Operating System

Definition

- An **Operating System (OS)** is a software that acts as an interface between computer hardware components and the user.
- Every computer system must have at least one operating system to run other programs.
- Applications like Browsers, MS Office, Notepad Games, etc., need some environment to run and perform its tasks.
- The OS helps you to communicate with the computer without knowing how to speak the computer's language.
- It is not possible for the user to use any computer or mobile device without having an operating system.

History Of OS

- Operating systems were first developed in the late 1950s to manage tape storage
- The General Motors Research Lab implemented the first OS in the early 1950s for their IBM 701
- In the mid-1960s, operating systems started to use disks
- In the late 1960s, the first version of the Unix OS was developed
- The first OS built by Microsoft was DOS. It was built in 1981 by purchasing the 86-DOS software from a Seattle company
- The present-day popular OS Windows first came to existence in 1985 when a GUI was created and paired with MS-DOS.

Following are the Operating System examples with the latest Market Share

OS Name	Share
Windows	40.34
Android	37.95
iOS	15.44
Mac OS	4.34
Linux	0.95
Chrome OS	0.14
Windows Phone OS	0.06

Types of Operating System (OS)

Following are the popular types of OS (Operating System):

- Batch Operating System
- Multitasking/Time Sharing OS
- Multiprocessing OS
- Real Time OS
- Distributed OS
- Network OS
- Mobile OS

Batch Operating System

- Some computer processes are very lengthy and time-consuming. To speed the same process, a job with a similar type of needs are batched together and run as a group.
- The user of a batch operating system never directly interacts with the computer. In this type of OS, every user prepares his or her job on an offline device like a punch card and submit it to the computer operator.

Multi-Tasking/Time-sharing Operating systems

- Time-sharing operating system enables people located at a different terminal(shell) to use a single computer system at the same time.
- The processor time (CPU) which is shared among multiple users is termed as time sharing.

Real time OS

- A real time operating system time interval to process and respond to inputs is very small. Examples: Military Software Systems, Space Software Systems are the Real time OS example.

Distributed Operating System

- Distributed systems use many processors located in different machines to provide very fast computation to its users.

Network Operating System

- Network Operating System runs on a server. It provides the capability to serve to manage data, user, groups, security, application, and other networking functions.

Mobile OS

- Mobile operating systems are those OS which is especially that are designed to power smartphones, tablets, and wearables devices.
- Some most famous mobile operating systems are Android and iOS.

Functions of Operating System

- **Process management**:- Process management helps OS to create and delete processes. It also provides mechanisms for synchronization and communication among processes.
- **Memory management:-** Memory management module performs the task of allocation and de-allocation of memory space to programs in need of this resources.
- **File management**:- It manages all the file-related activities such as organization storage, retrieval, naming, sharing, and protection of files.
- **Device Management**: Device management keeps tracks of all devices. This module also responsible for this task is known as the I/O controller. It also performs the task of allocation and de-allocation of the devices.

- **I/O System Management:** One of the main objects of any OS is to hide the peculiarities of that hardware devices from the user.
- **Secondary-Storage Management**: Systems have several levels of storage which includes primary storage, secondary storage, and cache storage. Instructions and data must be stored in primary storage or cache so that a running program can reference it.
- **Security**:- Security module protects the data and information of a computer system against malware threat and authorized access.
- **Command interpretation**: This module is interpreting commands given by the and acting system resources to process that commands.
- **Networking:** A distributed system is a group of processors which do not share memory, hardware devices, or a clock. The processors communicate with one another through the network.
- **Job accounting**: Keeping track of time & resource used by various job and users.
- **Communication management**: Coordination and assignment of compilers, interpreters, and another software resource of the various users of the computer systems.

Advantage of using Operating System

- Allows you to hide details of hardware by creating an abstraction
- Easy to use with a GUI
- Offers an environment in which a user may execute programs/applications
- The operating system must make sure that the computer system convenient to use
- Operating System acts as an intermediary among applications and the hardware components
- It provides the computer system resources with easy to use format
- Acts as an intermediator between all hardware's and software's of the system

Windows operating system

- **Windows operating system** is made by Microsoft and it has delivered numerous variants.
- The most renowned and utilized rendition of Microsoft Windows will be windows 7.
- The primary working framework (OS) which was created by Microsoft was MS-DOS. MS-DOS was a basic OS and has an order line interface.
- The new form of Microsoft OS has likewise MS-DOS work as a utility. MS-DOS is as yet utilized in doing some regular undertakings like making records, envelopes, introducing modules in some products, utilizing GIT and different uses also.

Contd..

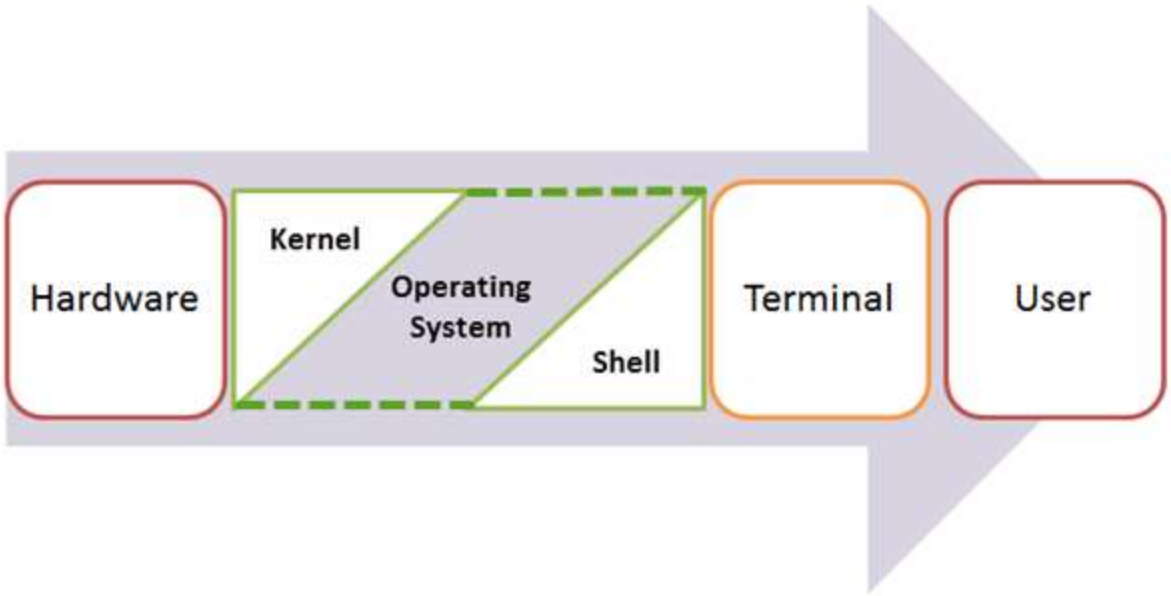
- After Windows 7, Microsoft dispatched Windows 8. Windows 8 was not that much achievement on account of the new interface and missing the beginning menu.
- In the following arrival of Windows 10, the organization again added a start menu which was absent in Windows 8. The start menu in Windows 10 is a mix of Windows 7 and a little bit of windows 8.
- In Windows 10, Cortana which is a snappy inquiry instrument was presented. Cortana still has a few bugs which should be fixed. Cortana is enlivened by google voice search.
- You can voice search in windows 10 by Cortana. This internet searcher stores bunches of information about individuals and it make a high burden on the framework. You can alternatively kill the Cortana in the event that you don’t prefer to utilize it.

Advantage of windows OS

- **Backing for all equipment :**
  As windows OS is utilized by 95% of clients so the majority of the equipment merchants make drivers for windows.
- **Convenience :**
  All forms of Microsoft Windows have something regular in it which makes it clients simple to move starting with one form then onto the next. Windows 7 clients have no trouble in moving to Windows 10 in light of the fact that a large portion of the highlights of Windows 10 is equivalent to Windows 7. The UI of windows is additionally simple to use than UNIX and MAC.
- **Programming support :**
  Windows stage is most appropriate for game and programming engineers. Windows have a huge number crowd so designers want to make utilities, games, and programming for windows OS. Linux clients can't make windows applications, so it is smarter to utilize windows for creating applications.
- **Fitting and play highlight :**
  Most equipment can be distinguished naturally by attachment and play include. You don't have to physically introduce the equipment however it is prepared to utilize when connected for example webcam, console, mouse, cell phone, and so forth.
- **Work area and contact screen:**
  Windows 10 is made for both touch screen gadgets and PCs. The UI of Windows 10 is made so that it turns out better for a windows gadget.

What is Kernel in Operating System?

- The kernel is the central component of a computer operating systems.
- The only job performed by the kernel is to the manage the communication between the software and the hardware.
- A Kernel is at the nucleus of a computer. It makes the communication between the hardware and software possible.
- While the Kernel is the innermost part of an operating system, a shell is the outermost one.


Hardware
Kernel
Operating System
Shell
Terminal
User

Features of Kennel

- Low-level scheduling of processes
- Inter-process communication
- Process synchronization
- Context switching

Features

- Protected and supervisor mode
- Allows disk access and file systems Device drivers Networking Security
- Program Execution
- Memory management Virtual Memory Multitasking
- Handling I/O operations
- Manipulation of the file system
- Error Detection and handling
- Resource allocation
- Information and Resource Protection

Windows XP Requirements

CPU
233MHz or 1.6GHz for Windows XP Media Center

RAM
64MB or 256MB for Windows XP Media Center

Disk Space
2GB for both OS

Media
CD ROM or DVD ROM

Windows 10 Recommended Requirements

CPU: 2 GHz or faster
RAM: 4 GB
HDD: 100 GB of storage space
GPU: Integrated GPU from Intel HD Graphics/Iris Graphics families
OS: Windows 7 SP1, Windows 8.1
DirectX: Version 9
Screen Resolution: 720p
Network: Broadband Internet connection

OS Services

- **User interface** -Almost all operating systems have a user interface (UI),Varies between Command-Line (CLI), Graphics User Interface (GUI), Batch
- **Program execution** -The system must be able to load a program into memory and to run that program, end execution, either normally or abnormally (indicating error)
- **I/O operations** -A running program may require I/O, which may involve a file or an I/O device
- **File-system manipulation** -The file system is of particular interest. Obviously, programs need to read and write files and directories, create and delete them, search them, list file Information, permission management.
- **Communications** –Processes may exchange information, on the same computer or between computers over a network .Communications may be via shared memory or through message passing (packets moved by the OS)
- **Error detection** –OS needs to be constantly aware of possible errors
  May occur in the CPU and memory hardware, in I/O devices, in user program
  For each type of error, OS should take the appropriate action to ensure correct and consistent computing . Debugging facilities can greatly enhance the user's and programmer's abilities to efficiently use the system

Contd..

- **Resource allocation** -When multiple users or multiple jobs running concurrently, resources must be allocated to each of them Many types of resources -Some (such as CPU cycles, main memory, and file storage) may have special allocation code, others (such as I/O devices) may have general request and release code
- **Accounting** -To keep track of which users use how much and what kinds of computer resources
- **Protection and security** -The owners of information stored in a multiuser or networked computer system may want to control use of that information, concurrent processes should not interfere with each other Protection involves ensuring that all access to system resources is controlled.Security of the system from outsiders requires user authentication, extends to defending external I/O devices from invalid access attempts.If a system is to be protected and secure, precautions must be instituted throughout it. A chain is only as strong as its weakest link.

System Call

- In computing, a **system call** is the programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on. A system call is a way for programs to **interact with the operating system**.
- A computer program makes a system call when it makes a request to the operating system's kernel.
- System call **provides** the services of the operating system to the user programs via Application Program Interface(API).
- It provides an interface between a process and operating system to allow user-level processes to request services of the operating system.
- System calls are the only entry points into the kernel system. All programs needing resources must use system calls.

Services Provided by System Calls :

- Process creation and management
- Main memory management
- File Access, Directory and File system management
- Device handling(I/O)
- Protection
- Networking, etc

Types of System Calls

- There are 5 different categories of system calls –
- **Process control:** end, abort, create, terminate, allocate and free memory.
- **File management:** create, open, close, delete, read file etc.
- Device management
- Information maintenance
- Communication

	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup())	chmod() umask() chown()

Process Management

- **Process** is the execution of a program that performs the actions specified in that program.
- It can be defined as an execution unit where a program runs. The OS helps you to create, schedule, and terminates the processes which is used by CPU.
- A process created by the main process is called a child process.
- Process operations can be easily controlled with the help of PCB(Process Control Block).
- You can consider it as the brain of the process, which contains all the crucial information related to processing like process id, priority, state, CPU registers, etc.

Cond..

- Process management involves various tasks like creation, scheduling, termination of processes, and a dead lock.
- Process is a program that is under execution, which is an important part of modern-day operating systems. The OS must allocate resources that enable processes to share and exchange information.
- It also protects the resources of each process from other methods and allows synchronization among processes.
- It is the job of OS to manage all the running processes of the system. It handles operations by performing tasks like process scheduling and such as resource allocation.

Process Architecture

Stack: The Stack stores temporary data like function parameters, returns addresses, and local variables.
Heap Allocates memory, which may be processed during its run time.
Data: It contains the variable.
Text: Text Section includes the current activity, which is represented by the value of the Program Counter.

Process Control Block

- In an Operating System, we have a number of processes present in it.
- Each process has some information that is needed by the CPU for the execution of the process.
- So, we need some kind of data structure to store information about a particular process.
- *Process Control Block or simple PCB is a data structure that is used to store the information of a process that might be needed to manage the scheduling of a particular process.*

Attributes of a Process Control Block

- **Process Id:** A process id is a unique identity of a process. Each process is identified with the help of the process id.
- **Program Counter:** The program counter, points to the next instruction that is to be executed by the CPU. It is used to find the next instruction that is to be executed.
- **Process State:** A process can be in any state out of the possible states of a process. So, the CPU needs to know about the current state of a process, so that its execution can be done easily.
- **Priority:** There is a priority associated with each process. Based on that priority the CPU finds which process is to be executed first. Higher priority process will be executed first.

Contd..

- **General-purpose Registers:** During the execution of a process, it deals with a number of data that are being used and changed by the process. But in most of the cases, we have to stop the execution of a process to start another process and after some times, the previous process should be resumed once again. Since the previous process was dealing with some data and had changed the data so when the process resumes then it should use that data only. These data are stored in some kind of storage units called registers.
- **CPU Scheduling Information:** It indicates the information about the process scheduling algorithms that are being used by the CPU for the process.
- **List of opened files:** A process can deal with a number of files, so the CPU should maintain a list of files that are being opened by a process to make sure that no other process can open the file at the same time.
- **List of I/O devices:** A process may need a number of I/O devices to perform various tasks. So, a proper list should be maintained that shows which I/O device is being used by which process.

Process states

As a process executes, it changes state

- new: The process is being created.
- running: Instructions are being executed.
- waiting: The process is waiting for some event to occur.
- ready: The process is waiting to be assigned to a process.
- terminated: The process has finished execution.

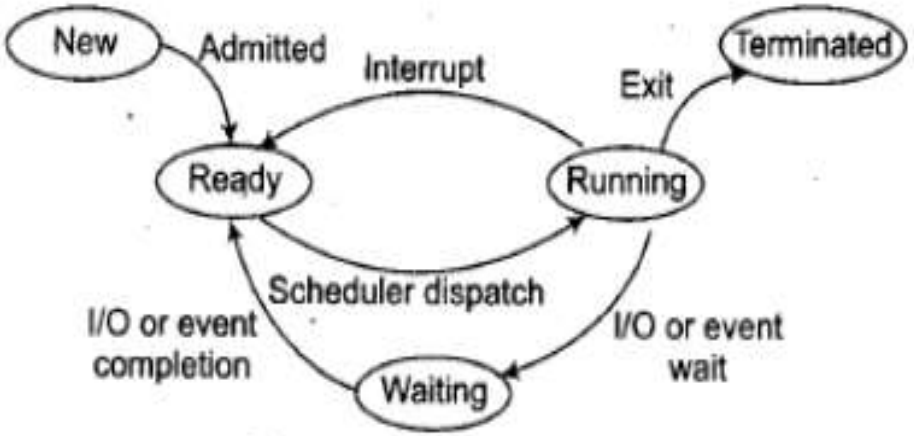


Diagram of a process state

Schedulers: A process migrates among various scheduling queues throughout its lifetime. The OS must select for scheduling purposes, processes from those queues in some fashion. The selection process is carried out by the appropriate scheduler.

Long Term Scheduler: A long term scheduler or job scheduler selects processes from job pool (mass storage device, where processes are kept for later execution) and loads them into memory for execution. The long term scheduler controls the degree of multiprogramming (the number of processes in memory).

Short Term Scheduler: A short term scheduler or CPU scheduler selects from the main memory among the processes that are ready to execute and allocates the CPU to one of them.

Medium Term Scheduler: The medium term scheduler available in all systems which is responsible for the swapping in and out operations which means loading the process into, main memory from secondary memory (swap in) and take out the process from main memory and store it into the secondary memory (swap out).

Aspects :	Long Term Scheduler	Medium Term Scheduler	Short Term Scheduler
Called as	It is a job scheduler	It is a process swapping	It is a CPU scheduler
Speed	Speed is lesser than short term scheduler	Speed is in between both short and long term scheduler	Speed is fastest among two other scheduler
Multiprogramming	It controls the degree of multiprogramming	It reduces the degree of multiprogramming	It provides lesser control over degree of multiprogramming
Time-sharing system	It is almost absent or minimal in time sharing system	It is a part of Time sharing system	It is also minimal in time sharing system
Processes	It selects processes from pool and loads them into memory for execution	It can reintroduce the process into memory and execution can be continued	It selects those processes which are ready to execute

Dispatcher:

- It is the module that gives control of the CPU to the process selected by the short term scheduler.
- Functions of Dispatcher: Switching context, Switching to user mode, and Jumping to the proper location in the user program to restart that program.

Single user OS

- In a single user operating system, a single user can access the computer system at a time.
- These types of operating systems are commonly found in home computers.
- There are two types of single user operating systems called single user, single task operating system and single user, multi-task operating system.

Contd..

- In a single user, single task operating system, a single user can perform only one task at a time.
- Palm OS for Palm handheld computers is an example for a single user, single task operating system.
- In a single user multitask system, a single user can perform multiple tasks at the same time. Microsoft Windows and Apple Mac OS allow a single user to work on multiple programs at the same time.
- For example, a user can work on a word document and browser the World Wide Web simultaneously. Most modern personal computers and laptops are single user multi-tasking operating systems.

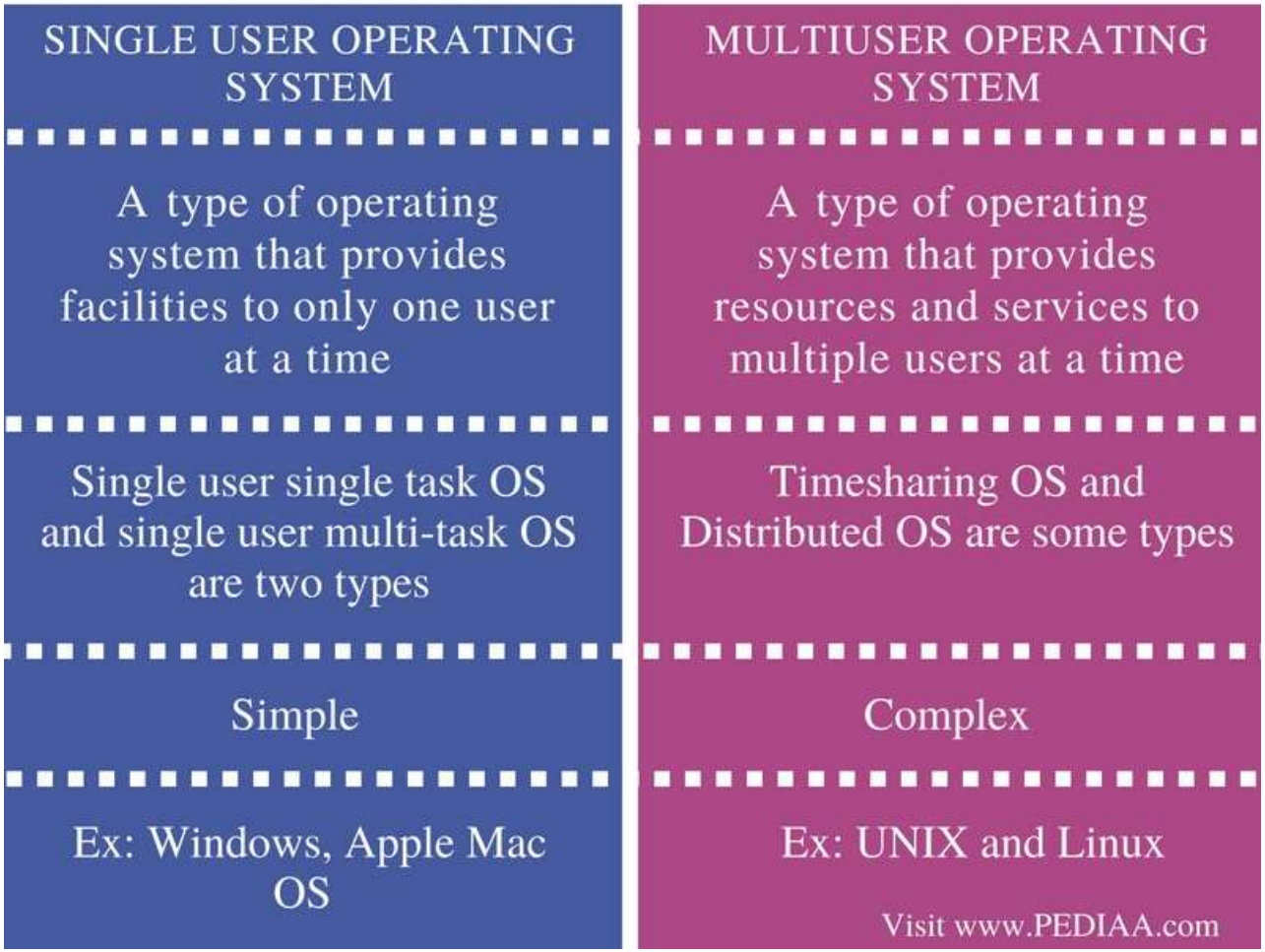
Multi-user OS

- A multi user operating system allows multiple users to access the computer at the same time. The operating system manages the memory and resources among the various users according to the requirements.
- The task of one user will not affect the tasks of the other users. UNIX and Linux are two examples of multi user operating systems.
- A time sharing operating system allows multiple users in different locations to use a particular computer system concurrently.
- In distributed operating system, the data processing task is divided among the processors accordingly. It is also a multiuser operating system.

SINGLE USER OPERATING SYSTEM

VERSUS

MULTIUSER OPERATING SYSTEM



CPU Scheduling

- CPU Scheduling is **a process of determining which process will own CPU for execution while another process is on hold**.
- The main task of CPU scheduling is to make sure that whenever the CPU remains idle, the OS at least select one of the processes available in the ready queue for execution

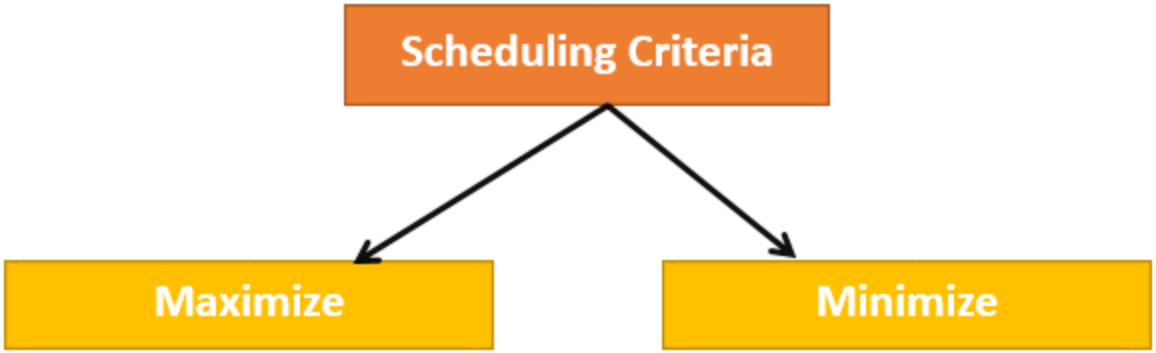
Why do we need scheduling?

- A typical process involves both I/O time and CPU time.
- In a uni programming system like MS-DOS, time spent waiting for I/O is wasted and CPU is free during this time.
- In multi programming systems, one process can use CPU while another is waiting for I/O.
- This is possible only with process scheduling.

Objectives of Process Scheduling Algorithm

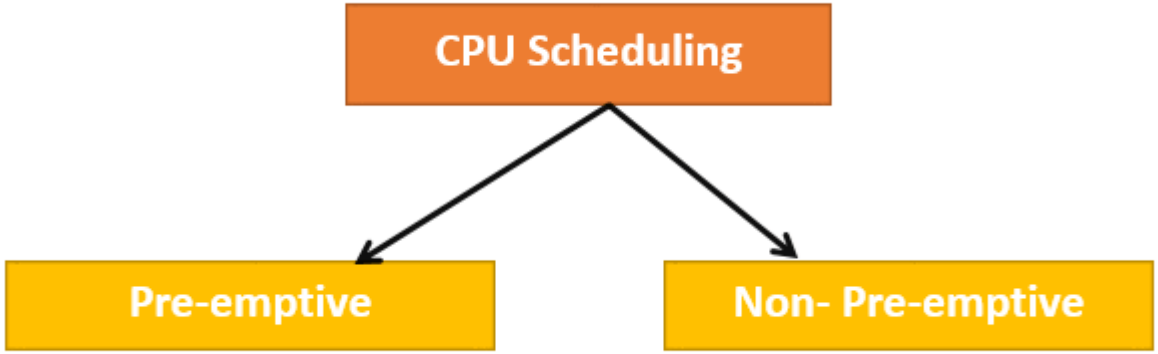
- Max CPU utilization [Keep CPU as busy as possible]
- Fair allocation of CPU.
- Max throughput [Number of processes that complete their execution per time unit]
- Min turnaround time [Time taken by a process to finish execution]
- Min waiting time [Time a process waits in ready queue]
- Min response time [Time when a process produces first response]

Arrival Time: *Time at which the process arrives in the ready queue.*
Completion Time: *Time at which process completes its execution.*
Burst Time: *Time required by a process for CPU execution.*
Turn Around Time: *Time Difference between completion time and arrival time.*
Turn Around Time = Completion Time – Arrival Time
Waiting Time(W.T): *Time Difference between turn around time and burst time.*
Waiting Time = Turn Around Time – Burst Time
Response time: It is an amount to time in which the request was submitted until the first response is produced.



Maximize:
CPU Utilization
Throughput

Minimize:
Turnaround time
Waiting time
Response time



Preemptive Scheduling

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

Non-Preemptive Scheduling

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

When scheduling is Preemptive or Non-Preemptive?

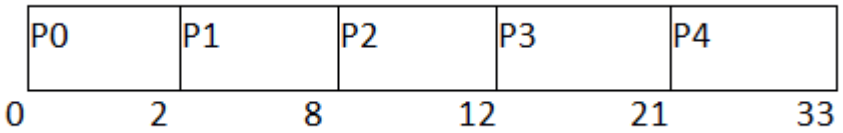
- To determine if scheduling is preemptive or non-preemptive, consider these four parameters:
- A process switches from the running to the waiting state.
- Specific process switches from the running state to the ready state.
- Specific process switches from the waiting state to the ready state.
- Process finished its execution and terminated.
- **Only conditions 1 and 4 apply, the scheduling is called non- preemptive.**
- **All other scheduling are preemptive.**

FCFS

- **First come first serve** (FCFS) scheduling algorithm simply schedules the jobs according to their arrival time.
- The job which comes first in the ready queue will get the CPU first.
- The lesser the arrival time of the job, the sooner will the job get the CPU.
- FCFS scheduling may cause the problem of starvation if the burst time of the first process is the longest among all the jobs.

Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
0	0	2	2	2	0
1	1	6	8	7	1
2	2	4	12	10	6
3	3	9	21	18	9
4	6	12	33	29	17

Avg Waiting Time=33/5

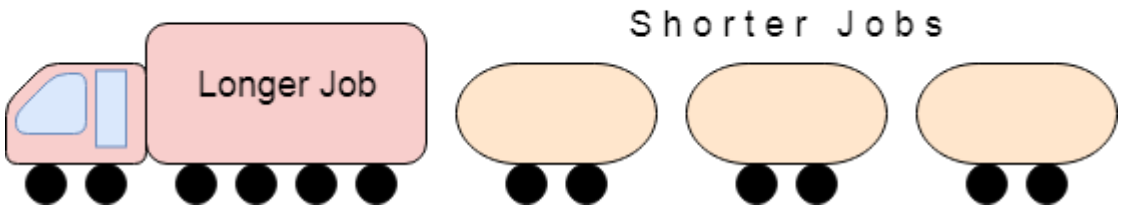


Gantt chart

Convoy Effect in FCFS

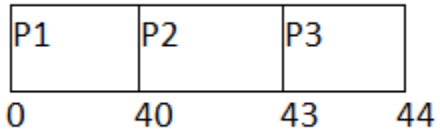
- FCFS may suffer from the **convoy effect** if the burst time of the first job is the highest among all.
- As in the real life, if a convoy is passing through the road then the other persons may get blocked until it passes completely. This can be simulated in the Operating System also.
- If the CPU gets the processes of the higher burst time at the front end of the ready queue then the processes of lower burst time may get blocked which means they may never get the CPU if the job in the execution has a very high burst time. This is called **convoy effect** or **starvation**.

The Convoy Effect, Visualized Starvation

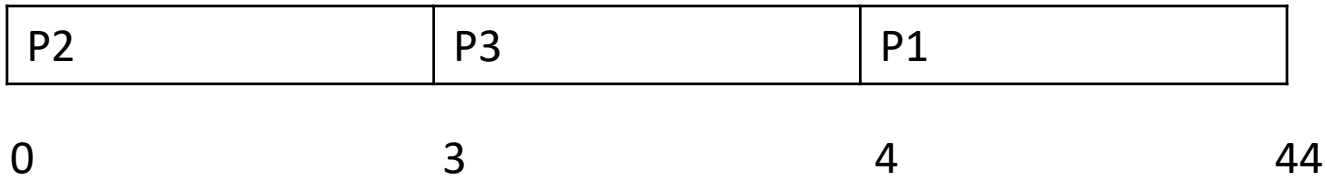


Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
1	0	40	40	40	0
2	1	3	43	42	39
3	1	1	44	43	42

Avg waiting Time = 81/3



Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
1	1	40	44	43	3
2	0	3	3	3	0
3	0	1	4	4	3



Avg Waiting Time=6/3

Advantages of FCFS

- Simple
- Easy
- First come, First serv

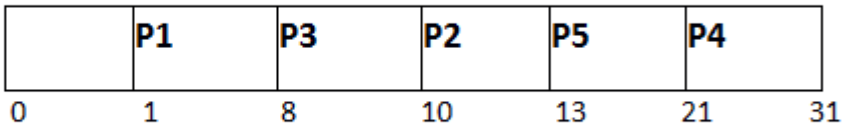
Disadvantages of FCFS

- The scheduling method is non preemptive, the process will run to the completion.
- Due to the non-preemptive nature of the algorithm, the problem of starvation may occur.
- Although it is easy to implement, but it is poor in performance since the average waiting time is higher as compare to other scheduling algorithms.

Shortest Job First (SJF) Scheduling

- Till now, we were scheduling the processes according to their arrival time (in FCFS scheduling).
- However, SJF scheduling algorithm, schedules the processes according to their burst time.
- In SJF scheduling, the process with the lowest burst time, among the list of available processes in the ready queue, is going to be scheduled next.
- However, it is very difficult to predict the burst time needed for a process hence this algorithm is very difficult to implement in the system.

PID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
1	1	7	8	7	0
2	3	3	13	10	7
3	6	2	10	4	2
4	7	10	31	24	14
5	9	8	21	12	4

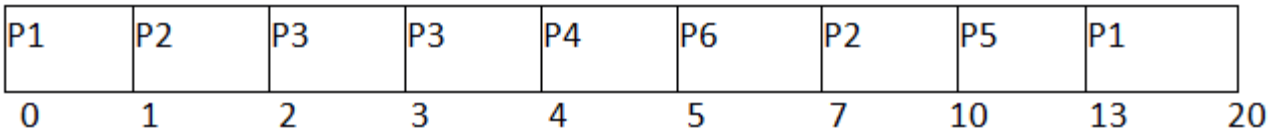


Avg Waiting Time = 27/5

Shortest Remaining Time First (SRTF) Scheduling Algorithm

- This Algorithm is the **preemptive version** of **SJF scheduling**. In SRTF, the execution of the process can be stopped after certain amount of time
- At the arrival of every process, the short term scheduler schedules the process with the least remaining burst time among the list of available processes and the running process.
- Once all the processes are available in the **ready queue**, No preemption will be done and the algorithm will work as **SJF scheduling**.
- The context of the process is saved in the **Process Control Block** when the process is removed from the execution and the next process is scheduled. This PCB is accessed on the **next execution** of this process.

Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time	Response Time
1	0	8	20	20	12	0
2	1	4	10	9	5	1
3	2	2	4	2	0	2
4	3	1	5	2	1	4
5	4	3	13	9	6	10
6	5	2	7	2	0	5



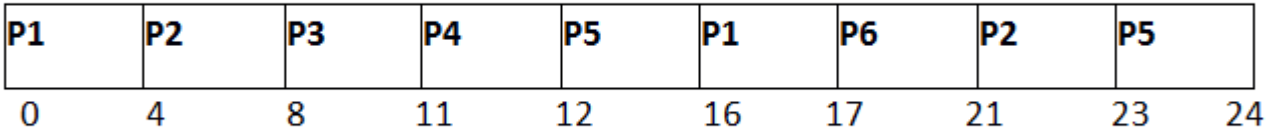
Avg Waiting Time = 24/6

Round Robin Scheduling Algorithm

- Round Robin scheduling algorithm is one of the most popular scheduling algorithm which can actually be implemented in most of the operating systems.
- This is the **preemptive version** of first come first serve scheduling. The Algorithm focuses on Time Sharing. In this algorithm, every process gets executed in a **cyclic way**.
- A certain time slice is defined in the system which is called time **quantum**.
- Each process present in the ready queue is assigned the CPU for that time quantum, if the execution of the process is completed during that time then the process will **terminate** else the process will go back to the **ready queue** and waits for the next turn to complete the execution.

Time quantum=4

Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
1	0	5	17	17	12
2	1	6	23	22	16
3	2	3	11	9	6
4	3	1	12	9	8
5	4	5	24	20	15
6	6	4	21	15	11



Avg Waiting Time = (12+16+6+8+15+11)/6 = 76/6 units

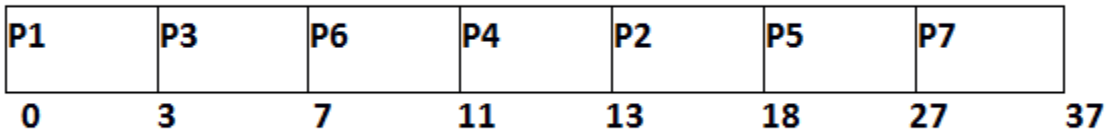
Priority Scheduling

- In Priority scheduling, there is a priority number assigned to each process. In some systems, the lower the number, the higher the priority.
- While, in the others, the higher the number, the higher will be the priority. The Process with the higher priority among the available processes is given the CPU.
- There are two types of priority scheduling algorithm exists. One is **Preemptive** priority scheduling while the other is **Non Preemptive** Priority scheduling.

Non Preemptive Priority Scheduling

- In the Non Preemptive Priority scheduling, The Processes are scheduled according to the priority number assigned to them.
- Once the process gets scheduled, it will run till the completion.
- Generally, the lower the priority number, the higher is the priority of the process.

Process Id	Priority	Arrival Time	Burst Time	Completion Time	Turnaround Time	Waiting Time	Response Time
1	2	0	3	3	3	0	0
2	6	2	5	18	16	11	13
3	3	1	4	7	6	2	3
4	5	4	2	13	9	7	11
5	7	6	9	27	21	12	18
6	4	5	4	11	6	2	7
7	10	7	10	37	30	18	27



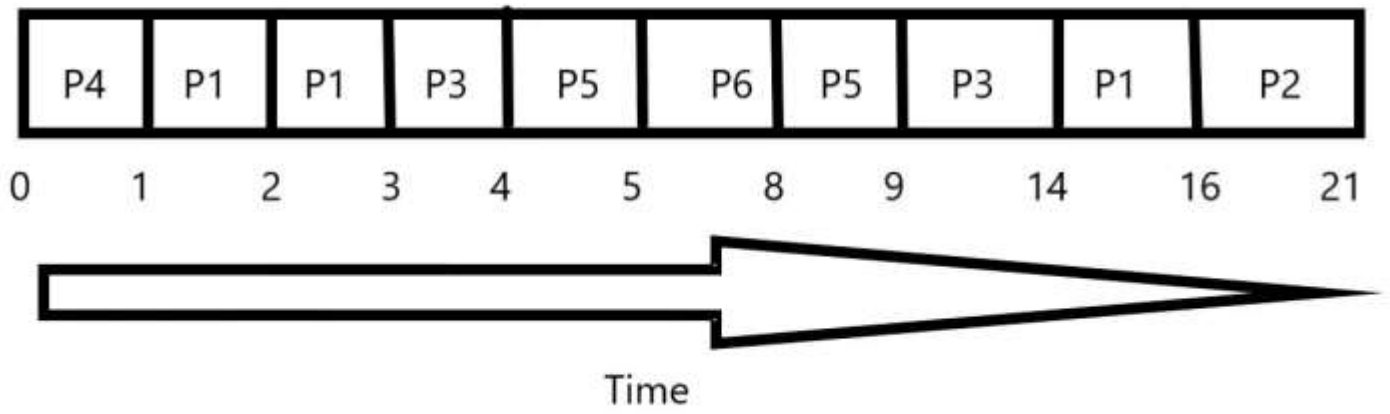
Preemptive Priority Scheduling

- In Preemptive Priority Scheduling, at the time of arrival of a process in the ready queue, its Priority is compared with the priority of the other processes present in the ready queue as well as with the one which is being executed by the CPU at that point of time.
- The One with the highest priority among all the available processes will be given the CPU next.
- The difference between preemptive priority scheduling and non preemptive priority scheduling is that, in the preemptive priority scheduling, the job which is being executed can be stopped at the arrival of a higher priority job.
- Once all the jobs get available in the ready queue, the algorithm will behave as non-preemptive priority scheduling, which means the job scheduled will run till the completion and no preemption will be done.

Process	Arrival Time	Priority	Burst Time
P1	1	5	4
P2	2	2 (lowest)	5
P3	3	6	6
P4	0	4	1
P5	4	7	2
P6	5	8 (highest)	3

1. At 0 arrival time we have process P4 which will execute for 1ms. Now, remaining time for P4 is 0ms so it won't be added to table T.
2. At arrival time 1 we have process P1 which will execute for 1ms. Now, remaining time for P1 is 3ms so it will be added to table T.
3. At arrival time 2 we have process P2. Compare priority of P2 and P1. Clearly, P1 has higher priority so P1 will be scheduled again for 1ms.Now, remaining time for P1 is 2ms. P2 will also be added to table T with remaining time 5ms.
4. At arrival time 3 we have process P3. Compare priority of P2, P1, P3. Clearly, P3 has higher priority so P3 will be scheduled for 1ms.Now, remaining time for P3 is 5ms. P3 will also be added to table T with remaining time 5ms.

5. At arrival time 4 we have process P5. Compare priority of P2, P1, P3, P5. Clearly, P5 has higher priority so P5 will be scheduled for 1ms.Now, remaining time for P5 is 1ms. P5 will also be added to table T with remaining time 1ms.
6. At arrival time 5 we have process P6. Compare priority of P2, P1, P3, P5, P6. Clearly, P6 has higher priority so P6 will be scheduled.Since no process have arrived after P6 thus P6 will execute for complete 3ms.
7. P6 has completed execution.
8. Remaining process in table T are P2, P1, P3, P5. Clearly, P5 has highest priority so P5 will be scheduled for its remaining time which is 1ms.
9. P5 has completed execution.
10. Remaining process in table T are P2, P1, P3. Clearly, P3 has highest priority so P3 will be scheduled for its remaining time which is 5ms.
11. P3 has also completed execution.
12. Remaining process in table T are P2, P1. Clearly, P1 has highest priority so P1 will be scheduled for its remaining time which is 2ms.
13. P1 has also completed execution.
14. Remaining process in table T are P2 so P2 will be executed for 5ms.
15. P2 has also completed execution.

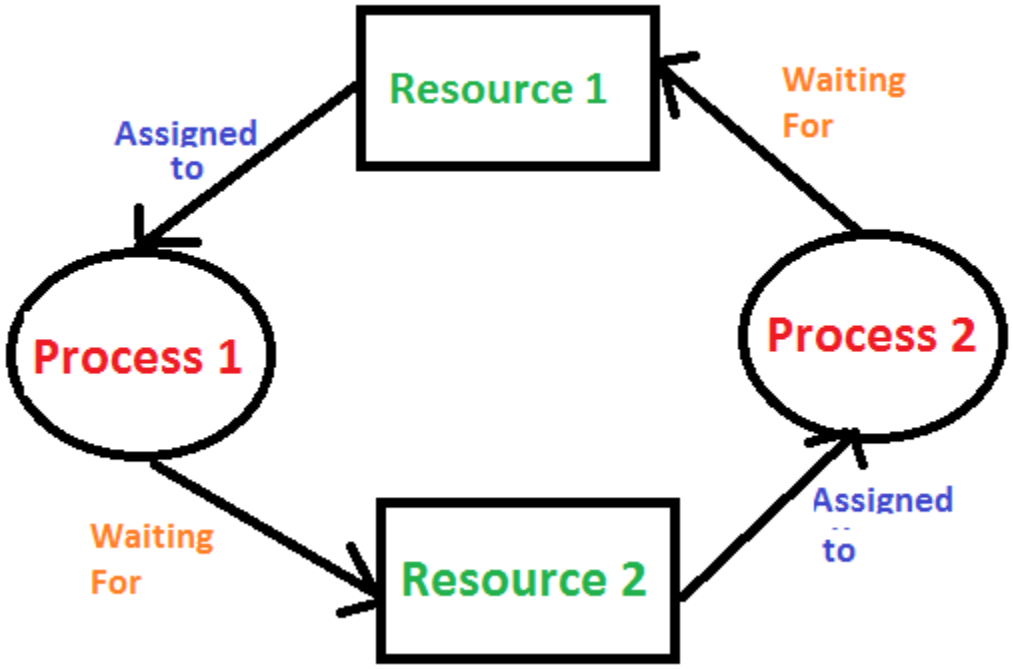


Process	AT	Priority	BT	CT	TA	WT
P1	1	5	4	16	15	11
P2	2	2	5	21	19	14
P3	3	6	6	14	11	5
P4	0	4	1	1	1	0
P5	4	7	2	9	5	3
P6	5	8	3	8	3	0

Introduction to Dead lock

- A process in operating systems uses different resources and uses resources in the following way.
  1) Requests a resource
  2) Use the resource
  3) Releases the resource

Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.



Deadlock can arise if the following four conditions hold simultaneously (Necessary Conditions)

Mutual Exclusion: One or more than one resource are non-shareable (Only one process can use at a time)

Hold and Wait: A process is holding at least one resource and waiting for resources.

No Preemption: A resource cannot be taken from a process unless the process releases the resource.

Circular Wait: A set of processes are waiting for each other in circular form.

Strategies for handling Deadlock

- Deadlock prevention
- Deadlock avoidance
- Deadlock detection and recovery

Deadlock prevention

- To prevent a deadlock, we must ensure that any one of the following four conditions listed below should not hold.

1. Mutual exclusion condition
2. Hold and wait condition
3. Non-Preemption condition
4. Circular wait condition

Deadlock avoidance

- Deadlock avoidance can be done with Banker's Algorithm.
- **Banker'sAlgorithm**
  Bankers's Algorithm is resource allocation and deadlock avoidance algorithm which test all the request made by processes for resources, it checks for the safe state, if after granting request system remains in the safe state it allows the request and if there is no safe state it doesn't allow the request made by the process.

releases the requested

We need the following data structures:

1. **Available:** A vector of length m indicates the number of available resources of each type. If Available$[j] = k$, there are k instances available of resource type R_j.
2. **Max:** An $n \times m$ matrix defines the maximum demand of each process, if Max$[i, j] = k$, then process P_i may request at most 'k' instances of resource type R_j.
3. **Allocation:** An $n \times m$ matrix defines the number of resources of each type currently allocated to each process.
   - If Allocation$[i, j] = k$
   - Then, process P_i is currently allocated 'k' instances of resource type R_j.
4. **Need:** An $n \times m$ matrix indicates the remaining resources needed for each process.
   - If Need$[i,j] = k$
   - Then, process P_i may need 'k' more instances of resource type R_j to complete its task.
   - Further, Need$[i, j]$ = Max$[i, j]$ – Allocation$[i, j]$

Problem 7.2: Consider a system having five processes and three resources types A, B and C. Resource type A has nine instances, B has four instances and C has seven instances. Initially, the snapshot of the given system is as follows:

Process	Allocated	Maximum	Available
	ABC	ABC	ABC
P_0	0,1,0	7,5,3	2,2,2
P_1	2,0,0	3,2,2	
P_2	3,0,2	9,0,2	
P_3	2,1,1	2,2,2	
P_4	0,0,2	4,3,3	

Using Banker's algorithm, how can we avoid the deadlock?

Solution: We will start with the first part of the Banker's algorithm, as follows:

Step 1.1: We will first calculate the Need matrix

Need[i] = Max[i] − Allocation [i]

Process	Need
	ABC
P_0	7,4,3
P_1	1,2,2
P_2	6,0,0
P_3	0,1,1
P_4	4,3,1

Step 1.2: Now, we will decide which process should execute first, that is, what are the processes that need resources which are less than available.

Process	Need	Available
	ABC	ABC
P_0	7,4,3	2,2,2
P_1	1,2,2	

So, we will start with P_1 (because its resource need is less than the available resource) and after its completion all the resources will be free; hence, after completion of P_1 the available matrix contents will be

Available
2 2 2
\+ 2 0 0 (resources held by P_1)
4 2 2

Step 1.3: Again let us see the status

Process	Need	Available
	ABC	ABC
P_0	7,4,3	4,2,2
P_1	Complete	
P_2	6,0,0	
P_3	0,1,1	
P_4	4,3,1	

Now, we observe that P_3 can be executed, and is considered to be executed.

Step 1.4: Again, let us see the status

Process	Need	Available
	ABC	ABC
P_0	7,4,3	6,3,3
P_1	Complete	
P_2	6,0,0	
P_3	Complete	
P_4	4,3,1	

So, P_4 executes and similarly we can say that P_2 and then P_0 will be executed, and as a result, the available matrix after execution of each process would be:

Available	6,3,3
P_4 executes	0,0,2 (+)
	6,3,5

After the execution of P_2

Available	6,3,5
P_2 executes	3,0,2
	9,3,7

Step 1.2: Now, we will decide which process should execute first, that is, what are the processes that need resources which are less than available.

Process	Need	Available
	ABC	ABC
P_0	7,4,3	2,3,2
P_1	1,2,2	

So, we will start with P_1 (because its resource need is less than the available resource) and after its completion all the resources will be free; hence, after completion of P_1 the available matrix contents will be

Available
2 2 2
\+ 2 0 0 (resources held by P_1)
4 2 2

Step 1.3: Again let us see the status

Process	Need	Available
	ABC	ABC
P_0	7,4,3	4,2,2
P_1	Complete	
P_2	6,0,0	
P_3	0,1,1	
P_4	4,3,1	

Now, we observe that P_3 can be executed, and is considered to be executed.

Step 1.4: Again, let us see the status

Process	Need	Available
	ABC	ABC
P_0	7,4,3	6,3,3
P_1	Complete	
P_2	6,0,0	
P_3	Complete	
P_4	4,3,1	

So, P_4 executes and similarly we can say that P_2 and then P_0 will be executed, and as a result, the available matrix after execution of each process would be:

Available	6,3,3
P_4 executes	0,0,2 (+)
	6,3,5

After the execution of P_2

Available	6,3,5
P_2 executes	3,0,2
	9,3,7

After the execution of P_0

Available	9,3,7
P_0 executes	0,1,0
	9,4,7

Thus, the final status of the system under consideration will be:

Process	Need	Available
	ABC	ABC
P_0	Complete	9,4,7
P_1	Complete	
P_2	Complete	
P_3	Complete	
P_4	Complete	

Therefore, the proposed sequence of execution is $[P_1, P_3, P_4, P_2, P_0]$. Now, we will find whether this sequence of execution is a 'safe sequence'. We will now consider the resource request algorithm (the second portion) of the Banker's algorithm. In the beginning, we have:

Step 2.1:

Work = [2,2,2] (i.e., avail at the start)

Finish = [F, F, F, F, F] (the finish value of all processes is F meaning False)

So, P_1 is executed first, and we observe that

$Need_1$ < Work: [1,2,2] < [2,2,2] holds good, and as a result we have

Work = [4,2,2]

Finish = [F,T,F,F,F] (the finish value of P_1 is T meaning True)

Step 2.2:

Next, we have P_3 executed and in this case the condition $Need_3$ < Work holds good, and as a result we have,

Work = [6,3,3]

Finish = [F,T,F,T,F]

After the execution of P_4, we have

Step 2.3:

Work = [6,3,5]

Finish = [F,T,F,T,T]

After the execution of P_2, we have

Step 2.4:

Work = [9,3,7]

Finish = [F,T,T,T,T]

After the execution of P_0, we have

Step 2.5:

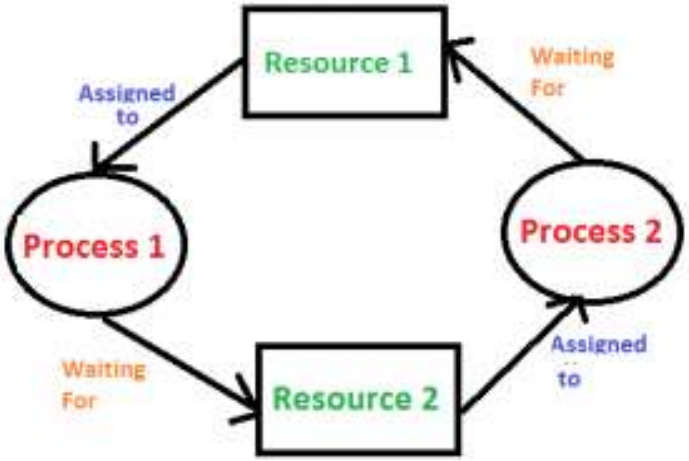
Work = [9,4,7]

Finish = [T,T,T,T,T]

In the whole process, we have never encountered a situation where the condition $Need_i$ < Work does not hold good. Therefore, we can execute our processes in the order $[P_1, P_3, P_4, P_2, P_0]$ without encountering the deadlock.

Deadlock detection and recovery

- **Deadlock Detection**
- If resources have single instance:
  In this case for Deadlock detection we can run an algorithm to check for cycle in the Resource Allocation Graph. Presence of cycle in the graph is the sufficient condition for deadlock.



In the above diagram, resource 1 and resource 2 have single instances.
There is a cycle R1 → P1 → R2 → P2. So, Deadlock is Confirmed.

If there are multiple instances of resources:
Detection of the cycle is necessary but not sufficient condition for deadlock detection, in this case, the system may or may not be in deadlock varies according to different situations.

Deadlock Recovery

- A traditional operating system such as Windows doesn't deal with deadlock recovery as it is time and space consuming process. Real-time operating systems use Deadlock recovery.

Recovery method

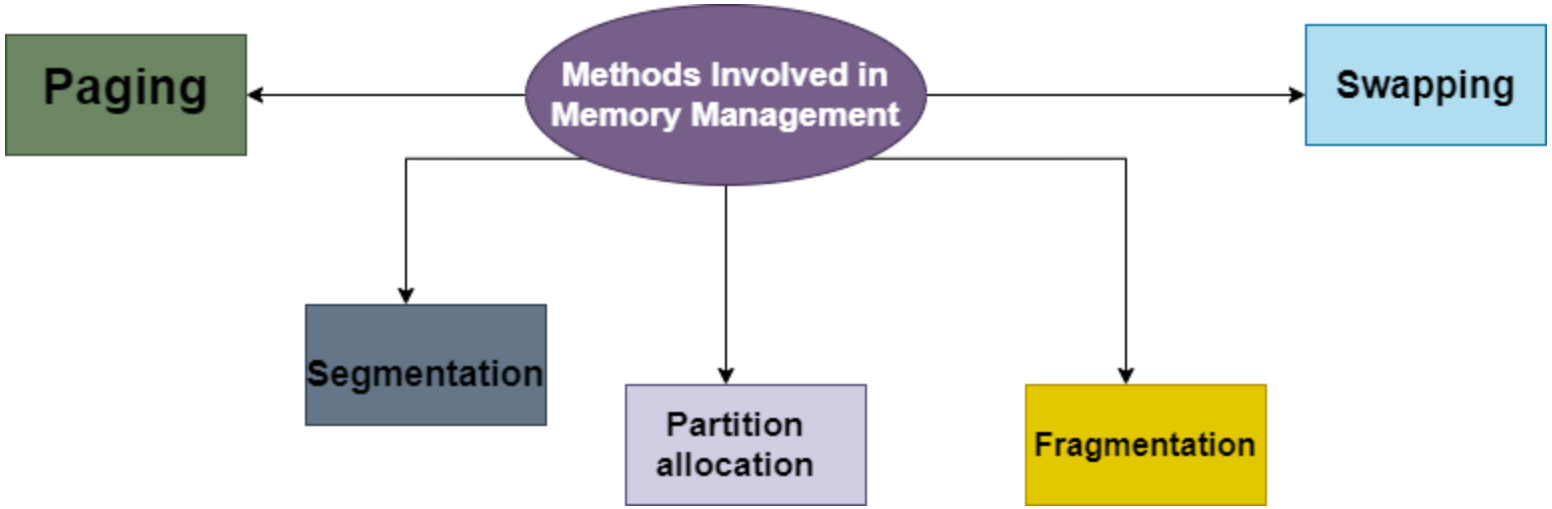
- **Killing the process:** killing all the process involved in the deadlock. Killing process one by one. After killing each process check for deadlock again keep repeating the process till system recover from deadlock.
- **Resource Preemption:** Resources are preempted from the processes involved in the deadlock, preempted resources are allocated to other processes so that there is a possibility of recovering the system from deadlock. In this case, the system goes into starvation.

Memory Management

- **Computer Memory** is basically known as a collection of data that is represented in binary format**. Main Memory** refers to a physical memory that is the internal memory of the computer. The word main is used to distinguish it from external mass storage devices such as disk drives.
- Main memory is also known as RAM. The computer is able to change only data that is in the main memory. Therefore, every program we execute and every file we access must be copied from a storage device into main memory.
- All the programs are loaded in the main memory for execution. Sometimes the complete program is loaded into the memory, but sometimes a certain part or routine of the program is loaded into the main memory only when it is called by the program, this mechanism is called **Dynamic Loading**, which enhances the performance.
- Also, at times one program is dependent on some other program. In such a case, rather than loading all the dependent programs, the CPU links the dependent programs to the main executing program when required. This mechanism is known as **Dynamic Linking**.

Introduction

- Memory Management is the process of coordinating and controlling the memory in a computer, Blocks are assigning portions that are assigned to various running programs in order to optimize the overall performance of the system.
- This technique helps in keeping the track of each and every memory location, whether the memory location is allocated to some process or it is free.

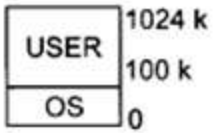

Paging
Methods Involved in Memory Management
Swapping
Segmentation
Partition allocation
Fragmentation

Swapping in OS

- A process needs to be in memory for execution. But sometimes there is not enough main memory to hold all the currently active processes in a timesharing system.
- So, the excess process is kept on disk and brought in to run dynamically. Swapping is the process of bringing in each process in the main memory, running it for a while, and then putting it back to the disk.

Multiple Partition Allocation: The multiple partition allocation may be further classified as

Single Partition Allocation: The memory is divided into two parts. One to be used by os and the other one is for user programs. The as code and data is protected from being modified by user programs using a base register.



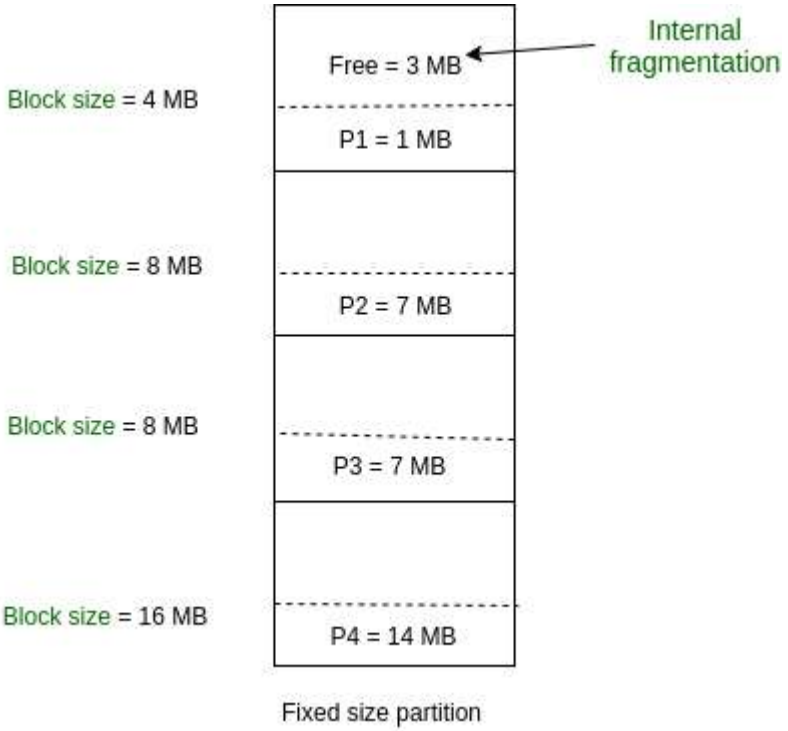
Single partition allocation

Multiple Partition Allocation: The multiple partition allocation may be further classified as

1. Fixed partition scheme
2. Variable partition scheme

Fixed Partition Scheme

- This is the oldest and simplest technique used to put more than one processes in the main memory.
- 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 or during system configure.



As illustrated 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) = 3MB.
Sum of Internal Fragmentation in every block = (4-1)+(8-7)+(8-7)+(16-14)= 3+1+1+2 = 7MB.

Suppose process P5 of size 7MB comes. But this process cannot be accommodated inspite of available free space because of contiguous allocation (as spanning is not allowed). Hence, 7MB becomes part of External Fragmentation.

Advantages of Fixed Partitioning –

- **Easy to implement:**
  Algorithms needed to implement Fixed Partitioning are easy to implement. It simply requires putting a process into certain partition without focussing on the emergence of Internal and External Fragmentation.
- **Little OS overhead:**
  Processing of Fixed Partitioning require lesser excess and indirect computational power.

Disadvantages of Fixed Partitioning –

- **Internal Fragmentation:**
  Main memory use is inefficient. Any program, no matter how small, occupies an entire partition. This can cause internal fragmentation.
- **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).
- **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. Hence, process size of 32MB in above stated example is invalid.
- **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. Suppose if there are partitions in RAM and are the number of processes, then condition must be fulfilled. Number of processes greater than number of partitions in RAM is invalid in Fixed Partitioning

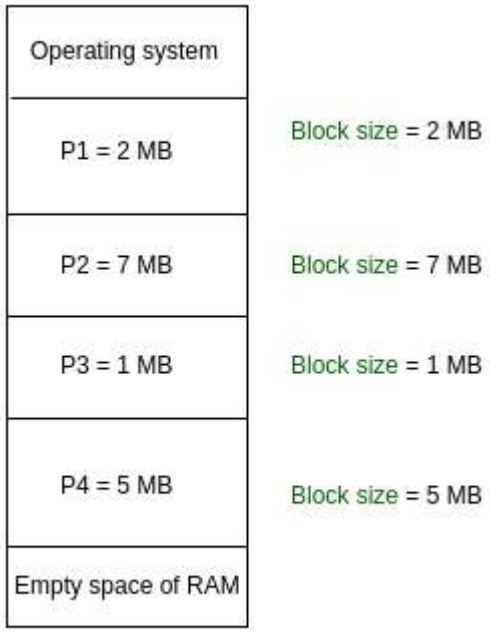
Overlays

- The main problem in Fixed partitioning is the size of a process has to be limited by the maximum size of the partition, which means a process can never be span over another.In order to solve this problem, earlier people have used some solution which is called as Overlays.
- 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 done, then you just unload it, means just pull it back and get the new part you required and run it.
  Formally,
  “The process of **transferring a block** of program code or other data into internal memory, replacing what is already stored”.

Variable Partitioning

- It is a part of Contiguous allocation technique. It is used to alleviate the problem faced by Fixed Partitioning. In contrast with fixed partitioning, partitions are not made before the execution or during system configure. Various **features** associated with variable Partitioning-
- Initially RAM is empty and partitions are made during the run-time according to process's need instead of partitioning during system configure.
- The size of partition will be equal to incoming process.
- The partition size varies according to the need of the process so that the internal fragmentation can be avoided to ensure efficient utilisation of RAM.
- Number of partitions in RAM is not fixed and depends on the number of incoming process and Main Memory's size.

Dynamic partitioning



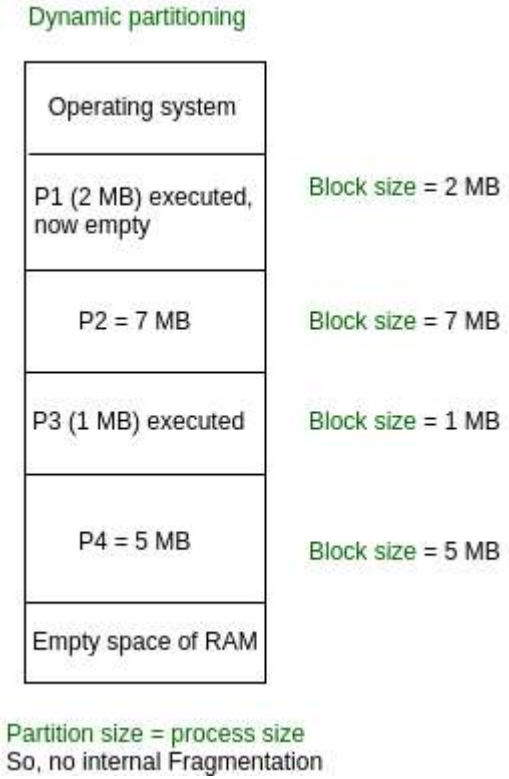
Partition size = process size
So, no internal Fragmentation

Advantages of Variable Partitioning

- **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.
- **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.
- **No Limitation on the size of the process:**
  In Fixed partitioning, the process with the size greater than the size of the largest partition could not be loaded and process can not be divided as it is invalid in contiguous allocation technique. Here, 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

- **Difficult 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.
- **External Fragmentation:**
  There will be external fragmentation inspite of absence of internal fragmentation.
- For example, suppose in above example- process P1(2MB) and process P3(1MB) completed their execution. Hence two spaces are left i.e. 2MB and 1MB. Let's suppose process P5 of size 3MB comes. The empty space in memory cannot be allocated as no spanning is allowed in contiguous allocation. The rule says that process must be contiguously present in main memory to get executed. Hence it results in External Fragmentation



Now P5 of size 3 MB cannot be accommodated in spite of required available space because in contiguous no spanning is allowed.

Compaction

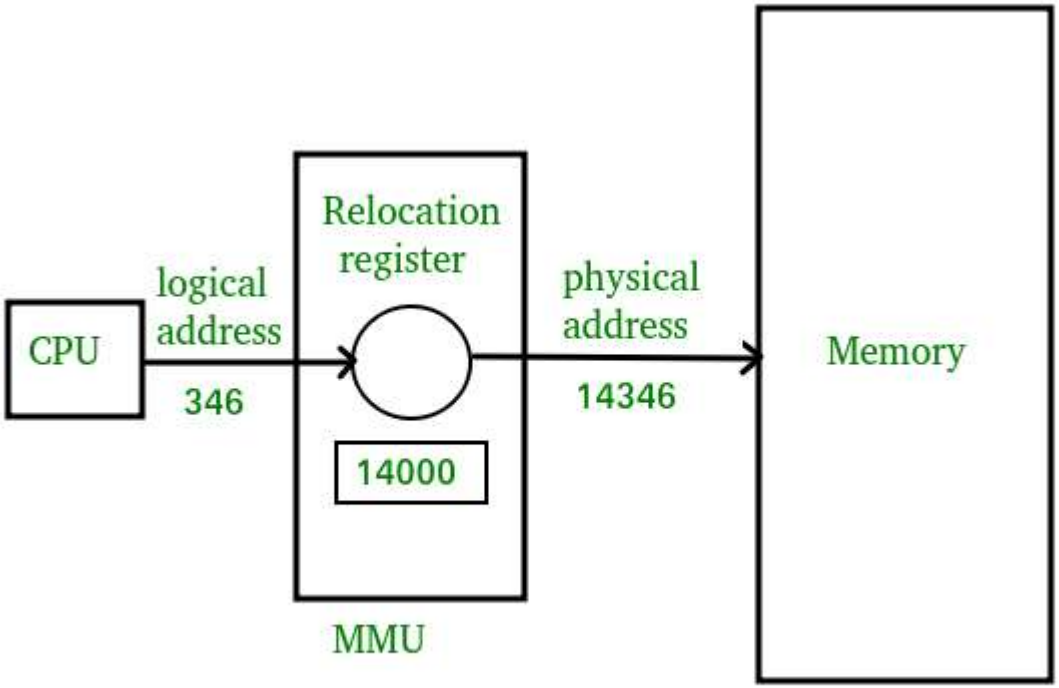
- This is strategy to solve the problem of external fragmentation.
- All free memory is placed together by moving processes to new locations

Non-Contiguous Allocation in Operating System

- Paging and Segmentation are the two ways which allow a process's physical address space to be non-contiguous.
- It has **advantage** of reducing memory wastage but it increases the overheads due to address translation.
- It slows the execution of the memory because time is consumed in address translation.
- In non-contiguous allocation, Operating system needs to maintain the table which is called **Page Table** for each process which contains the base address of the each block which is acquired by the process in memory space.
- In non-contiguous memory allocation, different parts of a process is allocated different places in Main Memory. Spanning is allowed which is not possible in other techniques like Dynamic or Static Contiguous memory allocation.
- That's why paging is needed to ensure effective memory allocation. Paging is done to remove External Fragmentation.

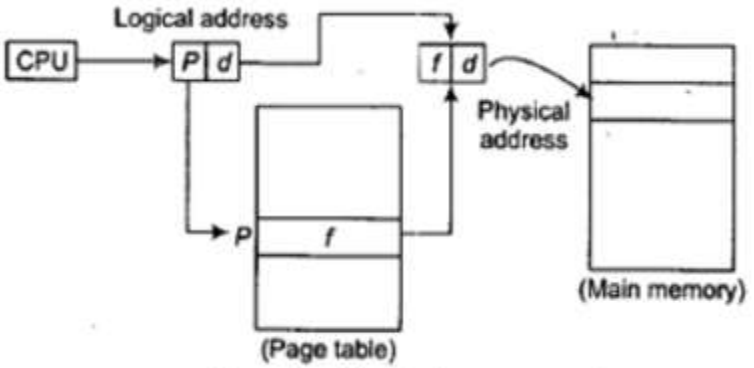
Logical and Physical Address in Operating System

- **Logical Address** is generated by CPU while a program is running. The logical address is virtual address as it does not exist physically, therefore, it is also known as Virtual Address.
- This address is used as a reference to access the physical memory location by CPU. The term Logical Address Space is used for the set of all logical addresses generated by a program's perspective. The hardware device called Memory-Management Unit is used for mapping logical address to its corresponding physical address.
- **Physical Address** identifies a physical location of required data in a memory. The user never directly deals with the physical address but can access by its corresponding logical address.
- The user program generates the logical address and thinks that the program is running in this logical address but the program needs physical memory for its execution, therefore, the logical address must be mapped to the physical address by MMU before they are used. The term Physical Address Space is used for all physical addresses corresponding to the logical addresses in a Logical address space.


Relocation
register
logical
address
CPU
346
14000
MMU
physical
address
14346
Memory

Paging in Operating System

- **Paging:** It is a memory management technique, which allows the memory to be allocated to the process wherever it is available.
- Physical memory is divided into fixed size blocks called frames. Logical memory is broken into blocks of same size called pages.
- The backing store is also divided into same size blocks.
- When a process is to be executed its pages are loaded into available page frames.
- A frame is a collection of contiguous pages. Every logical address generated by the CPU is divided into two parts. The page number (P) and the page offset (d).
- The page number is used as an index into a page table.



A paging block diagram

Each entry in the page table contains the base address of the page in physical memory (f).
The base address of the Pth entry is then combined with the offset (d) to give the actual address in memory.

Logical Address or Virtual Address (represented in bits): An address generated by the CPU

Logical Address Space or Virtual Address Space(represented in words or bytes): The set of all logical addresses generated by a program

Physical Address (represented in bits): An address actually available on memory unit

Physical Address Space (represented in words or bytes): The set of all physical addresses corresponding to the logical addresses

Example:

If Logical Address = 31 bit, then Logical Address Space = 2^{31} words = 2 G words

(1 G = 2^{30})

If Logical Address Space = 128 M words = $2^7 * 2^{20}$ words, then

Logical Address = $\log_2 2^{27}$ = 27 bits

If Physical Address = 22 bit, then Physical Address Space = 2^{22} words = 4 M words

(1 M = 2^{20})

If Physical Address Space = 16 M words = $2^4 * 2^{20}$ words, then

Physical Address = $\log_2 2^{24}$ = 24 bits

The mapping from virtual to physical address is done by the memory management unit (MMU) which is a hardware device and this mapping is known as paging technique.

The Physical Address Space is conceptually divided into a number of fixed-size blocks, called **frames**.

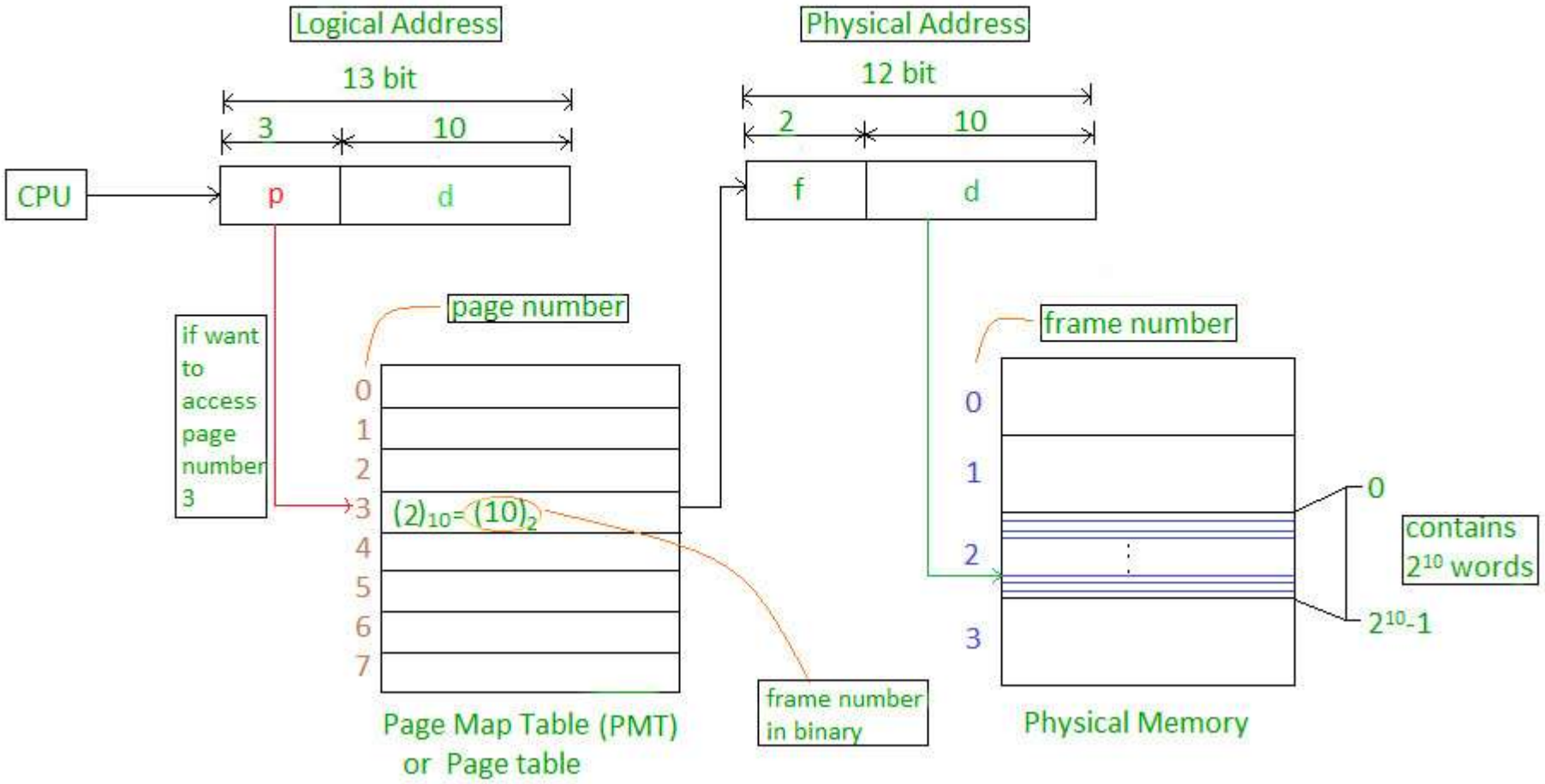
The Logical address Space is also splitted into fixed-size blocks, called **pages**.

Page Size = Frame Size

- Let us consider an example:
- Physical Address = 12 bits, then Physical Address Space = 4 K words
- Logical Address = 13 bits, then Logical Address Space = 8 K words
- Page size = frame size = 1 K words (assumption)

Number of frames = Physical Address Space / Frame size = 4 K / 1 K = 4 = 2^2

Number of pages = Logical Address Space / Page size = 8 K / 1 K = 8 = 2^3



Address generated by CPU is divided into

Page number(p): Number of bits required to represent the pages in Logical Address Space or Page number

Page offset(d): Number of bits required to represent particular word in a page or page size of Logical Address Space or word number of a page or page offset.

Physical Address is divided into

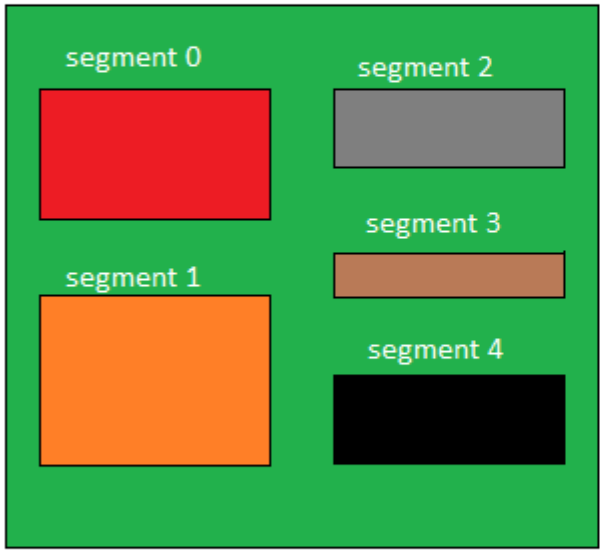
Frame number(f): Number of bits required to represent the frame of Physical Address Space or Frame number.

Frame offset(d): Number of bits required to represent particular word in a frame or frame size of Physical Address Space or word number of a frame or frame offset

Segmentation

- In Operating Systems, Segmentation is a memory management technique in which the memory is divided into the variable size parts. Each part is known as a segment which can be allocated to a process.
- The details about each segment are stored in a table called a segment table. Segment table is stored in one (or many) of the segments.
- Segment table contains mainly two information about segment:
- Base: It is the base address of the segment
- Limit: It is the length of the segment.

Logical View of Segmentation

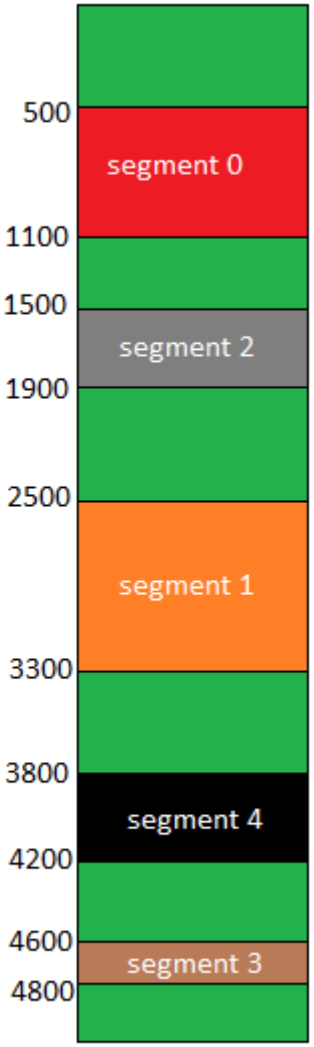


Logical Address Space

Segment Number

	base address	Limit
0	500	600
1	2500	800
2	1500	400
3	4600	200
4	3800	400

Segment Table



Physical Address Space

Advantages of Segmentation

- No internal fragmentation
- Average Segment Size is larger than the actual page size.
- Less overhead
- It is easier to relocate segments than entire address space.
- The segment table is of lesser size as compared to the page table in paging.

Disadvantages

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

Sr No.	Paging	Segmentation
1	Non-Contiguous memory allocation	Non-contiguous memory allocation
2	Paging divides program into fixed size pages.	Segmentation divides program into variable size segments.
3	OS is responsible	Compiler is responsible.
4	Paging is faster than segmentation	Segmentation is slower than paging
5	Paging is closer to Operating System	Segmentation is closer to User
6	It suffers from internal fragmentation	It suffers from external fragmentation
7	There is no external fragmentation	There is no external fragmentation
8	Logical address is divided into page number and page offset	Logical address is divided into segment number and segment offset
9	Page table is used to maintain the page information.	Segment Table maintains the segment information
10	Page table entry has the frame number and some flag bits to represent details about pages.	Segment table entry has the base address of the segment and some protection bits for the segments.

Virtual Memory

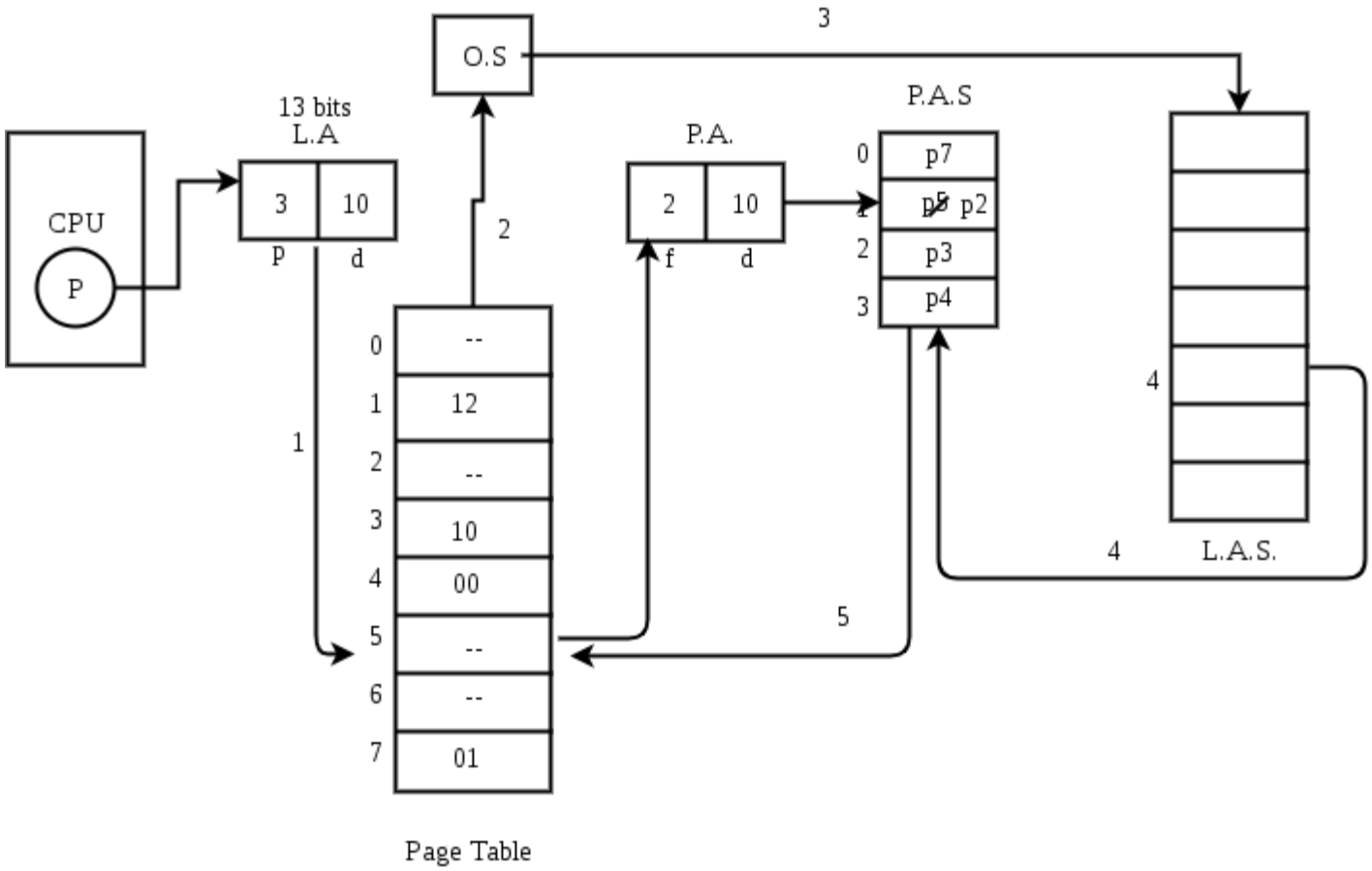
- Separation of user logical memory from physical memory. It is a technique to run process size more than main memory. Virtual memory is a memory management scheme which allows the execution of a partially loaded process.

Advantages of Virtual Memory

- The advantages of virtual memory can be given as
- Logical address space can therefore be much larger than physical address space.
- Allows address spaces to be shared by several processes.
- Less I/O is required to load or swap a process in memory, so each user can run faster.

Demand Paging :

The process of loading the page into memory on demand (whenever page fault occurs) is known as demand paging.



Contd..

- If CPU try to refer a page that is currently not available in the main memory, it generates an interrupt indicating memory access fault.
- The OS puts the interrupted process in a blocking state. For the execution to proceed the OS must bring the required page into the memory.
- The OS will search for the required page in the logical address space.
- The required page will be brought from logical address space to physical address space. The page replacement algorithms are used for the decision making of replacing the page in physical address space.
- The page table will updated accordingly.
- The signal will be sent to the CPU to continue the program execution and it will place the process back into ready state.
- Hence whenever a page fault occurs these steps are followed by the operating system and the required page is brought into memory.

- **Page Fault Service Time :**
  The time taken to service the page fault is called as page fault service time. The page fault service time includes the time taken to perform all the above six steps.
- Let Main memory access time is: m
- Page fault service time is: s
- Page fault rate is : p
- Then, Effective memory access time = (p*s) + (1-p)*m

Page Replacement Algorithms

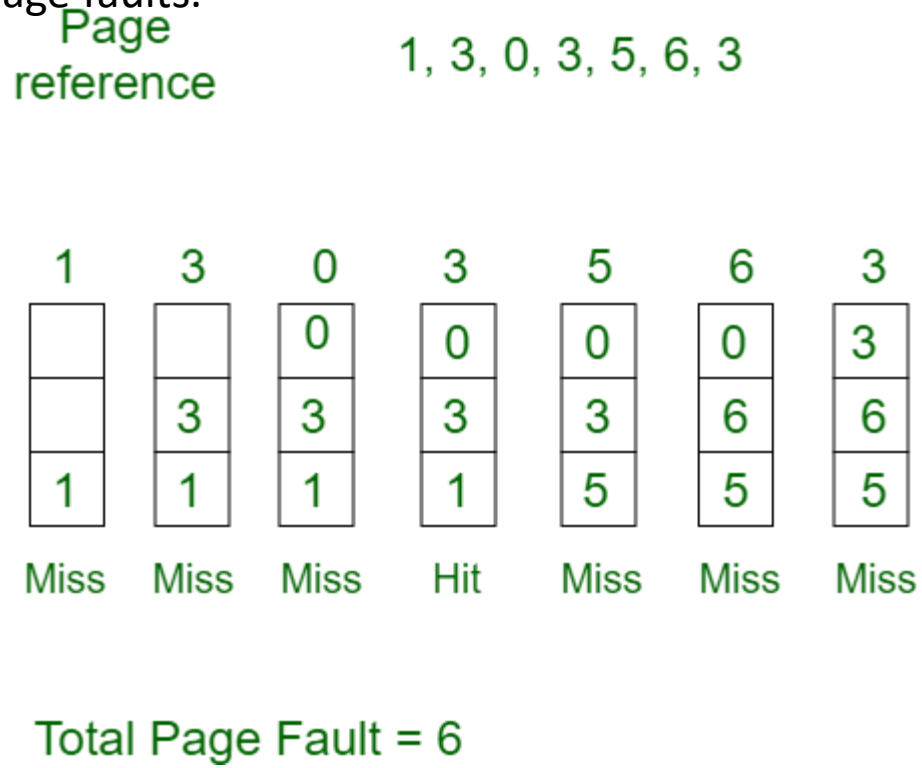
Introduction

- In an operating system that uses paging for memory management, a page replacement algorithm is needed to decide which page needs to be replaced when new page comes in.
- **Page Fault –** A page fault happens when a running program accesses a memory page that is mapped into the virtual address space, but not loaded in physical memory.
- Since actual physical memory is much smaller than virtual memory, page faults happen. In case of page fault, Operating System might have to replace one of the existing pages with the newly needed page. Different page replacement algorithms suggest different ways to decide which page to replace. The target for all algorithms is to reduce the number of page faults.

1. First In First Out (FIFO) –

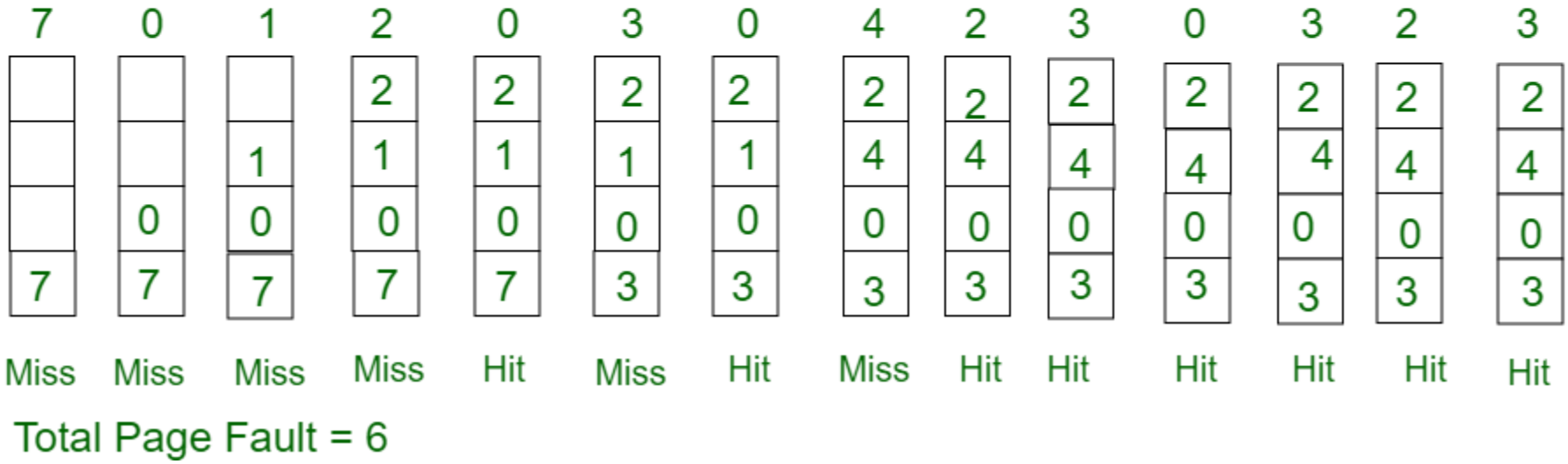
This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal

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



Optimal Page replacement –
In this algorithm, pages are replaced which would not be used for the longest duration of time in the future.
Example-2: Consider the page references 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, with 4 page frame. Find number of page fault.

Page reference 7,0,1,2,0,3,0,4,2,3,0,3,2,3 No. of Page frame - 4



Total Page Fault = 6

Least Recently Used –

In this algorithm page will be replaced which is least recently used.

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

Page reference 7,0,1,2,0,3,0,4,2,3,0,3,2,3 No. of Page frame - 4

Total Page Fault = 6

Here LRU has same number of page fault as optimal but it may differ according to question.

Disk Scheduling Algorithms

Introduction

- **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.

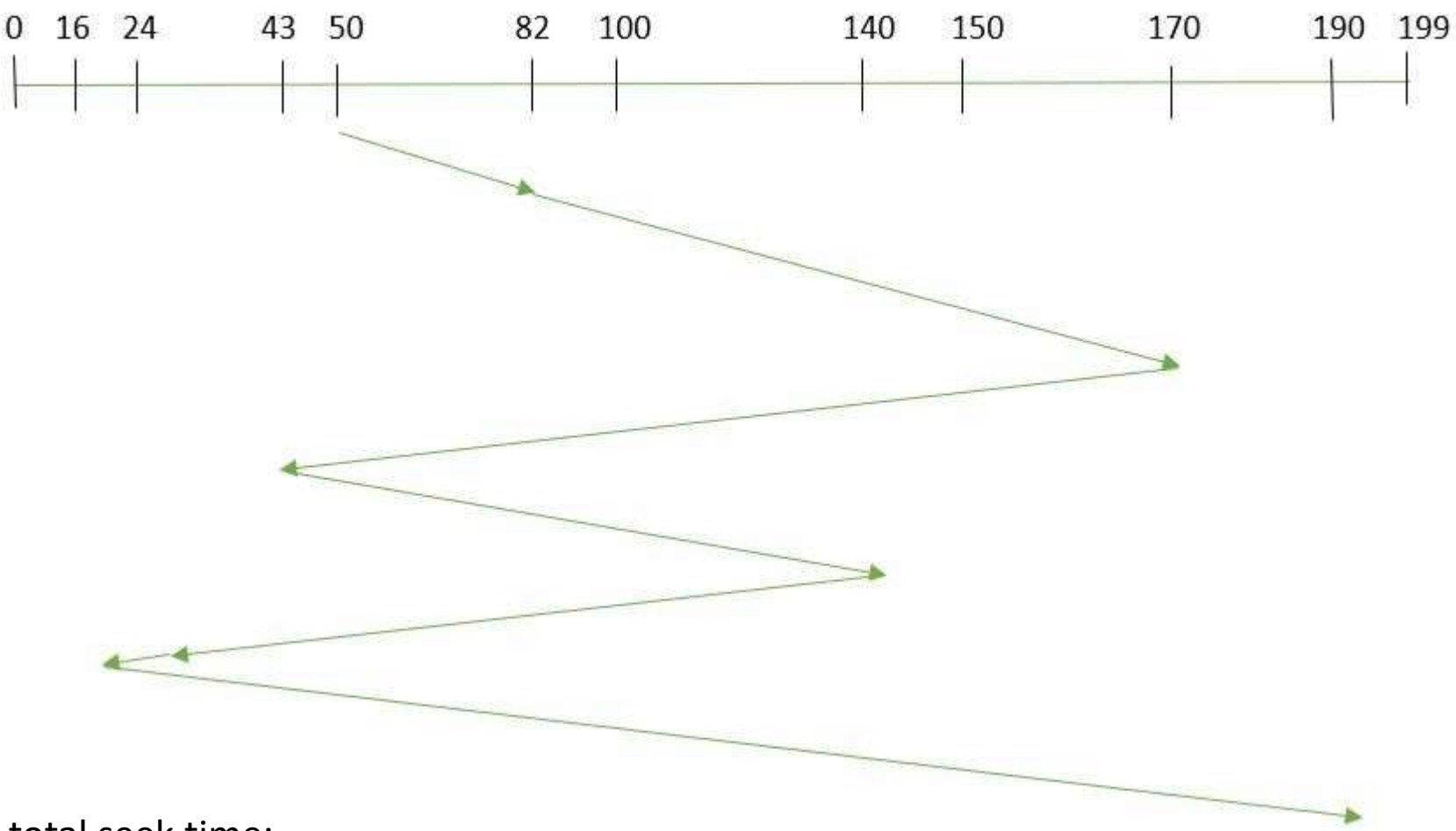
Disk scheduling is important because:

- 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.
- Hard drives are one of the slowest parts of the computer system and thus need to be accessed in an efficient manner.

- **Seek Time:**Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or write. So the disk scheduling algorithm that gives minimum average seek time is better.
- **Rotational Latency:** Rotational Latency is the time taken by the desired sector of disk to rotate into a position so that it can access the read/write heads. So the disk scheduling algorithm that gives minimum rotational latency is better.
- **Transfer Time:** Transfer time is the time to transfer the data. It depends on the rotating speed of the disk and number of bytes to be transferred.
- Disk Access Time = Seek Time + Rotational Latency + Transfer Time
- **Disk Response Time:** Response Time is the average of time spent by a request waiting to perform its I/O operation. *Average Response time* is the response time of the all requests. *Variance Response Time* is measure of how individual request are serviced with respect to average response time. So the disk scheduling algorithm that gives minimum variance response time is better.

FCFS

- **FCFS:** FCFS is the simplest of all the Disk Scheduling Algorithms. In FCFS, the requests are addressed in the order 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:
=(82-50)+(170-82)+(170-43)+(140-43)+(140-24)+(24-16)+(190-16)
=642

Advantages:

- Every request gets a fair chance
- No indefinite postponement

Disadvantages:

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

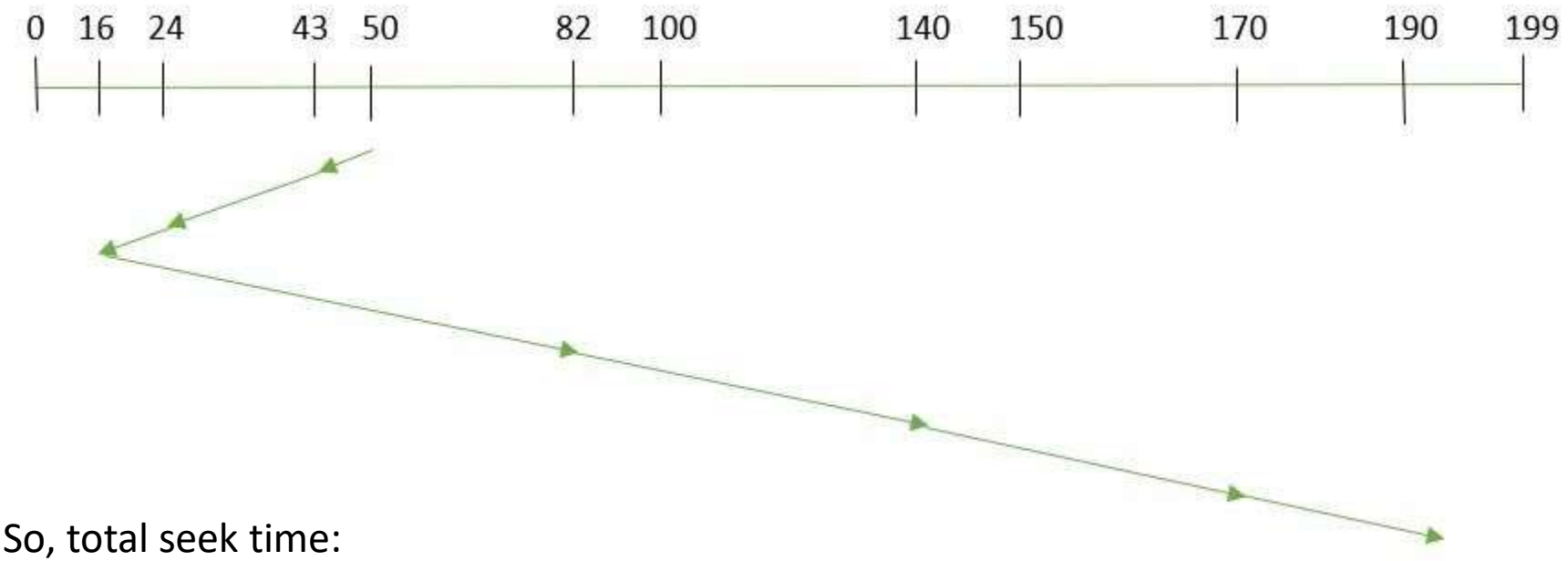
SSTF

- 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 of system.

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:

=(50-43)+(43-24)+(24-16)+(82-16)+(140-82)+(170-40)+(190-170)

=208

Advantages:

- Average Response Time decreases
- Throughput increases

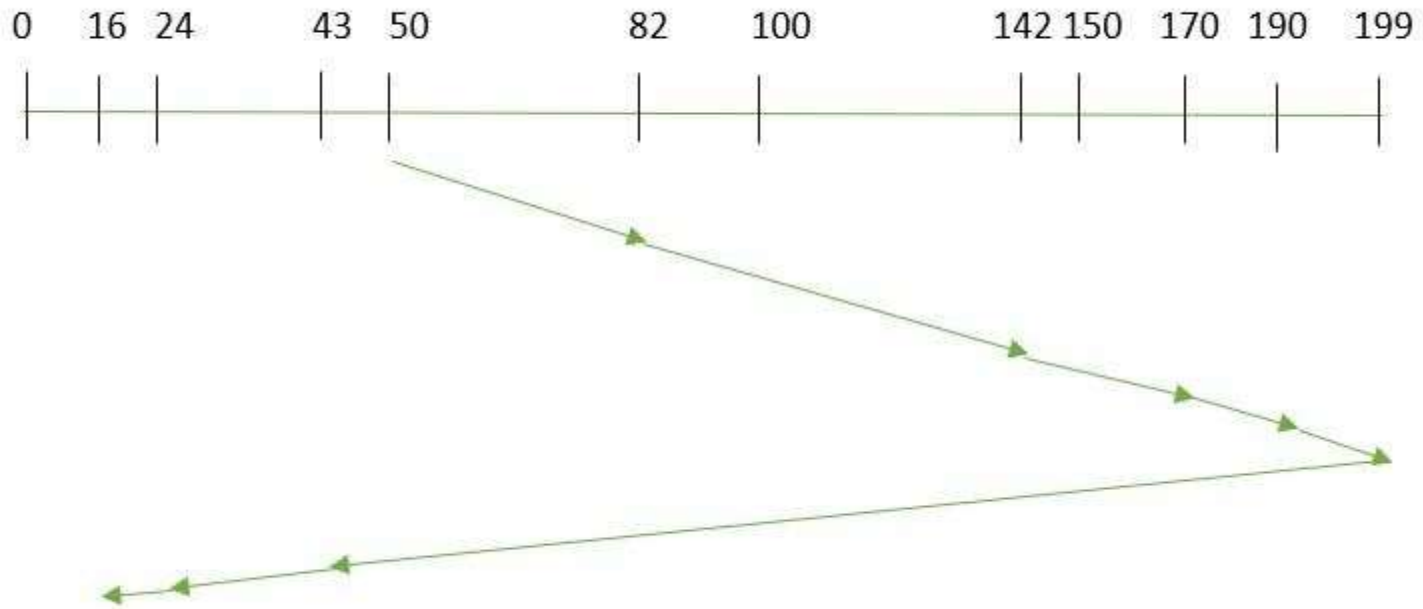
Disadvantages:

- Overhead to calculate seek time in advance
- Can cause Starvation for a request if it has higher seek time as compared to incoming requests
- High variance of response time as SSTF favours only some requests

SCAN

- 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.

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”.**



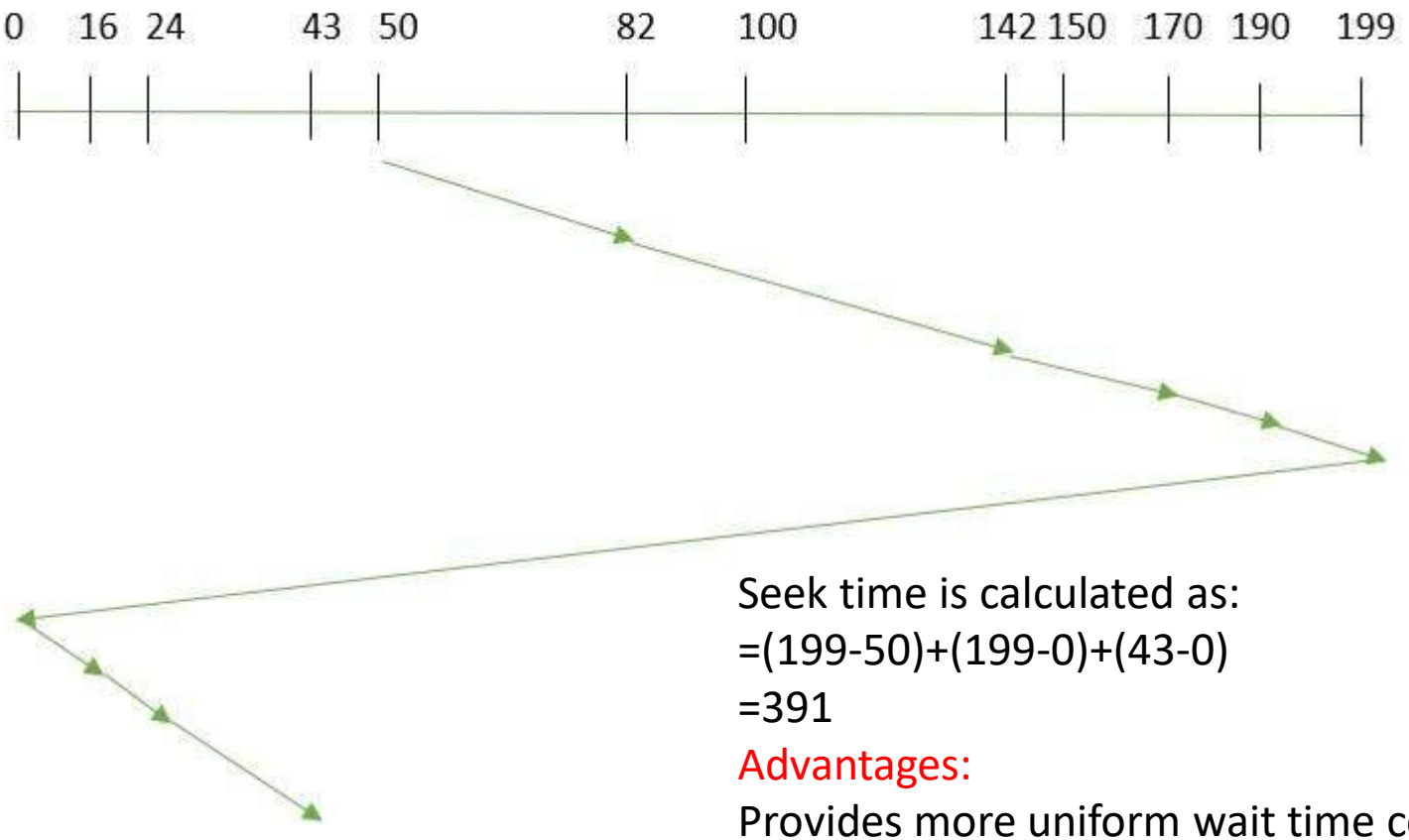
Seek time=(199-50)+(199-16)
=332

- Advantages:
- High throughput
- Low variance of response time
- Average response time
- Disadvantages:
- Long waiting time for requests for locations just visited by disk arm

CSCAN

- 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 or there may be zero or few requests pending at the scanned area.
- These situations are avoided in *CSCAN* algorithm in which the disk arm instead of reversing its direction goes to the other end of the disk and starts servicing the requests from there.
- So, 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).

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:
=(199-50)+(199-0)+(43-0)
=391

Advantages:

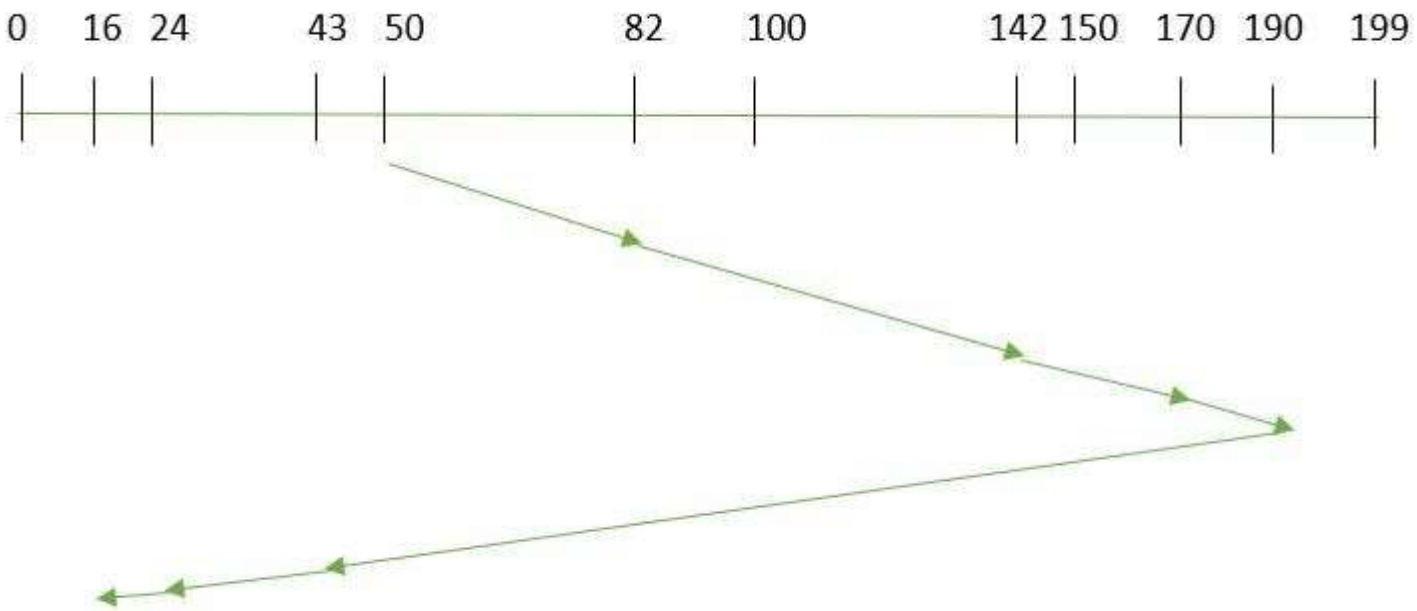
Provides more uniform wait time compared to SCAN

LOOK

- It is similar to the SCAN disk scheduling algorithm except for the difference that the disk arm in spite of going to the end of the disk goes only to the last request to be serviced in front of the head and then reverses its direction from there only.
- Thus it prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

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".**



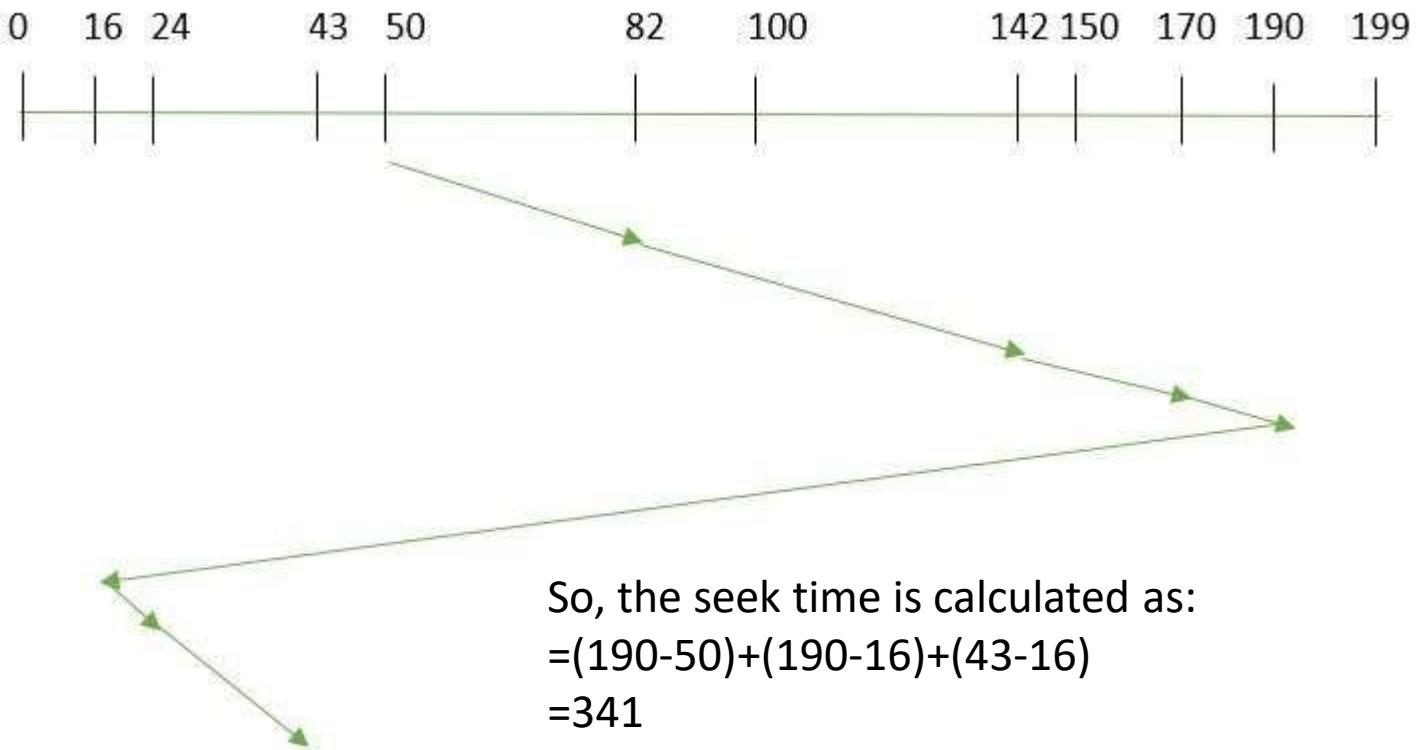
So, the seek time is calculated as:
=(190-50)+(190-16)
=314

CLOOK

- As LOOK is similar to SCAN algorithm, in similar way, CLOOK is similar to CSCAN disk scheduling algorithm.
- In CLOOK, the disk arm in spite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end's last request.
- Thus, it also prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

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"**



So, the seek time is calculated as:
=(190-50)+(190-16)+(43-16)
=341

Free Space Management

A file system is responsible to allocate the free blocks to the file therefore it has to keep track of all the free blocks present in the disk. There are mainly two approaches by using which, the free blocks in the disk are managed.

1. 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.

LAs the space allocation proceeds, the file system starts allocating blocks to the files and setting the respective bit to 0.

The given instance of disk blocks on the disk in *Figure 1* (where green blocks are allocated) can be represented by a bitmap of 16 bits as: **0000111000000110**.

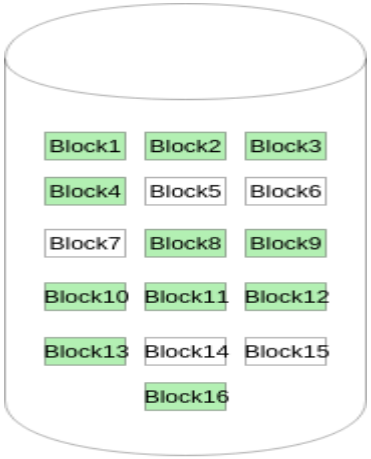


Figure - 1

Linked List –

In this approach, the free disk blocks are linked together i.e. a free block contains a pointer to the next free block. The block number of the very first disk block is stored at a separate location on disk and is also cached in memory.

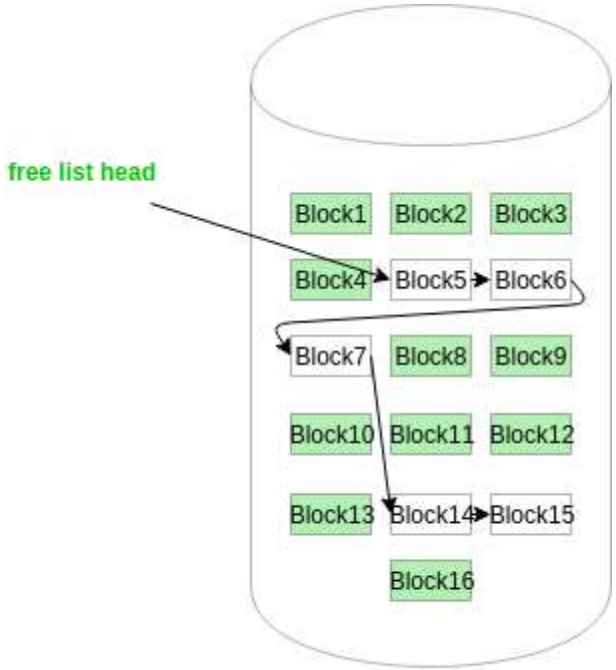


Figure - 2

In *Figure-2*, the free space list head points to Block 5 which points to Block 6, the next free block and so on. The last free block would contain a null pointer indicating the end of free list.

A drawback of this method is the I/O required for free space list traversal.

- **Grouping**
  This approach stores the address of the free blocks in the first free block. The first free block stores the address of some, 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.
  An **advantage** of this approach is that the addresses of a group of free disk blocks can be found easily.
- **Counting**
  This approach stores the address of the first free disk block and a number n of free contiguous disk blocks that follow the first block. Every entry in the list would contain:
  - Address of first free disk block
  - A number n
- For example, *in Figure-1*, the first entry of the free space list would be: ([Address of Block 5], 2), because 2 contiguous free blocks follow block 5.

Interprocess Communication

Introduction

- **Inter process communication (IPC)** is used for exchanging data between multiple threads in one or more processes or programs. The Processes may be running on single or multiple computers connected by a network.
- It is a set of programming interface which allow a programmer to coordinate activities among various program processes which can run concurrently in an operating system. This allows a specific program to handle many user requests at the same time.


Pipes
Message ques
FIFO
Sharred Memmory
Message Passing
Direct Commnication
Indirect Commication
ICP

Pipes

- Pipe is widely used for communication between two related processes. This is a half-duplex method, so the first process communicates with the second process. However, in order to achieve a full-duplex, another pipe is needed.

Message Passing:

- It is a mechanism for a process to communicate and synchronize. Using message passing, the process communicates with each other without resorting to shared variables.
- IPC mechanism provides two operations:
- Send (message)- message size fixed or variable
- Received (message)

Message Queues:

- A message queue is a linked list of messages stored within the kernel. It is identified by a message queue identifier. This method offers communication between single or multiple processes with full-duplex capacity.

Direct Communication:

- In this type of inter-process communication process, should name each other explicitly. In this method, a link is established between one pair of communicating processes, and between each pair, only one link exists.

Indirect Communication:

- Indirect communication establishes like only when processes share a common mailbox each pair of processes sharing several communication links. A link can communicate with many processes. The link may be bi-directional or unidirectional.

Shared Memory:

- Shared memory is a memory shared between two or more processes that are established using shared memory between all the processes. This type of memory requires to protected from each other by synchronizing access across all the processes.

FIFO:

- Communication between two unrelated processes. It is a full-duplex method, which means that the first process can communicate with the second process, and the opposite can also happen.

Why IPC?

- Here, are the reasons for using the interprocess communication protocol for information sharing:
- It helps to speedup modularity
- Computational
- Privilege separation
- Convenience
- Helps operating system to communicate with each other and synchronize their actions.

Process Synchronization

- On the basis of synchronization, processes are categorized as one of the following two types:
- **Independent Process** : Execution of one process does not affects the execution of other processes.
- **Cooperative Process** : Execution of one process affects the execution of other processes.
- Process synchronization problem arises in the case of Cooperative process also because resources are shared in Cooperative processes.

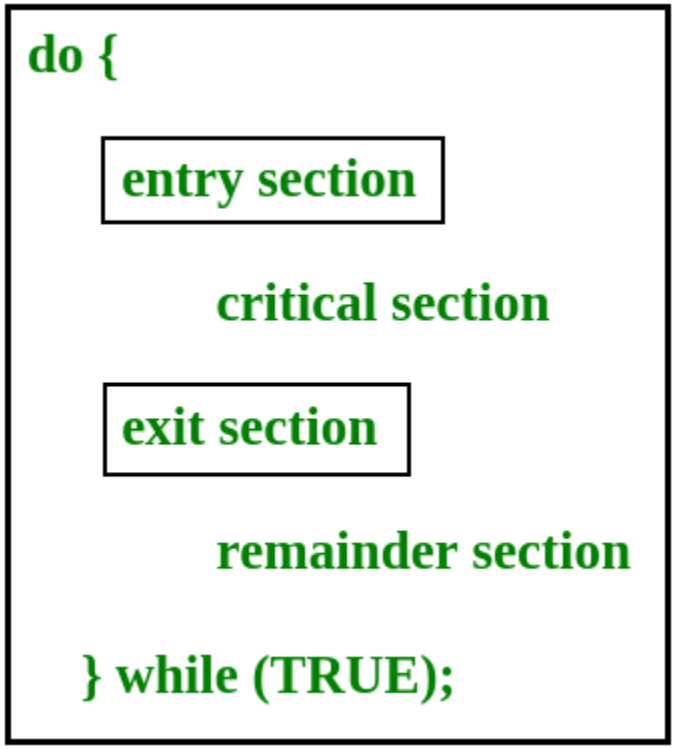
Race Condition
When more than one processes are executing the same code or accessing the same memory or any shared variable in that condition there is a possibility that the output or the value of the shared variable is wrong so for that all the processes doing the race to say that my output is correct this condition known as a race condition.

Contd..

- A race condition is a situation that may occur inside a critical section. This happens when the result of multiple thread execution in the critical section differs according to the order in which the threads execute.
- Race conditions in critical sections can be avoided if the critical section is treated as an atomic instruction. Also, proper thread synchronization using locks or atomic variables can prevent race conditions.

Critical Section Problem

Critical section is a code segment that can be accessed by only one process at a time. Critical section contains shared variables which need to be synchronized to maintain consistency of data variables.



The solution to the Critical Section Problem

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

1. Mutual Exclusion

- Out of a group of cooperating processes, only one process can be in its critical section at a given point of time.

2. Progress

- If no process is in its critical section, and if one or more threads want to execute their critical section then any one of these threads must be allowed to get into its critical section.

3. Bounded Waiting

- After a process makes a request for getting into its critical section, there is a limit for how many other processes can get into their critical section, before this process's request is granted. So after the limit is reached, the system must grant the process permission to get into its critical section.

Peterson's Solution

Peterson's Solution is a classical software based solution to the critical section problem.
In Peterson's solution, we have two shared variables:
boolean flag[i] :Initialized to FALSE, initially no one is interested in entering the critical section
int turn : The process whose turn is to enter the critical section.

```
do {

    flag[i] = TRUE ;
    turn = j ;
    while (flag[j]  &&  turn == j) ;

        critial section

    flag[i] = FALSE ;

        remainder section

} while (TRUE) ;
```

Disadvantages of Peterson's Solution
It involves Busy waiting
It is limited to 2 processes.

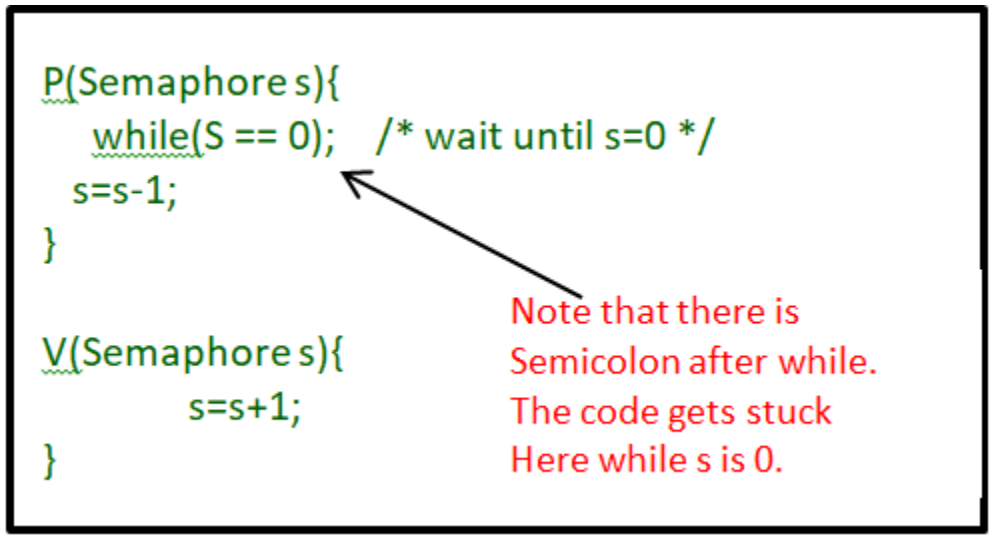
TestAndSet

- TestAndSet is a hardware solution to the synchronization problem. In TestAndSet, we have a shared lock variable which can take either of the two values, 0 or 1.
- 0 Unlock 1 Lock Before entering into the critical section, a process inquires about the lock. If it is locked, it keeps on waiting until it becomes free and if it is not locked, it takes the lock and executes the critical section
- In TestAndSet, Mutual exclusion and progress are preserved but bounded waiting cannot be preserved.

Semaphores

- Semaphore was proposed by Dijkstra in 1965 which is a very significant technique to manage concurrent processes by using a simple integer value, which is known as a semaphore. Semaphore is simply a variable that is non-negative and shared between threads. This variable is used to solve the critical section problem and to achieve process synchronization in the multiprocessing environment.
  Semaphores are of two types:

- **Binary Semaphore –**
  This is also known as mutex lock. It can have only two values – 0 and 1. Its value is initialized to 1. It is used to implement the solution of critical section problems with multiple processes.
- **Counting Semaphore –**
  Its value can range over an unrestricted domain. It is used to control access to a resource that has multiple instances.



Some point regarding P and V operation

P operation is also called wait, sleep, or down operation, and V operation is also called signal, wake-up, or up operation.
Both operations are atomic and semaphore(s) is always initialized to one. Here atomic means that variable on which read, modify and update happens at the same time/moment with no pre-emption i.e. in-between read, modify and update no other operation is performed that may change the variable.
A critical section is surrounded by both operations to implement process synchronization. See the below image. The critical section of Process P is in between P and V operation.

Problem-01:

A counting semaphore S is initialized to 10. Then, 6 P operations and 4 V operations are performed on S. What is the final value of S?

Solution-

We know-
P operation also called as wait operation decrements the value of semaphore variable by 1.
V operation also called as signal operation increments the value of semaphore variable by 1.

Thus,
Final value of semaphore variable S
= 10 – (6 x 1) + (4 x 1)
= 10 – 6 + 4=8

A counting semaphore S is initialized to 7. Then, 20 P operations and 15 V operations are performed on S. What is the final value of S?

Solution-

We know-
P operation also called as wait operation decrements the value of semaphore variable by 1.
V operation also called as signal operation increments the value of semaphore variable by 1.

Thus,
Final value of semaphore variable S
= 7 – (20 x 1) + (15 x 1)
= 7 – 20 + 15
= 2

- Semaphore can be used in other synchronization problems besides Mutual Exclusion.
- Below are some of the classical problem depicting flaws of process synchronaization in systems where cooperating processes are present.
- We will discuss the following three problems:
- Bounded Buffer (Producer-Consumer) Problem
- Dining Philosophers Problem
- The Readers Writers Problem