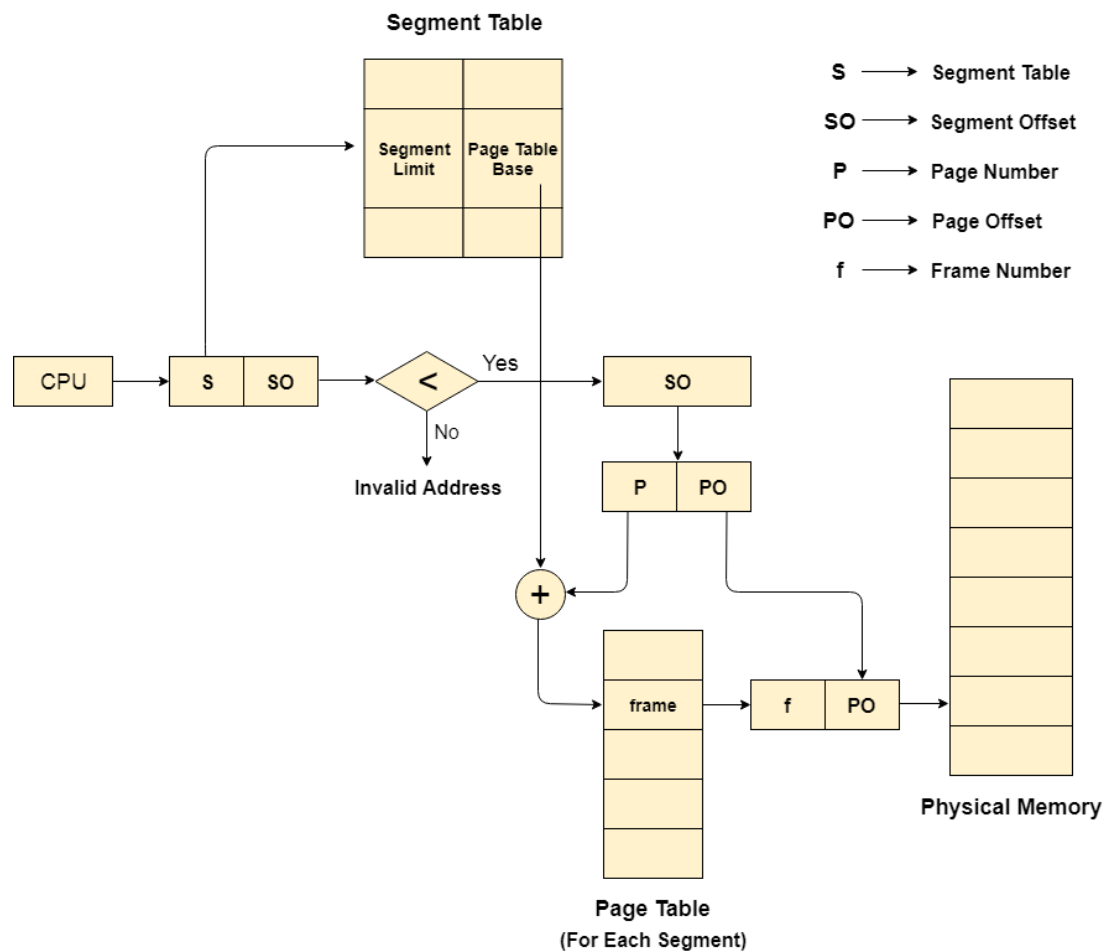


### Translation of logical address to physical address:-

The CPU generates a logical address which is divided into two parts: Segment Number and Segment Offset. The Segment Offset must be less than the segment limit. Offset is further divided into Page number and Page Offset. To map the exact page number in the page table, the page number is added into the page table base.



The actual frame number with the page offset is mapped to the main memory to get the desired word in the page of the certain segment of the process.



### Advantages of Segmented Paging

1. It reduces memory usage.
2. Page table size is limited by the segment size.
3. Segment table has only one entry corresponding to one actual segment.
4. External Fragmentation is not there.
5. It simplifies memory allocation.

## **Disadvantages of Segmented Paging**

1. Internal Fragmentation will be there.
2. The complexity level will be much higher as compare to paging.
3. Page Tables need to be contiguously stored in the memory.

### **Problem-01:**

A certain computer system has the segmented paging architecture for virtual memory. The memory is byte addressable. Both virtual and physical address spaces contain  $2^{16}$  bytes each. The virtual address space is divided into 8 non-overlapping equal size segments. The memory management unit (MMU) has a hardware segment table, each entry of which contains the physical address of the page table for the segment. Page tables are stored in the main memory and consists of 2 byte page table entries. What is the minimum page size in bytes so that the page table for a segment requires at most one page to store it?

### **Solution-**

Given-

- Virtual Address Space = Process size =  $2^{16}$  bytes
- Physical Address Space = Main Memory size =  $2^{16}$  bytes
- Process is divided into 8 equal size segments
- Page table entry size = 2 bytes

Let page size = n bytes.

Now, since page table has to be stored into a single page, so we must have-

$$\text{Size of page table} \leq \text{Page size}$$

### **Size of Each Segment-**

Size of each segment

$$= \text{Process size} / \text{Number of segments}$$

$$\begin{aligned}
&= 2^{16} \text{ bytes} / 8 \\
&= 2^{16} \text{ bytes} / 2^3 \\
&= 2^{13} \text{ bytes} \\
&= 8 \text{ KB}
\end{aligned}$$

### **Number of Pages Of Each Segment-**

Number of pages each segment is divided

$$\begin{aligned}
&= \text{Size of segment} / \text{Page size} \\
&= 8 \text{ KB} / n \text{ bytes} \\
&= (8K / n) \text{ pages}
\end{aligned}$$

### **Size of Each Page Table-**

Size of each page table

$$\begin{aligned}
&= \text{Number of entries in page table} \times \text{Page table entry size} \\
&= \text{Number of pages the segment is divided} \times 2 \text{ bytes} \\
&= (8K / n) \times 2 \text{ bytes} \\
&= (16K / n) \text{ bytes}
\end{aligned}$$

### **Page Size-**

Substituting values in the above condition, we get-

$$\begin{aligned}
(16K / n) \text{ bytes} &\leq n \text{ bytes} \\
(16K / n) &\leq n \\
n^2 &\geq 16K \\
n^2 &\geq 2^{14} \\
n &\geq 2^7
\end{aligned}$$

Thus, minimum page size possible =  $2^7$  bytes = 128 bytes.

## Virtual Memory:-

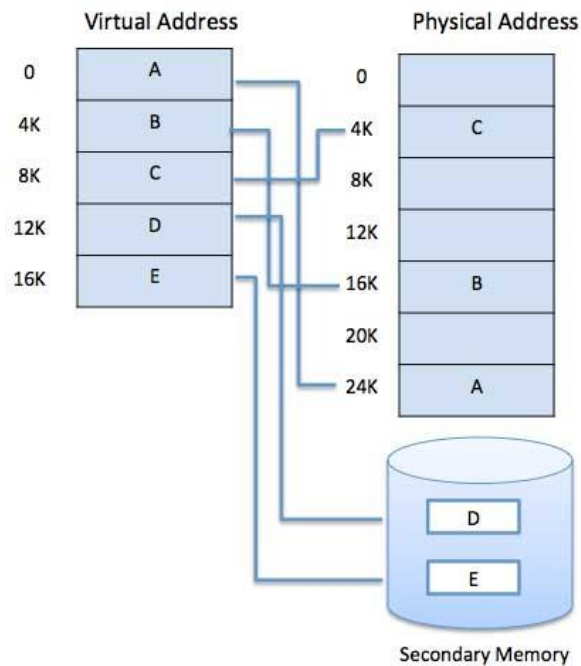
A computer can address more memory than the amount physically installed on the system. This extra memory is actually called **virtual memory** and it is a section of a hard disk that's set up to emulate the computer's RAM.

The main visible advantage of this scheme is that programs can be larger than physical memory. Virtual memory serves two purposes. First, it allows us to extend the use of physical memory by using disk. Second, it allows us to have memory protection, because each virtual address is translated to a physical address.

Following are the situations, when entire program is not required to be loaded fully in main memory.

- User written error handling routines are used only when an error occurred in the data or computation.
- Certain options and features of a program may be used rarely.
- Many tables are assigned a fixed amount of address space even though only a small amount of the table is actually used.
- The ability to execute a program that is only partially in memory would counter many benefits.
- Less number of I/O would be needed to load or swap each user program into memory.
- A program would no longer be constrained by the amount of physical memory that is available.
- Each user program could take less physical memory, more programs could be run the same time, with a corresponding increase in CPU utilization and throughput.

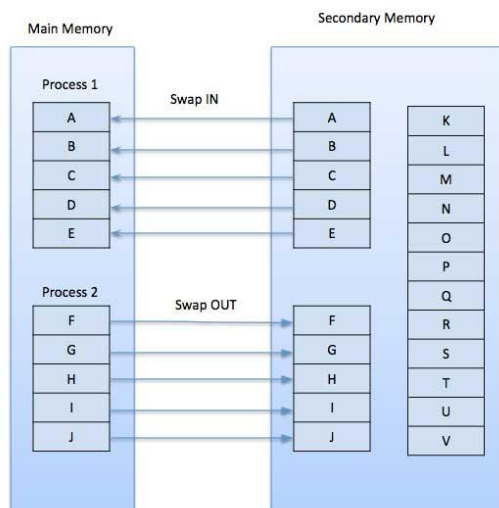
Modern microprocessors intended for general-purpose use, a memory management unit, or MMU, is built into the hardware. The MMU's job is to translate virtual addresses into physical addresses. A basic example is given below –



Virtual memory is commonly implemented by demand paging. It can also be implemented in a segmentation system. Demand segmentation can also be used to provide virtual memory.

### Demand Paging:-

A demand paging system is quite similar to a paging system with swapping where processes reside in secondary memory and pages are loaded only on demand, not in advance. When a context switch occurs, the operating system does not copy any of the old program's pages out to the disk or any of the new program's pages into the main memory. Instead, it just begins executing the new program after loading the first page and fetches that program's pages as they are referenced.



While executing a program, if the program references a page which is not available in the main memory because it was swapped out a little ago, the processor treats this invalid memory reference as a **page fault** and transfers control from the program to the operating system to demand the page back into the memory.

### Advantages

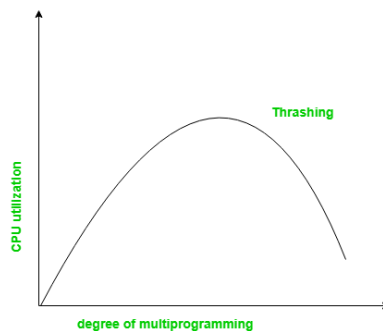
Following are the advantages of Demand Paging –

- Large virtual memory.
- More efficient use of memory.
- There is no limit on degree of multiprogramming.

### Disadvantages

- Number of tables and the amount of processor overhead for handling page interrupts are greater than in the case of the simple paged management techniques.

**Thrashing:-** Thrashing is a condition or a situation when the system is spending a major portion of its time in servicing the page faults, but the actual processing done is very negligible.



The basic concept involved is that if a process is allocated too few frames, then there will be too many and too frequent page faults. As a result, no useful work would be done by the CPU and the CPU utilization would fall drastically. The long-term scheduler would then try to improve the CPU utilization by loading some more processes into the memory thereby increasing the degree of multiprogramming. This would result in a further decrease in the CPU utilization triggering a chained reaction of higher page faults followed by an increase in the degree of multiprogramming, called Thrashing.

# OPERATING SYSTEM

## UNIT- 4

### File System

File system is the part of the operating system which is responsible for file management. It provides a mechanism to store the data and access to the file contents including data and programs. Some Operating systems treats everything as a file for example Ubuntu.

A file is a collection of correlated information which is recorded on secondary or non-volatile storage like magnetic disks, optical disks, and tapes. It is a method of data collection that is used as a medium for giving input and receiving output from that program.

In general, a file is a sequence of bits, bytes, or records whose meaning is defined by the file creator and user. Every File has a logical location where they are located for storage and retrieval.

The File system takes care of the following issues

- o **File Structure**

We have seen various data structures in which the file can be stored. The task of the file system is to maintain an optimal file structure.

- o **Recovering Free space**

Whenever a file gets deleted from the hard disk, there is a free space created in the disk. There can be many such spaces which need to be recovered in order to reallocate them to other files.

- o **Disk space assignment to the files**

The major concern about the file is deciding where to store the files on the hard disk. There are various disks scheduling algorithm.

- o **Tracking data location**

A File may or may not be stored within only one block. It can be stored in the non contiguous blocks on the disk. We need to keep track of all the blocks on which the part of the files reside.

## File Access Methods in Operating System

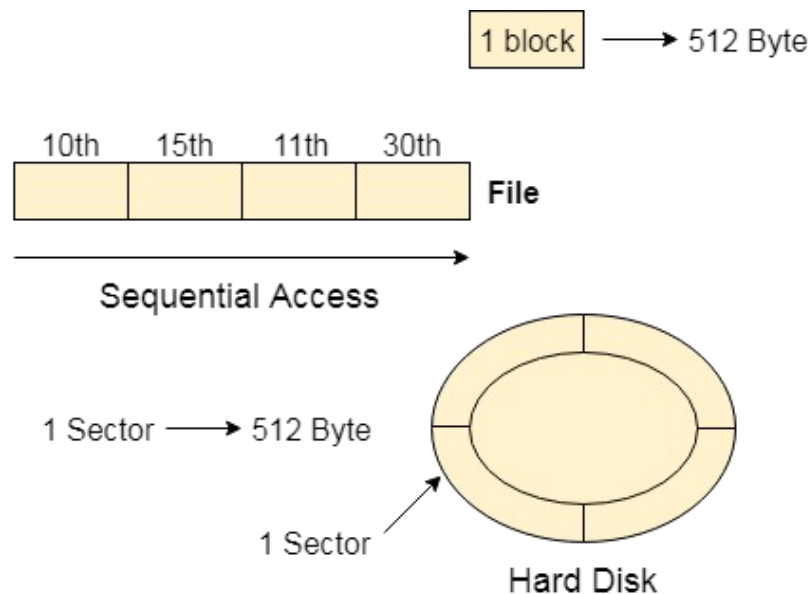
When a file is used, information is read and accessed into computer memory and there are several ways to access this information of the file. Some systems provide only one access method for files. Other systems, such as those of IBM, support many access methods, and choosing the right one for a particular application is a major design problem.

There are three ways to access a file into a computer system: Sequential-Access, Direct Access, Index sequential Method.

### Sequential Access –

It is the simplest access method. Information in the file is processed in order, one record after the other. This mode of access is by far the most common; for example, editor and compiler usually access the file in this fashion.

Read and write make up the bulk of the operation on a file. A read operation *-read next-* read the next position of the file and automatically advance a file pointer, which keeps track I/O location. Similarly, for the write *write next* append to the end of the file and advance to the newly written material.



Most of the operating systems access the file sequentially. In other words, we can say that most of the files need to be accessed sequentially by the operating system.



In sequential access, the OS read the file word by word. A pointer is maintained which initially points to the base address of the file. If the user wants to read first word of the file then the pointer provides that word to the user and increases its value by 1 word. This process continues till the end of the file.

**Key points:**

- Data is accessed one record right after another record in an order.
- When we use read command, it move ahead pointer by one
- When we use write command, it will allocate memory and move the pointer to the end of the file
- Such a method is reasonable for tape.

**Direct Access –**

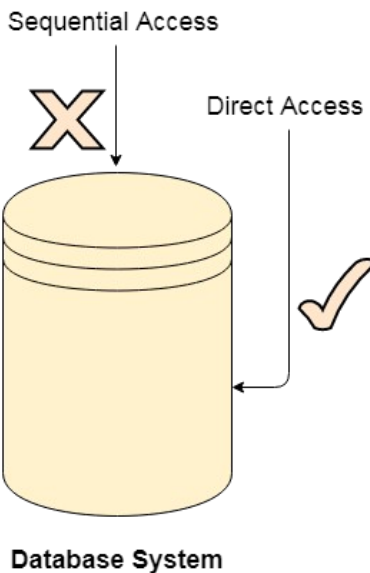
Another method is *direct access method* also known as *relative access method*. A fixed-length logical record that allows the program to read and write record rapidly. in no particular order. The direct access is based on the disk model of a file since disk allows random access to any file block. For direct access, the file is viewed as a numbered sequence of block or record. Thus, we may read block 14 then block 59 and then we can write block 17. There is no restriction on the order of reading and writing for a direct access file.

A block number provided by the user to the operating system is normally a *relative block number*, the first relative block of the file is 0 and then 1 and so on.

The Direct Access is mostly required in the case of database systems. In most of the cases, we need filtered information from the database. The sequential access can be very slow and inefficient in such cases.

Suppose every block of the storage stores 4 records and we know that the record we needed is stored in 10th block. In that case, the sequential access will not be implemented because it will traverse all the blocks in order to access the needed record.

Direct access will give the required result despite of the fact that the operating system has to perform some complex tasks such as determining the desired block number. However, that is generally implemented in database applications.



### **Index sequential method –**

It is the other method of accessing a file which is built on the top of the direct access method. These methods construct an index for the file. The index, like an index in the back of a book, contains the pointer to the various blocks. To find a record in the file, we first search the index and then by the help of pointer we access the file directly.

- Provides solutions to problems of contiguous and linked allocation.
- A index block is created having all pointers to files.
- Each file has its own index block which stores the addresses of disk space occupied by the file.
- Directory contains the addresses of index blocks of files.

### **Key points:**

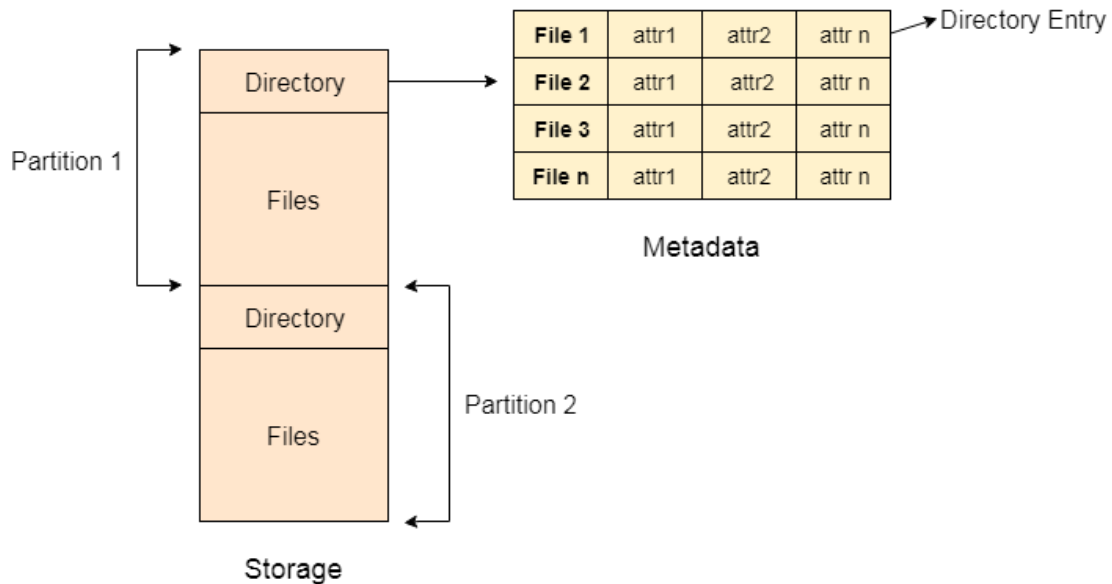
- It is built on top of Sequential access.
- It control the pointer by using index.

### **Directory Structure:-**

Directory can be defined as the listing of the related files on the disk. The directory may store some or the entire file attributes.

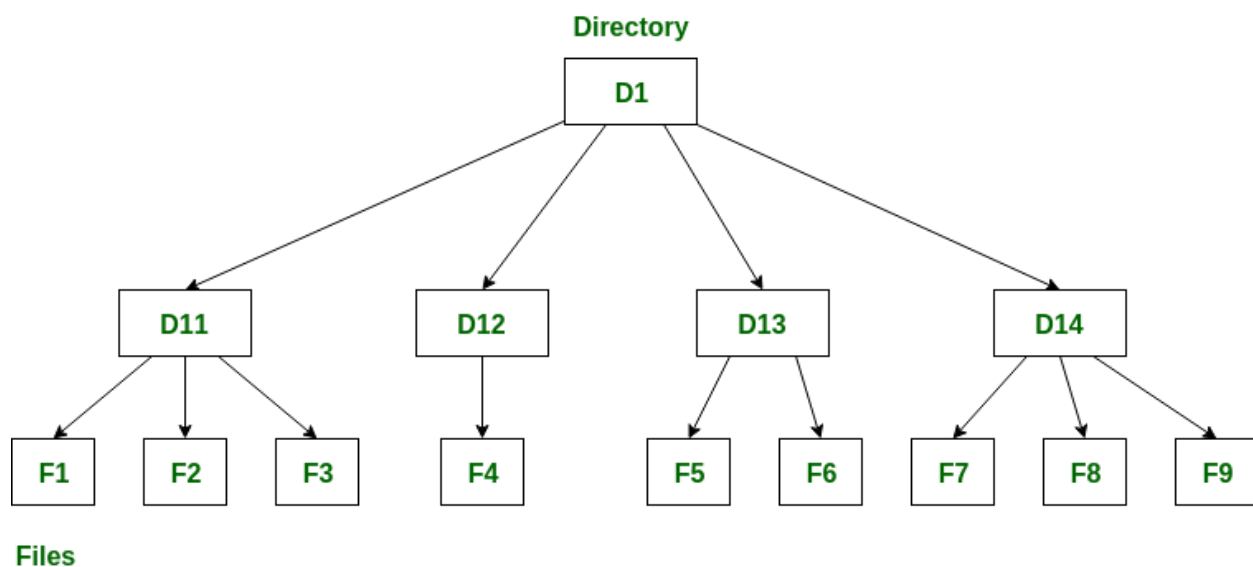
To get the benefit of different file systems on the different operating systems, A hard disk can be divided into the number of partitions of different sizes. The partitions are also called volumes or mini disks.

Each partition must have at least one directory in which, all the files of the partition can be listed. A directory entry is maintained for each file in the directory which stores all the information related to that file.



A directory can be viewed as a file which contains the Meta data of the bunch of files.

A **directory** is a container that is used to contain folders and file. It organizes files and folders into a hierarchical manner.

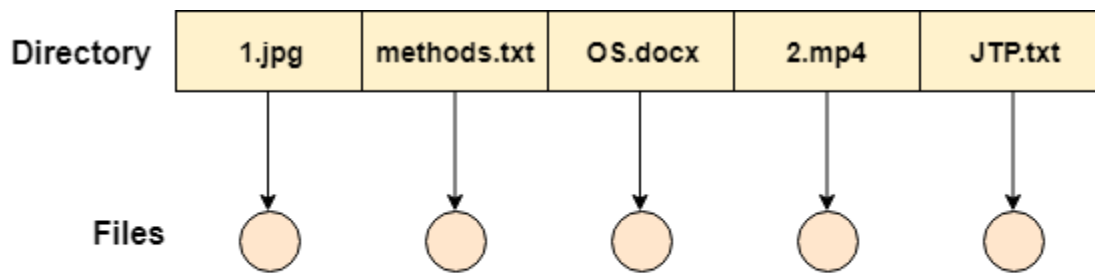


Every Directory supports a number of common operations on the file:

1. File Creation
2. Search for the file
3. File deletion
4. Renaming the file
5. Traversing Files
6. Listing of files

### 1. Single Level Directory

The simplest method is to have one big list of all the files on the disk. The entire system will contain only one directory which is supposed to mention all the files present in the file system. The directory contains one entry per each file present on the file system.



**Single Level Directory**

This type of directories can be used for a simple system.

### Advantages

1. Implementation is very simple.
2. If the sizes of the files are very small then the searching becomes faster.
3. File creation, searching, deletion is very simple since we have only one directory.

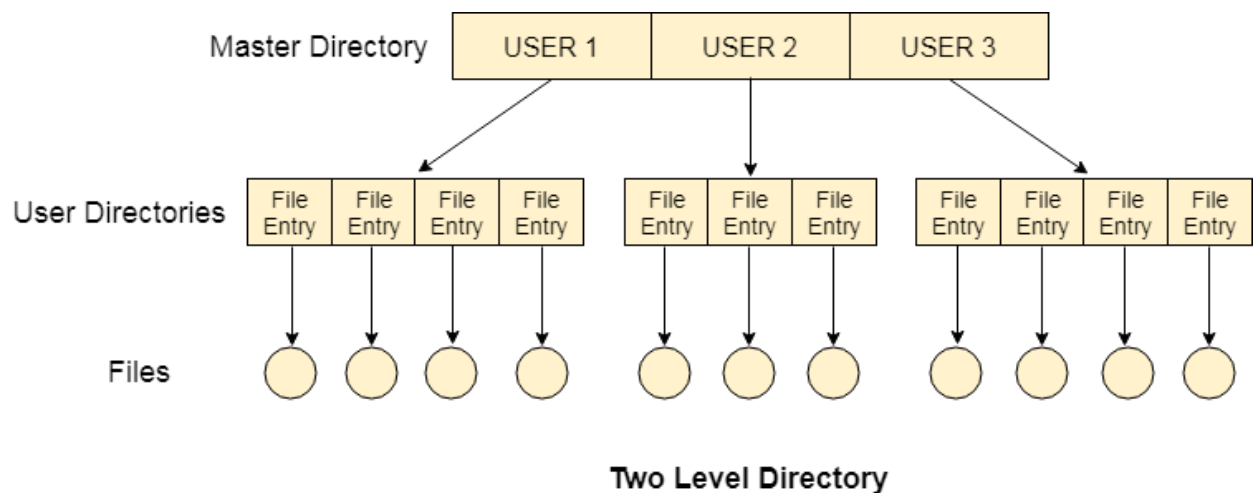
### Disadvantages

1. We cannot have two files with the same name.
2. The directory may be very big therefore searching for a file may take so much time.

3. Protection cannot be implemented for multiple users.
4. There are no ways to group same kind of files.
5. Choosing the unique name for every file is a bit complex and limits the number of files in the system because most of the Operating System limits the number of characters used to construct the file name.

## Two Level Directory

In two level directory systems, we can create a separate directory for each user. There is one master directory which contains separate directories dedicated to each user. For each user, there is a different directory present at the second level, containing group of user's file. The system doesn't let a user to enter in the other user's directory without permission.



### Characteristics of two level directory system

1. Each files has a path name as */User-name/directory-name/*
2. Different users can have the same file name.
3. Searching becomes more efficient as only one user's list needs to be traversed.
4. The same kind of files cannot be grouped into a single directory for a particular user.

Every Operating System maintains a variable as **PWD** which contains the present directory name (present user name) so that the searching can be done appropriately.

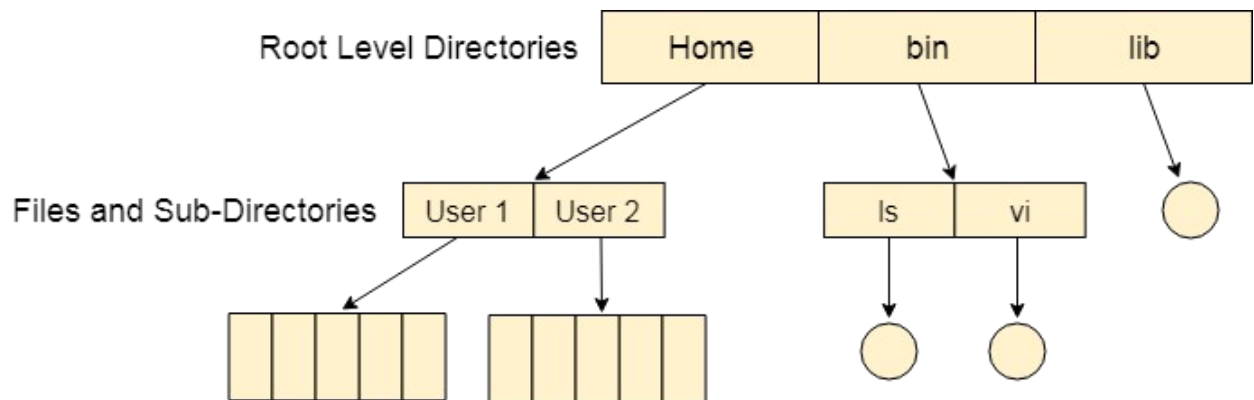
## Tree Structured Directory

In Tree structured directory system, any directory entry can either be a file or sub directory. Tree structured directory system overcomes the drawbacks of two level directory system. The similar kind of files can now be grouped in one directory.

Each user has its own directory and it cannot enter in the other user's directory. However, the user has the permission to read the root's data but he cannot write or modify this. Only administrator of the system has the complete access of root directory.

Searching is more efficient in this directory structure. The concept of current working directory is used. A file can be accessed by two types of path, either relative or absolute.

Absolute path is the path of the file with respect to the root directory of the system while relative path is the path with respect to the current working directory of the system. In tree structured directory systems, the user is given the privilege to create the files as well as directories.



**The Structured Directory System**

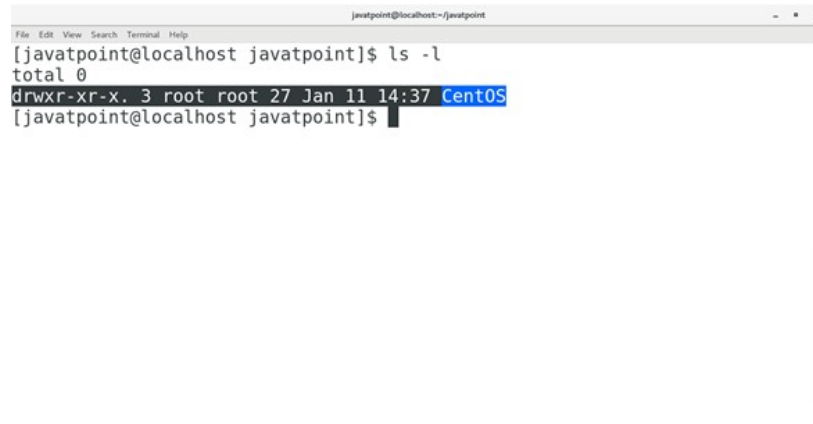
## Permissions on the file and directory

A tree structured directory system may consist of various levels therefore there is a set of permissions assigned to each file and directory.

The permissions are **R W X** which are regarding reading, writing and the execution of the files or directory. The permissions are assigned to three types of users: owner, group and others.

There is a identification bit which differentiate between directory and file. For a directory, it is **d** and for a file, it is dot (**.**)

The following snapshot shows the permissions assigned to a file in a Linux based system. Initial bit **d** represents that it is a directory.

A terminal window titled 'javatpoint@localhost:~/javatpoint' showing the output of the 'ls -l' command. The output is: 'total 0' followed by 'drwxr-xr-x. 3 root root 27 Jan 11 14:37 CentOS'. The 'd' in 'drwxr-xr-x' is highlighted in blue, and the 'CentOS' text is highlighted in blue. The prompt '[javatpoint@localhost javatpoint]\$' is visible at the bottom.

```
javatpoint@localhost:~/javatpoint
[javatpoint@localhost javatpoint]$ ls -l
total 0
drwxr-xr-x. 3 root root 27 Jan 11 14:37 CentOS
[javatpoint@localhost javatpoint]$
```

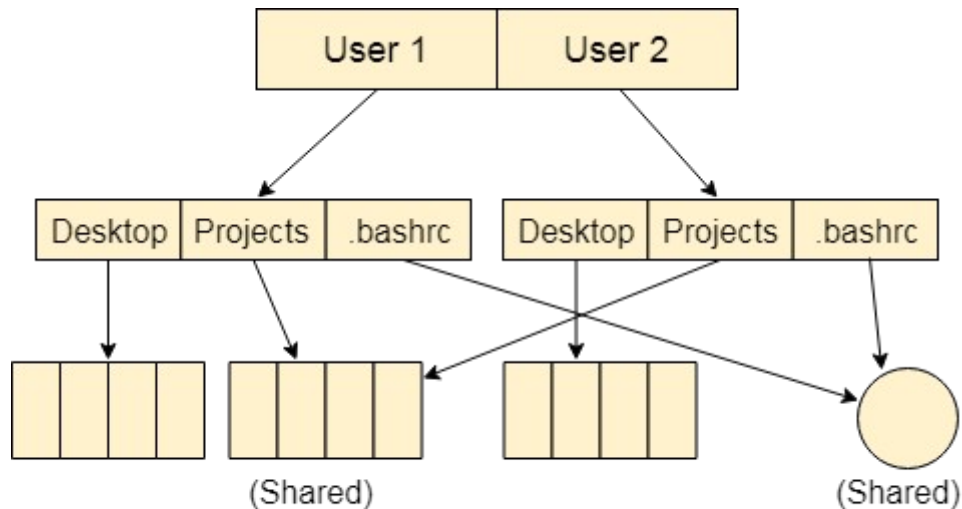
## Acyclic-Graph Structured Directories

The tree structured directory system doesn't allow the same file to exist in multiple directories therefore sharing is major concern in tree structured directory system. We can provide sharing by making the directory an acyclic graph. In this system, two or more directory entry can point to the same file or sub directory. That file or sub directory is shared between the two directory entries.

These kinds of directory graphs can be made using links or aliases. We can have multiple paths for a same file. Links can either be symbolic (logical) or hard link (physical).

If a file gets deleted in acyclic graph structured directory system, then

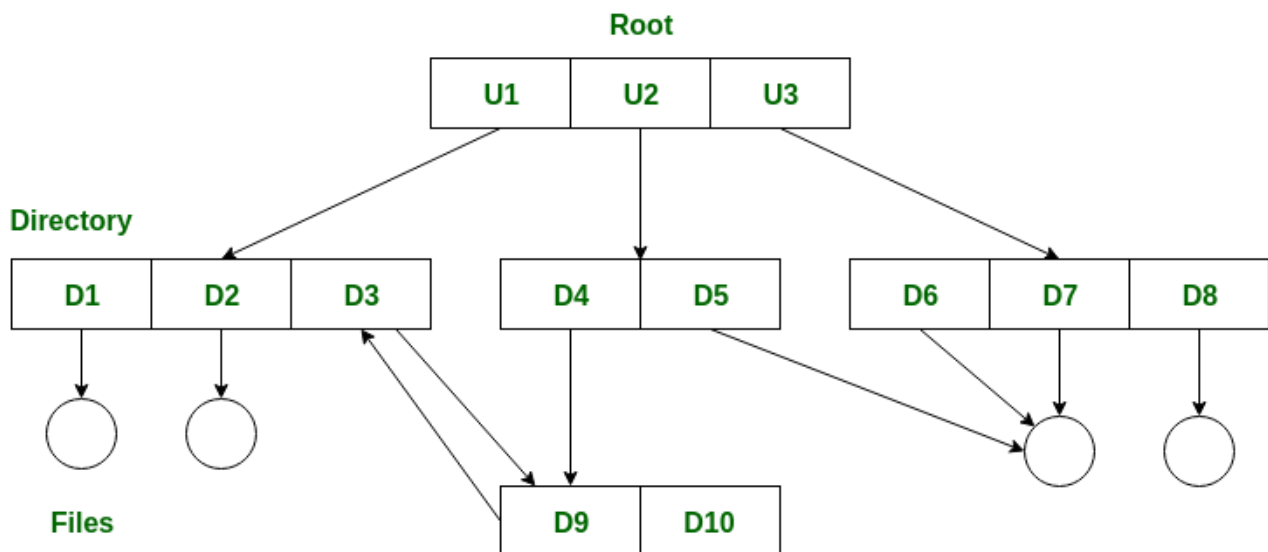
1. In the case of soft link, the file just gets deleted and we are left with a dangling pointer.
2. In the case of hard link, the actual file will be deleted only if all the references to it gets deleted.



Acyclic-Graph Structured Directory System

### General graph directory structure –

In general graph directory structure, cycles are allowed within a directory structure where multiple directories can be derived from more than one parent directory. The main problem with this kind of directory structure is to calculate total size or space that has been taken by the files and directories.



### Advantages:

- It allows cycles.
- It is more flexible than other directories structure.



### Disadvantages:

- It is more costly than others.
- It needs garbage collection.

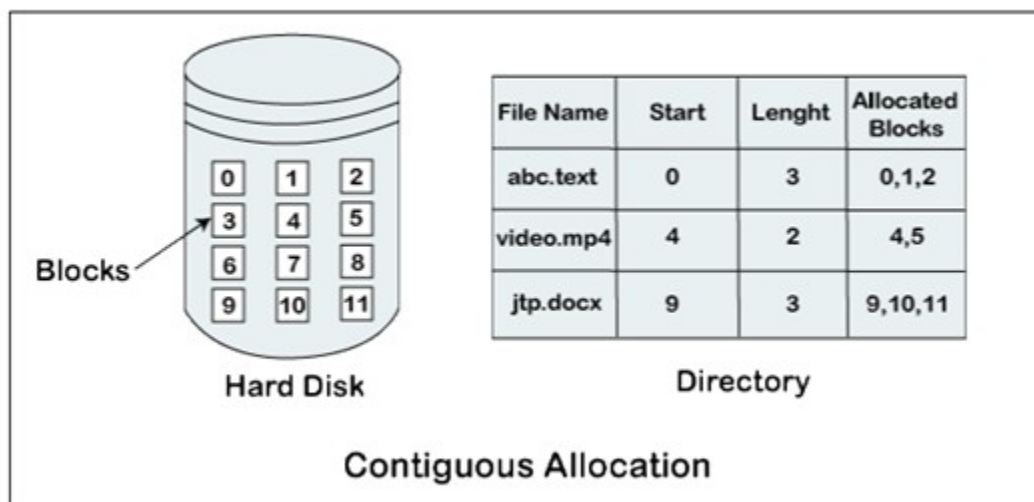
## Structures of Directory in Operating System

### Allocation Method

The allocation method defines how the files are stored in the disk blocks. The direct access nature of the disks gives us the flexibility to implement the files. In many cases, different files or many files are stored on the same disk. The main problem that occurs in the operating system is that how we allocate the spaces to these files so that the utilization of disk is efficient and the quick access to the file is possible. There are mainly three methods of file allocation in the disk. Each method has its advantages and disadvantages. Mainly a system uses one method for all files within the system.

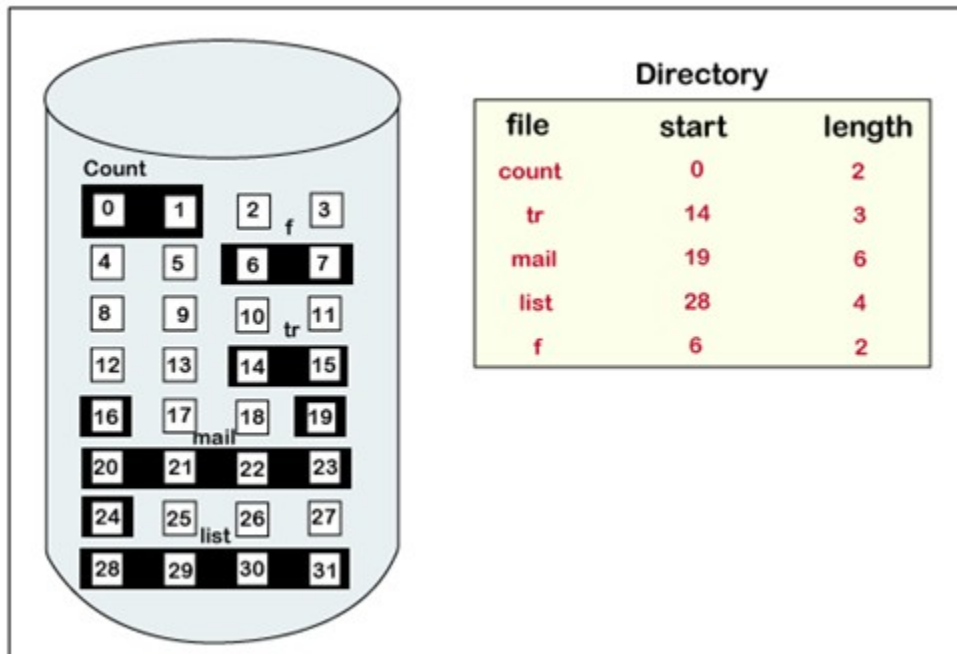
**Contiguous Allocation:** – Contiguous allocation is one of the most used methods for allocation. Contiguous allocation means we allocate the block in such a manner, so that in the hard disk, all the blocks get the contiguous physical block.

We can see in the below figure that in the directory, we have three files. In the table, we have mentioned the starting block and the length of all the files. We can see in the table that for each file, we allocate a contiguous block.



## Example of contiguous allocation

We can see in the given diagram, that there is a file. The name of the file is 'mail.' The file starts from the 19<sup>th</sup> block and the length of the file is 6. So, the file occupies 6 blocks in a contiguous manner. Thus, it will hold blocks 19, 20, 21, 22, 23, 24.



## Advantages of Contiguous Allocation

The advantages of contiguous allocation are:

1. The contiguous allocation method gives excellent read performance.
2. Contiguous allocation is easy to implement.
3. The contiguous allocation method supports both types of file access methods that are sequential access and direct access.
4. The Contiguous allocation method is fast because, in this method number of seeks is less due to the contiguous allocation of file blocks.

## Disadvantages of Contiguous allocation

The disadvantages of contiguous allocation method are:

1. In the contiguous allocation method, sometimes disk can be fragmented.

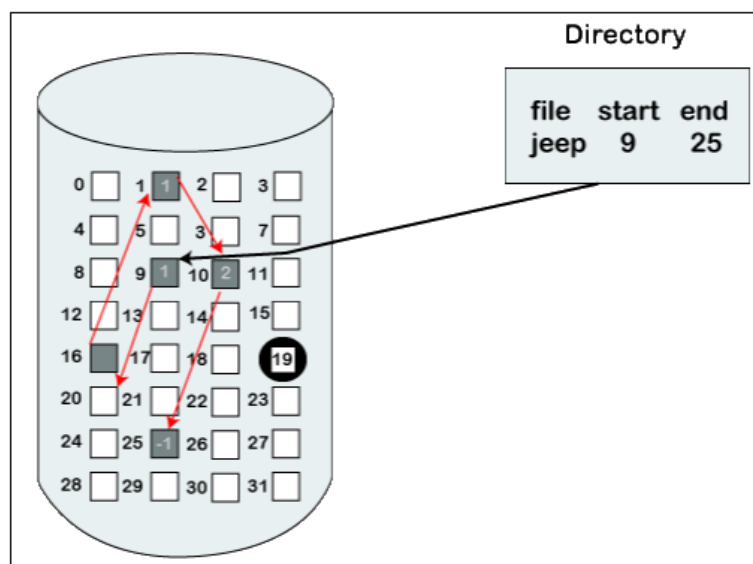
2. In this method, it is difficult to increase the size of the file due to the availability of the contiguous memory block.

## Linked List Allocation

The linked list allocation method overcomes the drawbacks of the contiguous allocation method. In this file allocation method, each file is treated as a linked list of disks blocks. In the linked list allocation method, it is not required that disk blocks assigned to a specific file are in the contiguous order on the disk. The directory entry comprises of a pointer for starting file block and also for the ending file block. Each disk block that is allocated or assigned to a file consists of a pointer, and that pointer points to the next block of the disk, which is allocated to the same file.

### Example of linked list allocation

We can see in the below figure that we have a file named 'jeep.' The value of the start is 9. So, we have to start the allocation from the 9<sup>th</sup> block, and blocks are allocated in a random manner. The value of the end is 25. It means the allocation is finished on the 25<sup>th</sup> block. We can see in the below figure that the block (25) comprised of -1, which means a null pointer, and it will not point to another block.



### Advantages of Linked list allocation

There are various advantages of linked list allocation:

1. In linked list allocation, there is no external fragmentation. Due to this, we can utilize the memory better.
2. In linked list allocation, a directory entry only comprises of the starting block address.
3. The linked allocation method is flexible because we can quickly increase the size of the file because, in this to allocate a file, we do not require a chunk of memory in a contiguous form.

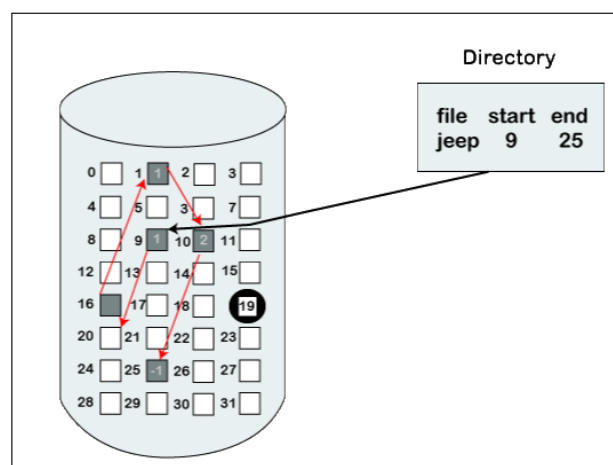
## Disadvantages of Linked list Allocation

There are various disadvantages of linked list allocation:

1. Linked list allocation does not support direct access or random access.
2. In linked list allocation, we need to traverse each block.
3. If the pointer in the linked list break in linked list allocation, then the file gets corrupted.
4. In the disk block for the pointer, it needs some extra space.

## Indexed Allocation

The Indexed allocation method is another method that is used for file allocation. In the index allocation method, we have an additional block, and that block is known as the index block. For each file, there is an individual index block. In the index block, the  $i$ th entry holds the disk address of the  $i$ th file block. We can see in the below figure that the directory entry comprises of the address of the index block.



## **Advantages of Index Allocation**

The advantages of index allocation are:

1. The index allocation method solves the problem of external fragmentation.
2. Index allocation provides direct access.

## **Disadvantages of Index Allocation**

The disadvantages of index allocation are:

1. In index allocation, pointer overhead is more.
2. We can lose the entire file if an index block is not correct.
3. It is totally a wastage to create an index for a small file.

A single index block cannot hold all the pointer for files with large sizes.

To resolve this problem, there are various mechanism which we can use:

1. Linked scheme
2. Multilevel Index
3. Combined Scheme

1. **Linked Scheme:** – In the linked scheme, to hold the pointer, two or more than two index blocks are linked together. Each block contains the address of the next index block or a pointer.
2. **Multilevel Index:** – In the multilevel index, to point the second-level index block, we use a first-level index block that in turn points to the blocks of the disk, occupied by the file. We can extend this up to 3 or more than 3 levels depending on the maximum size of the file.
3. **Combined Scheme:** – In a combined scheme, there is a special block which is called an information node (Inode). The inode comprises of all the information related to the file like authority, name, size, etc. To store the disk block addresses that contain the actual file, the remaining space of inode is used. In inode, the starting pointer is used to point the direct blocks. This means the pointer comprises of the addresses of the disk blocks, which consist of the file data. To indicate the indirect blocks, the next few pointers are used. The indirect blocks are of three types, which are single indirect, double indirect, and triple indirect.

## **Protection in File System:-**

In computer systems, a lot of user's information is stored, the objective of the operating system is to keep safe the data of the user from the improper access to the system. Protection can be provided in number of ways. For a single laptop system, we might provide protection by locking the computer in a desk drawer or file cabinet. For multi-user systems, different mechanisms are used for the protection.

## Types of Access :

The files which have direct access of the any user have the need of protection. The files which are not accessible to other users doesn't require any kind of protection. The mechanism of the protection provide the facility of the controlled access by just limiting the types of access to the file. Access can be given or not given to any user depends on several factors, one of which is the type of access required. Several different types of operations can be controlled:

- **Read** – Reading from a file.
- **Write** – Writing or rewriting the file.
- **Execute** – Loading the file and after loading the execution process starts.
- **Append** – Writing the new information to the already existing file, editing must be end at the end of the existing file.
- **Delete** – Deleting the file which is of no use and using its space for the another data.
- **List** – List the name and attributes of the file.

Operations like renaming, editing the existing file, copying; these can also be controlled. There are many protection mechanism. each of them mechanism have different advantages and disadvantages and must be appropriate for the intended application.

## Access Control :

There are different methods used by different users to access any file. The general way of protection is to associate *identity-dependent access* with all the files and directories an list called access-control list (ACL) which specify the names of the users and the types of access associate with each of the user. The main problem with the access list is their length. If we want to allow everyone to read a file, we must list all the users with the read access. This technique has two undesirable consequences: Constructing such a list may be tedious and unrewarding task, especially if we do not know in advance the list of the users in the system.

Previously, the entry of the any directory is of the fixed size but now it changes to the variable size which results in the complicates space management. These problems can be resolved by use of a condensed version of the access list. To condense the length of the access-control list, many systems recognize three classification of users in connection with each file:

- **Owner** – Owner is the user who has created the file.
- **Group** – A group is a set of members who has similar needs and they are sharing the same file.
- **Universe** – In the system, all other users are under the category called universe.

The most common recent approach is to combine access-control lists with the normal general owner, group, and universe access control scheme. For example: Solaris uses the three categories of access by default but allows access-control lists to be added to specific files and directories when more fine-grained access control is desired.

### **Other Protection Approaches:**

The access to any system is also controlled by the password. If the use of password are is random and it is changed often, this may be result in limit the effective access to a file. The use of passwords has a few disadvantages:

- The number of passwords are very large so it is difficult to remember the large passwords.
- If one password is used for all the files, then once it is discovered, all files are accessible; protection is on all-or-none basis.

### **Disk scheduling:-**

- As we know, a process needs two type of time, CPU time and IO time. For I/O, it requests the Operating system to access the disk.
- However, the operating system must be fare enough to satisfy each request and at the same time, operating system must maintain the efficiency and speed of process execution.
- The technique that operating system uses to determine the request which is to be satisfied next is called disk scheduling.

Let's discuss some important terms related to disk scheduling.

- **Seek Time** - Seek time is the time taken in locating the disk arm to a specified track where the read/write request will be satisfied.
- **Rotational Latency** - It is the time taken by the desired sector to rotate itself to the position from where it can access the R/W heads.
- **Transfer Time** - It is the time taken to transfer the data.
- **Disk Access Time** - Disk access time is given as,  

$$\text{Disk Access Time} = \text{Rotational Latency} + \text{Seek Time} + \text{Transfer Time}$$
- **Disk Response Time** - It is the average of time spent by each request waiting for the IO operation.
- **Purpose of Disk Scheduling** - The main purpose of disk scheduling algorithm is to select a disk request from the queue of IO requests and decide the schedule when this request will be processed.

### **Goal of Disk Scheduling Algorithm**

- Fairness
- High throughput
- Minimal traveling head time

### **Disk Scheduling Algorithms:-**

- The list of various disks scheduling algorithm is given below. Each algorithm is carrying some advantages and disadvantages. The limitation of each algorithm leads to the evolution of a new algorithm.
- FCFS scheduling algorithm
- SSTF (shortest seek time first) algorithm
- SCAN scheduling
- C-SCAN scheduling
- LOOK Scheduling
- C-LOOK scheduling

### **FCFS Scheduling Algorithm:-**

- It is the simplest Disk Scheduling algorithm. It services the IO requests in the order in which they arrive. There is no starvation in this algorithm, every request is serviced.

### **Disadvantages**

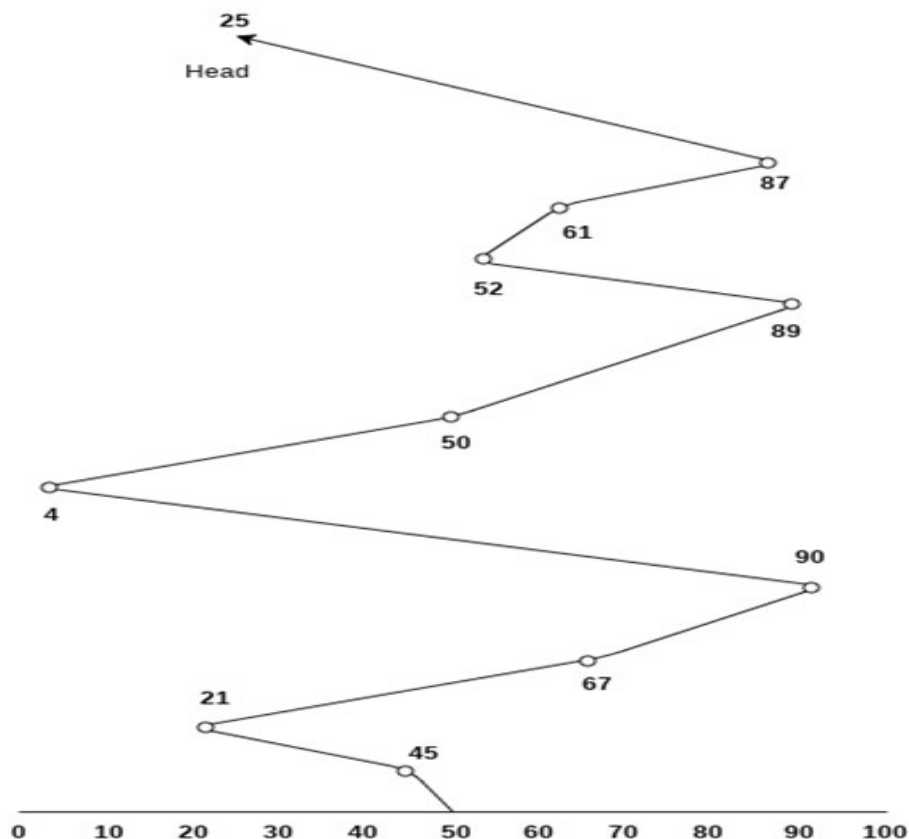


- The scheme does not optimize the seek time.
- The request may come from different processes therefore there is the possibility of inappropriate movement of the head.

**Example:-** Consider the following disk request sequence for a disk with 100 tracks 45, 21, 67, 90, 4, 50, 89, 52, 61, 87, 25 Head pointer starting at 50 and moving in left direction. Find the number of head movements in cylinders using FCFS scheduling.

**Solution:-** Number of cylinders moved by the head

$$\begin{aligned}
 &= (50-45)+(45-21)+(67-21)+(90-67)+(90-4)+(50-4)+(89-50)+(61-52)+(87-61)+(87-25) \\
 &= 5 + 24 + 46 + 23 + 86 + 46 + 49 + 9 + 26 + 62 \\
 &= 376
 \end{aligned}$$



### C-SCAN Algorithm:-

In C-SCAN algorithm, the arm of the disk moves in a particular direction servicing requests until it reaches the last cylinder, then it jumps to the last cylinder of the opposite direction without

servicing any request then it turns back and start moving in that direction servicing the remaining requests.

**Example:** - Consider the following disk request sequence for a disk with 100 tracks 98, 137, 122, 183, 14, 133, 65, 78 Head pointer starting at 54 and moving in left direction. Find the number of head movements in cylinders using C-SCAN scheduling.

**Solution:-** No. of cylinders crossed =  $40 + 14 + 199 + 16 + 46 + 4 + 11 + 24 + 20 + 13 = 387$

