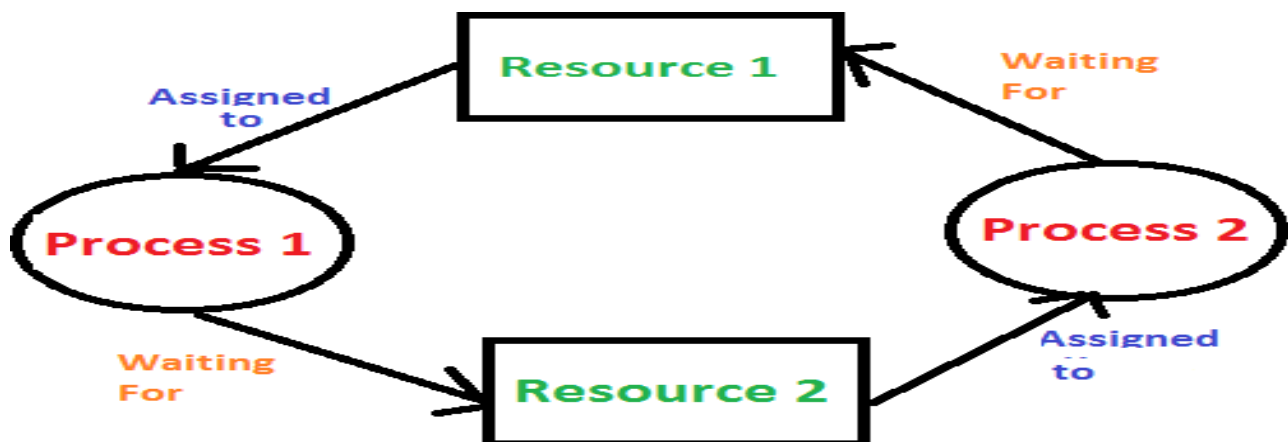


# Deadlocks

BCA-II(2024-25)

## Deadlocks:

- We know that, program in execution is called 'Process' and a process in operating systems uses different resources in following way:
  - 1) Requests a resource.
  - 2) Use the resource.
  - 3) Releases the resource.
- A deadlock happens in operating system when two or more processes needs some resource to complete their execution but that resource is already held by the other process.
- Deadlock is a situation that occurs in OS when any process enters a waiting state because another process is holding the demanded resource. Deadlock is a common problem in multi-processing where several processes share a specific type of mutually exclusive resource known as a 'soft lock' or 'software.'
- Or we can say that Deadlock is a situation where a set of processes are blocked (or in waiting state) because each process is wants to access a resource which is already acquired by some other process.
- **Example-** When two trains are coming toward each other on the same track and there is only one track, none of the trains can move back once they are in front of each other. A similar situation occurs in operating systems when there are two or more processes that hold some resources and wait for resources held by other(s). For example, in the below diagram, Process 1 is holding Resource 1 and waiting for resource 2 which is acquired by process 2, and process 2 is waiting for resource 1.



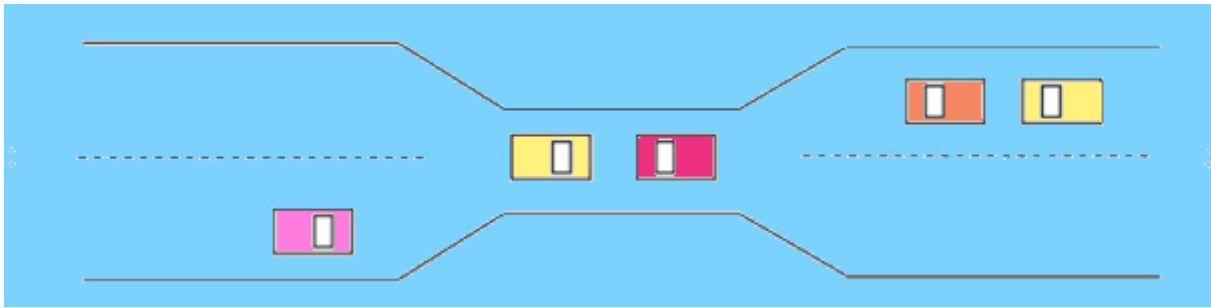
A real-world example of deadlock would be traffic found on a narrow bridge is available.

- Here, a bridge is considered a resource.
- So, when Deadlock happens, it can be easily resolved if one car backs up (Preempt resources and rollback).
- Several cars may have to be backed up if a deadlock situation occurs.

# Deadlocks

BCA-II(2024-25)

- So starvation is possible.



## Deadlock Characterization: (Necessary Conditions for deadlock)

- We know that a deadlock happens in operating system when two or more processes need some resource to complete their execution that is held by the other process.
- A deadlock occurs if the four conditions (Mutual exclusion, hold and wait, no preemption, circular wait) are true. They are given as follows –

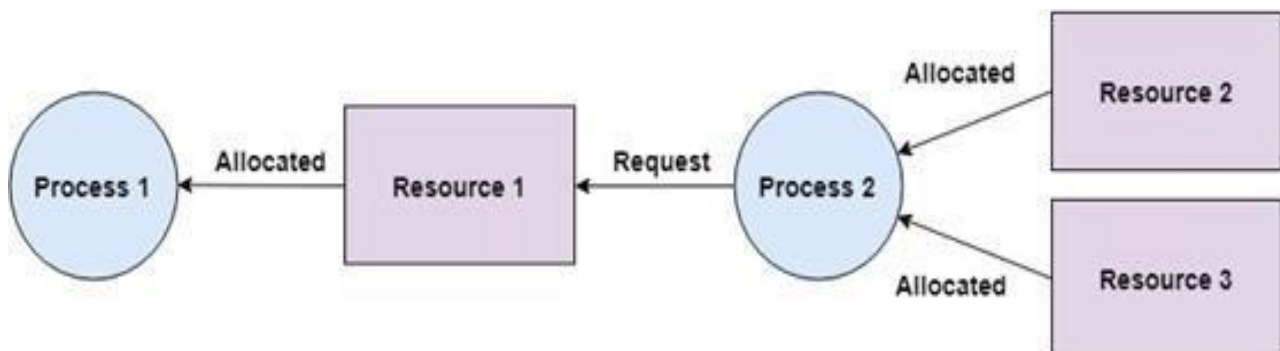
### 1) Mutual Exclusion:

- 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 Process1 only.



### 2) Hold and Wait:

- A process can hold multiple resources and still request more resources from other processes which are holding them to complete its execution. In the diagram given below, Process 2 holds Resource 2 and Resource 3 and is requesting the Resource1 which is held by Process 1.

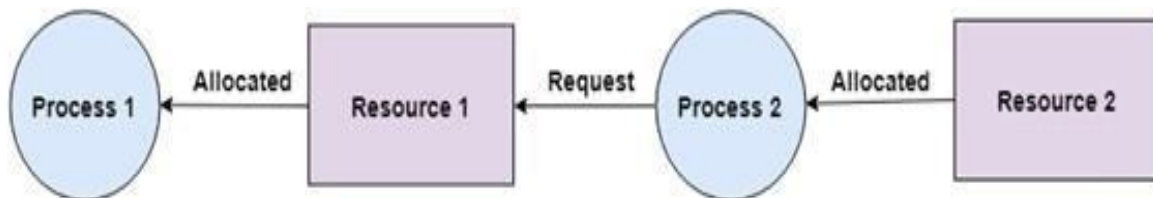


# Deadlocks

BCA-II(2024-25)

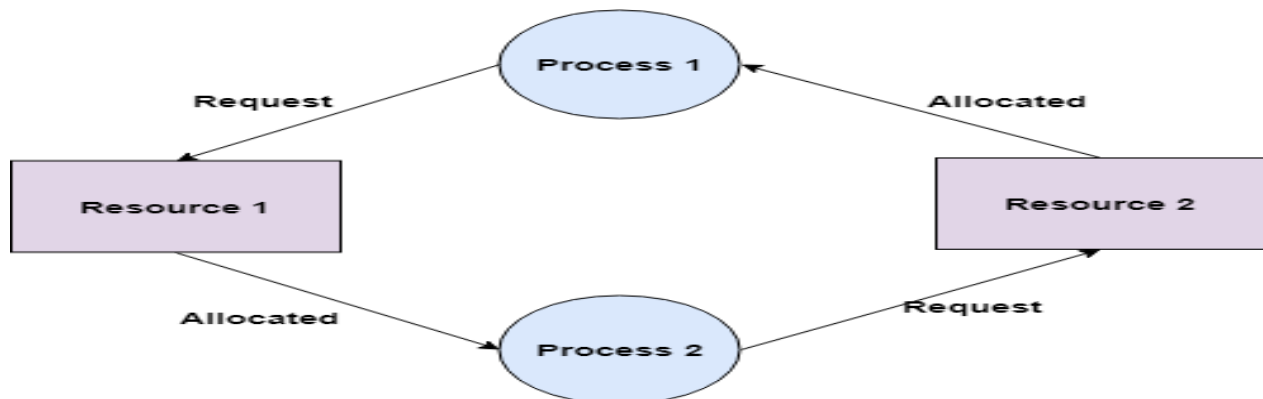
## 3) No Preemption:

- A resource cannot be preempted from a process by force. A process can only release a resource voluntarily (own free). In the diagram below, Process 2 can not preempt Resource1 from Process
- It will only be released when Process
- 1 relinquishes it voluntarily after its execution is complete.



## 4) Circular Wait:

- 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 wait.
- For example: Process 1 is allocated Resource 2 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



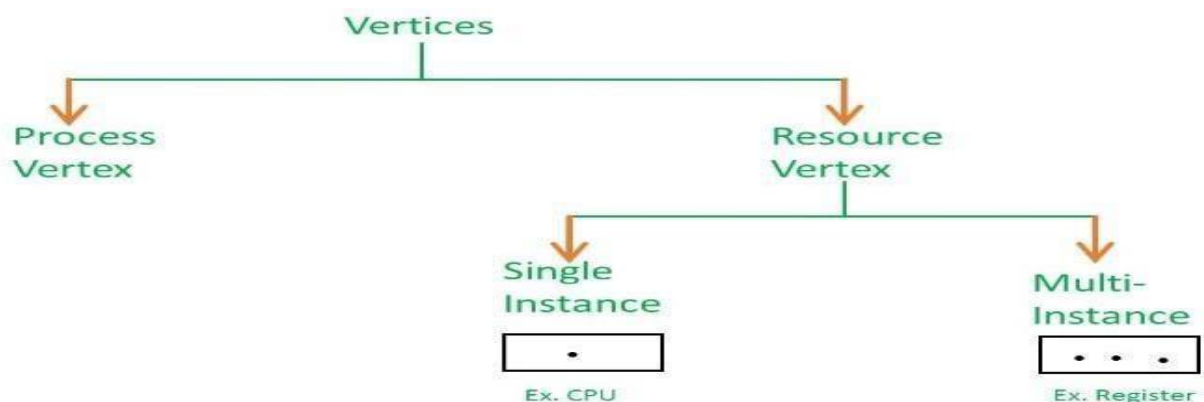
# Deadlocks

BCA-II(2024-25)

## Resource allocation graph: (RAG)

- 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.
- Resource allocation graph is explained to us- what is the state of the system in terms of processes and resources. Like how many resources are available, how many are allocated and what is the request of each process. Everything can be represented in terms of the diagram and that diagram is nothing but RAG.
- One of the advantages of RAG, it is possible to see a deadlock directly.
- We know that any graph contains vertices and edges. So RAG also contains vertices and edges. In RAG vertices are two type –
  1. **Process vertex** – Every process in a system is represented by a process vertex. Generally, the process will be represented with a circle symbol.
  2. **Resource vertex** – Every resource in a system is represented by resource vertex. Generally, the resource will be represented with a rectangle or box symbol.It has two types –
  - Single instance resource vertex– It represents as a box or rectangle. Inside the box, there will be one dot.
  - Multi-instance resource vertex– It also represents as a box or rectangle. Inside the box, there will be many dots present. So the number of dots indicates how many instances are present of each resource type.

Following diagram shows types of vertices in RAG.

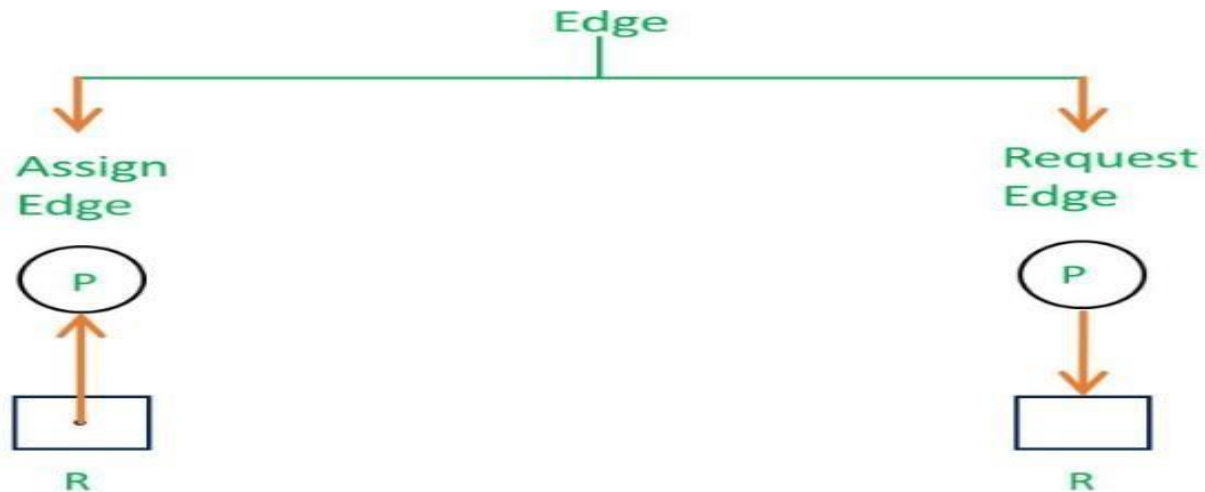


Now coming to the edges of RAG. There are two types of edges in RAG –

1. **Assign Edge** – If you already assign a resource to a process then it is called “Assign edge”.
2. **Request Edge** – It means in future the process might want some resource to complete the execution that is called “request edge”.

# Deadlocks

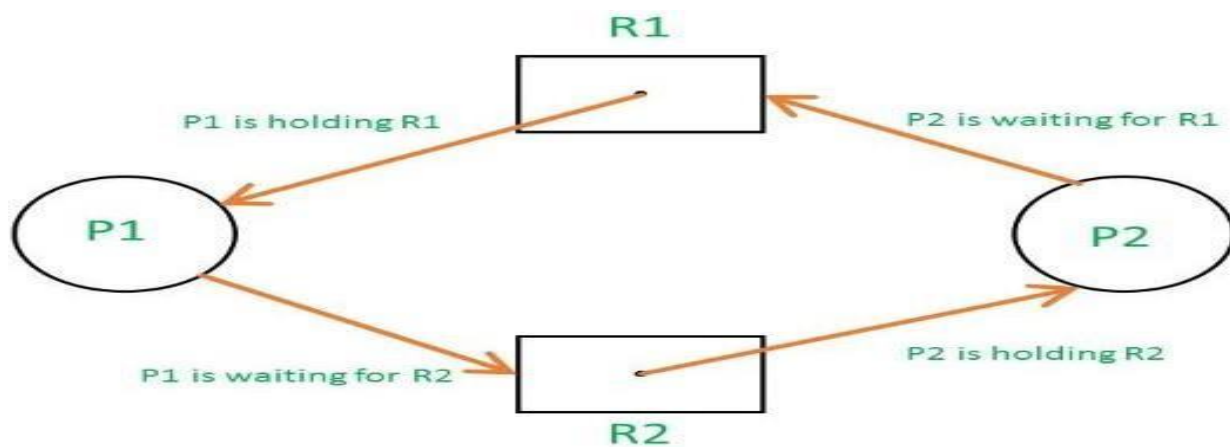
BCA-II(2024-25)



So, If a process is using a resource, an arrow is drawn from the resource vertex to the process vertex.

If a process is requesting a resource, an arrow is drawn from the process node to the resource node.

**Example 1:** (Single instances RAG) –



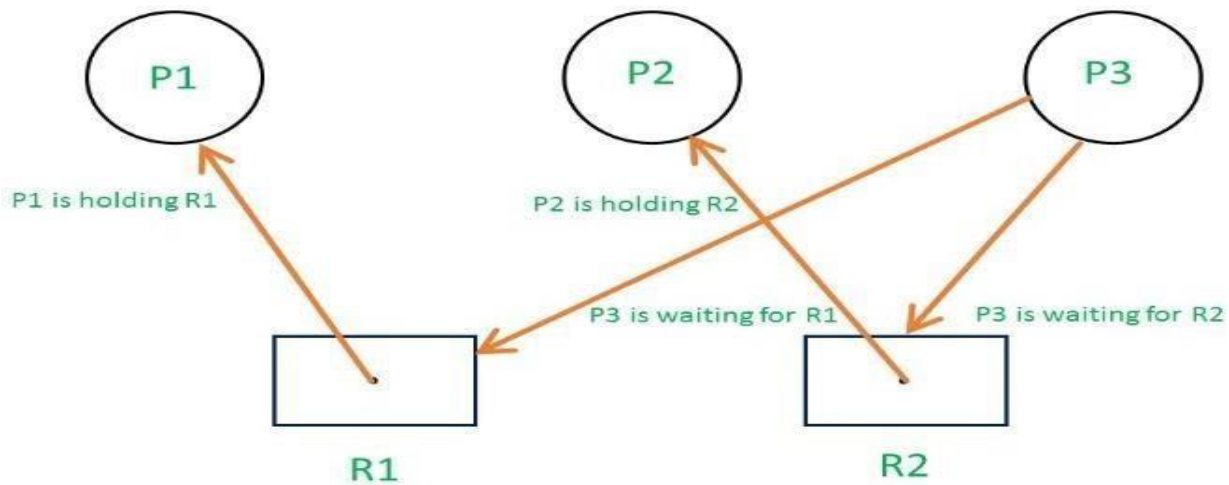
## SINGLE INSTANCE RESOURCE TYPE WITH DEADLOCK

If there is a cycle in the Resource Allocation Graph and each resource in the cycle provides only one instance, then the processes will be in deadlock. For example, if process P1 holds resource R1, process P2 holds resource R2 and process P1 is waiting for R2 and process P2 is waiting for R1, then process P1 and process P2 will be in deadlock.

Here's another example, which shows Processes P1 and P2 acquiring resources R1 and R2 while

# Deadlocks

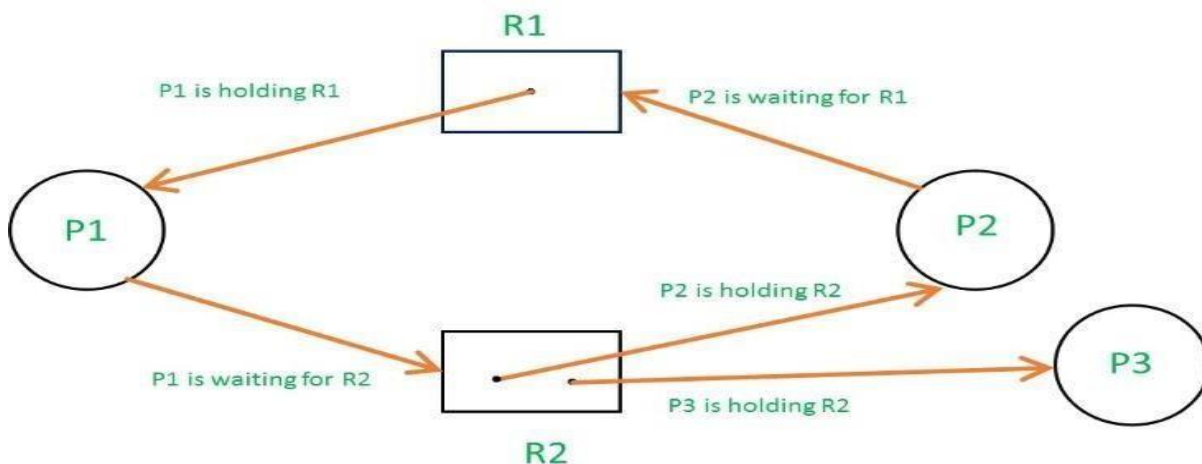
BCA-II(2024-25)



SINGLE INSTANCE RESOURCE TYPE WITHOUT DEADLOCK

process P3 is waiting to acquire both resources. In this example, there is no deadlock because there is no circular dependency. So cycle in single-instance resource type is the sufficient condition for deadlock.

**Example 2** (Multi-instances RAG) –



MULTI INSTANCES WITHOUT DEADLOCK

Note: In above diagram, there is no circular wait between process P1, P2 and P3 therefore deadlock not happen.

## Methods of Handling Deadlocks:

- We know that, Deadlock is a situation where a process or a set of processes is blocked or waiting for some other resource that is held by some other waiting process. It is an undesirable (unwanted) state of the system.
- There are four conditions mutual exclusion, hold and wait, no preemption and circular wait that must hold simultaneously for a deadlock to occur.

There are four strategies or methods to handle the deadlocks:

### 1) Deadlock Ignorance.

# Deadlocks

BCA-II(2024-25)

## 2) Deadlock Prevention.

## 3) Deadlock Avoidance.

## 4) Deadlock detection & recovery.

Let's see these methods in details-

### 1) Deadlock Ignorance:

- Deadlock Ignorance is the most widely used approach among all the mechanism. This is being used by many operating systems mainly for end user uses.
- In this approach, the Operating system assumes that deadlock never occurs. It simply ignores deadlock. This approach is best suitable for a single end user system where User uses the system only for browsing and all other normal stuff.
- The operating systems like Windows and Linux mainly focus upon performance. However, the performance of the system decreases if it uses deadlock handling mechanism all the time if deadlock happens 1 out of 100 times then it is completely unnecessary to use the deadlock handling mechanism all the time.
- In these types of systems, the user has to simply restart the computer in the case of deadlock. Windows and Linux are mainly using this approach.

### 2) Deadlock Prevention:

- We know that, deadlock happens only when Mutual Exclusion, hold and wait, No preemption and circular wait holds simultaneously.
- If it is possible to violate or prevent one of the four conditions at any time and don't let them occur together then we can prevent the deadlock.
- The idea behind the approach is very simple that we have to fail one of the four conditions but there can be a big challenge on its physical implementation in the system.
- Let's see how we can prevent each of the conditions.

#### 1. Mutual Exclusion:

- We know that in mutual exclusion, only one resource is allocated to only one process at a time and this is the main reason behind the deadlock. If a resource could have been used by more than one process at the same time then the process would have never been waiting for any resource.
- However, if we can be able to violate resources behaving in the mutually exclusive manner then the deadlock can be prevented.

#### 2. Hold and Wait:

- We know that, Hold and wait condition occurs 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)**

# Deadlocks

BCA-II(2024-25)

- 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 starvation due to the fact that some process may hold a resource for a very long time.

### 3) **No preemption:**

- We know that, once a resource is allocated to a process then we cannot force fully preempts that resource from a process & that causes deadlock. However, if we take the resource away (preempt) from the process 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 effective less and also the fact that the process can't start printing again from where it has left last time.

### 4) **Circular Wait:**

- To violate circular wait, we have to break the circular dependency of resources used by processes for that we can assign a priority number to each of the resource.
- Here, a process requests for a resource by incremental order of priorities of resources that causes circular wait doesn't happen. Note that, a process can't request for a lesser priority resource. This ensures that a process cannot request are source 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. Thus, we can prevent the deadlock.



# Deadlocks

BCA-II(2024-25)

## 3) Deadlock Avoidance:

- In deadlock avoidance, at every step the operating system checks whether the system is in safe state or in unsafe state. The process continues until the system is in safe state. Once the system moves to unsafe state, the OS has to backtrack one step.
- In simple words, The OS continue reviews for each resource allocation so that the allocation doesn't cause the deadlock in the system.
- 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 for the process so that there can never be a circular wait condition.
- **Resource allocation state (Safe and Unsafe States):**
- The resource allocation state of a system can be defined by available and allocated instances of 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 A	Type B	Type C	Type D
P1	3	0	2	2
P2	0	0	1	1
P3	1	1	1	0
P4	2	1	4	0

Resources still needed to complete execution:

Process	Type A	Type B	Type C	Type D
P1	1	1	0	0
P2	0	1	1	2
P3	1	2	1	0
P4	2	1	1	2

Total resource E = (7 6 8 4) (Let, It is initially considered in system)

Allocated resource P= (6 2 8 3)

Available resource A= (1 4 0 1)

# Deadlocks

BCA-II(2024-25)

- 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 that each process still needs.
- Vector E represents total number of 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.
- If system is in unsafe state & resource allocation request is granted then it again leads to 'Deadlock' therefore in deadlock avoidance, resource allocation request is granted only if system is in 'Safe' state such that deadlock never occurs.

## Banker's Algorithm in Operating System

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.

### Characteristics of Banker's Algorithm

#### The characteristics of Banker's algorithm are as follows:

- If any process requests for a resource, then it has to wait.
- This algorithm consists of advanced features for maximum resource allocation.
- There are limited resources in the system we have.
- In this algorithm, if any process gets all the needed resources, then it is that it should

# Deadlocks

BCA-II(2024-25)

- return the resources in a restricted period.
- Various resources are maintained in this algorithm that can fulfill the needs of at least one client.

Let us assume that there are  $n$  processes and  $m$  resource types.

Data Structures used to implement the Banker's Algorithm

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  $\text{Available}[j] = k$ , 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  $\text{Max}[i][j] = k$ , then the process  $P_i$  can request at most  $k$  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  $\text{Allocation}[i][j] = k$ , then process  $P_i$  is currently allocated  $k$  instances of resource type  $R_j$ .

## 4. Need

It is a two-dimensional array. It is an  $n \times m$  matrix which indicates the remaining resource needs of each process. If  $\text{Need}[i][j] = k$ , then process  $P_i$  may need  $k$  more instances of resource type  $R_j$  to complete its task.

$$\text{Need}[i][j] = \text{Max}[i][j] - \text{Allocation}[i][j]$$

Banker's algorithm comprises of two algorithms:

1. Safety algorithm
2. Resource request algorithm

## Safety Algorithm

A safety algorithm is an algorithm used to find whether or not a system is in its safe state. The algorithm is as follows:

1. Let Work and Finish be vectors of length  $m$  and  $n$ , respectively. Initially,

# Deadlocks

25)

2. `Work = Available`

3. `Finish[i] = false` for  $i = 0, 1, \dots, n - 1$ .

This means, initially, no process has finished and the number of available resources is represented by the **Available** array

4. Find an index **i** such that both

5. `Finish[i] == false`

6. `Needi <= Work`

If there is no such **i** present, then proceed to step 4.

It means, we need to find an unfinished process whose need can be satisfied by the available resources. If no such process exists, just go to step 4.

7. Perform the following:

8. `Work = Work + Allocationi`

9. `Finish[i] = true`

Go to step 2.

When an unfinished process is found, then the resources are allocated and the process is marked finished. And then, the loop is repeated to check the same for all other processes.

10. If `Finish[i] == true` for all **i**, then the system is in a safe state.

That means if all processes are finished, then the system is in safe state.

This algorithm may require an order of  **$m \times n^2$  operations** in order to determine whether a state is safe or not.

Resource Request Algorithm

Now the next algorithm is a resource-request algorithm and it is mainly used to determine whether requests can be safely granted or not.

Let `Requesti` be the request vector for the process  $P_i$ . If `Requesti[j] == k`, then process  $P_i$  wants  $k$  instance of Resource type  $R_j$ . When a request for resources is made by the process  $P_i$ , the following are the actions that will be taken:

1. If `Requesti <= Needi`, then go to step 2; else raise an error condition, since the process has

# Deadlocks

BCA-II(2024-25)

exceeded its maximum claim.

2. If  $\text{Request}_i \leq \text{Available}_i$  then go to step 3; else  $P_i$  must have to wait as resources are not available.

3. Now we will assume that resources are assigned to process  $P_i$  and thus perform the following steps:

$\text{Available} = \text{Available} - \text{Request}_i$ ;

$\text{Allocation}_i = \text{Allocation}_i + \text{Request}_i$ ;  $\text{Need}_i$

$= \text{Need}_i - \text{Request}_i$ ;

If the resulting resource allocation state comes out to be safe, then the transaction is completed and, process  $P_i$  is allocated its resources. But in this case, if the new state is unsafe, then  $P_i$  waits for  $\text{Request}_i$ , and the old resource-allocation state is restored.

## Disadvantages of Banker's Algorithm

Some disadvantages of this algorithm are as follows:

1. During the time of Processing, this algorithm does not permit a process to change its maximum need.
2. Another disadvantage of this algorithm is that all the processes must know in advance about the maximum resource needs.
3. This algorithm permits the requests to be provided in constrained time, but for one year which is a fixed period.

Now its time to take a look at the Example of Banker's Algorithm:

### Example:

**Let us consider the following snapshot for understanding the banker's algorithm:**

1. calculate the content of the need matrix?
2. Check if the system is in a safe state?
3. Determine the total sum of each type of resource?

### Solution:

The Content of the need matrix can be calculated by using the formula given below:

# Deadlocks

BCA-II(2024-25)

Processes	Allocation A B C	Max A B C	Available A B C
P0	1 1 2	4 3 3	2 1 0
P1	2 1 2	3 2 2	
P2	4 0 1	9 0 2	
P3	0 2 0	7 5 3	
P4	1 1 2	1 1 2	

**Need = Max – Allocation**

Process	Need		
	A	B	C
P <sub>0</sub>	3	2	1
P <sub>1</sub>	1	1	0
P <sub>2</sub>	5	0	1
P <sub>3</sub>	7	3	3
P <sub>4</sub>	0	0	0

**1.** Let us now check for the safe state.

**Safe sequence:**

1. For process P<sub>0</sub>, Need = (3, 2, 1) and Available = (2, 1, 0)

Need <= Available = False

So, the system will move to the next process.

2. For Process P<sub>1</sub>, Need = (1, 1, 0) Available = (2, 1, 0)

Need <= Available = True Request of P<sub>1</sub> is

granted. Available = Available + Allocation

= (2, 1, 0) + (2, 1, 2)

= (4, 2, 2) (New Available)

# Deadlocks

BCA-II(2024-25)

3. For Process P2, Need = (5, 0, 1)

Available = (4, 2, 2)

Need <= Available = False

So, the system will move to the next process.

4. For Process P3, Need = (7, 3, 3) Available = (4, 2, 2)

Need <= Available = False

So, the system will move to the next process.

5. For Process P4, Need = (0, 0, 0) Available = (4, 2, 2)

Need <= Available = True Request of P4 is

granted. Available = Available + Allocation

= (4, 2, 2) + (1, 1, 2)

= (5, 3, 4) now, (New Available)

6. Now again check for Process P2, Need = (5, 0, 1) Available = (5, 3, 4)

Need <= Available = True Request of P2 is

granted. Available = Available + Allocation

= (5, 3, 4) + (4, 0, 1)

= (9, 3, 5) now, (New Available)

7. Now again check for Process P3, Need = (7, 3, 3) Available = (9, 3, 5)

Need <= Available = True The request

for P3 is granted.

Available = Available + Allocation

= (9, 3, 5) + (0, 2, 0) = (9, 5, 5)

8. Now again check for Process P0, = Need (3, 2, 1)  
= Available (9, 5, 5)

# Deadlocks

BCA-II(2024-25)

Need  $\leq$  Available = True

So, the request will be granted to P0.Safe

sequence: < P1, P4, P2, P3, P0>

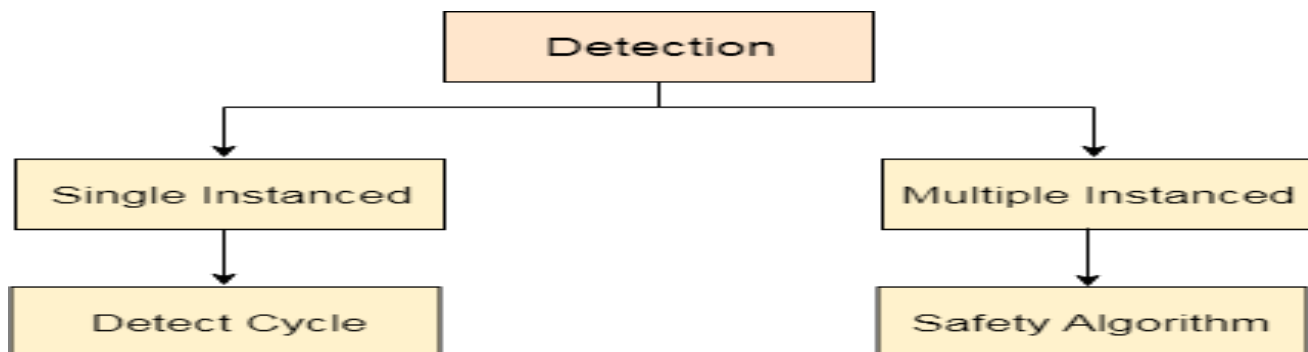
**The system allocates all the needed resources to each process. So, we can say that the system is in a safe state.**

The total amount of resources will be calculated by the following formula:

**The total amount of resources= sum of columns of allocation +Available**  
 $= [8 \ 5 \ 7] + [2 \ 1 \ 0] = [10 \ 6 \ 7]$

## 4) Deadlock Detection and Recovery:

- This approach let the processes fall in deadlock and then periodically check whether deadlock occur in the system or not. If it occurs then it applies some of the recovery methods on the system to resolve deadlock.
- 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. And we know that, OS can detect the deadlocks with the help of Resource allocation graph.



- In single instance 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 (Not both considered at a time)



# Deadlocks

BCA-II(2024-25)

## Deadlock recovery by considering Resource-

### ➤ Preempt the resource:

We can snatch or take 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:

- We know that, a system passes through various states to get into the deadlock state. The operating system can roll back 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 resource allocations to get into the previous safe state.

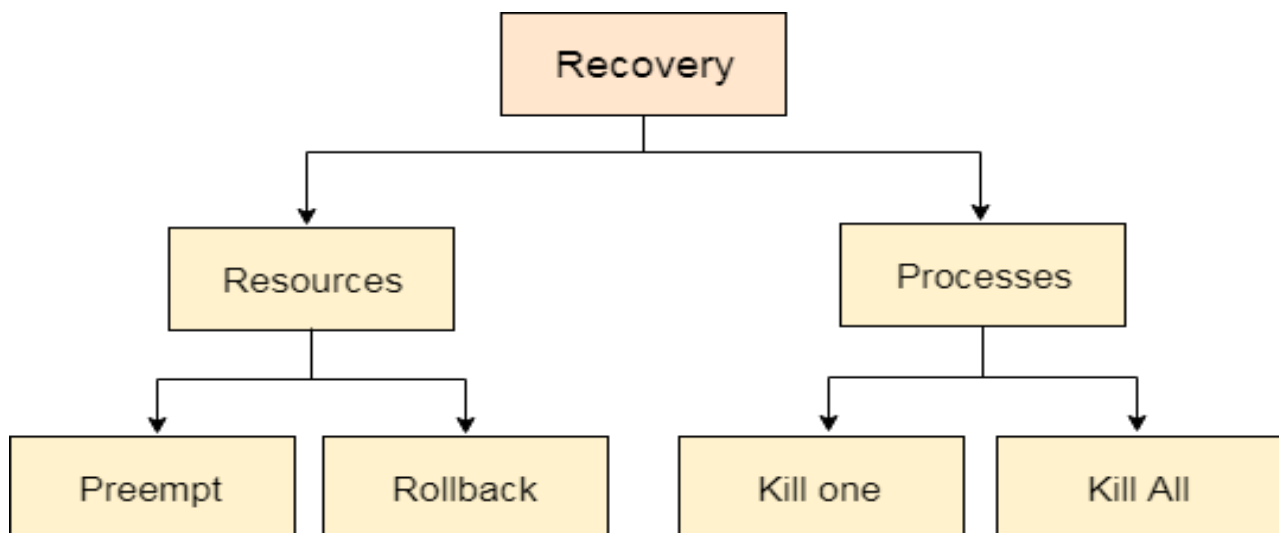
## Deadlock recovery by considering Process-

### ➤ Kill a process:

Killing a process can solve our deadlock 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.



## Unit-II. Memory Management

### Memory Management:

- Memory management is the most important functionality of an operating system which handles or manages primary memory and moves processes back and forward between main memory and disk during execution.
- It helps OS to keep track of every memory location, irrespective of whether it is allocated to some process or it remains free.
- Memory Management is the process of controlling and coordinating computer memory, assigning memory portions (blocks) to various running programs to optimize the overall performance of the system.
- In memory management, OS 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.

### Use of Memory Management:

Here, are reasons for using memory management:

- 1) It allows you to check how much memory needs to be allocated to processes that decide which process should get memory at what time.
- 2) It update the status of memory whenever memory is get allocated or freed (unallocated)
- 3) It allocates the space to application routines.
- 4) It also make sure that these applications do not interfere with each other.
- 5) It places the programs in memory so that memory is fully utilized.

### Process Address Space:

- The process address space is the set of logical addresses of a process in main memory allocated by OS while come into main memory.
- The operating system takes care of mapping (to keep record) the logical addresses to physical addresses at the time of memory allocation to the program.

**There are three types of addresses used in a program before and after memory is allocated –**

#### 1) Symbolic addresses:

- The addresses used in a source code of a program is called ‘Symbolic addresses’.
- In source code of program, addresses given to different variables, constants, and instruction labels are referred as symbolic address space.

#### 2) Relative addresses:

- At the time of compilation, a compiler converts symbolic addresses into relative addresses.

#### 3) Physical addresses:

- When a program is loaded into main memory, then loader generates physical addresses for the loaded program i.e. at the time of program execution.

#### Logical Address space:

- The set of all logical addresses generated by CPU during execution program execution is referred to as a logical address space.

#### Physical address space:

- The set of all physical addresses corresponding to these logical addresses is referred to as a physical address space.

### Address binding:

- Address binding is the process of mapping (to keep record) from one address space to another address space.
- Logical address is an address generated by the CPU during program execution, whereas Physical Address refers to the actual location in the memory unit (the one that is loaded into memory).
- The logical address undergoes translation by the MMU (Memory Management Unit) or Address Translation Unit. The output of this process is the appropriate physical address or the location of code/data in RAM.

**An address binding can be done in three different ways:**

**1) Compile Time –**

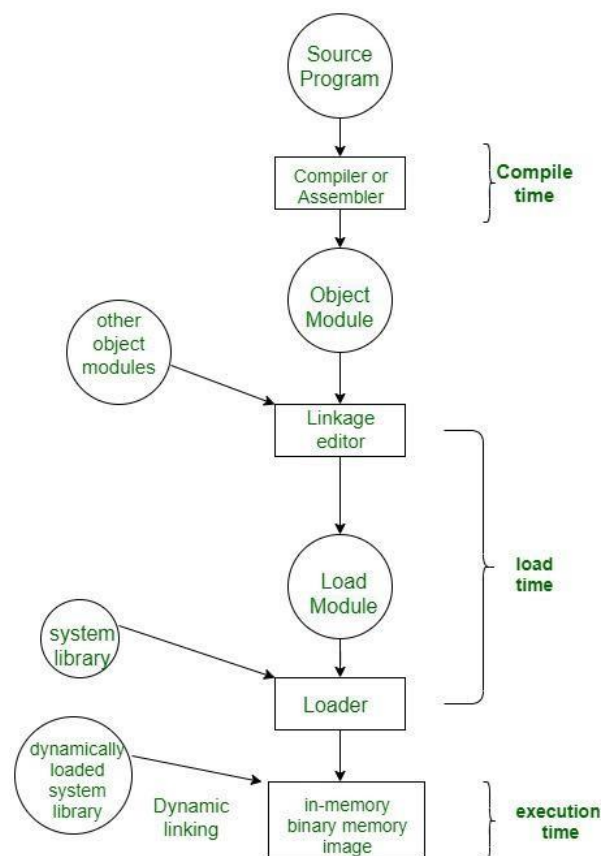
- If the address binding is done at compile time of program is called ‘Compile time address binding’.
- During compile time, where process will reside in memory, then an absolute address is generated. i.e the physical address is embedded to the executable of the program during compilation.
- Loading the executable program as a process in memory is very fast. But if the generated address space is preoccupied by other processes, then the program crashes then it is necessary to recompile the program again to change the address space.

**2) Load time –**

- If the address binding is done at Load time of program is called ‘Load time address binding’.
- If it is not known at the compile time where the process will reside in memory, then a relocatable address will be generated by loader.
- The loader translates the relocatable address to an absolute address.
- To generate an absolute address, the loader adds the base address of the process in main memory to all logical addresses. In this, if the base address of the process changes, then we need to reload the process again.

**3) Execution time –**

- If the address binding is done at execution time of program is called ‘execution time address binding’.
- In this case, the instructions are in main memory and are being processed by the CPU. Here, additional memory may be allocated and/or deallocated at this time. This is used if a process can be moved from one memory to another during execution (dynamic linking-Linking that is done during load or run time).  
e.g – Compaction.



**Linking & Loading:**

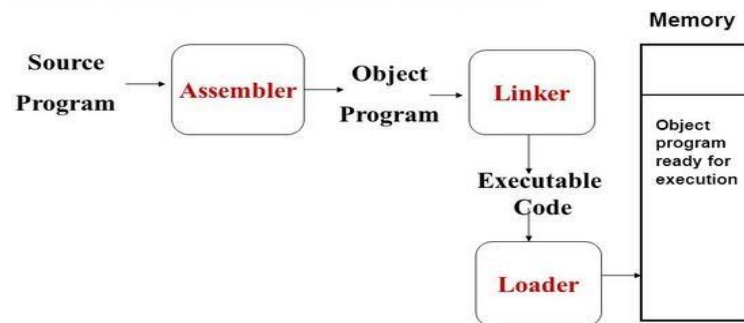
- Linker and Loader are the utility programs that play an important role in the execution of a program.
- Linker takes the object codes as input generated by the compiler/assembler and combines them to generate the executable module.
- On the other hand, the loader loads this executable module to the main memory for execution.

➤ **Loading:**

- Bringing the program from secondary memory to main memory is called Loading.

➤ **Linking:**

- Establishing the linking between all the modules or all the functions of the program in order to continue the program execution is called linking.
- Following diagram shows linking and loading of program-



**Types of Loading:**

There are two types of loading viz.- 1) Static Loading 2) Dynamic Loading

Let's see these types in details-

**1) Static Loading-**

- **Loading the entire program into the main memory before start of the program execution is called as 'static loading'.**
- In this loading, complete program is linked and compiled without dependency of an external program.
- In static loading, absolute data and program are just loaded into the memory to start execution.
- In this case, linker combines the object program with other object modules to make a single program.
- In static loading, processing speed is faster because no files are updated during the processing time. Also code may or may not be executed once it is loaded into the memory.
- If the static loading is used then accordingly static linking is applied.
- Statically linked program takes constant load time to load program into the memory for execution.
- Static linking is performed by programs called linkers as the last step in compiling a program.
- Static loading is done only in the case of structured programming languages such as C.

**Drawback of static loading:**

- The primary drawback is the wastage of memory since, once the code is loaded, it might or might not be executed.
- Inefficient utilization of memory because whether it is required or not required entire program is brought into the main memory.
- Program execution is faster.

**2) Dynamic Loading:**

- **Loading the program into the main memory on demand is called as 'dynamic loading'.**
- In dynamic loading, the modules are loaded dynamically whenever required. The developer provides a reference of module to the rest of the work which being in execution state.
- In this case, loading of data and information takes bit by bit to the running program.
- Here, the linking process takes place dynamically in relocatable form. Also, data or information is loaded into the memory only when it is needed in the program.
- In dynamic loading, processing speed is slower because files are uploaded at the time of processing.
- In dynamic loading, dynamic linking is performed at run time by the operating system.
- In dynamic loading, program execution will be slower.
- Dynamic loading takes place in the case of object-oriented programming languages such as C++, Java, C# etc.

**Advantages of Dynamic loading:**

- In dynamic linking, individual shared modules can be updated and recompiled at run time of program.
- In dynamic linking, load time might be reduced if the shared library code is already present in memory.
- The primary benefit of dynamic loading is efficient memory utilization.

## Overlays:

- Overlay is a technique to run a program which is bigger in size than the physical memory (RAM) by keeping only those instructions and data that are needed at any given time.
- In overlays, entire program is divided into several parts (modules) in such a way that not all parts (modules) need to be in the physical memory at the same time.
- We know that, in fixed partitioning of memory the size of partition is fixed. Therefore a process of higher size than partition size never execute properly and solution over such problem is 'Overlays' where entire process can be divided into several sub-parts of smaller size such that it would then execute on fixed partitioned memory.
- The concept of overlays is that whenever a process is running it will not use the complete program at the same time, it will use only some part of it. Then overlays concept says that whatever part you required, you load it an once the part is used, then you just unload it, means just pull it back and get the new part you required and run it.

### Advantage of overlays–

- Overlays reduce the memory requirement.
- Overlays reduce time requirement.

### Disadvantage of overlays –

- Programming design of overlays structure is complex and not possible in all cases
- In overlays, programmer must know memory requirement.

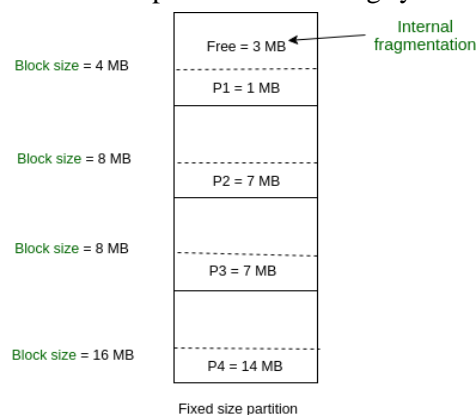
## Memory Partitioning:

- The process of dividing computers main memory into different parts (memory blocks) is called as 'Memory Partitioning'.
- We know that, memory management is responsible for allocating and managing computer's main memory. Memory Management function keeps track of the status of each memory location, either allocated or free to ensure effective and efficient use of Primary Memory.
- There are two Memory Management Techniques: Contiguous, and Non-Contiguous. In Contiguous Technique, executing process must be loaded entirely in main-memory. Contiguous Technique can be divided into:
  - 1) **Fixed (or static) partitioning.**
  - 2) **Variable(ordynamic)partitionig.**

Let's see these types in details-

### 1) Fixed (or static) partitioning:

- This is the oldest and simplest technique that divides total RAM size into several fixed sized parts or memory blocks such that it would then use to put more than one processes in the main memory depending size of processes. In this partitioning, number of partitions (non-overlapping) in RAM are fixed but size of each partition may or may not be same. As it is contiguous allocation, hence no spanning is allowed. Here partition are made before execution of processes or during system configure. As shown in fig.



- As shown in above figure, first process is only consuming 1MB out of 4MB in the main memory.
- Hence, Internal Fragmentation in first block is  $(4-1) = 3\text{MB}$ .
- Sum of Internal Fragmentation in every block =  $(4-1)+(8-7)+(8-7)+(16-14) = 3+1+1+2 = 7\text{MB}$ .
- Suppose process P5 of size 7MB comes. But this process cannot be accommodated in spite of available free space because of contiguous allocation (as spanning is not allowed). Hence, 7MB becomes part of External Fragmentation.

#### Advantages of Fixed Partitioning –

- 1) Easy to implement:  
Algorithms needed to implement Fixed Partitioning are easy to implement.
- 2) Processing of Fixed Partitioning require lesser excess and indirect computational efforts.

#### Disadvantages of Fixed Partitioning –

- 1) Internal Fragmentation:  
Main memory use is inefficient. Any program, no matter how small, occupies an entire partition. This can cause internal fragmentation.
- 2) External Fragmentation:  
The total unused space (as stated above) of various partitions cannot be used to load the processes even though there is space available but not in the contiguous form (as spanning is not allowed).
- 3) Limit process size:  
Process of size greater than size of partition in Main Memory cannot be accommodated. Partition size cannot be varied according to the size of incoming process's size.
- 4) Limitation on Degree of Multiprogramming:  
Partition in Main Memory are made before execution or during system configure. Main Memory is divided into fixed number of partition. Number of processes greater than number of partitions in RAM is invalid in Fixed Partitioning.

#### 2) Variable (or dynamic) partitioning:

- Variable partitioning is also a part of Contiguous allocation technique. It is used to overcome the problem faced by Fixed Partitioning.
- In variable partitioning, partitions are **not** made before the execution or during system configure.
- Variable (Dynamic) Partitioning done by following way-
  1. Initially RAM is empty and partitions are made during the run-time according to process's need instead of partitioning during system configure.
  2. The size of partition will be equal to incoming process.
  3. The partition size varies according to the need of the process so that the internal fragmentation can be avoided to ensure efficient utilization of RAM.
  4. Number of partitions in RAM is not fixed and depends on the number of incoming process and Main Memory's size.

##### Dynamic partitioning

Operating system	
P1 = 2 MB	Block size = 2 MB
P2 = 7 MB	Block size = 7 MB
P3 = 1 MB	Block size = 1 MB
P4 = 5 MB	Block size = 5 MB
Empty space of RAM	

Partition size = process size  
So, no internal Fragmentation

### Advantages of Variable Partitioning –

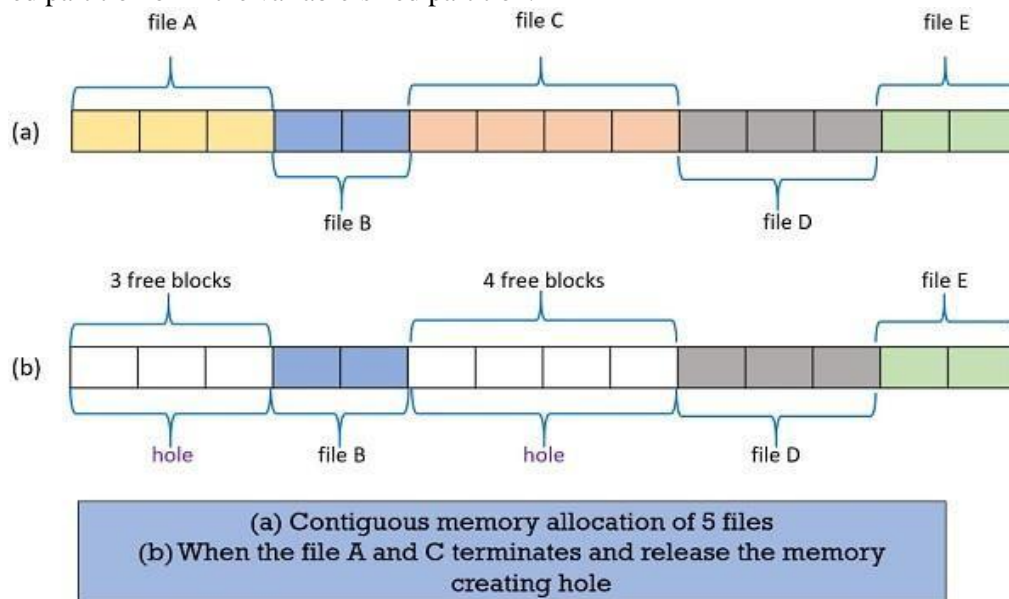
- 1) No Internal Fragmentation:  
In variable Partitioning, space in main memory is allocated strictly according to the need of process, hence there is no case of internal fragmentation. There will be no unused space left in the partition.
- 2) No restriction on Degree of Multiprogramming:  
More number of processes can be accommodated due to absence of internal fragmentation. A process can be loaded until the memory is empty.
- 3) No Limitation on the size of the process:  
In variable partitioning, the process size can't be restricted since the partition size is decided according to the process size.

### Disadvantages of Variable Partitioning –

- 1) Difficult in Implementation:  
Implementing variable Partitioning is difficult as compared to Fixed Partitioning as it involves allocation of memory during run-time rather than during system configure.
- 2) External Fragmentation:  
There will be external fragmentation in spite of absence of internal fragmentation.

### Contiguous Memory allocation-

- Contiguous memory allocation is a method of allocating the memory for process or file in a single contiguous section/part of memory.
- Because of this, all the available memory space resides at the same place together, which means that the free or unused available memory partitions are not distributed in a random fashion here.
- In the Contiguous Memory Allocation, each process is contained in a single contiguous section of memory. In this memory allocation, all the available memory space remains together in one place which implies that the freely available memory partitions are not spread over here.
- In Contiguous memory allocation which is a memory management technique, whenever there is a request by the user process for the memory then a single section of the contiguous memory block is given to that process according to its requirement.
- In this case, memory can be divided either in the fixed-sized partition or in the variable-sized partition in order to allocate contiguous space to user processes.
- In Contiguous memory allocation, when the process arrives from the ready queue to the main memory for execution, the contiguous memory blocks are allocated to the process according to its requirement. Now, to allocate the contiguous space to user processes, the memory can be divide either in the fixed-sized partition or in the variable-sized partition.

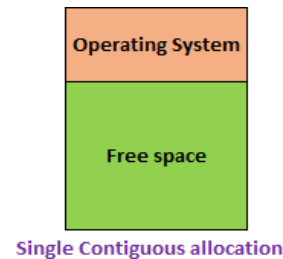


## Memory allocation Strategies: (Partition Allocation Methods in Memory Management)

In the operating system, the following are four common memory management techniques.

- 1) **Single contiguous allocation:** This is simplest allocation method used by MS-DOS. In this case some memory reserved for OS and all remaining is available to a processes.

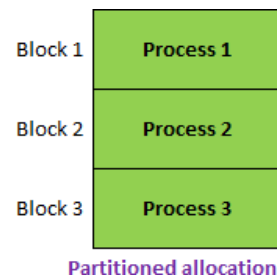
Following diagram shows Single contiguous allocation-



- 2) **Partitioned allocation:**

- In this allocation, entire memory is divided into different blocks or partitions. Then each process is allocated block of memory according to the requirement.
- In Partition Allocation, when there is more than one partition freely available to accommodate a process's request, a partition must be selected. To choose a particular partition, a partition allocation method is needed. A partition allocation method is considered better if it avoids internal fragmentation.

Following diagram shows partitioned allocation-



- 3) **Paged memory management:** In this case, memory is divided into fixed-sized units called 'page frames' which is used in a virtual memory environment.

Following diagram shows paged memory management

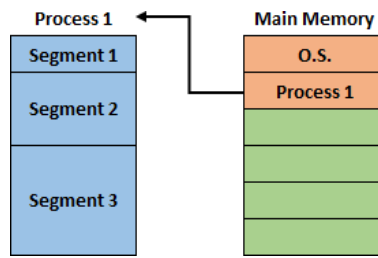
P1 PMT		Memory	
Page	Frame	Frame	Contents
0	5	0	
1	12	1	P2/Page2
2	15	2	
3	7	3	
4	22	4	
		5	P1/Page0
		6	
		7	P1/Page3
		8	
		9	
		10	P2/Page0
		11	P2/Page3
		12	P1/Page1
		13	
		14	
		15	P1/Page2
			⋮
			⋮

- 4) **Segmented memory management:** In this case, memory is divided into different segments (a segment is a logical grouping of the process' data or code).

In this management, allocated memory doesn't have to be contiguous.

Following diagram shows segmented memory management:





Most of the operating systems (for example Windows and Linux) use Segmentation with Paging. A process is divided into segments and individual segments have pages.

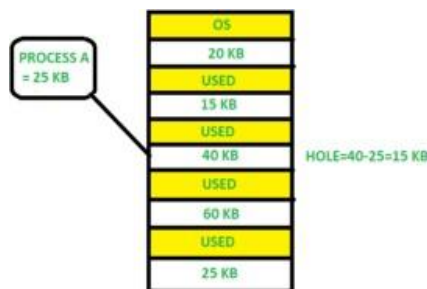
### Process Placement Algorithms:

- When it is time to load a process into the main memory and if there is more than one free block of memory of sufficient size then the OS decides which free block is allocate to process.
- For that different process placement algorithms are used which are as follow-
  1. First Fit
  2. Best Fit
  3. Worst Fit
  4. Next Fit

Let's see all these in details-

#### 1) First Fit:

- In the first fit algorithm, the partition is allocated to process which is the first sufficient block from the top of Main Memory.
- It scans memory from the beginning and chooses the first available block that is large enough for process. Thus it allocates the first hole that is large enough and sufficient to process.



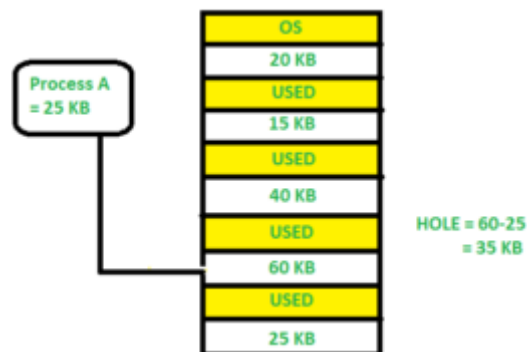
#### 2) Best Fit:

- The best fit algorithm allocates the process to such partition which is the smallest sufficient partition among the free available partition.
- It searches the entire list of holes to find the smallest hole whose size is greater than or equal to the size of the process.



### 3) Worst Fit:

- The worst fit algorithm allocates the process to such partition which is the **largest sufficient among the freely available partitions in the main memory.**
- It is opposite to the best-fit algorithm. It searches the entire list of holes to find the largest hole and allocate it to process.



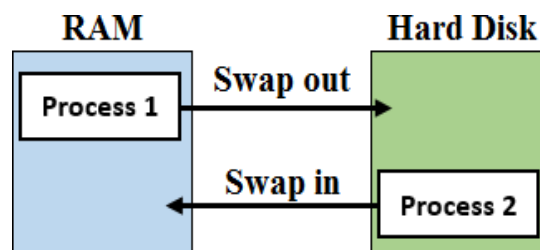
### 4) Next Fit:

- Next fit is similar to the first fit but it will search for the first sufficient partition from the last allocation point.

### Swapping:

- Swapping is a mechanism in which a process can be swapped out temporarily from main memory and moved to the secondary storage (disk) and make that memory available to other processes. At some later time, the system swaps back the process from the secondary storage to main memory.
- Though performance of operating system is usually affected by swapping process but it helps to run multiple and big processes in parallel and that's why Swapping is also known as a technique for memory compaction.
- Swapping is a memory management scheme in which any process can be temporarily swapped from main memory to secondary memory so that the main memory can be made available for other processes. It is used to improve main memory utilization. In secondary memory, the place where the swapped-out process is stored is called 'swap space'.
- Also, swapping in operating system can be used to access the data present in the hard disk and bring it to RAM so that the application programs can use it. The thing to remember is that swapping is used only when data is **not** present in RAM.
- The concept of swapping has divided into two more concepts: Swap-in and Swap-out.
- Swap-out: It is a method of removing a process from RAM and adding it to the hard disk.
- Swap-in: It is a method of removing a process from a hard disk and putting it back into the RAM.

Following diagram shows swapping process:



**Example:** Suppose the new process size is 2048KB and there is a standard hard disk having data transfer rate of 1Mbps. Now we will calculate how long it will take to transfer from main memory to secondary memory and vice versa.

- ➔ User process size is 2048Kb  
Data transfer rate is 1Mbps = 1024 kbps  
Time = process size / transfer rate  
Time = 2048 / 1024

Time = 2 seconds

Time = 2000 milliseconds

Now for swap-in and swap-out, the process will take 4000 milliseconds.

### Advantages of Swapping:

- 1) It helps the CPU to manage multiple processes within a single main memory.
- 2) It helps to create and use virtual memory.
- 3) Swapping allows the CPU to perform multiple tasks simultaneously. Therefore, processes do not have to wait very long before they are executed.
- 4) It improves the main memory utilization.

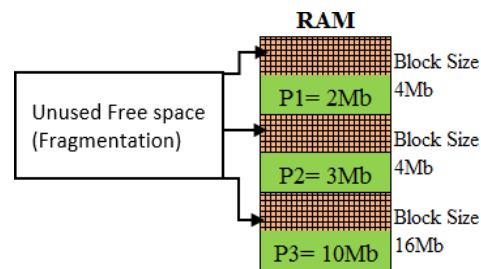
### Disadvantages of Swapping:

- 1) While performing swapping activity, if the computer system loses power then there is loss of all information related with the program/process.
- 2) If the swapping algorithm is not good, the composite method can increase the number of Page Fault and decrease the overall processing performance.

### Fragmentation:

- As processes are loaded and removed from main memory, the free memory space is broken into little pieces. It happens after sometimes that processes cannot be allocated to memory blocks considering their small size and memory blocks remains unused. This problem is known as 'Fragmentation'.
- Fragmentation is an unwanted problem where the memory blocks cannot be allocated to the processes due to their small size and the blocks remain unused.
- It can also be done when the processes are loaded and removed from the memory they create free space or hole in the memory and these small blocks cannot be allocated to new upcoming processes and results in inefficient use of memory called 'Fragmentation'.

Following diagram shows fragmentation-



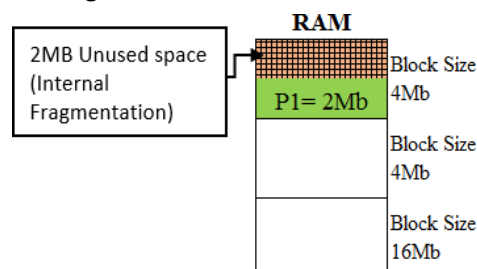
Basically, there are two types of fragmentation:

- 1) Internal Fragmentation
- 2) External Fragmentation

Let's see these types in details-

#### 1) Internal Fragmentation-

- In this fragmentation, the small size process is allocated to a memory block having big size. Due to this some part of the memory is left unused and this cause 'internal fragmentation.'
- Example: Suppose there is fixed partitioning (i.e. the memory blocks are of fixed sizes) is used for memory allocation in RAM. The sizes of memory blocks are 4MB, 4MB, and 16MB.
- Now, suppose a process P1 of size 2MB comes and it gets allocated in first memory block of size 4MB. So, the 2MB memory remains free in this block & is wasted and this space can't be utilized for allocating memory to some other process (*Since, in fixed partitioning only one process is allocated within only one block*). This is called 'internal fragmentation'. Which is shown in following diagram-

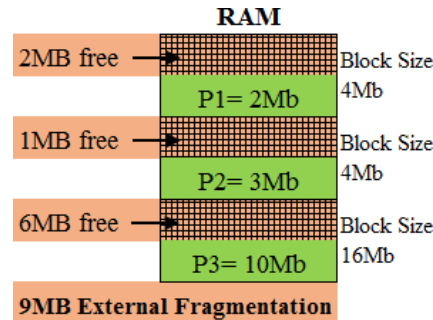


## 2) External Fragmentation-

- In this fragmentation, although we have total space available that is needed by a process still we are not able to put that process in the memory because that space is not contiguous. This is called external fragmentation.

**Example:** Consider following diagram, if there are processes P1, P2, and P3 with sizes 2MB, 3MB, and 10MB respectively are allocated memory blocks of size 2MB, 4MB and 8MB respectively.

So, now if we closely analyze this situation then process P1 (unused 2MB), P2 (unused 1MB) and P3 (unused 10MB) are causing internal fragmentation. So, a **total of 9MB is unused** due to internal fragmentation in every block. Thus, 9MB is external fragmentation.

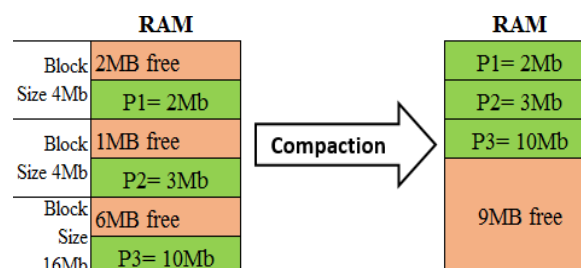


Now, suppose a new process of 9 MB comes. Though we have a total space of 9MB still we can't allocate this memory to the process because this memory is not contiguous. This is called external fragmentation.

## Compaction:

- We know that, in memory management, swapping creates multiple fragments in the memory because of the processes moving in and out. There will be more unused or free spaces available in memory but this free space not usable to allocate new process since it is not in contiguous form.
- Compaction is a process in which such free spaces are collected or combined in a large memory chunk to make some contiguous space available for new processes.
- Compaction refers to combining all empty or unused spaces of memory together and make available contiguous memory space for new processes.
- Compaction helps to solve the problem of fragmentation, but it requires too much of CPU time.
- It moves all occupied memory areas to one end and leaves one large free space for incoming process.
- The compaction technique overcomes the problem of fragmentation. Due to this reason the, compaction is known as 'defragmentation'.
- The goal of compaction is to shuffle the memory content such that all free memory comes together in one large block.
- Definition – The process of putting the used partitions at one end and creating one big free area at the other end for the new process is called 'Compaction'**

Following diagram shows compaction-



**Example-** In above diagram,

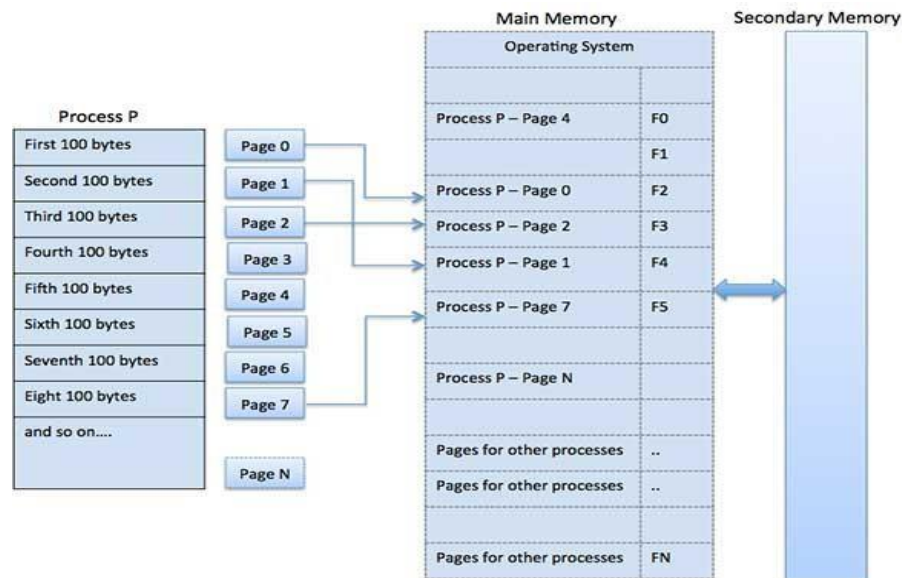
Before compaction- A new process of size 9MB cannot allocate memory space because the remaining (unused) memory is not contiguous.

After compaction- A new process of size 9MB easily allocate memory space because of compaction all remaining (unused) memory becomes contiguous.

## Paging:

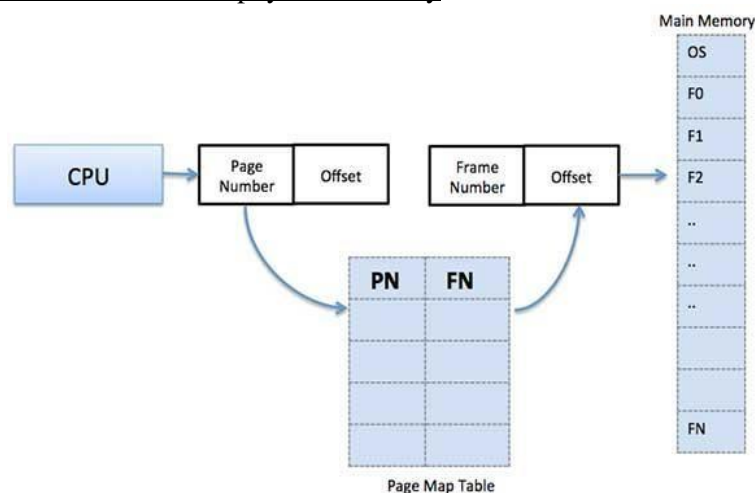
- A computer can address or show more memory than the amount physically installed memory on the system. This extra memory is actually called virtual memory and such virtual memory is hard to match with the computer's RAM. Paging technique plays an important role in implementing such virtual memory.
- Paging is a memory management technique in which process address space is broken into blocks of the same size called pages
- In paging, the size of the process is measured in the number of pages.
- Similarly, main memory is also divided into small fixed-sized blocks of (physical) memory called frames and the size of a frame is kept the same as that of a page to have optimum utilization of the main memory and to avoid external fragmentation.

Following diagram shows paging:



### Basic terminology used in paging:

- Page address is also called logical address and represented by page number and the page offset (page size). Therefore we write:  $\text{Logical Address} = \text{Page number} + \text{page offset}$
- Frame address is also called physical address and represented by a frame number and the page offset. Therefore we write:  $\text{Physical Address} = \text{Frame number} + \text{page offset}$
- In paging a data structure called 'page map' table is also used to keep track of the relation between a page of a process to a frame in physical memory.



- When the system allocates a frame to any page, it translates this logical address into a physical address and create entry into the page table to be used throughout execution of the program.
- When a process is to be executed, its corresponding pages are loaded into any available memory frames. Suppose you have a program of 8Kb but your memory can accommodate only 5Kb at a given point in

time, then the paging concept will come into picture. In such situation, the operating system (OS) will move idle or unwanted pages of memory to secondary memory to free up RAM for other processes and brings them back when needed by the program.

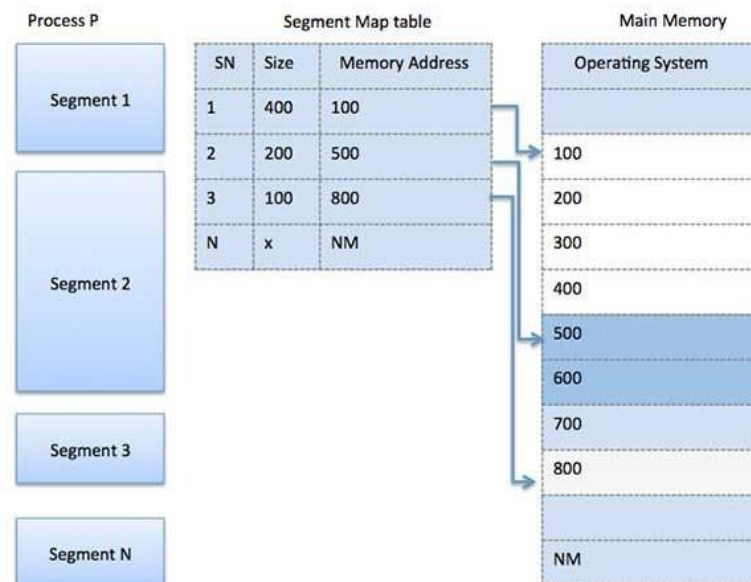
- This process continues during the whole execution of the program where the OS keeps removing idle pages from the main memory and write them onto the secondary memory and bring them back when required by the program.

### Points to remember about paging-

- Paging concept basically used to manage virtual memory.
- Paging reduces external fragmentation, but still suffer from internal fragmentation.
- Paging is simple to implement and it is efficient memory management technique.
- Due to equal size of the pages and frames, swapping becomes very easy.
- Page table requires extra memory space, so may not be good for a system having small RAM.

### Segmentation:

- Segmentation is also memory management technique in which each job or process is divided into several segments of different sizes, one each segment contains process code pieces that perform specific 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 available block of 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 process segment contains the program's main function, utility functions, data structures, and so on
- In this case, the operating system maintains a segment map table for every process and that contains segment numbers, their size and corresponding memory locations in main memory.
- For each segment, the segment map 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.
- Following diagram shows segmentation-



### Advantages of Segmentation:

- In segmentation there is no internal fragmentation.
- In segmentation there is less overhead memory.
- It is easier to relocate segments than entire address space.
- The segment table is of lesser size as compare to the page table in paging.

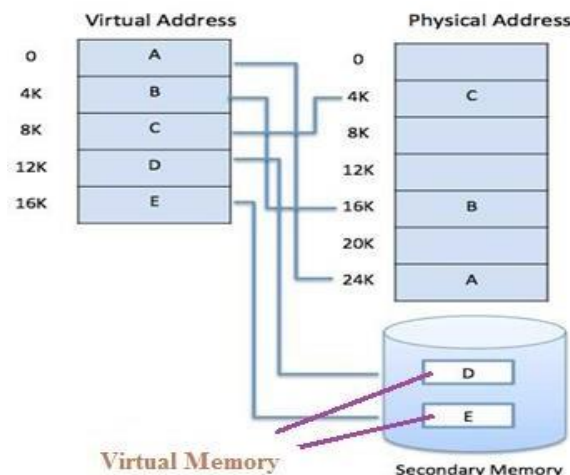
### Disadvantages of segmentation:

- It can have external fragmentation.
- It is difficult to allocate contiguous memory to variable sized partition.
- Segmentation have costly memory management algorithms.



## Virtual Memory:

- A computer can address or show more memory than the amount of physically installed on the system. This extra memory is actually called 'virtual memory'.
- Basically virtual memory is a section of a hard disk that setup's to match the computer's RAM.
- The main advantage of virtual memory is if any program is larger than RAM then it is not capable to load directly into RAM but it can be load on to such virtual memory from where it will then load on to RAM using any memory management technique.
- **The main purpose of Virtual memory is:**
  - It allows us to extend the physical memory (RAM) by using disk space.
  - It plays important role in memory protection, because each virtual address is translated to a physical address.
  - User written error handling routines which are rarely used only when an error occurred therefore such routines can be stored on virtual memory.
  - Many tables are assigned a fixed amount of address space even though only a small amount of the table is actually used such tables can also be stored on virtual memory.
  - Because of virtual memory, a program will execute successfully still it is loaded partially into main memory.
  - Because of virtual memory, less number of I/O would be needed to load or swap each user program into memory.
  - Virtual memory increases CPU utilization as virtual memory has capability to load user programs. Therefore each user program could take less physical memory & hence more programs could be run at the same time.
- The Modern microprocessors system use a memory management unit (MMU) which is built into the hardware. The MMU's job is to translate virtual addresses into physical addresses. Which is shown 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.

### Advantages of Virtual Memory:

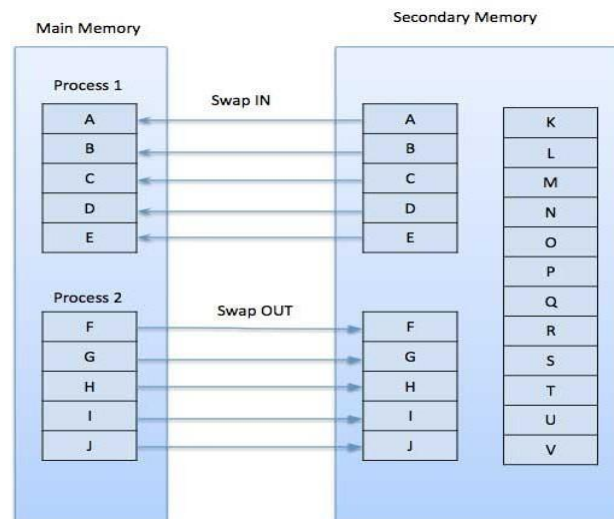
- The degree of Multiprogramming will be increased. (i.e. multiple programs runs simultaneously)
- User can run large application with less use of RAM.
- There is no need to buy more memory RAMs.

### Disadvantages of Virtual Memory:

- The system becomes slower since swapping takes time.
- It takes more time in switching between applications.
- The user will have the lesser hard disk space for its use.

## 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 from the disk or any of the new program's pages from the main memory.
- Operating system just begins executing the new program after loading the first page and fetches that program's pages on demand for execution which is shown in following diagram-



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

### Advantages of demand paging-

- 1) Demand paging allow to use large virtual memory.
- 2) Demand paging is more efficient method to use memory.
- 3) In demand paging there is no limit on degree of multiprogramming.

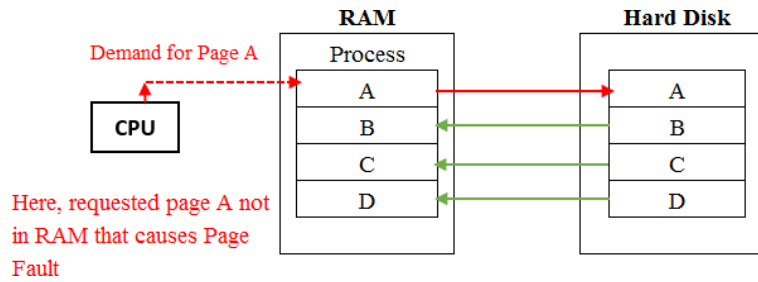
### Disadvantages of demand paging-

- 1) In demand paging, number of tables and the amount of processor overhead for handling page interrupts are greater.

## Page Fault:

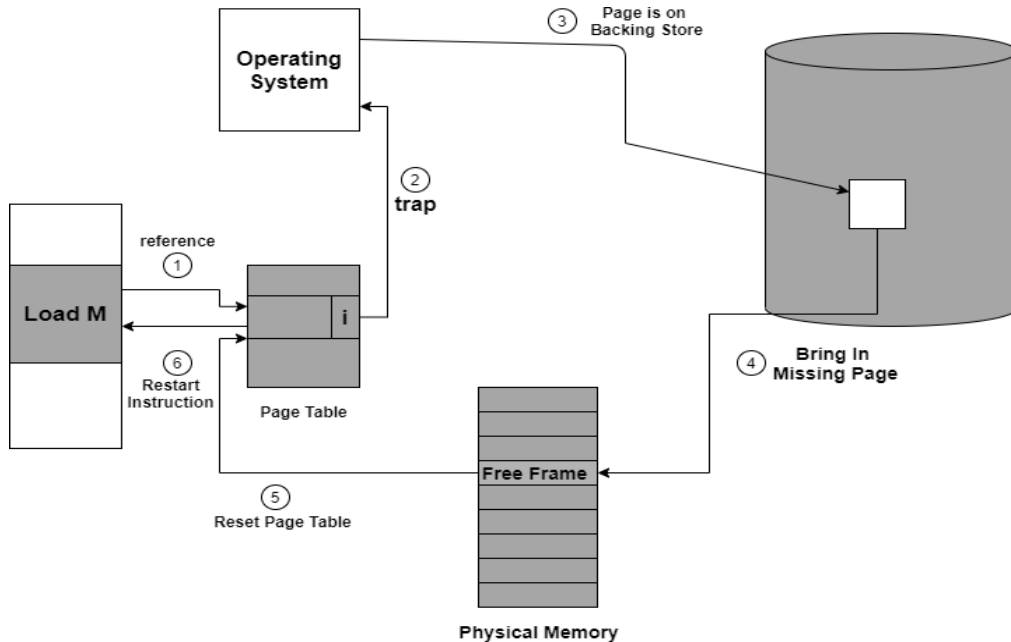
- While executing a program, if the processor demands for a page which is not available in the main memory because it was swapped out a little ago, then processor treats this as invalid memory reference called 'page fault' .
- Page fault is more likely run time error (exception) which is mainly occurs when processor tries to access the data or the code that is in the address space of the program, but that data is not currently located in the RAM of the system.
- Whenever any page fault occurs, then the required page has to be fetched from the secondary memory into the main memory. In case if again required page is not loaded into the memory, then a page fault trap arises.
- The page fault mainly generates an exception, which is used to notify the operating system that it must have to retrieve the "pages" from the virtual memory in order to continue the execution. Once all the data is moved into the physical memory then program continues its execution normally.
- The Page fault process takes place in the background and thus goes unnoticed by the user.
- Following diagram shows page fault-





### Handling Page Fault:

Given below is the simple procedure to handle the page fault:

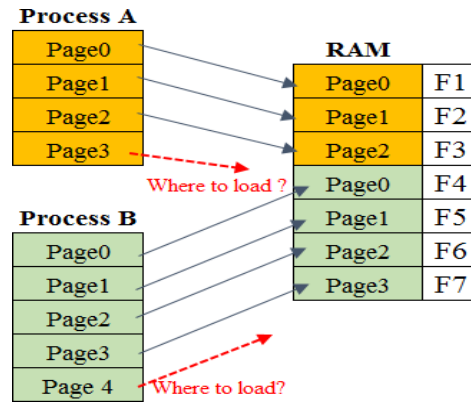


- We know that, if the processor demands for a page which is not available in the main memory because it was swapped out a little ago, then processor treats this as invalid memory reference called 'page fault'.
  - If CPU try to access a page again that is already marked as invalid then it also causes a Page Fault.
  - And such page fault exception should be handle properly otherwise it causes Operating system failure.
- Let us understand the procedure to handle the page fault as shown in above diagram:

- 1) First of all, scanning is done on internal table (that is usually the process control block) for this process in order to determine whether the reference was valid or invalid memory access.
- 2) If the scanned reference is invalid, then it will terminate the process. If the reference is valid, but CPU have not unable to access that page then it notify to operating system about page address.
- 3) Then operating system search for the demanded page on secondary storage device.
- 4) After getting demanded page, it will then bring or load that page to frame of RAM.
- 5) Now the internal page table of process is modified and that mainly indicates the page is now in memory.
- 6) Thus, now the processor can access the page as it now in main memory.

### Page replacement policies:

- If an operating system that uses paging for memory management then a page replacement algorithm is needed to decide which page should to be replaced when new page comes in?
- As we have studied in Demand Paging, only certain pages of a process are loaded initially into the memory. This allows us to get more processes into memory at the same time, but what happens when a process requests for more pages and no free memory is available to bring them in? this problem is shown in following figure-



- Following steps can be taken to solve this problem :
  - 1) Put a process in the wait queue, until any other process finishes its execution thereby freeing frames.
  - 2) Or, remove some other process completely from the memory to free frames.
  - 3) Or, find some pages that are not being used right now, move them to the disk to get free frames. This technique is called 'Page replacement' and it is most commonly used.

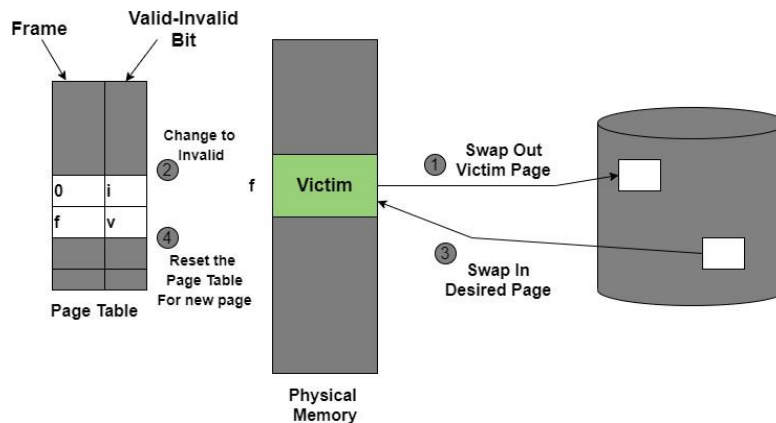
### Page replacement:

- If a process requests a new page and supposes there are no free frames available in RAM, then the Operating system needs to decide which existing page to replace from frame. Here, then operating system use any page replacement algorithm in order to select the victim frame (frame from which page can be replaced) and such process is called 'Page replacement'.
- The Operating system must then write the victim frame to the disk and then read the upcoming page & write into that frame then update the page tables. And all these require double the disk access time.
- Page replacement prevents the over-allocation of the memory by modifying the page-fault service routine.
- To reduce the overhead of page replacement a modify bit (dirty bit) is used in order to indicate whether each page is modified.
- Page replacement technique provides complete separation between logical memory and physical memory.
- In Virtual Memory Management, Page Replacement Algorithms play an important role. The main objective of all the Page replacement policies is to decrease the maximum number of page faults.

### Basic of Page Replacement Algorithm in OS:

- If main memory is full i.e. no frame is available to load new page but to continue execution of process, existing page from frame needs to be replace by new page. For that, page replacement algorithms are used.
- Page replacement algorithm provide a way to replace existing page from frame and make that frame available to new upcoming page.
- Page Replacement technique or algorithm uses the following approach to replace the page-
  - 1) First of all, find the location of the desired page on the disk.
  - 2) Find a free Frame:
    - a) If there is a free frame, then use it for new page.
    - b) If there is no free frame then make use of the page-replacement algorithm in order to select the victim frame.
    - c) Then write the victim frame to the disk and then make the changes in the page table and frame table accordingly.
  - 3) After that put the desired page into the newly freed frame and then change the page and frame tables.
  - 4) Restart the process.

The overall process of page replacement is shown in following diagram-



## Page replacement algorithms:

- This algorithm helps to decide which pages must be swapped out from the main memory in order to create a room for the incoming page.
- The main objective of all the Page replacement algorithm is to decrease the maximum number of page faults.
- Various Page Replacement algorithms used in the Operating system are as follows:
  - 1) FIFO page replacement algorithm
  - 2) LRU (Last Recently Used) page replacement algorithm
  - 3) Optimal page replacement algorithm (OPR)

Let's see all these in details-

### 1) FIFO page replacement algorithm:

- This is the simplest page replacement algorithm.
- In this algorithm, the operating system keeps track of all pages which are in queue of the memory.
- In this case, the oldest page is at front of the queue. When a new page demands for frame then the page which is at front of the queue is selected for removal.
- This algorithm mainly replaces the oldest page that has been present in the main memory for the longest time (i.e. old page).
- As new pages are requested and are swapped in, they are added to the tail (rear part) of a queue and the page which is at the head (front part) becomes the victim.
- This is not an effective way of page replacement but it can be used for small systems.

#### Advantages-

- This algorithm is simple and easy to use.
- FIFO does not cause more overhead.

#### Disadvantages-

- There is an increase in page faults as page frames increases.
- The performance of this algorithm is the worst.

#### Example-

1) Consider page reference string 1, 3, 0, 3, 5, 6, 3 with 3 page frames. Find number of page faults?

#### Solution-

Given Things- Page reference string 1, 3, 0, 3, 5, 6, 3      Frame Size=3

	1	3	0	3	5	6	3
			0	0	0	0	3
		3	3	3	3	6	6
	1	1	1	1	5	5	5
Empty Frame	miss	miss	miss	hit	miss	miss	miss
Page Fault count-	1	1	1	0	1	1	1
Total page fault = 6							

Initially all slots are empty, so when 1, 3, 0 came they are allocated to the empty slots → **3 Page Faults.**

Next, CPU is demanding for page 3, it is already in memory so → **0 Page Fault**

Next, CPU is demanding for page 5, it is not available in memory so it replaces the oldest page slot i.e 1 → **1 Page Fault.**

Next, CPU is demanding for page 6, it is not available in memory so it replaces the oldest page slot i.e 3 → **1 Page Fault.**

Lastly, CPU is demanding for page 3, it is not available so it replaces the oldest page slot i.e 0 → **1 page Fault**

**Thus, Total number of page faults= 3 + 0 + 1 + 1 + 1 = 6**

## 2) Least Recently Used (LRU) page replacement algorithm:

- In this algorithm page will be replaced which is least recently used.
- This algorithm helps the Operating system to search those pages that are used over a short duration of time frame.
- That is the page that has not been used for the longest time in the main memory will be selected for replacement.
- This algorithm is easy to implement.
- This algorithm makes use of the counter along with the even-page.

### Advantages of LRU-

- It is an efficient technique for page replacement.
- With this algorithm, it becomes easy to identify the faulty pages that are not needed for a long time.

### Disadvantages of LRU-

- It is expensive and has more complexity.
- There is a need for an additional data structure.

**Example-** Consider the page reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 5 with 4 page frames. Find number of page faults.

### Solution-

Given Things- Page reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 5      Frame Size=4

	7	0	1	2	0	3	0	4	2	3	0	3	5
				2	2	2	2	2	2	2	2	2	2
			1	1	1	1	1	4	4	4	4	4	5
		0	0	0	0	0	0	0	0	0	0	0	0
	7	7	7	7	7	3	3	3	3	3	3	3	3
Empty Frame	miss	miss	miss	miss	hit	miss	hit	miss	hit	hit	hit	hit	miss
Page Fault count-	1	1	1	1	0	1	0	1	0	0	0	0	1
Total page fault = 7													

Initially all slots are empty, so when 7, 0, 1, 2 came they are allocated to the empty slots → **4 Page Faults.**

Next, CPU is demanding for page 0, it is already in memory so → **0 Page Fault**

Next, CPU is demanding for page 3, it is not available in memory so it replaces least recently used page i.e 7 → **1 Page Fault.**

Next, CPU is demanding for page 0, it is already in memory so → **0 Page Fault**

Next, CPU is demanding for page 4, it is not available in memory so it replaces least recently used page i.e 1 → **1 Page Fault.**

Next, CPU is demanding for page 2, it is already in memory so → **0 Page Fault**

Next, CPU is demanding for page 3, it is already in memory so → **0 Page Fault**

Next, CPU is demanding for page 0, it is already in memory so → **0 Page Fault**

Next, CPU is demanding for page 3, it is already in memory so → **0 Page Fault**

Lastly, CPU is demanding for page 5, it is not available so it replaces least recently used page i.e 4

→ **1 page Fault**

Thus, **Total number of page faults= 7**

### 3) Optimal page replacement algorithm (OPR):

- This algorithm mainly replaces the page that will not be used for the longest time in the future.
- Practical implementation is not possible because we cannot predict in advance those pages that will not be used for the longest time in the future.
- This algorithm leads to less number of page faults and thus is the best-known algorithm
- Also, this algorithm can be used to measure the performance of other algorithms.

#### Advantages of OPR-

- This algorithm is easy to use.
- This algorithm provides excellent efficiency and is less complex.
- For the best result, the implementation of data structures is very easy

#### Disadvantages of OPR-

- In this algorithm future awareness of the program is needed.
- Practical Implementation is not possible because the operating system is unable to track the future request

**Example:** Consider the page references 4,7,6,1,7,6,1,2,7,2,1 with 3 page frame. Find number of page fault?

**Solution-** Given Things- Page reference string 4,7,6,1,7,6,1,2,7,2,1 Frame Size=3

	4	7	6	1	7	6	1	2	7	2	1
			6	6	6	6	6	2	2	2	2
		7	7	7	7	7	7	7	7	7	7
	4	4	4	1	1	1	1	1	1	1	1
<b>Empty Frame</b>	<b>miss</b>	<b>miss</b>	<b>miss</b>	<b>miss</b>	<b>hit</b>	<b>hit</b>	<b>hit</b>	<b>miss</b>	<b>hit</b>	<b>hit</b>	<b>hit</b>
<b>Page Fault count-</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>
<b>Total page fault = 5</b>											

Initially all slots are empty, so when 4, 7, 6 came they are allocated to the empty slots → **3 Page Faults.**

Next, CPU is demanding for page 1, it is not available in memory so it replaces the page that will not be used for the longest time in the future i.e page 4 → **1 Page Fault.**

Next, CPU is demanding for page 7, it is already in memory so → **0 Page Fault**

Next, CPU is demanding for page 6, it is already in memory so → **0 Page Fault**

Next, CPU is demanding for page 1, it is already in memory so → **0 Page Fault**

Next, CPU is demanding for page 2, it is not available in memory so it replaces the page that will not be used for the longest time in the future i.e page 6 → **1 Page Fault.**

Next, CPU is demanding for page 7, it is already in memory so → **0 Page Fault**

Next, CPU is demanding for page 2, it is already in memory so → **0 Page Fault**

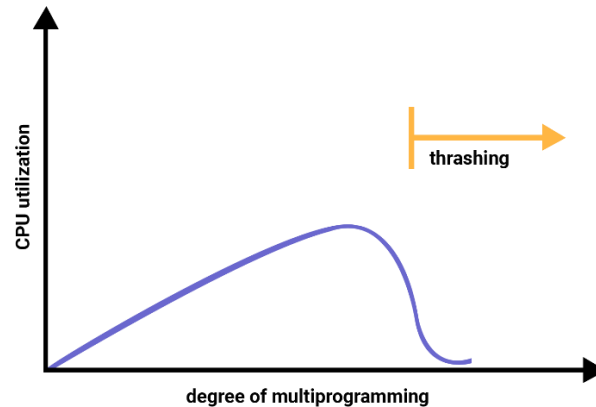
Next, CPU is demanding for page 1, it is already in memory so → **0 Page Fault**

Thus, **Total number of page faults= 5**

- Optimal page replacement is perfect, but not possible in practice as the operating system cannot know future requests. The use of Optimal Page replacement is to set up a benchmark so that other replacement algorithms can be analyzed against it.

### Thrashing:

- In case, if the page fault and swapping happens very frequently at a higher rate, then the operating system has to spend more time swapping these pages. This state in the operating system is termed **thrashing**. Because of thrashing the CPU utilization is going to be reduced.
- Let's understand by an example, if any process does not have the number of frames that it needs to support pages in active use then it will quickly page fault. And at this point, the process must replace some pages. As all the pages of the process are actively in use, it must replace a page that will be needed again right away. Consequently, the process will quickly fault again, and again, and again, replacing pages that it must bring back in immediately. This high paging activity by a process is called thrashing.
- During thrashing, the CPU spends less time on some actual productive work spend more time swapping.



## **Unit-IV. Storage Management, Disk Management**

**Storage Management :-** refers to the processes that help make data storage easier through software or techniques. It tries to improve and maximize the efficiency of data storage resources. Storage management processes can deal with local or external storage such as USBs, SDDs, HDD, the Cloud, etc.,

Storage management techniques or software can be divided into the following four subsets:

1. Performance,
2. Availability,
3. Recoverability, and
4. Capacity.

### **Feature of Storage management:**

Features of storage management are given below:

- Storage management is a process that is used to optimize the use of storage devices.
- Storage management is generally a basic system component of information systems.
- Storage management is used to improve the performance of their data storage resources.

### **Advantage of storage management:**

There are some advantage of storage management which are given below:

1. It is very simple to manage a storage capacity.
2. It is generally take a less time.
3. It is improve the performance of storage system.

### **File concept:**

- A file is a large collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks.
- In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the files creator and user.
- Computer store information in storage media such as disk, tape drives, and optical disks. The operating system provides a logical view of the information stored in the disk. This logical storage unit is a “file”.
- The information stored in files are non-volatile, means they are not lost during power failures. A file is named collection of related information that is stored on physical storage.
- Every file has a structure defined by its owner or creator and depends on the file type.
- Also, file is a method of data collection that is used as a medium for giving input and receiving output from the program.
- Every File has a logical location where they are located for storage and retrieval.

### **Objective of File management System:**

The file management system is used to manage the files.

Here are the main objectives of the file management system:

1. It provides I/O support for a variety of storage devices.
2. It minimizes the chances of lost or destroyed data.
3. It helps OS to standardized I/O interface routines for user processes.
4. It provides I/O support for multiple users in a multiuser systems environment.

## Properties of a File System:

Here, are important properties of a file system:

1. Files are stored on disk or other storage and do not disappear when a user logs off.
2. Files have names and have access permission that controls the access rights of file.
3. Every file has a structure defined by its owner or creator and depends on the file type.

## File Structure:

- File Structures is the Organization of Data in Secondary Storage Device in such a way that, it minimizes the file access time and the file storage space.
- A File Structure is a combination of representations for data in files and operations for accessing the data.
- A File Structure allows applications to read, write and modify the data.
- A File Structure should be according to a required format that the operating system can understand.
- A File has a certain defined structure according to its type like-
  - A text file is a sequence of characters organized into lines.
  - A source file is a sequence of procedures and functions.
  - An object file is a sequence of bytes organized into blocks that are understandable by the machine.
- When operating system defines different file structures, it also contains the code to support these file structure.
- Unix, MS-DOS support minimum number of file structure whereas LINUX, MS-Windows supports for large number of file structure.

## File Type:

- File type refers to the ability of the operating system to distinguish different types of file such as text files, source files, picture file, audio file, binary files etc.
- In short- depending on file type, operating system knows owner or creator of file. That is if file type has extension .doc then OS recognizes its owner is MS-Word, if file type is .cpp then OS recognizes it is CPP source program etc.
- Operating system like MS-Windows and LINUX operating systems support many types of files.
- Operating system like MS-DOS and UNIX have the following types of files –

### 1) Ordinary files:

- These are the files that contain user information.
- These may have text, databases or executable program.
- The user can apply various operations on such files like add, modify, delete or even remove the entire file.

### 2) Directory files:

- These files contain list of file names and other information related to these files.

### 3) Special files:

- These files are also known as device files.
- These files represent physical device like disks, terminals, printers, networks, tape drive etc.
- These files are of two types –
  - **Character special files** – In this case, data is handled character by character as in case of terminals or printers.
  - **Block special files** – In this case, data is handled in blocks as in the case of disks and tapes.



## File Attributes:

- File Attributes are data that define a file, and are used by the operating system and other software to determine how a file is accessed and used.
- File attributes, consisting of information such as file name, file data, and its security information.
- File attributes are a type of meta-data that describes and modifies the files or directories in a file system.
- Every files are distinct with one another by considering file attributes such as dates and times, filename, file extensions or file system permissions.

The file attributes of a file can be different on many systems, however, some of them are common:

- Name – unique name for a human to recognize the file.
- Identifier– A unique number tag that identifies the file in the file system and it is non-human readable
- Type – Information about the file to get support from the system.
- Size – The current size of the file in bytes, kilobytes, or words.
- Access Control – Defines who could read, write, execute, change, or delete the file in the system.
- Time, Date, and User identification – This information kept for date created, last modified, or accessed and so on.

## File Operations:

As you know that files are used to store the required information for its later uses.

There are many file operations that can be perform by the computer system such as-

Create operation	Delete operation	Open operation
Close operation	Read operation	Write operation
Append operation	Seek operation	Get attribute operation
Set attribute operation	Rename operation	

Let's see all these operations in details-

### 1) File Create Operation:

- This operation creates new file with no data.
- Without creating any file, there is no any operation can be performed, therefore file create operation is the first operation of the file.

### 2) File Open Operation:

- This operation opens the file before using it.

### 3) File Close Operation:

- This operation closes the opened file.
- When all the accesses are finished and the attributes and the disk addresses are no longer needed then file must be closed to free up the internal space.

### 4) File Delete Operation:

- This operation deletes the file when it is no longer needed.
- File delete operation free up the disk space that file holds.
- After deleting the file, it doesn't exist.

### 5) File Read Operation:

- This read operation is used to just read the data that are stored in the file.

### 6) File Write Operation:

- This write operation is used to write the data to the file.
- The data generally write at the current position of the file.

### 7) File Append Operation:

- The file append operation is same as the file write operation except that the file append operation only write or add the data at the end of the file.

### 8) File Seek Operation:

- This operation is performed on random access files.

- This operation is used to specify from which location of file to take the data.
- 9) File Get Attribute Operation:
- The file get attributes operation are performed by the processes when they need to read the file attributes to do their required work.
- 10) File Set Attribute Operation:
- The file set attribute operation used to set some of the attributes (user settable attributes) after the file has been created.
- 11) File Rename Operation:
- The file rename operation is used to change the name of the existing file.

## **File Access Mechanisms:**

- File access mechanism refers to the manner in which the data or information or records of a file can be accessed.

There are several ways to access files –

1. Sequential access.
2. Direct/Random access.
3. Indexed sequential access.

Let's see types in details-

### **1) Sequential access:**

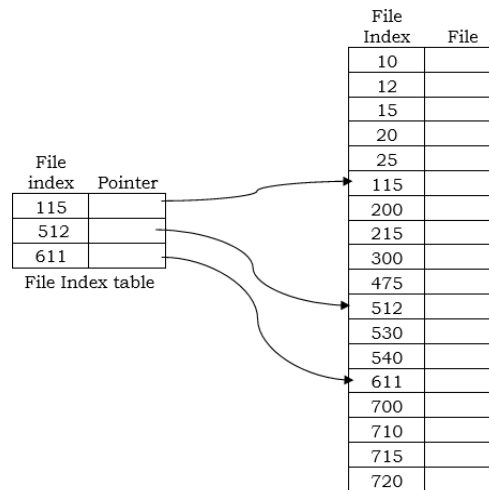
- A sequential access is file access mechanism in which the data or records of file are accessed in sequential manner, i.e. the information in the file is processed in sequential order, one record, after then next record, and so on. This access method is the most primitive one.
- If files data stored on magnetic tape then it is accessed in sequential manner by tape reader.
- Also, Compilers usually access files in sequential manner.
- This sequential access mechanism is slow because file's data is accessed in sequential manner.

### **2) Direct/Random access:**

- A Random access is file access mechanism in which the data or records of file are accessed directly i.e. random manner.
- In file, each record has its own address. With the help of such address, random access mechanism can directly access record for reading or writing.
- In this case, records need not be in any sequence within the file and they need not be in adjacent locations on the storage medium.
- The data stored on hard disk or CD or DVD can allow random access mechanism.

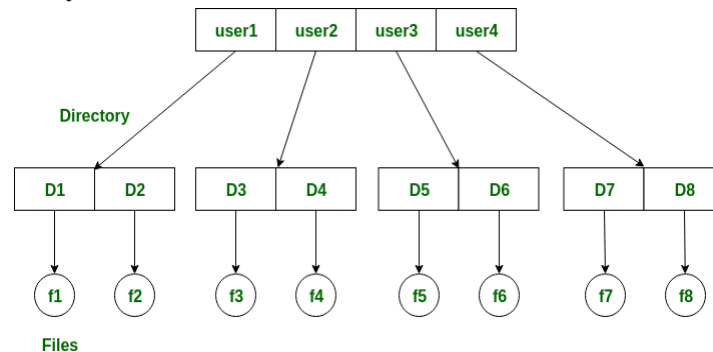
### **3) Indexed sequential access:**

- Indexed sequential mechanism is built up on basis of sequential access mechanism.
- In this mechanism, an index is created for each file which contains pointers to various blocks of memory.
- In this case, index is searched sequentially and its pointer is used to access the file directly as shown in following figure:



## Directory structure: (File directory)

- Collection of files referred as file directory. That is, in directory number of different files can be stored.
- The directory contains information about the files, including attributes, location and ownership.
- The directory is itself a file, accessible by various file management routines and is managed by the operating system.
- Following diagram shows directory structure of four different user-



In above directory structure,

‘user1’ creates two directories D1 and D2. The D1 directory contains ‘f1’ file and D2 directory contains ‘f2’  
‘user2’ creates two directories D3 and D4. The D3 directory contains ‘f3’ file and D4 directory contains ‘f4’  
‘user3’ creates two directories D5 and D6. The D5 directory contains ‘f5’ file and D6 directory contains ‘f6’  
‘user4’ creates two directories D7 and D8. The D7 directory contains ‘f7’ file and D8 directory contains ‘f8’

## Attributes of directory-

- 1) Name: It specifies the name of directory.
- 2) Type: It specifies the type of directory i.e. it is system directory or user directory.
- 3) Address: It specifies the physical location of directory on storage device.
- 4) Current length: It specifies the current storage or size of directory.
- 5) Maximum length: It specifies the maximum size of directory.
- 6) Date last accessed: It specifies the last date when directory was accessed.
- 7) Date last updated: It specifies the last date when directory was updated.
- 8) Owner id: It specifies the owner of directory.
- 9) Protection information: It specifies the protection information of directory.

## Operations performed on directory are:

- 1) Search for a file: This operation search for particular file in directory.
- 2) Create a file: This operation creates new file in directory.
- 3) Delete a file: This operation deleted file from directory.
- 4) List a directory: This operation displays all the directories and files available in directory.
- 5) Rename a file: This operation renames the file present in directory.

### Advantages of maintaining directories are:

- Grouping: Making logical grouping of files becomes easy. e.g. Java programs, OS notes, C notes etc.
- Because of grouping of file in directory makes quickly access to a file.
- It becomes convenient for users to locate or store same copy file in different directories this achieve file backup concept.

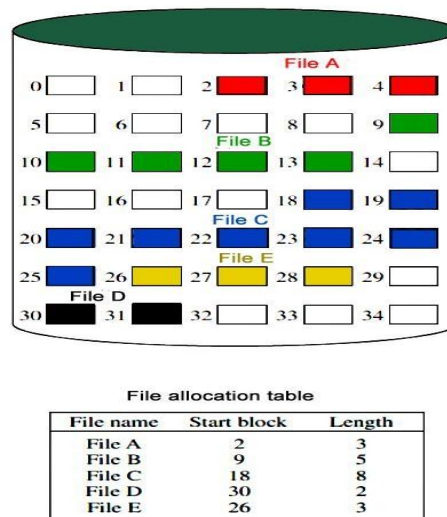
### File allocation methods:

- The process of allocating or assigning memory space for files on disk space is called as 'File allocation'.
- Files are allocated on disk spaces by operating system.
- Operating systems uses following three main ways to allocate disk space for files.-
  1. Contiguous Allocation method.
  2. Linked Allocation method.
  3. Indexed Allocation method.

Let's see these types in details-

#### 1) Contiguous Allocation method.-

- In contiguous file allocation, OS allocates contiguous address space (memory blocks) for each file on disk whenever file is just created. (i.e. pre-allocation of memory blocks)
- In this method, OS assigns disk address for every file in linear order.
- In this scheme, file allocation table is maintained for each file and that shows the starting block and the length of the file.
- The contiguous allocation is of fixed-size blocks for a file.
- It is pre-allocation method, therefore it is necessary to declare the size of the file at the time of creation.
- It supports for sequential access of file still multiple blocks can be read at a time to improve I/O performance for sequential processing.
- In this method, it is also easy to retrieve a single block of a particular file. For example, if a file starts at block 'b', and we want to access it's the  $i^{\text{th}}$  block, then its  $i^{\text{th}}$  block is found at  $b+i-1$  location on disk.
- Following diagram shows, contiguous file allocation-



#### Advantage of contiguous allocation-

- This method is easy to implement.
- This method is best for individual sequential file.

#### Disadvantage of contiguous allocation-

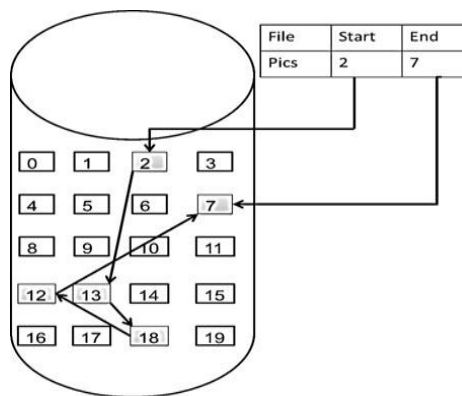
- In this method there is problem of external fragmentation (i.e. memory wastage)
- In this method, it is difficult to find contiguous blocks of space of sufficient length for a file.

- Compaction algorithm will be necessary to free up additional space on disk.
- It is pre-allocation method, therefore it is necessary to declare the size of the file at the time of creation.

## 2) Linked Allocation method-

- In linked allocation, allocation is done on an individual block basis where each block contains a pointer to the next block in the chain. As like linked list data structure.
- In this case, each file has a list of links (pointers) to disk blocks.
- In this scheme, file allocation table is also maintained for each file is that shows the starting block and the ending block of the file.
- Although pre-allocation is possible still it is more simply to allocate blocks for file as needed i.e. any free block can easily be added to the chain. Here, the blocks need not be continuous.
- In this case, increase in file size is always possible if free disk block is available. Here just need to assign link to next empty block.
- There is no external fragmentation because only one block at a time is needed but there can be internal fragmentation but it exists only in the last disk block of file.

Following diagram shows linked allocation-



### Advantages linked allocation method-

- In linked allocation, there is no external fragmentation.
- It is effectively used in sequential access file.
- It is better allocation than contiguous allocation as memory point of view.

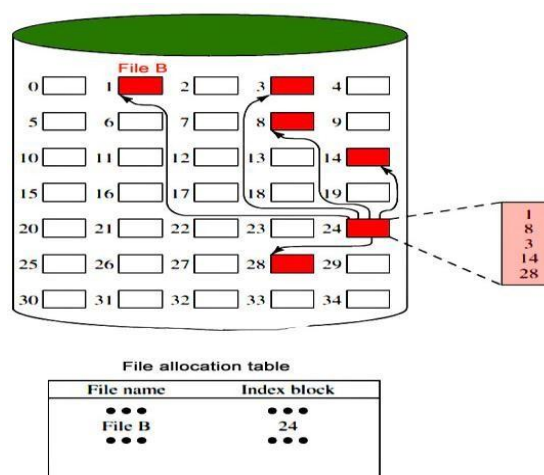
### Disadvantages linked allocation method-

- Internal fragmentation exists in last disk block of file.
- There is an overhead of maintaining the pointer in every disk block.
- If the pointer of any disk block is lost, the file will be truncated.
- It only supports for sequential access of files.
- Inefficient in case of direct access file.

## 3) Indexed Allocation method:

- The indexed allocation solves the problems of contiguous and linked allocation.
- In indexed allocation, each file has its own index block which stores the addresses of disk space occupied by the file. An index block is having all pointers to files.
- In this allocation, directory contains the addresses of such index blocks of files.
- In this case, the file allocation table contains a separate one-level index for each file. The index has one entry for each block allocated to the file.
- In this scheme, allocation may be on the basis of fixed-size blocks or variable-sized blocks. Allocation by fixed-size blocks eliminates external fragmentation, whereas allocation by variable-size blocks improves locality access.
- This allocation technique supports both sequential and direct access to the file and thus is the most popular form of file allocation.

- Following diagram shows indexed allocation-



### Advantages of indexed allocation method-

- This allocation technique supports both sequential and direct access to the file and thus is the most popular form of file allocation.
- No problem of external fragmentation.
- This file access mechanism is fast as compared to contiguous and linked allocation.

### Free space management:

- We know that, as file created, file system (i.e. OS) allocates some memory blocks to file. And when file is deleted then file system free or release allocated memory blocks such that it would be used for another file in future. Thus, file system is responsible to keep track of all the allocated and free blocks present on the hard disk.
- Managing such free spaces which were found after deleting file is crucial task and it is called as 'Free space management'.
- In free space management, the file system maintains a free space list which keeps track of the disk blocks that are not allocated to some file or directory.
- The free space list can be implemented using following methods: (Free space management Approach)
  - 1) Bitmap or Bit vector
  - 2) Linked list
  - 3) Grouping

Let's see these in details-

#### 1) Bitmap or Bit vector-

- In this approach, the free space list is implemented as a bit map vector.
- It contains the number of bits where each bit represents each block.
- If the block is empty then the bit is 1 otherwise it is 0. Initially all the blocks are empty therefore each bit in the bit map vector contains 1.
- When file is created, the file system starts allocating blocks to the files and setting the respective bit to 0.
- A Bitmap or Bit Vector is series or collection of bits where each bit corresponds to a disk block. The bit can take two values: 0 and 1. 0 indicates that the block is allocated and 1 indicates a free block.
- Consider following Figure- that shows some green blocks and some white blocks. Green blocks indicates allocated blocks to file and white block indicates free blocks. And it can be represented by a bitmap of 16 bits as: 0000111000000110.
- Following diagram free space management using Bitmap vector-

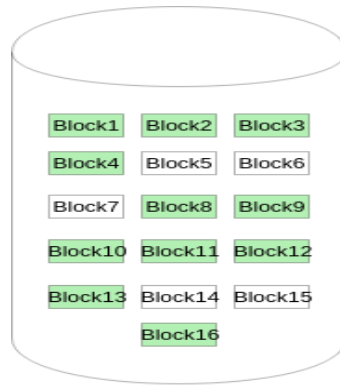


Figure - 1

### Advantages bitmap vector –

1. This method is simple to understand and easy to implement.
2. Finding the free blocks and allocated blocks is easy by scanning all bits in a word (8 bits)

### 2) Linked list:

- It is another approach for free space management. In this approach all free blocks are linked together and keeping a pointer in the cache which points to the first free block.
- Therefore, all the free blocks on the disks will be linked together with a pointer. Whenever a block gets allocated, its previous free block will be linked to its next free block.
- Whenever the memory is allocated to a block, its corresponding pointers are updated.
- Following diagram free space management using linked list-

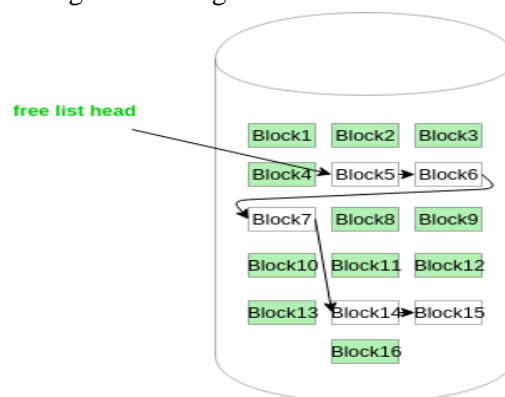
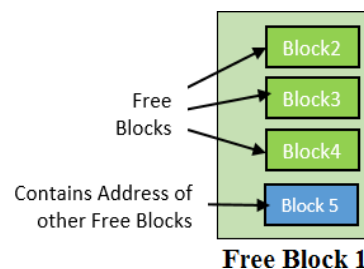


Figure - 2

### 3) Grouping:

- This is another free space management technique. In this approach, the addresses of all free blocks are maintained in a group such that in future free block will easily available for allocation.
- The first free block stores the address of some other free blocks say 'n' free blocks. Out of these n blocks, the first 'n-1' blocks are actually free and the last block contains the address of next free 'n' blocks. As shown in following figure:



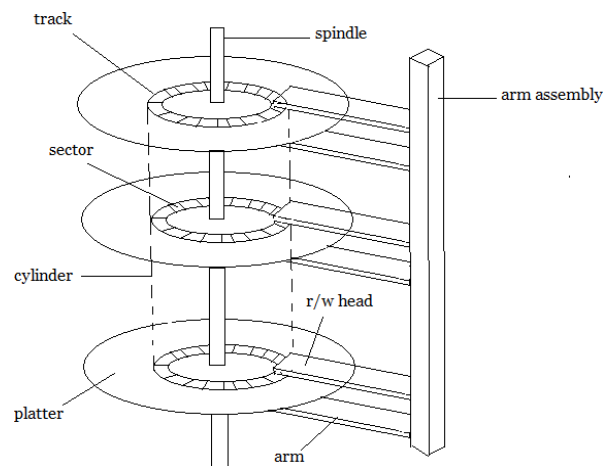
- The main advantage of this approach is that the addresses of a group of free disk blocks can be found easily.

## Disk Management:

- Concept: We know that, in a computer there are different processes requests for I/O operation (Disk request) at a time. In such request some processes demands for write operation and some processes demands for read operation. Such multiple Read/Write operations request needs to be managed & it is called as 'Disk management'. Disk management is done by disk controller i.e. by operating system.
- **Basic of Disk (Secondary Memory):**
- We know that, secondary storage devices are those devices whose memory is nonvolatile i.e. stored data will not lost even if the system is turned off.
- Examples of such devices are magnetic disks, magnetic tapes, removable drives (Pen drive) etc.
- Here are few important things about secondary storage-
  - Secondary storage is also called auxiliary storage.
  - Secondary storage is less expensive when compared to primary memory like RAMs.
  - The speed of the secondary storage is also lesser than that of primary storage. Hence, the data which is less frequently accessed is kept in the secondary storage.

## Disk Structure:

In modern computers, most of the secondary storage is in the form of magnetic disks. Hence, knowing the structure of a magnetic disk is necessary to understand how the data in the disk is accessed by the computer. Following diagram shows disk structure-



- A magnetic disk contains several platters. Each platter is divided into circular shaped tracks. The length of the tracks near the center is less whereas the length of the tracks far from the center is bigger. Each track is further divided into sectors, as shown in the figure.
- The point to be noted here is that outer tracks are bigger in size than the inner tracks but they contain the same number of sectors and have equal storage capacity. This is because the storage density is high in sectors of the inner tracks whereas the bits are sparsely arranged in sectors of the outer tracks. Some space of every sector is used for formatting. So, the actual capacity of a sector is less than the given capacity.
- Tracks of the same distance from center forms a cylinder.
- A read-write head is used to read data from a sector of the magnetic disk.
- Read-Write(R-W) head moves over the rotating hard disk. This Read-Write head performs all the read and write operations on the disk and hence, position of the R-W head is a major concern.
- To perform a read or write operation on a memory location, we need to place the R-W head over that position.

## Some important terminologies-

- 1) **Seek time** – The time taken by the R-W head to reach the desired track from its current position.
- 2) **Rotational latency** – Time taken by the sector to come under the R-W head.



- 3) **Data transfer time** – Time taken to transfer the required amount of data. It depends upon the rotational speed.
  - 4) **Controller time** – The processing time taken by the controller.
  - 5) **Average Access time:** seek time + Average Rotational latency + data transfer time + controller time.
- Note: Average Rotational latency is mostly  $\frac{1}{2} \times (\text{Rotational latency})$ .*

## Disk Scheduling:

- Disk scheduling is done by operating systems to schedule I/O requests arriving for the disk. Disk scheduling is also known as I/O scheduling.

### Importance of disk scheduling:

- Multiple I/O requests may arrive by different processes and only one I/O request can be served at a time by the disk controller. Thus other I/O requests need to wait in the waiting queue and need to be scheduled.
- Two or more request may be far from each other so can result in greater disk arm movement.
- We know that, hard drives are one of the slowest parts of the computer system and thus need to be accessed in an efficient manner & that's why disk scheduling is necessary.

## Disk scheduling Algorithms:

- Disk scheduling algorithms are useful to the disk controller to schedule multiple I/O requests.

We have to learn following disk scheduling algorithms-

- 1) First Come First Serve (FCFS) algorithm
- 2) Shortest Seek Time First (SSTF) algorithm
- 3) Elevator (SCAN)
- 4) Circular SCAN (CSCAN)

Let's see these algorithms in details-

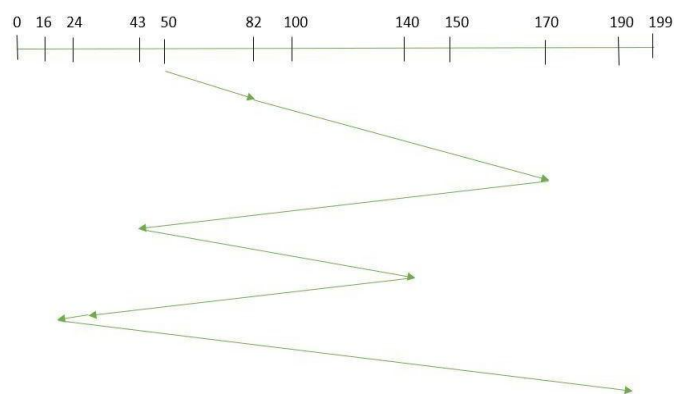
### 1) First Come First Serve (FCFS) algorithm:

- FCFS is the simplest of all the Disk Scheduling Algorithms.
- In FCFS algorithm, the I/O requests are granted in the FIFO order as they arrive in the disk queue.
- Let us understand this with the help of an example.

#### Example:

Suppose the order of request is- (82, 170, 43, 140, 24, 16, 190)

And current position of Read/Write head is: 50



So, total seek time (The time taken by the R-W head to reach the desired track from its current position)  

$$= (82-50) + (170-82) + (170-43) + (140-43) + (140-24) + (24-16) + (190-16)$$

$$= 642$$

### Advantages FCFS algorithm:

- In FCFS algorithm, every request gets a fair chance to access disk space.
- No request stay for long time in disk queue.

**Disadvantages FCFS algorithm:**

- Does not try to optimize seek time.
- May not provide the best possible service.

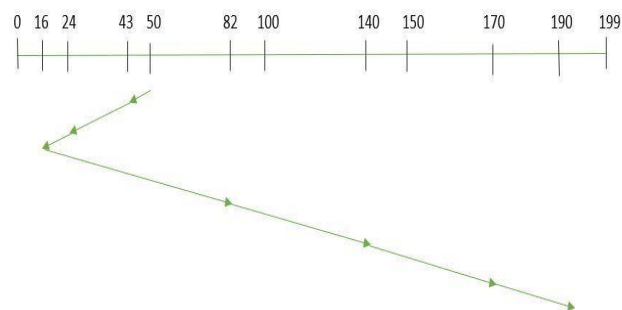
**2) Shortest Seek Time First (SSTF) algorithm:**

- In SSTF (Shortest Seek Time First), requests having shortest seek time are executed first.
- So, the seek time of every request is calculated in advance in the queue and then they are scheduled according to their calculated seek time.
- As a result, the request near the disk arm will get executed first. SSTF is certainly an improvement over FCFS as it decreases the average response time and increases the throughput (amount of time done in specific time unit) of system.
- Let us understand this with the help of an example.

**Example:**

Suppose the order of request is- (82, 170, 43, 140, 24, 16, 190)

And current position of Read/Write head is: 50



So, total seek time:

$$\begin{aligned} &= (50-43)+(43-24)+(24-16)+(82-16)+(140-82)+(170-140)+(190-170) \\ &= 208 \end{aligned}$$

**Advantages of SSTF algorithm:**

- Average Response Time decreases.
- Throughput increases.

**Disadvantages of SSTF algorithm:**

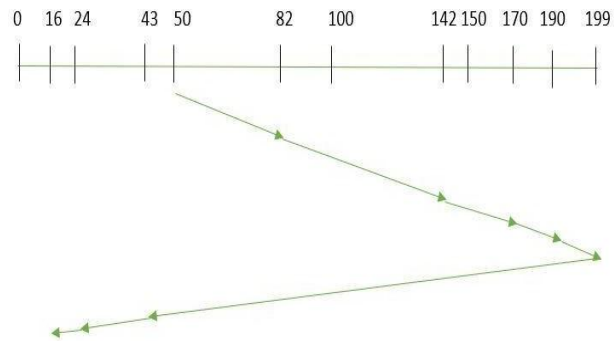
- May be possibility to occur overhead to calculate seek time in advance.
- Starvation may occur for a request if it has higher seek time as compared to incoming requests.

**3) SCAN algorithm: (Elevator algorithm)**

- In SCAN algorithm the disk arm moves into a particular direction and services the requests coming in its path and after reaching the end of disk, it reverses its direction and again services the request arriving in its path.
- So, this algorithm works as an elevator and hence also known as ‘elevator’ algorithm. As a result, the requests at the midrange are serviced more and those arriving behind the disk arm will have to wait.

**Example:**

Suppose the requests to be addressed are- 82,170,43,140,24,16,190. And the Read/Write arm is at 50, and it is also given that the disk arm (i.e. R/W head) should move “towards the larger value”.



Therefore, the seek time is calculated as:

$$= (199-50) + (199-16) \\ = 332$$

#### Advantages of SCAN algorithm:

- This algorithm has High throughput.
- It has better Average response time

#### Disadvantages of SCAN algorithm:

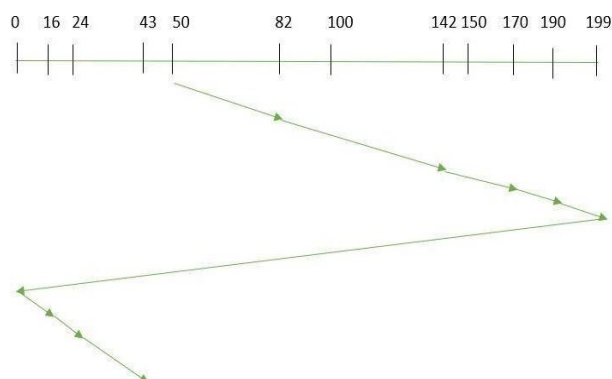
- Overhead of request happens while arm head reverses from one end to another end.

#### 4) CSCAN algorithm: (Circular SCAN algorithm)

- We know that, in SCAN algorithm, the disk arm again scans the path that has been scanned, after reversing its direction. So, it may be possible that too many requests are waiting at the other end and also generates overheads. These situations are avoided in CSCAN algorithm.
- In this algorithm, the disk arm (R/W head) first goes to one end of disk while doing so it process all requests that comes under this movement. After reaching at one end, it then goes towards other remaining end. After reaching other end it again start scanning & completes remaining requests towards the previous end.
- In this case, the disk arm moves in a circular fashion and this algorithm is also similar to SCAN algorithm and hence it is known as C-SCAN (Circular SCAN).

#### Example:

Suppose the requests to be addressed are-82, 170, 43, 140, 24, 16, 190. And the Read/Write arm is at 50, and it is also given that the disk arm should move “towards the larger value”.



Seek time is calculated as:      Seek time=  $(199-50) + (199-0) + (43-0) = 391$

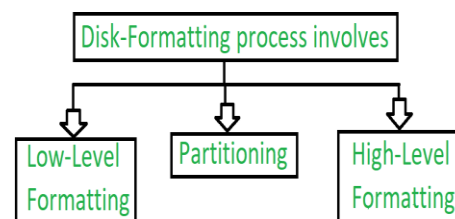
#### Advantages CSCAN algorithm-

- It provides more uniform wait time for requests compared to SCAN algorithm.
- It eliminates overheads that happens in SCAN algorithm.

## Disk Formatting:

- Disk formatting is a process to configure the data-storage devices such as hard-drive, floppy disk and flash drive when we are going to use them for the very first time.
- Disk formatting is usually required when new operating system is going to be used by the user.
- It is also done when there is space issue and we require additional space for the storage of more data in the drives. When we format the disk then the existing files within the disk is also erased.
- We can perform disk formatting on both magnetic platter hard-drives and solid-state drives.
- When we are going to use hard-drive for initial use it is necessary to search for virus and bad sectors within the drive. Disk formatting has ability to erase the bad applications, viruses and also repairs bad sectors of drive.
- As we know that disk formatting deletes data and removes all the programs installed within the drive. So it can be done with caution. We must have the backup of all the data and applications which we require before formatting.
- No-doubt disk formatting requires time. But the frequent formatting of the disk decreases the life of the hard-drive.

### Disk formatting process:



Let's see disk formatting process in details-

### 1. Low-level Formatting:

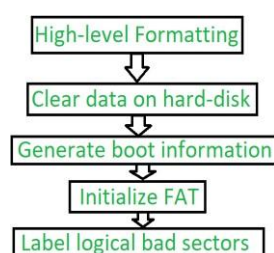
- Low level formatting is a type of physical formatting.
- In this low-level formatting, the process of marking of cylinders and tracks on the blank hard-disk.
- After making tracks, there is the division of tracks into sectors with the sector markers.
- Now-a-days low-level formatting is performed by the hard-disk manufactures companies.
- If we have data in our hard-disks and we perform low-level formatting in the presence of data then all the data have been erased and it is impossible to recover that data. Some users make such a format that they can avoid their privacy leakage.
- Low-level formatting will cause damage to hard-disk, therefore this formatting is not suggested to users.

### 2. Partitioning:

- As the name suggest, partitioning means divisions. Partitioning is the process of dividing the entire hard-disk into two or more regions. The regions are called as partitions.
- It can be performed by the users while installing of operating system.
- Partitioning will also affect the disk performance.

### 3. High-level Formatting:

- High-level formatting is the process of writing something on new created partitions of disk.
- High-level formatting writes on a file system (FAT/NTFS), cluster size, partition label, and so on for a newly created partition or volume. Following diagram shows steps involved in high-level formatting-



- Firstly High-level formatting clears the data on hard-disk, then it will generate boot information, then it will initialize FAT after this it will go for label logical bad sectors.
- Formatting done by the user is the high-level formatting.
- Generally, it does not harm the hard-disk. It can be done easily with the disk management tools such as Administrator, Windows snap-in, disk part, etc.
- We can use such a format to fix some problems like errors in the file system, corrupted hard-drive and developing of bad sectors.

### **Boot Blocks:**

- Boot blocks are those memory blocks located on hard disk that help to start computer initially i.e. after power ON.
- The boot blocks have initial program stored on it that helps a computer to start running. And this initial program is known as 'bootstrap'.
- Bootstrap program initializes all aspects of the system, from CPU registers to device controllers and the contents of the main memory, and then starts the operating system.
- To do this job the bootstrap program basically finds the operating system kernel on disk and then loads the kernel into memory and after this, it jumps to the initial address to begin the operating-system execution.
- For most of today's computer bootstrap is stored in Read Only Memory (ROM) because-
  - This location is good for storage because this place doesn't require initialization and this location is fixed in ROM so that processor can start executing when powered up or reset.
  - ROM is basically read-only memory and hence it cannot be affected by the computer virus.

### **Bad Blocks:**

- Bad Block is an area of storing devices that is no longer useful for the storage of data because it is completely damaged or corrupted.
- We know that to access data from the disk, disk moves fast and in this case it will have small tolerances, this causes failure of blocks and such block is called 'Bad block'

### **Cause of Bad Block:**

- Storage drives can ship from the factory with defective blocks that originated in the manufacturing process. The device with bad-blocks are marked as defective before leaving the factory.
- A physical damage to device also makes a device as bad block because sometimes operating system does not able to access the data. Dropping a laptop will also cause damage to the platter of the HDD's. Sometimes dust also cause damage to HDD's.
- When the memory transistor fails it will cause damage to the solid-state drive.
- Soft bad sectors are caused by software problems. For instance, if a computer unexpectedly shuts down, due to this, hard drive also turn off in the middle of writing to a block. Due to this, the data contained in the block doesn't match with the CRC detection error code and it would be marked as bad sector.

### **Types of Bad Blocks:**

There are two types of bad blocks –

#### **1) Physical or Hard bad block:**

- Physical or hard blocks are such bad blocks that happen because of damage to the storage medium.

#### **2) Soft or Logical bad block:**

- Soft or logical bad blocks occur when the operating system (OS) is unable to read data from a sector because of software failure.