OPERATING SYSTEM
U5CS25ST

Unit-1

Introduction – Types of Operating Systems – Operating System Services – System Calls And System Programs.

Unit-II

Process Management – Process Concepts – Process Scheduling – Operation On Process – Inter-Process Communication – CPU Scheduling – Scheduling Algorithms – Dead Locks.

Unit-III

Memory Management – Single and Multiple Partition Allocation – Paging – Segmentation – Virtual Memory Management – Demand Paging and Page Replacement Algorithms.

Unit-IV

Information Management – File Concepts – Access Methods – Directory Structure – Allocation Methods – Free Space Management – Disk Scheduling.

Unit-V

UNIX: Unix System –Introduction-User interface-File System-Interprocess Communication-A Case Study.

Text Books

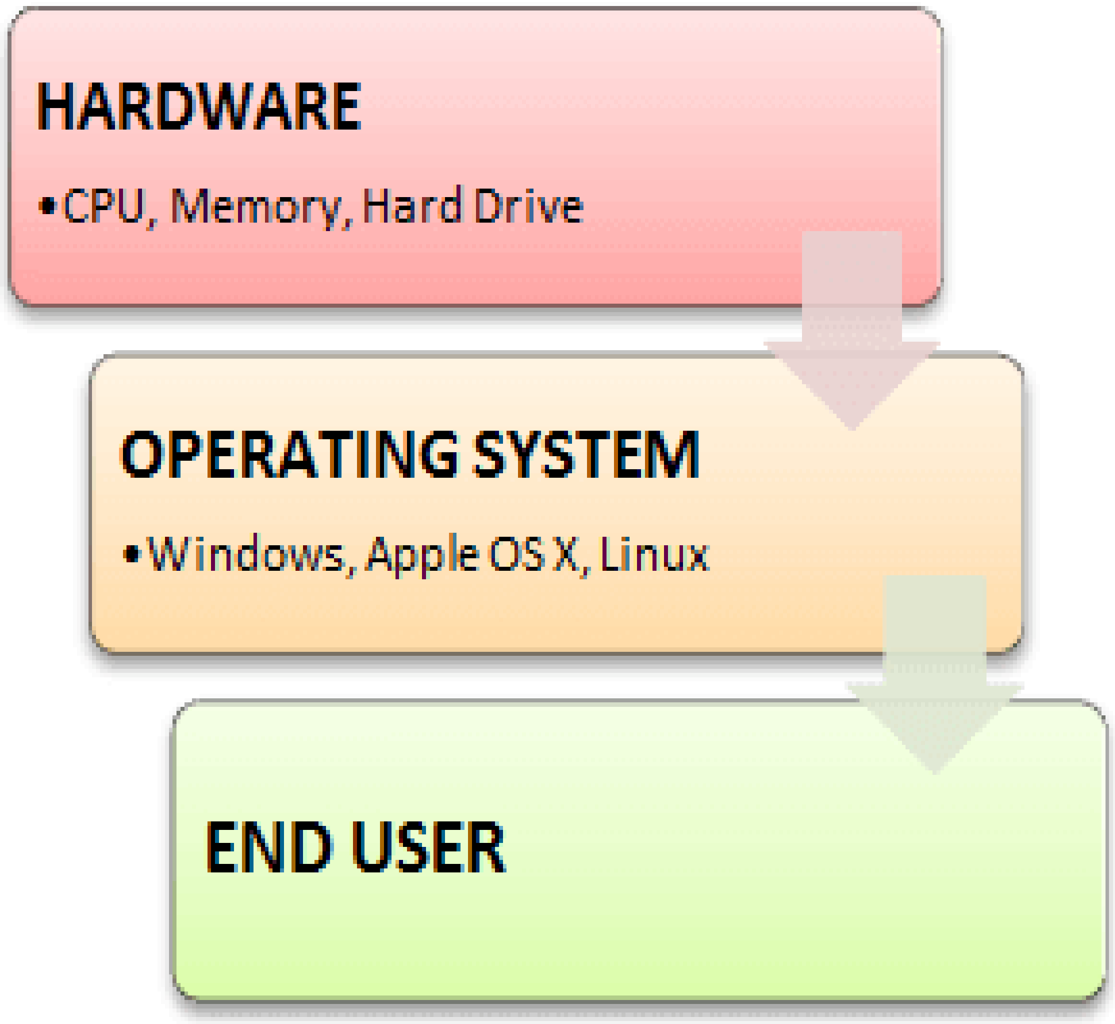
1. Abraham Silberschatz and P.B.Galvin – "operating System Concepts" – Addison Wesley Pub.

Operating system Unit no 1

What is an Operating System?

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.

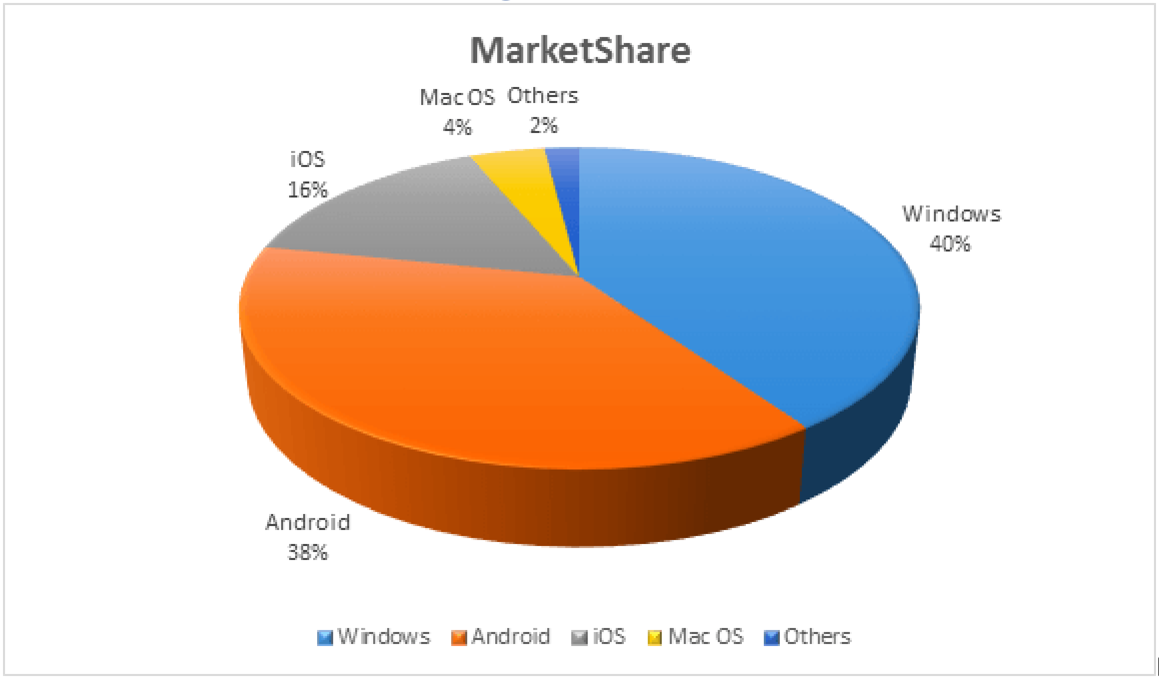
The OS helps you to communicate with the computer without knowing how to speak the computer's language. It is not possible for the user to use any computer or mobile device without having an operating system.



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.

Examples of Operating System with Market Share



Market Share of Operating Systems

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

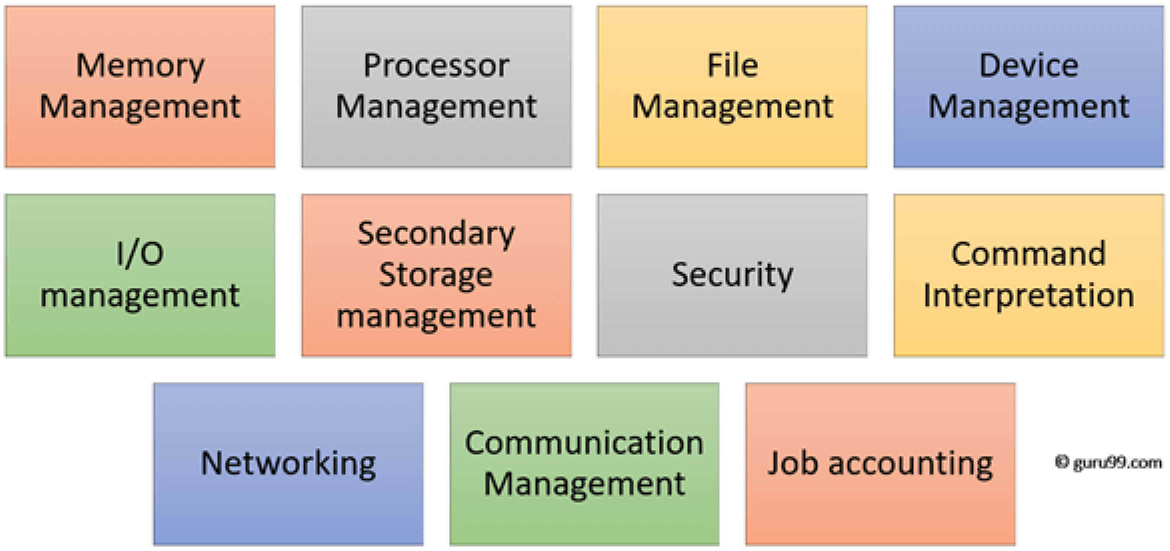
Mobile operating systems are those OS which is especially that are designed to power smartphones, tablets, and wearables devices.

Some most famous mobile operating systems are Android and iOS, but others include BlackBerry, Web, and watchOS.

Functions of Operating System

Some typical operating system functions may include managing memory, files, processes, I/O system & devices, security, etc.

Below are the main functions of Operating System:



Functions of Operating System

In an operating system software performs each of the function:

1. **Process management**:- Process management helps OS to create and delete processes. It also provides mechanisms for synchronization and communication among processes.

2. **Memory management:-** Memory management module performs the task of allocation and de-allocation of memory space to programs in need of this resources.

3. **File management**:- It manages all the file-related activities such as organization storage, retrieval, naming, sharing, and protection of files.

4. **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.

5. **I/O System Management:** One of the main objects of any OS is to hide the peculiarities of that hardware devices from the user.

6. **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.

7. **Security**:- Security module protects the data and information of a computer system against malware threat and authorized access.

8. **Command interpretation**: This module is interpreting commands given by the and acting system resources to process that commands.

9. **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.

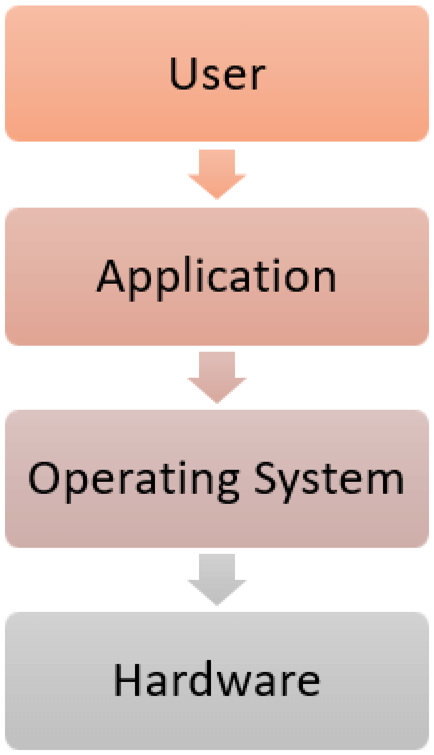
10. **Job accounting**: Keeping track of time & resource used by various job and users.

11. **Communication management**: Coordination and assignment of compilers, interpreters, and another software resource of the various users of the computer systems.

Features of Operating System (OS)

Here is a list important features of OS:

- 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



Advantage of using Operating System

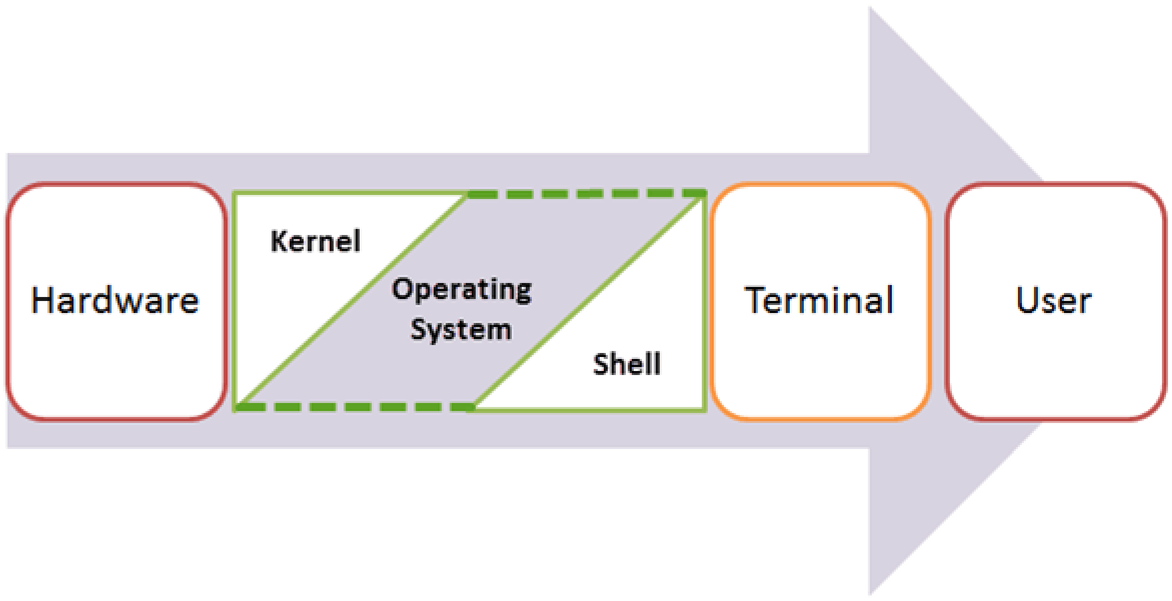
- Allows you to hide details of hardware by creating an abstraction
- Easy to use with a GUI
- Offers an environment in which a user may execute programs/applications
- The operating system must make sure that the computer system convenient to use
- Operating System acts as an intermediary among applications 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

Disadvantages of using Operating System

- If any issue occurs in OS, you may lose all the contents which have been stored in your system
- Operating system's software is quite expensive for small size organization which adds burden on them. Example Windows
- It is never entirely secure as a threat can occur at any time

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.



Introduction to Kernel

Features of Kennel

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

Types of Kernels

There are many types of kernels that exists, but among them, the two most popular kernels are:

1.**Monolithic**

A monolithic kernel is a single code or block of the program. It provides all the required services offered by the operating system. It is a simplistic design which creates a distinct communication layer between the hardware and software.

2. Microkernels

Microkernel manages all system resources. In this type of kernel, services are implemented in different address space. The user services are stored in user address space, and kernel services are stored under kernel address space. So, it helps to reduce the size of both the kernel and operating system.

Difference between Firmware and Operating System

Firmware	**Operating System**
Define Firmware: Firmware is one kind of programming that is embedded on a chip in the device which controls that specific device.	Define Operating System: OS provides functionality over and above that which is provided by the firmware.
Firmware is programs that been encoded by the manufacture of the IC or something and cannot be changed.	OS is a program that can be installed by the user and can be changed.

It is stored on non-volatile memory.	OS is stored on the hard drive.

Difference between 32-Bit vs. 64 Bit Operating System

Parameters	32. Bit	64. Bit
Architecture and Software	Allow 32 bit of data processing simultaneously	Allow 64 bit of data processing simultaneously
Compatibility	32-bit applications require 32-bit OS and CPUs.	64-bit applications require a 64-bit OS and CPU.
Systems Available	All versions of Windows 8, Windows 7, Windows Vista, and Windows XP, Linux, etc.	Windows XP Professional, Vista, 7, Mac OS X and Linux.
Memory Limits	32-bit systems are limited to 3.2 GB of RAM.	64-bit systems allow a maximum 17 Billion GB of RAM.

Summary

- What is OS (Operating System definition) and its Types: An operating system is a software which acts as an interface between the end user and computer hardware. Different categories of Operating System in computer and other devices are: Batch Operating System, Multitasking/Time Sharing OS, Multiprocessing OS, Real Time OS, Distributed OS, Network OS & Mobile OS
- Personal Computer Operating Systems were first developed in the late 1950s to manage tape storage
- Explain Operating System working: OS works as an intermediate between the user and computer. It helps the user to communicate with the computer without knowing how to speak the computer's language.

- 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
- Two most popular kernels are Monolithic and MicroKernels
- Process, Device, File, I/O, Secondary-Storage, Memory management are various functions of an Operating System

Operating System – Definition:

- An operating system is a program that controls the execution of application programs and acts as an interface between the user of a computer and the computer hardware.
- A more common definition is that the operating system is the one program running at all times on the computer (usually called the kernel), with all else being application programs.
- An operating system is concerned with the allocation of resources and services, such as memory, processors, devices, and information. The operating system correspondingly includes programs to manage these resources, such as a traffic controller, a scheduler, a memory management module, I/O programs, and a file system.

An Operating System supplies different kinds of services to both the users and to the programs as well. It also provides application programs (that run within an Operating system) an environment to execute it freely. It provides users the services run various programs in a convenient manner.
Here is a list of common services offered by an almost all operating systems:

- User Interface
- Program Execution
- File system manipulation
- Input / Output Operations
- Communication
- Resource Allocation
- Error Detection
- Accounting
- Security and protection

This chapter will give a brief description of what services an operating system usually provide to users and those programs that are and will be running within it.

System services

User Interface of Operating System

Usually Operating system comes in three forms or types. Depending on the interface their types have been further subdivided. These are:

- Command line interface
- Batch based interface
- Graphical User Interface

Let's get to know in brief about each of them.
The command line interface (CLI) usually deals with using text commands and a technique for entering those commands. The batch interface (BI): commands and directives are used to manage those commands that are entered into files and those files get executed. Another type is the graphical user interface (GUI): which is a window system with a pointing device (like mouse or trackball) to point to the I/O, choose from menus driven interface and to make choices viewing from a number of lists and a keyboard to entry the texts.

Program Execution in Operating System

The operating system must have the capability to load a program into memory and execute that program. Furthermore, the program must be able to end its execution, either normally or abnormally / forcefully.

File System Manipulation in Operating System

Programs need has to be read and then write them as files and directories. File handling portion of operating system also allows users to create and delete files by specific name along with extension, search for a given file and / or list file information. Some programs comprise of permissions management for allowing or denying access to files or directories based on file ownership.

I/O operations in Operating System

A program which is currently executing may require I/O, which may involve file or other I/O device. For efficiency and protection, users cannot directly govern the I/O devices. So, the OS provide a means to do I/O Input / Output operation which means read or write operation with any file.

Communication System of Operating System

Process needs to swap over information with other process. Processes executing on same computer system or on different computer systems can communicate using operating system support. Communication between two processes can be done using shared memory or via message passing.

Resource Allocation of Operating System

When multiple jobs running concurrently, resources must need to be allocated to each of them. Resources can be CPU cycles, main memory storage, file storage and I/O devices. CPU scheduling routines are used here to establish how best the CPU can be used.

Error Detection

Errors may occur within CPU, memory hardware, I/O devices and in the user program. For each type of error, the OS takes adequate action for ensuring correct and consistent computing.

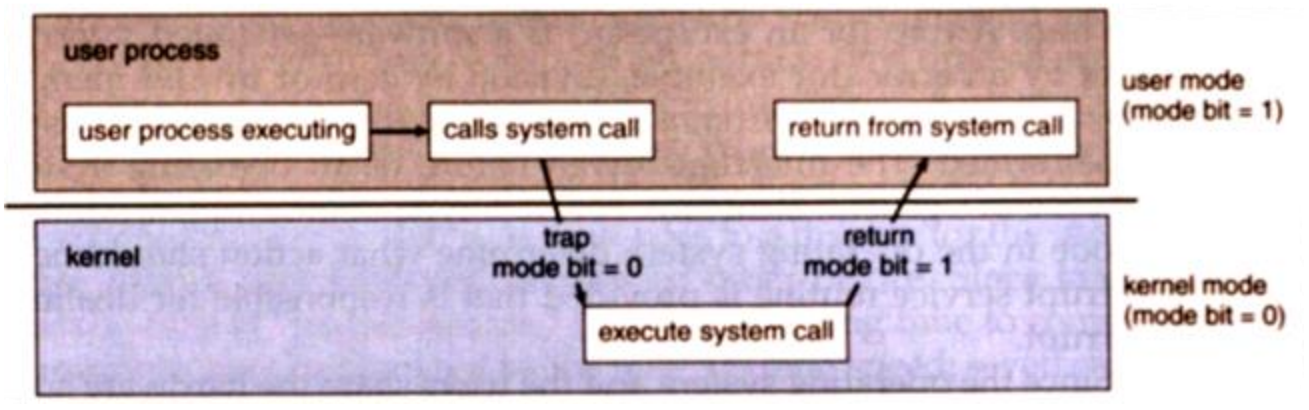
Accounting

This service of the operating system keeps track of which users are using how much and what kinds of computer resources have been used for accounting or simply to accumulate usage statistics.

Protection and Security

Protection includes in ensuring all access to system resources in a controlled manner. For making a system secure, the user needs to authenticate him or her to the system before using (usually via login ID and password)

System Calls



The system call provides an interface to the operating system services.

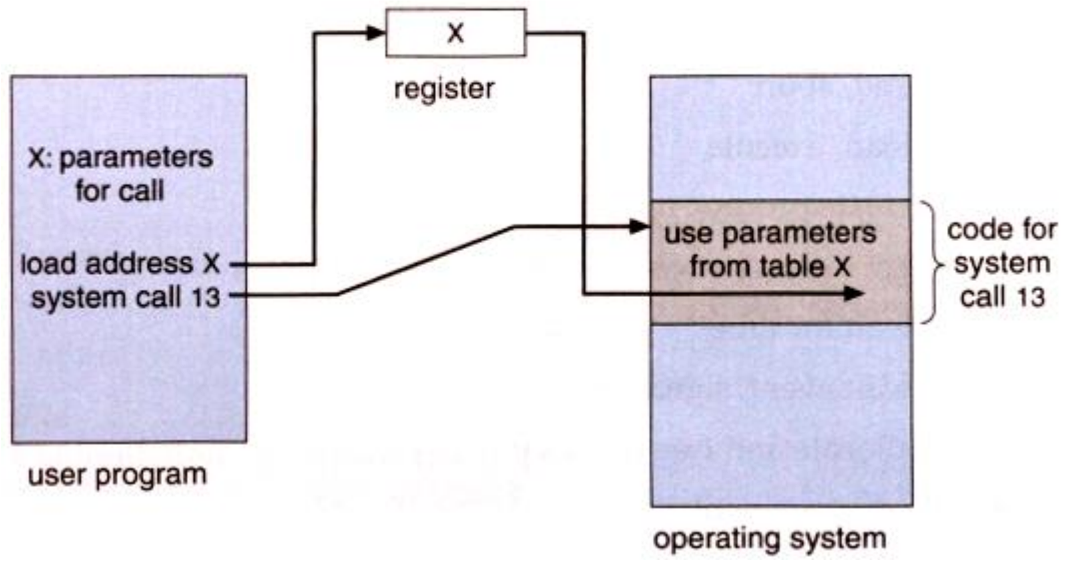
Application developers often do not have direct access to the system calls, but can access them through an application programming interface (API). The functions that are included in the API invoke the actual system calls. By using the API, certain benefits can be gained:

- Portability: as long a system supports an API, any program using that API can compile and run.
- Ease of Use: using the API can be significantly easier than using the actual system call.

System Call Parameters

Three general methods exist for passing parameters to the OS:

1. Parameters can be passed in registers.
2. When there are more parameters than registers, parameters can be stored in a block and the block address can be passed as a parameter to a register.
3. Parameters can also be pushed on or popped off the stack by the operating system.



Types of System Calls

There are 5 different categories of system calls:

process control, file manipulation, device manipulation, information maintenance, and communication.

Process Control

A running program needs to be able to stop execution either normally or abnormally. When execution is stopped abnormally, often a dump of memory is taken and can be examined with a debugger.

File Management

Some common system calls are *create*, *delete*, *read*, *write*, *reposition*, or *close*. Also, there is a need to determine the file attributes – *get* and *set* file attribute. Many times the OS provides an API to make these system calls.

Device Management

Process usually require several resources to execute, if these resources are available, they will be granted and control returned to the user process. These resources are also thought of as devices. Some are physical, such as a video card, and others are abstract, such as a file.

User programs *request* the device, and when finished they *release* the device. Similar to files, we can *read*, *write*, and *reposition* the device.

Information Management

Some system calls exist purely for transferring information between the user program and the operating system. An example of this is *time*, or *date*.

The OS also keeps information about all its processes and provides system calls to report this information.

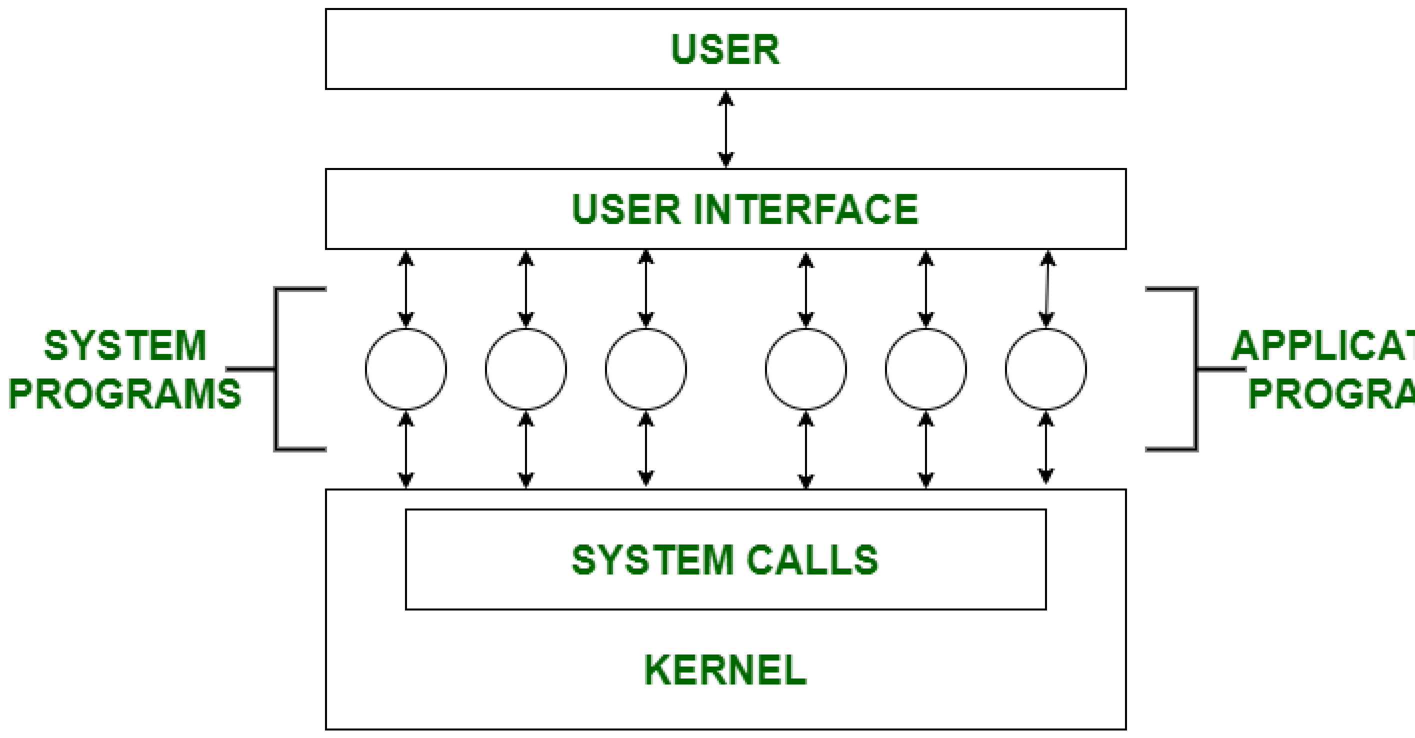
Communication

There are two models of interprocess communication, the message-passing model and the shared memory model.

- Message-passing uses a common mailbox to pass messages between processes.
- Shared memory use certain system calls to create and gain access to create and gain access to regions of memory owned by other processes. The two processes exchange information by reading and writing in the shared data

System Programs in Operating System

System Programming can be defined as the act of building Systems Software using System Programming Languages. According to Computer Hierarchy, one which comes at last is Hardware. Then it is Operating System, System Programs, and finally Application Programs. Program Development and Execution can be done conveniently in System Programs. Some of the System Programs are simply user interfaces, others are complex. It traditionally lies between the user interface and system calls.



So here, the user can only view up-to-the System Programs he can't see System Calls.
System Programs can be divided into these categories :

1. **File Management –**
   A file is a collection of specific information stored in the memory of a computer system. File management is defined as the process of manipulating files in the computer system, its management includes the process of creating, modifying and deleting files.
   - It helps to create new files in the computer system and placing them at specific locations.
   - It helps in easily and quickly locating these files in the computer system.

- It makes the process of sharing files among different users very easy and user-friendly.
- It helps to store files in separate folders known as directories.
- These directories help users to search files quickly or to manage files according to their types of uses.
- It helps users to modify the data of files or to modify the name of files in directories.

2. **Status Information –**
   Information like date, time amount of available memory, or disk space is asked by some users. Others providing detailed performance, logging, and debugging information which is more complex. All this information is formatted and displayed on output devices or printed. Terminal or other output devices or files or a window of GUI is used for showing the output of programs.

3. **File Modification –**
   For modifying the contents of files we use this. For Files stored on disks or other storage devices, we used different types of editors. For searching contents of files or perform transformations of files we use special commands.

4. **Programming-Language support –**
   For common programming languages, we use Compilers, Assemblers, Debuggers, and interpreters which are already provided to users. It provides all support to users. We can run any programming language. All languages of importance are already provided.

5. **Program Loading and Execution –**
   When the program is ready after Assembling and compilation, it must be loaded into memory for execution. A loader is part of an operating system that is responsible for loading programs and libraries. It is one of the essential stages for starting a program. Loaders, relocatable loaders, linkage editors, and Overlay loaders are provided by the system.

6. **Communications –**
   Virtual connections among processes, users, and computer systems are provided by programs. Users can send messages to another user on their screen, User can send e-mail, browsing on web pages, remote login, the transformation of files from one user to another.

Some examples of system program in O.S. are –

- Windows 10
- Mac OS X
- Ubuntu
- Linux
- Unix
- Android
- Anti-virus
- Disk formatting
- Computer language translators

Unit 2:

What is a Process?

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.

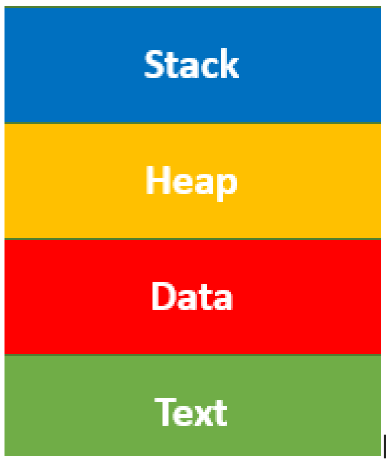
Process operations can be easily controlled with the help of PCB(Process Control Block). You can consider it as the brain of the process, which contains all the crucial information related to processing like process id, priority, state, CPU registers, etc.

What is Process Management?

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



Process architecture Image

Here, is an Architecture diagram of the Process

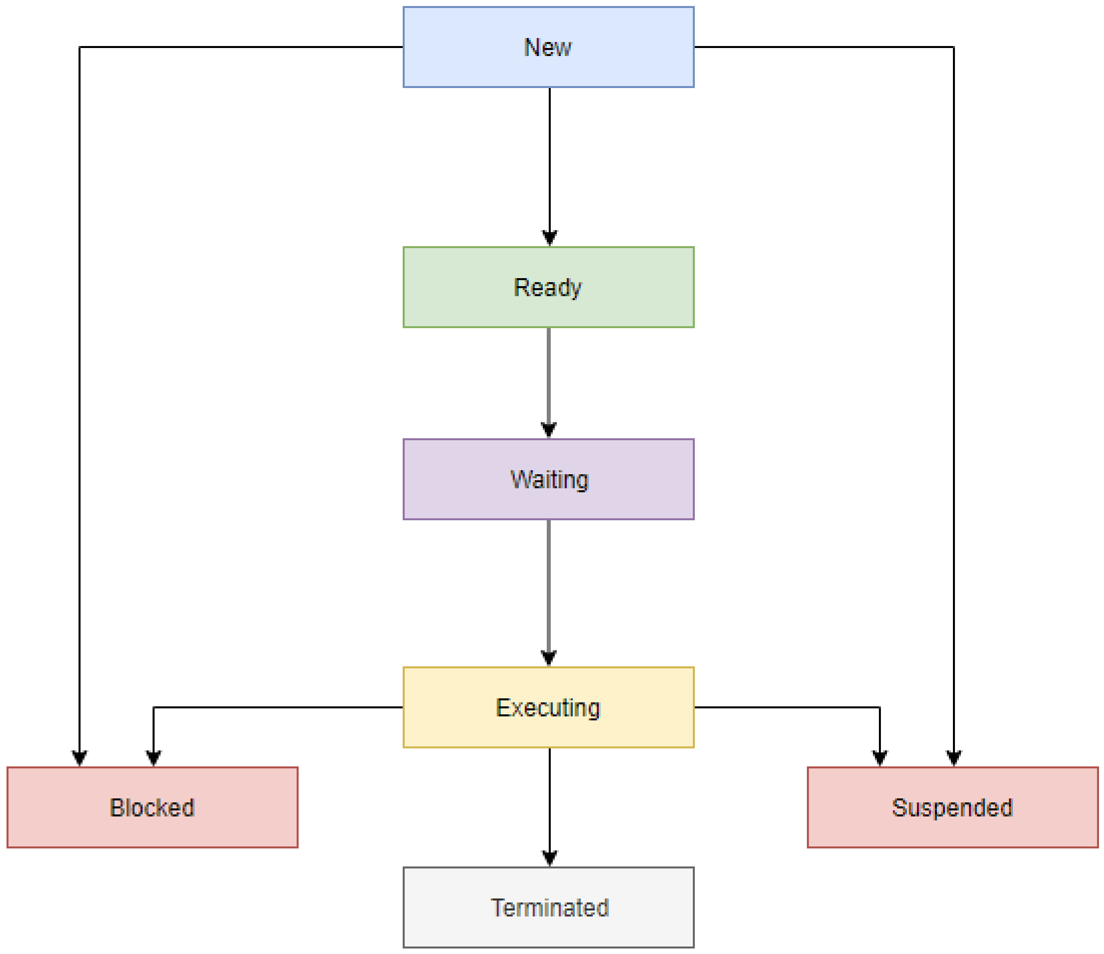
- Stack: The Stack stores temporary data like function parameters, returns addresses, and local variables.
- Heap Allocates memory, which may be processed during its run time.
- Data: It contains the variable.
- Text:
  Text Section includes the current activity, which is represented by the value of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

Process Control Blocks

PCB stands for Process Control Block. It is a data structure that is maintained by the Operating System for every process. The PCB should be identified by an integer Process ID (PID). It helps you to store all the information required to keep track of all the running processes.

It is also accountable for storing the contents of processor registers. These are saved when the process moves from the running state and then returns back to it. The information is quickly updated in the PCB by the OS as soon as the process makes the state transition.

Process States



Process States Diagram

A process state is a condition of the process at a specific instant of time. It also defines the current position of the process.

There are mainly seven stages of a process which are:

- **New: The new process is created when a specific program calls from secondary memory/ hard disk to primary memory/ RAM a**
- **Ready: In a ready state, the process should be loaded into the primary memory, which is ready for execution.**
- **Waiting: The process is waiting for the allocation of CPU time and other resources for execution.**

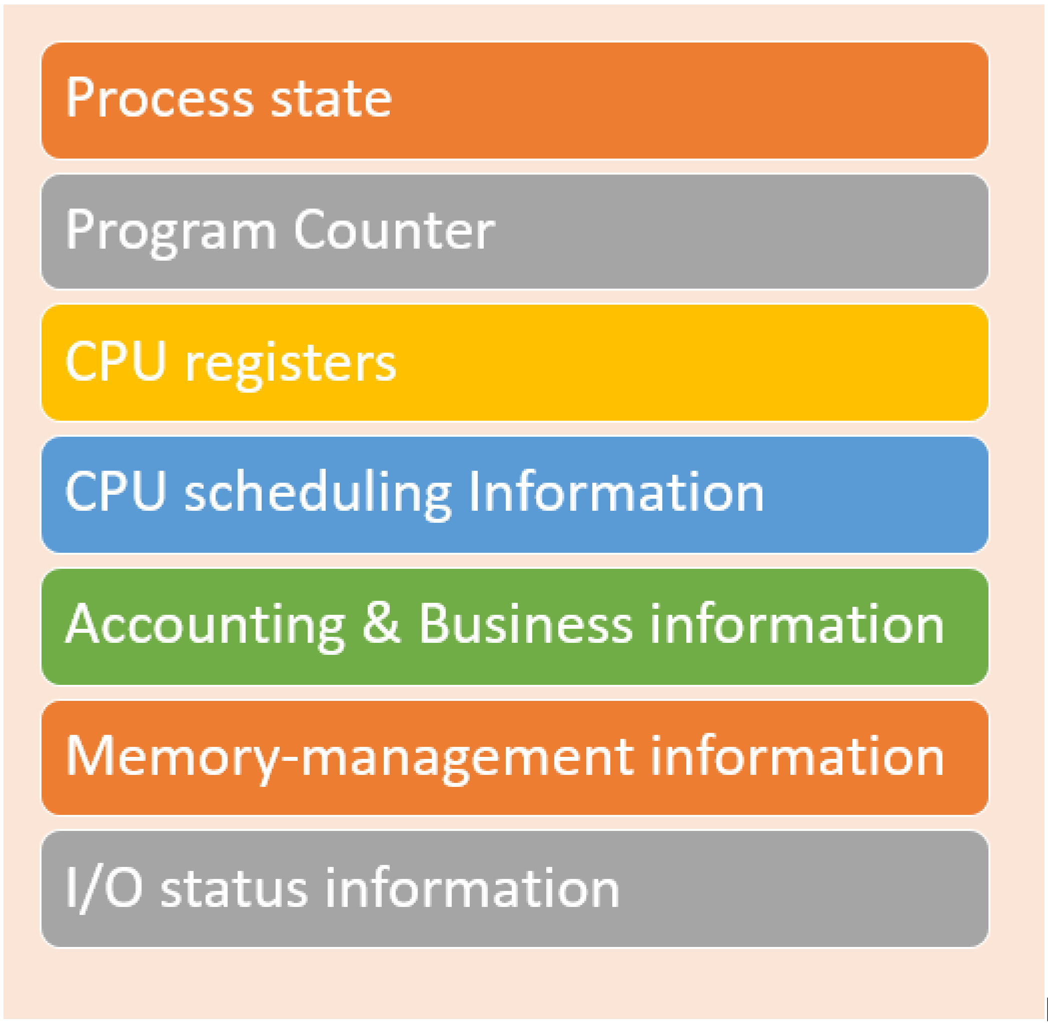
- **Executing: The process is an execution state.**
- **Blocked: It is a time interval when a process is waiting for an event like I/O operations to complete.**
- **Suspended: Suspended state defines the time when a process is ready for execution but has not been placed in the ready queue by OS.**
- **Terminated: Terminated state specifies the time when a process is terminated**

After completing every step, all the resources are used by a process, and memory becomes free.

Process Control Block(PCB)

Every process is represented in the operating system by a process control block, which is also called a task control block.

Here, are important components of PCB


Process state
Program Counter
CPU registers
CPU scheduling Information
Accounting & Business information
Memory-management information
I/O status information

Process Control Block(PCB)

- **Process state: A process can be new, ready, running, waiting, etc.**
- **Program counter: The program counter lets you know the address of the next instruction, which should be executed for that process.**
- **CPU registers: This component includes accumulators, index and general-purpose registers, and information of condition code.**
- **CPU scheduling information: This component includes a process priority, pointers for scheduling queues, and various other scheduling parameters.**
- **Accounting and business information: It includes the amount of CPU and time utilities like real time used, job or process numbers, etc.**
- **Memory-management information: This information includes the value of the base and limit registers, the page, or segment tables. This depends on the memory system, which is used by the operating system.**

- **I/O status information: This block includes a list of open files, the list of I/O devices that are allocated to the process, etc.**

Definition

The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.

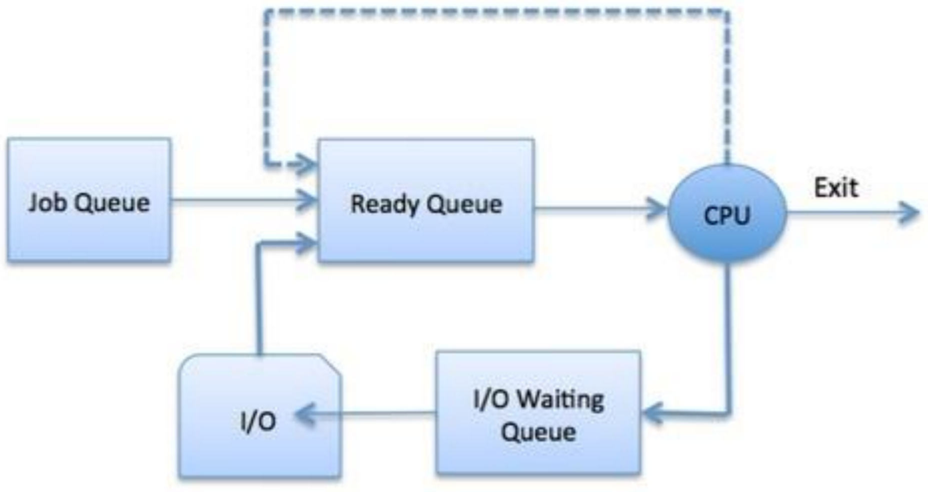
Process scheduling is an essential part of a Multiprogramming operating systems. Such operating systems allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU using time multiplexing.

Process Scheduling Queues

The OS maintains all PCBs in Process Scheduling Queues. The OS maintains a separate queue for each of the process states and PCBs of all processes in the same execution state are placed in the same queue. When the state of a process is changed, its PCB is unlinked from its current queue and moved to its new state queue.

The Operating System maintains the following important process scheduling queues −

- **Job queue −** This queue keeps all the processes in the system.
- **Ready queue −** This queue keeps a set of all processes residing in main memory, ready and waiting to execute. A new process is always put in this queue.
- **Device queues −** The processes which are blocked due to unavailability of an I/O device constitute this queue.



The OS can use different policies to manage each queue (FIFO, Round Robin, Priority, etc.). The OS scheduler determines how to move processes between the ready and run queues which can only have one entry per processor core on the system; in the above diagram, it has been merged with the CPU.

Two-State Process Model

Two-state process model refers to running and non-running states which are described below −

S.N.	State & Description
1	**Running** When a new process is created, it enters into the system as in the running state.
2	**Not Running** Processes that are not running are kept in queue, waiting for their turn to execute. Each entry in the queue is a pointer to a particular process. Queue is implemented by using linked list. Use of dispatcher is as follows. When a process is interrupted, that process is transferred in the waiting queue. If the process has completed or aborted, the process is discarded. In either case, the dispatcher then selects a process from the queue to execute.

Schedulers

Schedulers are special system software which handle process scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run. Schedulers are of three types −

- Long-Term Scheduler
- Short-Term Scheduler
- Medium-Term Scheduler

Long Term Scheduler

It is also called a **job scheduler**. A long-term scheduler determines which programs are admitted to the system for processing. It selects processes from the queue and loads them into memory for execution. Process loads into the memory for CPU scheduling.

The primary objective of the job scheduler is to provide a balanced mix of jobs, such as I/O bound and processor bound. It also controls the degree of multiprogramming. If the degree of multiprogramming is stable, then the average rate of process creation must be equal to the average departure rate of processes leaving the system.

On some systems, the long-term scheduler may not be available or minimal. Time-sharing operating systems have no long term scheduler. When a process changes the state from new to ready, then there is use of long-term scheduler.

Short Term Scheduler

It is also called as **CPU scheduler**. Its main objective is to increase system performance in accordance with the chosen set of criteria. It is the change of ready state to running state of the process. CPU scheduler selects a process among the processes that are ready to execute and allocates CPU to one of them.

Short-term schedulers, also known as dispatchers, make the decision of which process to execute next. Short-term schedulers are faster than long-term schedulers.

Medium Term Scheduler

Medium-term scheduling is a part of **swapping**. It removes the processes from the memory. It reduces the degree of multiprogramming. The medium-term scheduler is in-charge of handling the swapped out-processes.

A running process may become suspended if it makes an I/O request. A suspended processes cannot make any progress towards completion. In this condition, to remove the process from memory and make space for other processes, the suspended process is moved to the secondary storage. This process is called **swapping**, and the process is said to be swapped out or rolled out. Swapping may be necessary to improve the process mix.

Comparison among Scheduler

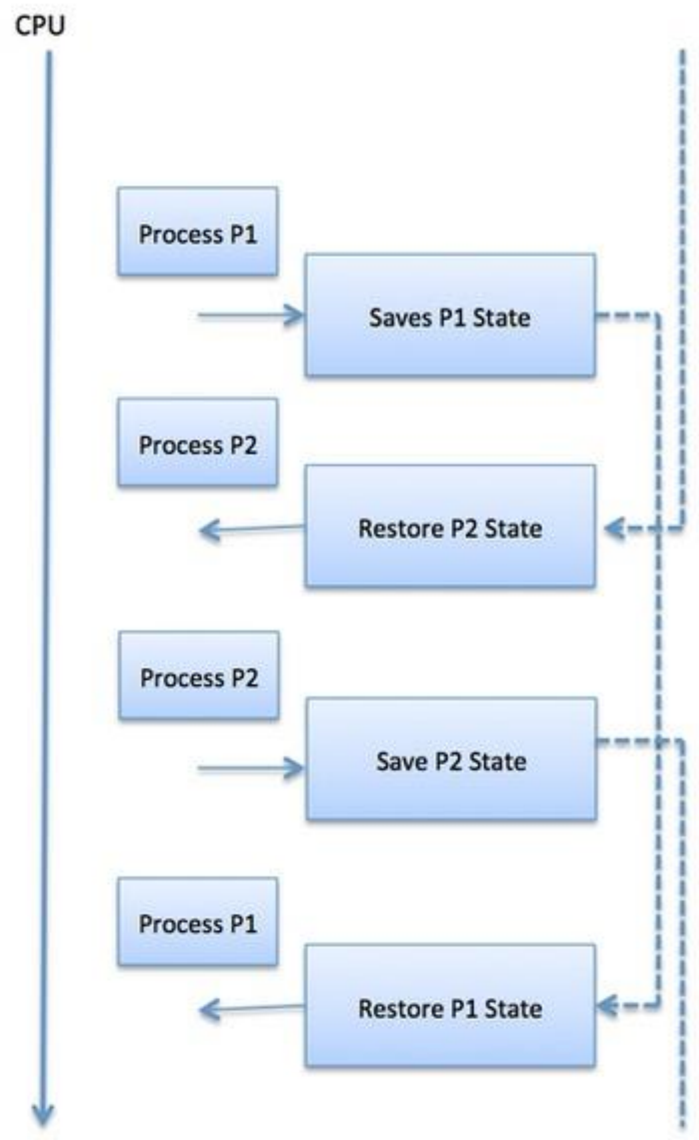
S.N.	Long-Term Scheduler	Short-Term Scheduler	Medium-Term Scheduler
1	It is a job scheduler	It is a CPU scheduler	It is a process swapping scheduler.
2	Speed is lesser than short term scheduler	Speed is fastest among other two	Speed is in between both short and long term scheduler.
3	It controls the degree of multiprogramming	It provides lesser control over degree of multiprogramming	It reduces the degree of multiprogramming.
4	It is almost absent or minimal in time sharing system	It is also minimal in time sharing system	It is a part of Time sharing systems.

5	It selects processes from pool and loads them into memory for execution	It selects those processes which are ready to execute	It can re-introduce the process into memory and execution can be continued.

Context Switch

A context switch is the mechanism to store and restore the state or context of a CPU in Process Control block so that a process execution can be resumed from the same point at a later time. Using this technique, a context switcher enables multiple processes to share a single CPU. Context switching is an essential part of a multitasking operating system features.

When the scheduler switches the CPU from executing one process to execute another, the state from the current running process is stored into the process control block. After this, the state for the process to run next is loaded from its own PCB and used to set the PC, registers, etc. At that point, the second process can start executing.



Context switches are computationally intensive since register and memory state must be saved and restored. To avoid the amount of context switching time, some hardware systems employ two or more sets of processor registers. When the process is switched, the following information is stored for later use.

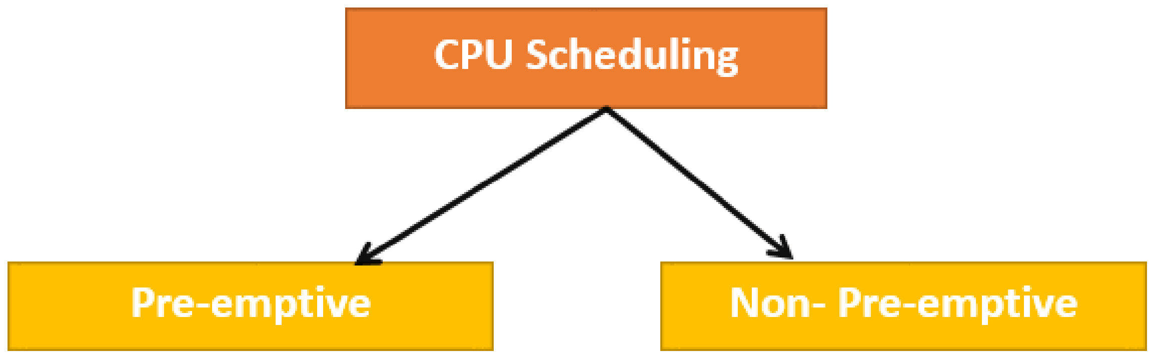
- Program Counter
- Scheduling information
- Base and limit register value
- Currently used register
- Changed State
- I/O State information
- Accounting information

What is 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. The selection process will be carried out by the CPU scheduler. It selects one of the processes in memory that are ready for execution.
In this CPU scheduling tutorial, you will learn:

Types of CPU Scheduling

Here are two kinds of Scheduling methods:



Preemptive Scheduling

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

Non-Preemptive Scheduling

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

When scheduling is Preemptive or Non-Preemptive?

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

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

Only conditions 1 and 4 apply, the scheduling is called non- preemptive.

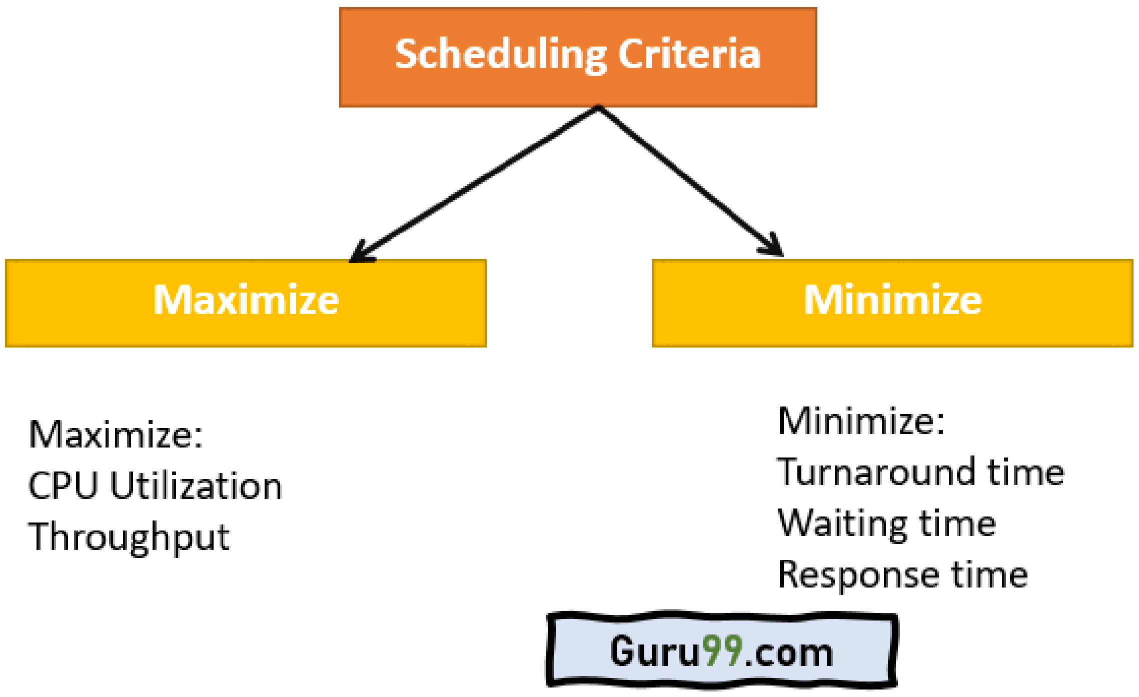
All other scheduling are preemptive.

Important CPU scheduling Terminologies

- **Burst Time/Execution Time: It is a time required by the process to complete execution. It is also called running time.**
- **Arrival Time: when a process enters in a ready state**
- **Finish Time: when process complete and exit from a system**
- **Multiprogramming: A number of programs which can be present in memory at the same time.**
- **Jobs: It is a type of program without any kind of user interaction.**
- **User: It is a kind of program having user interaction.**
- **Process: It is the reference that is used for both job and user.**
- **CPU/IO burst cycle: Characterizes process execution, which alternates between CPU and I/O activity. CPU times are usually shorter than the time of I/O.**

CPU Scheduling Criteria

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



Maximize:

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

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

Minimize:

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

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

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

Interval Timer

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

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

What is Dispatcher?

It is a module that provides control of the CPU to the process. The Dispatcher should be fast so that it can run on every context switch. Dispatch latency is

the amount of time needed by the CPU scheduler to stop one process and start another.

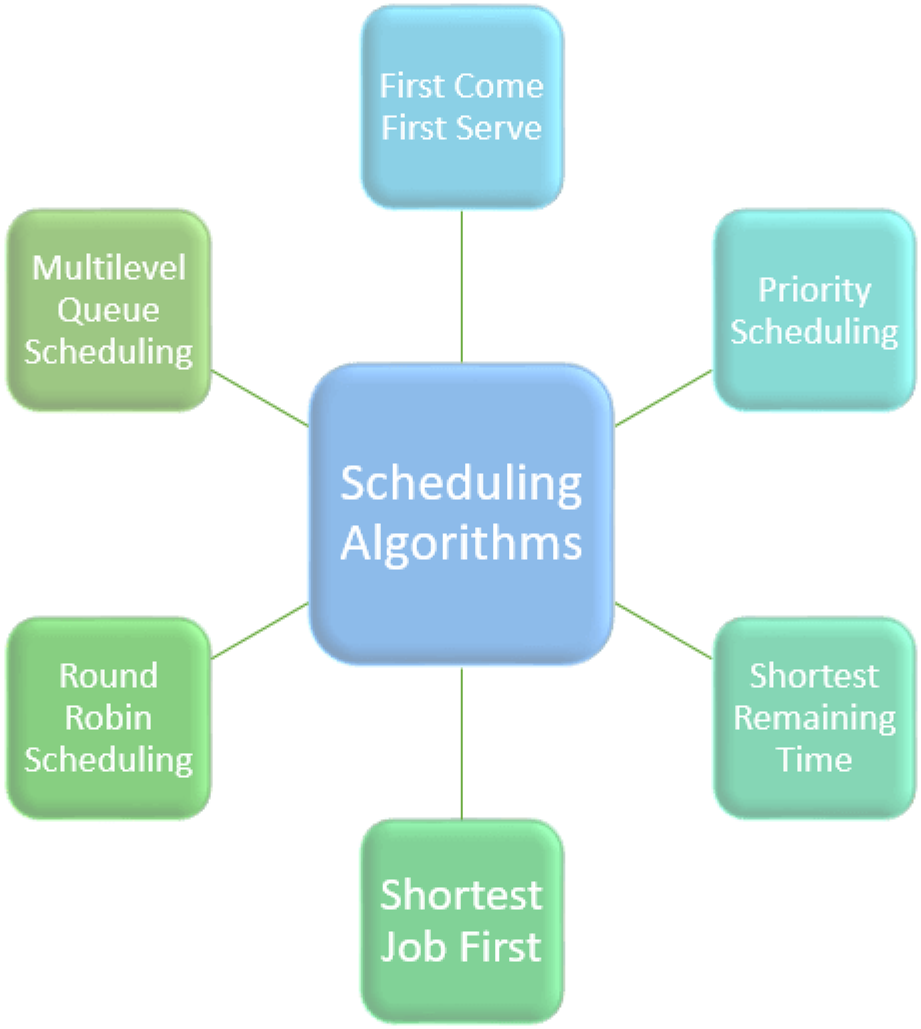
Functions performed by Dispatcher:

- **Context Switching**
- **Switching to user mode**
- **Moving to the correct location in the newly loaded program.**

Types of CPU scheduling Algorithm

There are mainly six types of process scheduling algorithms

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



Scheduling Algorithms

First Come First Serve

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

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

Characteristics of FCFS method:

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

Shortest Remaining Time

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

Characteristics of SRT scheduling method:

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

Priority Based Scheduling

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

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

Round-Robin Scheduling

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

Characteristics of Round-Robin Scheduling

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

Shortest Job First

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

Characteristics of SJF Scheduling

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

Multiple-Level Queues Scheduling

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

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

Characteristic of Multiple-Level Queues Scheduling:

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

The Purpose of a Scheduling algorithm

Here are the reasons for using a scheduling algorithm:

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

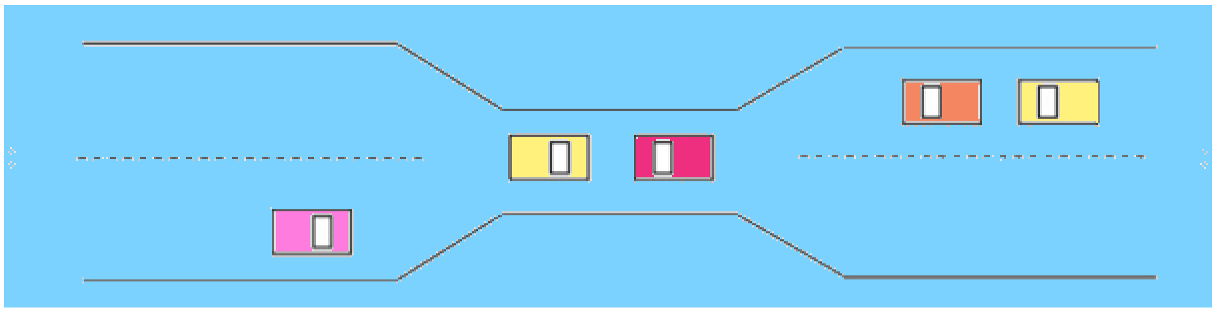
What is Deadlock?

Deadlock is a situation that occurs in OS when any process enters a waiting state because another waiting 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.
In this operating system tutorial, you will learn:

- What is Deadlock?
- Example of Deadlock

Example of Deadlock

- A real-world example would be traffic, which is going only in one direction.
- 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.
- So starvation is possible.


Example of deadlock

What is Circular wait?

One process is waiting for the resource, which is held by the second process, which is also waiting for the resource held by the third process etc. This will continue until the last process is waiting for a resource held by the first process. This creates a circular chain.

For example, Process A is allocated Resource B as it is requesting Resource A. In the same way, Process B is allocated Resource A, and it is requesting Resource B. This creates a circular wait loop.

Example of Circular wait

For example, a computer has three USB drives and three processes. Each of the three processes able to holds one of the USB drives. So, when each process requests another drive, the three processes will have the deadlock situation as each process will be waiting for the USB drive to release, which is currently in use. This will result in a circular chain.

Circular wait example

Deadlock Detection

A deadlock occurrence can be detected by the resource scheduler. A resource scheduler helps OS to keep track of all the resources which are allocated to different processes. So, when a deadlock is detected, it can be resolved using the below-given methods:

Deadlock Prevention:

It’s important to prevent a deadlock before it can occur. The system checks every transaction before it is executed to make sure it doesn’t lead the deadlock situations. Such that even a small change to occur dead that an operation which can lead to Deadlock in the future it also never allowed process to execute.

It is a set of methods for ensuring that at least one of the conditions cannot hold.

No preemptive action:

No Preemption – A resource can be released only voluntarily by the process holding it after that process has finished its task

- If a process which is holding some resources request another resource that can’t be immediately allocated to it, in that situation, all resources will be released.
- Preempted resources require the list of resources for a process that is waiting.
- The process will be restarted only if it can regain its old resource and a new one that it is requesting.
- If the process is requesting some other resource, when it is available, then it was given to the requesting process.
- If it is held by another process that is waiting for another resource, we release it and give it to the requesting process.

Mutual Exclusion:

Mutual Exclusion is a full form of Mutex. It is a special type of binary semaphore which used for controlling access to the shared resource. It includes a priority inheritance mechanism to avoid extended priority inversion problems. It allows current higher priority tasks to be kept in the blocked state for the shortest time possible.

Resources shared such as read-only files never lead to deadlocks, but resources, like printers and tape drives, needs exclusive access by a single process.

Hold and Wait:

In this condition, processes must be stopped from holding single or multiple resources while simultaneously waiting for one or more others.

Circular Wait:

It imposes a total ordering of all resource types. Circular wait also requires that every process request resources in increasing order of enumeration.

Deadlock Avoidance

It is better to avoid a deadlock instead of taking action after the Deadlock has occurred. It needs additional information, like how resources should be used. Deadlock avoidance is the simplest and most useful model that each process declares the maximum number of resources of each type that it may need.

Avoidance Algorithms

The deadlock-avoidance algorithm helps you to dynamically assess the resource-allocation state so that there can never be a circular-wait situation.

A single instance of a resource type.

- Use a resource-allocation graph
- Cycles are necessary which are sufficient for Deadlock

Multiples instances of a resource type.

- Cycles are necessary but never sufficient for Deadlock.
- Uses the banker’s algorithm

Difference Between Starvation and Deadlock

Here, are some important differences between Deadlock and starvation:

Deadlock	Starvation

The deadlock situation occurs when one of the processes got blocked.	Starvation is a situation where all the priority processes got blocked, and t priority processes execute.
Deadlock is an infinite process.	Starvation is a long waiting but not a process.
Every Deadlock always has starvation.	Every starvation does n't necessarily deadlock.
Deadlock happens then Mutual exclusion, hold and wait. Here, preemption and circular wait do not occur simultaneously.	It happens due to uncontrolled priori resource management.

Advantages of Deadlock

Here, are pros/benefits of using Deadlock method

- **This situation works well for processes which perform a single burst of activity**
- **No preemption needed for Deadlock.**
- **Convenient method when applied to resources whose state can be saved and restored easily**
- **Feasible to enforce via compile-time checks**
- **Needs no run-time computation since the problem is solved in system design**

Disadvantages of Deadlock method

Here, are cons/ drawback of using deadlock method

- **Delays process initiation**
- **Processes must know future resource need**
- **Pre-empts more often than necessary**
- **Dis-allows incremental resource requests**
- **Inherent preemption losses.**

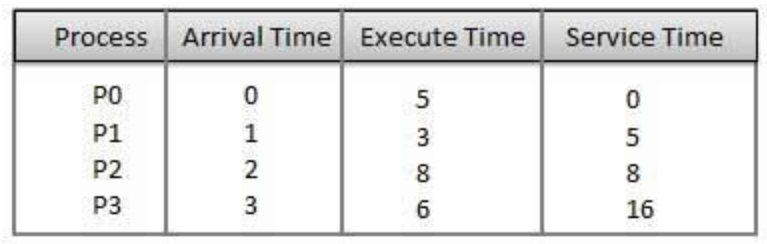
A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms. There are six popular process scheduling algorithms which we are going to discuss in this chapter −

- First-Come, First-Served (FCFS) Scheduling
- Shortest-Job-Next (SJN) Scheduling
- Priority Scheduling
- Shortest Remaining Time
- Round Robin(RR) Scheduling
- Multiple-Level Queues Scheduling

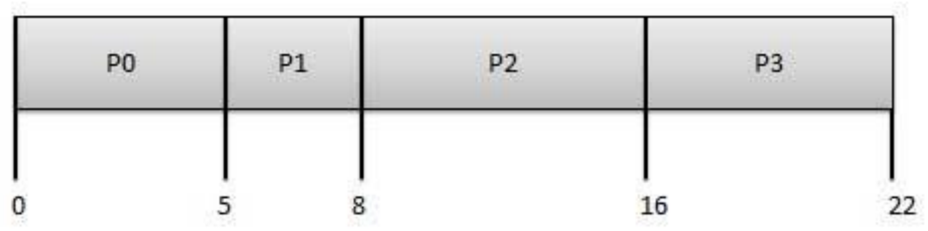
These algorithms are either **non-preemptive or preemptive**. Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time, whereas the preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

First Come First Serve (FCFS)

- Jobs are executed on first come, first serve basis.
- It is a non-preemptive, pre-emptive scheduling algorithm.
- Easy to understand and implement.
- Its implementation is based on FIFO queue.
- Poor in performance as average wait time is high.



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



Wait time of each process is as follows −

Process	Wait Time : Service Time - Arrival Time

P0	0 - 0 = 0
P1	5 - 1 = 4
P2	8 - 2 = 6
P3	16 - 3 = 13

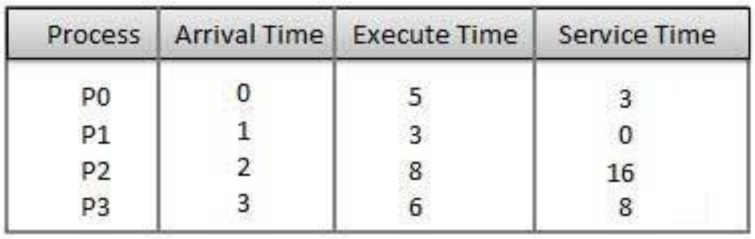
Average Wait Time: (0+4+6+13) / 4 = 5.75

Shortest Job Next (SJN)

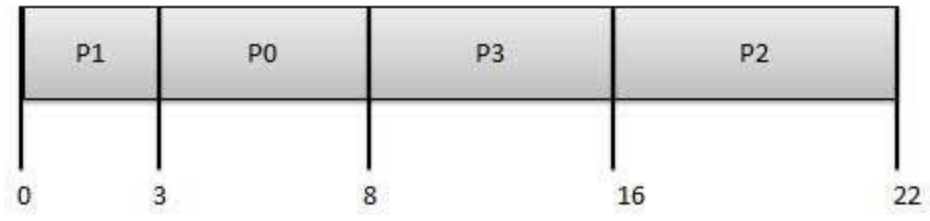
- This is also known as **shortest job first**, or SJF
- This is a non-preemptive, pre-emptive scheduling algorithm.
- Best approach to minimize waiting time.
- Easy to implement in Batch systems where required CPU time is known in advance.
- Impossible to implement in interactive systems where required CPU time is not known.
- The processer should know in advance how much time process will take.

Given: Table of processes, and their Arrival time, Execution time

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



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



Waiting time of each process is as follows −

Process	**Waiting Time**
P0	0 - 0 = 0
P1	5 - 1 = 4
P2	14 - 2 = 12
P3	8 - 3 = 5

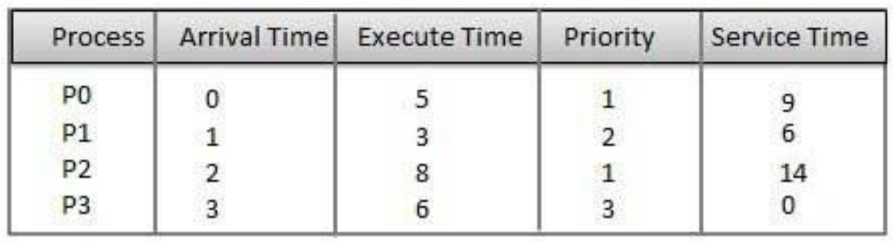
Average Wait Time: (0 + 4 + 12 + 5)/4 = 21 / 4 = 5.25

Priority Based Scheduling

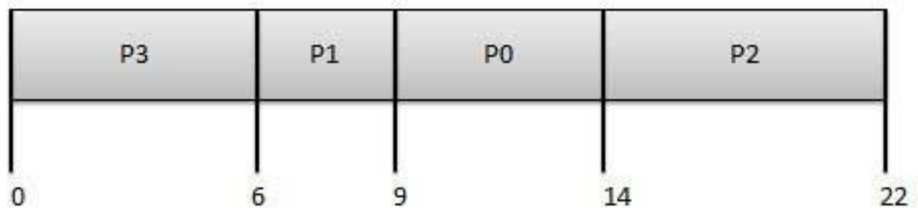
- Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.
- Each process is assigned a priority. Process with highest priority is to be executed first and so on.
- Processes with same priority are executed on first come first served basis.
- Priority can be decided based on memory requirements, time requirements or any other resource requirement.

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

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



Process	Arrival Time	Execute Time	Priority	Service Time
P0	0	5	1	9
P1	1	3	2	6
P2	2	8	1	14
P3	3	6	3	0



Waiting time of each process is as follows −

Process	Waiting Time
P0	0 - 0 = 0
P1	11 - 1 = 10

P2	14 - 2 = 12
P3	5 - 3 = 2

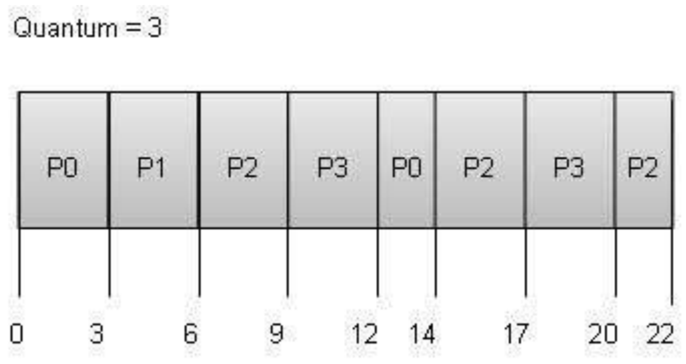
Average Wait Time: (0 + 10 + 12 + 2)/4 = 24 / 4 = 6

Shortest Remaining Time

- Shortest remaining time (SRT) is the preemptive version of the SJN algorithm.
- The processor is allocated to the job closest to completion but it can be preempted by a newer ready job with shorter time to completion.
- Impossible to implement in interactive systems where required CPU time is not known.
- It is often used in batch environments where short jobs need to give preference.

Round Robin Scheduling

- Round Robin is the preemptive process scheduling algorithm.
- Each process is provided a fix time to execute, it is called a **quantum**.
- Once a process is executed for a given time period, it is preempted and other process executes for a given time period.
- Context switching is used to save states of preempted processes.



Wait time of each process is as follows −

Process	Wait Time : Service Time - Arrival Time
P0	(0 - 0) + (12 - 3) = 9

P1	(3 - 1) = 2
P2	(6 - 2) + (14 - 9) + (20 - 17) = 12
P3	(9 - 3) + (17 - 12) = 11

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

Multiple-Level Queues Scheduling

Multiple-level queues are not an independent scheduling algorithm. They make use of other existing algorithms to group and schedule jobs with common characteristics.

- Multiple queues are maintained for processes with common characteristics.
- Each queue can have its own scheduling algorithms.
- Priorities are assigned to each queue.

For example, CPU-bound jobs can be scheduled in one queue and all I/O-bound jobs in another queue. The Process Scheduler then alternately selects jobs from each queue and assigns them to the CPU based on the algorithm assigned to the queue.

Unit 3:

What is Memory Management?

Memory Management is the process of controlling and coordinating computer memory, assigning portions known as blocks to various running programs to optimize the overall performance of the system.

It is the most important function of an operating system that manages primary memory. It helps processes to move back and forward between the main memory and execution disk. It helps OS to keep track of every memory location, irrespective of whether it is allocated to some process or it remains free.
In this, operating system tutorial you will learn:

Why Use Memory Management?

Here, are reasons for using memory management:

- **It allows you to check how much memory needs to be allocated to processes that decide which processor should get memory at what time.**
- **Tracks whenever inventory gets freed or unallocated. According to it will update the status.**
- **It allocates the space to application routines.**

- **It also make sure that these applications do not interfere with each other.**
- **Helps protect different processes from each other**
- **It places the programs in memory so that memory is utilized to its full extent.**

Memory Management Techniques

Here, are some most crucial memory management techniques:

Single Contiguous Allocation

It is the easiest memory management technique. In this method, all types of computer's memory except a small portion which is reserved for the OS is available for one application. For example, MS-DOS operating system allocates memory in this way. An embedded system also runs on a single application.

Partitioned Allocation

It divides primary memory into various memory partitions, which is mostly contiguous areas of memory. Every partition stores all the information for a specific task or job. This method consists of allotting a partition to a job when it starts & unallocate when it ends.

Paged Memory Management

This method divides the computer's main memory into fixed-size units known as page frames. This hardware memory management unit maps pages into frames which should be allocated on a page basis.

Segmented Memory Management

Segmented memory is the only memory management method that does not provide the user's program with a linear and contiguous address space.

Segments need hardware support in the form of a segment table. It contains the physical address of the section in memory, size, and other data like access protection bits and status.

What is Swapping?

Swapping is a method in which the process should be swapped temporarily from the main memory to the backing store. It will be later brought back into the memory for continue execution.

Backing store is a hard disk or some other secondary storage device that should be big enough inorder to accommodate copies of all memory images for all users. It is also capable of offering direct access to these memory images.

Benefits of Swapping

Here, are major benefits/pros of swapping:

- **It offers a higher degree of multiprogramming.**

- **Allows dynamic relocation. For example, if address binding at execution time is being used, then processes can be swap in different locations. Else in case of compile and load time bindings, processes should be moved to the same location.**
- **It helps to get better utilization of memory.**
- **Minimum wastage of CPU time on completion so it can easily be applied to a priority-based scheduling method to improve its performance.**

What is Memory allocation?

Memory allocation is a process by which computer programs are assigned memory or space.

Here, main memory is divided into two types of partitions

1. **Low Memory – Operating system resides in this type of memory.**
2. **High Memory– User processes are held in high memory.**

Partition Allocation

Memory is divided into different blocks or partitions. Each process is allocated according to the requirement. Partition allocation is an ideal method to avoid internal fragmentation.

Below are the various partition allocation schemes :

- **First Fit: In this type fit, the partition is allocated, which is the first sufficient block from the beginning of the main memory.**
- **Best Fit: It allocates the process to the partition that is the first smallest partition among the free partitions.**
- **Worst Fit: It allocates the process to the partition, which is the largest sufficient freely available partition in the main memory.**
- **Next Fit: It is mostly similar to the first Fit, but this Fit, searches for the first sufficient partition from the last allocation point.**

What is Paging?

Paging is a storage mechanism that allows OS to retrieve processes from the secondary storage into the main memory in the form of pages. In the Paging

method, the main memory is divided into small fixed-size blocks of physical memory, which is called frames. The size of a frame should be kept the same as that of a page to have maximum utilization of the main memory and to avoid external fragmentation. Paging is used for faster access to data, and it is a logical concept.

What is Fragmentation?

Processes are stored and removed from memory, which creates free memory space, which are too small to use by other processes.

After sometimes, that processes not able to allocate to memory blocks because its small size and memory blocks always remain unused is called fragmentation. This type of problem happens during a dynamic memory allocation system when free blocks are quite small, so it is not able to fulfill any request.

Two types of Fragmentation methods are:

1. External fragmentation
2. Internal fragmentation

- External fragmentation can be reduced by rearranging memory contents to place all free memory together in a single block.
- The internal fragmentation can be reduced by assigning the smallest partition, which is still good enough to carry the entire process.

What is Segmentation?

Segmentation method works almost similarly to paging. The only difference between the two is that segments are of variable-length, whereas, in the paging method, pages are always of fixed size.

A program segment includes the program’s main function, data structures, utility functions, etc. The OS maintains a segment map table for all the processes. It also includes a list of free memory blocks along with its size, segment numbers, and its memory locations in the main memory or virtual memory.

What is Dynamic Loading?

Dynamic loading is a routine of a program which is not loaded until the program calls it. All routines should be contained on disk in a relocatable load format. The main program will be loaded into memory and will be executed. Dynamic loading also provides better memory space utilization.

What is Dynamic Linking?

Linking is a method that helps OS to collect and merge various modules of code and data into a single executable file. The file can be loaded into memory and executed. OS can link system-level libraries into a program that combines the libraries at load time. In Dynamic linking method, libraries are linked at execution time, so program code size can remain small.

Difference Between Static and Dynamic Loading

Static Loading	Dynamic Loading
Static loading is used when you want to load your program statically. Then at the time of compilation, the entire program will be linked and compiled without need of any external module or program dependency.	In a Dynamically loaded program, references will be provided and the loading will be done at the time of execution.
At loading time, the entire program is loaded into memory and starts its execution.	Routines of the library are loaded into memory only when they are required in the program.

Difference Between Static and Dynamic Linking

Here, are main difference between Static vs. Dynamic Linking:

Static Linking	Dynamic Linking

Static linking is used to combine all other modules, which are required by a program into a single executable code. This helps OS prevent any runtime dependency.	**When dynamic linking is used, it does not need to link the actual module or library with the program. Instead of it use a reference to the dynamic module provided at the time of compilation and linking.**

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

This tutorial will teach you basic concepts related to Memory Management.

Process Address Space

The process address space is the set of logical addresses that a process references in its code. For example, when 32-bit addressing is in use, addresses can range from 0 to 0x7fffffff; that is, 2^31 possible numbers, for a total theoretical size of 2 gigabytes.

The operating system takes care of mapping 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 −

S.N.	Memory Addresses & Description
1	**Symbolic addresses** The addresses used in a source code. The variable names, constants, and instruction labels are the basic elements of the symbolic address space.

2	**Relative addresses** At the time of compilation, a compiler converts symbolic addresses into relative addresses.
3	**Physical addresses** The loader generates these addresses at the time when a program is loaded into main memory.

Virtual and physical addresses are the same in compile-time and load-time address-binding schemes. Virtual and physical addresses differ in execution-time address-binding scheme.

The set of all logical addresses generated by a program is referred to as a **logical address space**. The set of all physical addresses corresponding to these logical addresses is referred to as a **physical address space.**

The runtime mapping from virtual to physical address is done by the memory management unit (MMU) which is a hardware device. MMU uses following mechanism to convert virtual address to physical address.

- The value in the base register is added to every address generated by a user process, which is treated as offset at the time it is sent to memory. For example, if the base register value is 10000, then an attempt by the user to use address location 100 will be dynamically reallocated to location 10100.
- The user program deals with virtual addresses; it never sees the real physical addresses.

Static vs Dynamic Loading

The choice between Static or Dynamic Loading is to be made at the time of computer program being developed. If you have to load your program statically, then at the time of compilation, the complete programs will be compiled and linked without leaving any external program or module dependency. The linker combines the object program with other necessary object modules into an absolute program, which also includes logical addresses.

If you are writing a Dynamically loaded program, then your compiler will compile the program and for all the modules which you want to include dynamically, only references will be provided and rest of the work will be done at the time of execution.

At the time of loading, with **static loading**, the absolute program (and data) is loaded into memory in order for execution to start.

If you are using **dynamic loading**, dynamic routines of the library are stored on a disk in relocatable form and are loaded into memory only when they are needed by the program.

Static vs Dynamic Linking

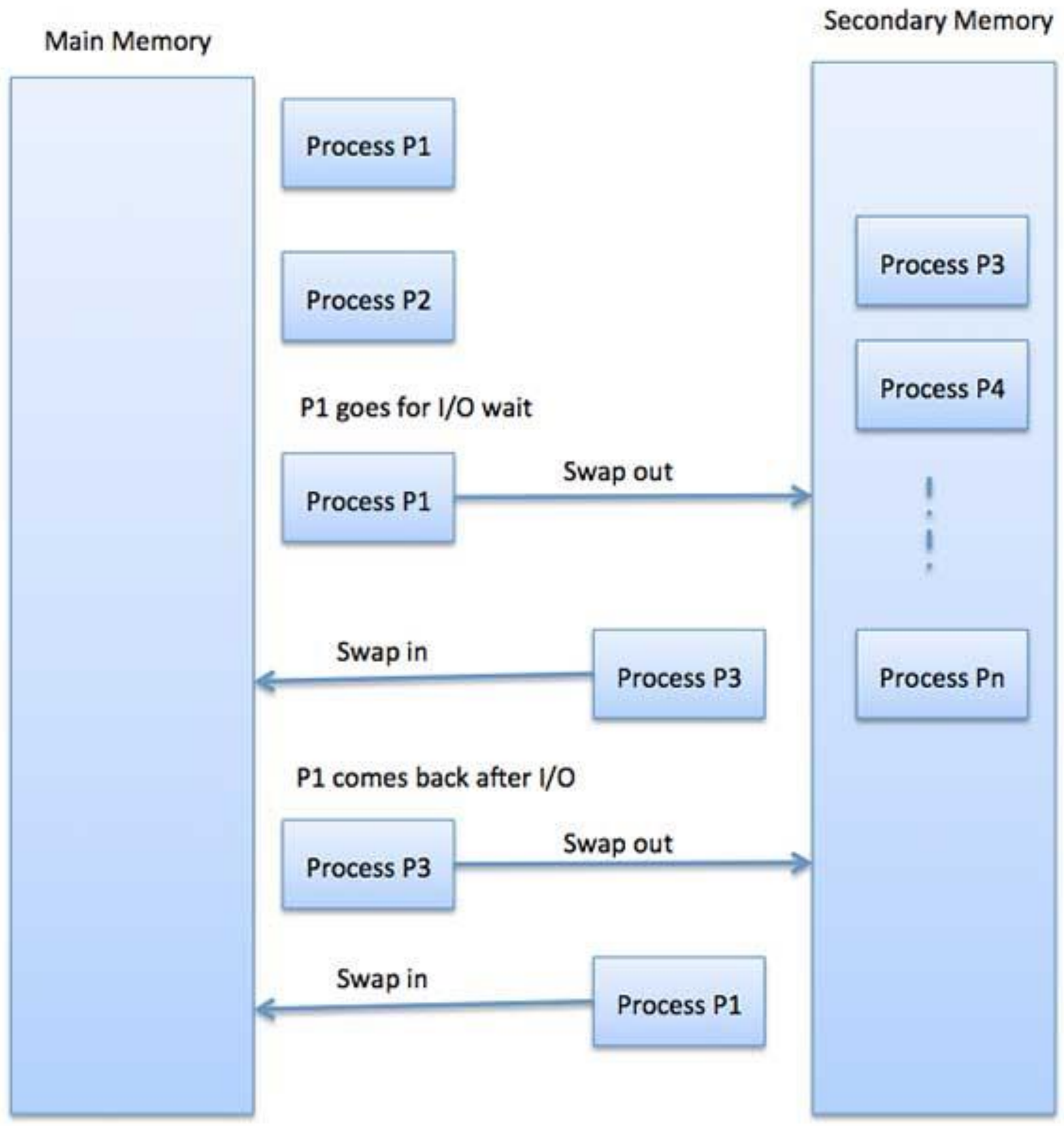
As explained above, when static linking is used, the linker combines all other modules needed by a program into a single executable program to avoid any runtime dependency.

When dynamic linking is used, it is not required to link the actual module or library with the program, rather a reference to the dynamic module is provided at the time of compilation and linking. Dynamic Link Libraries (DLL) in Windows and Shared Objects in Unix are good examples of dynamic libraries.

Swapping

Swapping is a mechanism in which a process can be swapped temporarily out of main memory (or move) to 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 is usually affected by swapping process but it helps in running multiple and big processes in parallel and that's the reason **Swapping is also known as a technique for memory compaction**.



The total time taken by swapping process includes the time it takes to move the entire process to a secondary disk and then to copy the process back to memory, as well as the time the process takes to regain main memory.

Let us assume that the user process is of size 2048KB and on a standard hard disk where swapping will take place has a data transfer rate around 1 MB per second. The actual transfer of the 1000K process to or from memory will take

```
2048KB / 1024KB per second
= 2 seconds
= 2000 milliseconds
```

Now considering in and out time, it will take complete 4000 milliseconds plus other overhead where the process competes to regain main memory.

Memory Allocation

Main memory usually has two partitions −

- **Low Memory −** Operating system resides in this memory.
- **High Memory −** User processes are held in high memory.

Operating system uses the following memory allocation mechanism.

S.N.	Memory Allocation & Description
1	**Single-partition allocation** In this type of allocation, relocation-register scheme is used to protect user processes from each other, and from changing operating-system code and data. Relocation register contains value of smallest physical address whereas limit register contains range of logical addresses. Each logical address must be less than the limit register.
2	**Multiple-partition allocation** In this type of allocation, main memory is divided into a number of fixed-sized partitions where each partition should contain only one process. When a partition is free, a process is selected from the input queue and is loaded into the free partition. When the process terminates, the partition becomes available for another process.

Fragmentation

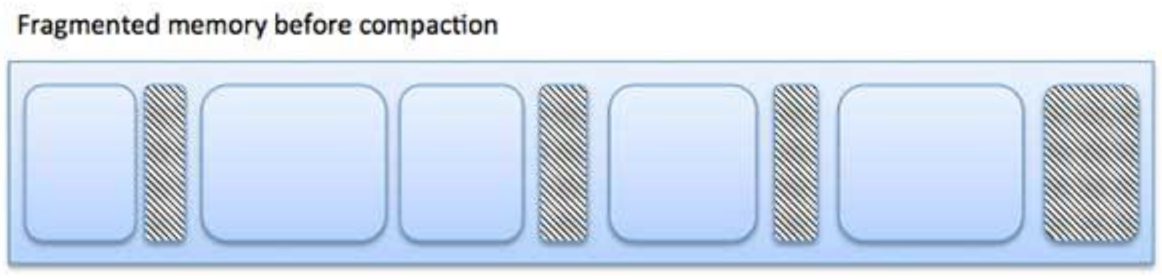
As processes are loaded and removed from memory, the free memory space is broken into little pieces. It happens after sometimes that processes cannot be allocated to memory blocks considering their small size and memory blocks remains unused. This problem is known as Fragmentation.

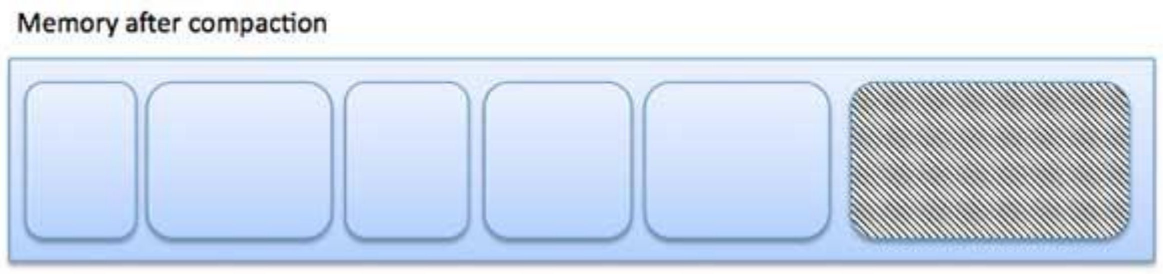
Fragmentation is of two types −

S.N.	Fragmentation & Description
1	**External fragmentation** Total memory space is enough to satisfy a request or to reside a process in it, but it is not contiguous, so it cannot be used.
2	**Internal fragmentation**

	Memory block assigned to process is bigger. Some portion of memory is left unused, as it cannot be used by another process.

The following diagram shows how fragmentation can cause waste of memory and a compaction technique can be used to create more free memory out of fragmented memory −





External fragmentation can be reduced by compaction or shuffle memory contents to place all free memory together in one large block. To make compaction feasible, relocation should be dynamic.

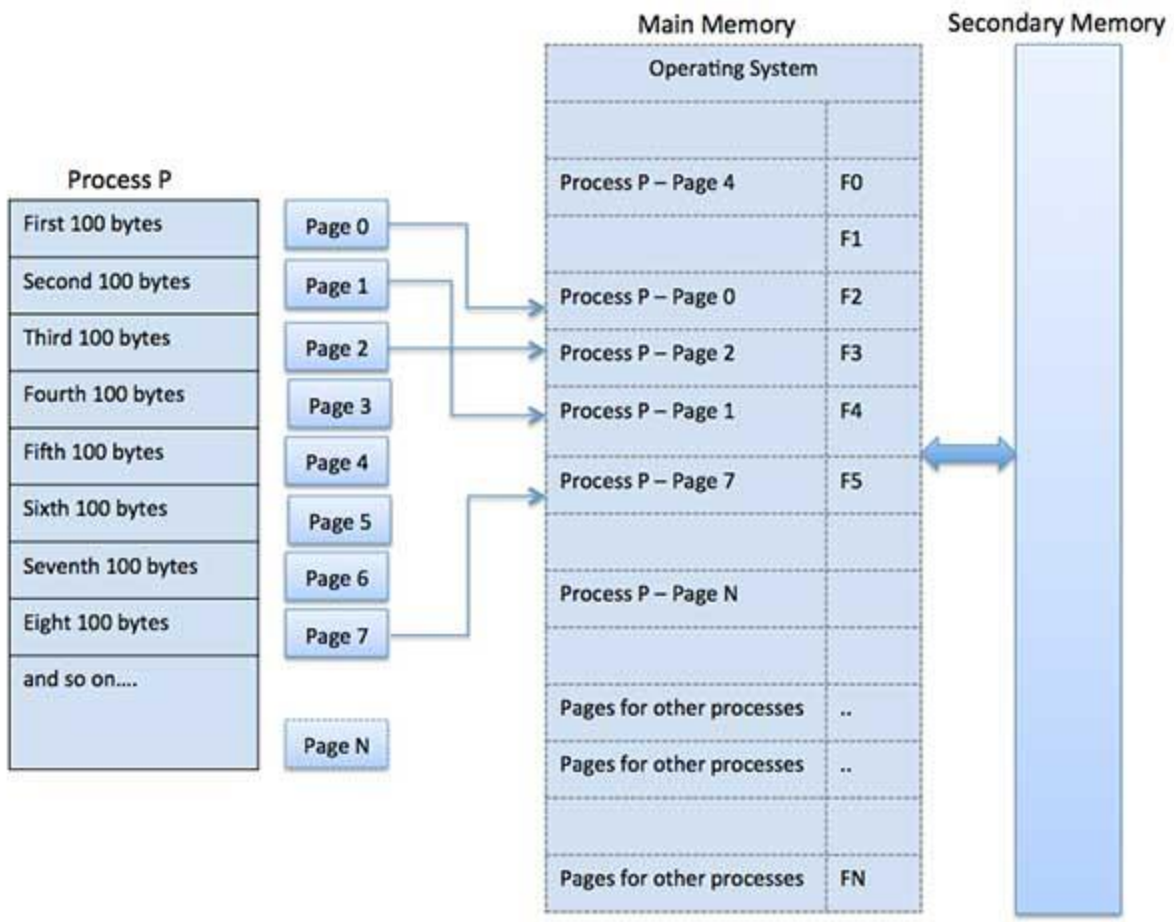
The internal fragmentation can be reduced by effectively assigning the smallest partition but large enough for the process.

Paging

A computer can address more memory than the amount physically installed on the system. This extra memory is actually called virtual memory and it is a section of a hard that's set up to emulate the computer's RAM. Paging technique plays an important role in implementing virtual memory.

Paging is a memory management technique in which process address space is broken into blocks of the same size called **pages** (size is power of 2, between 512 bytes and 8192 bytes). The size of the process is measured in the number of pages.

Similarly, main memory is 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.



Address Translation

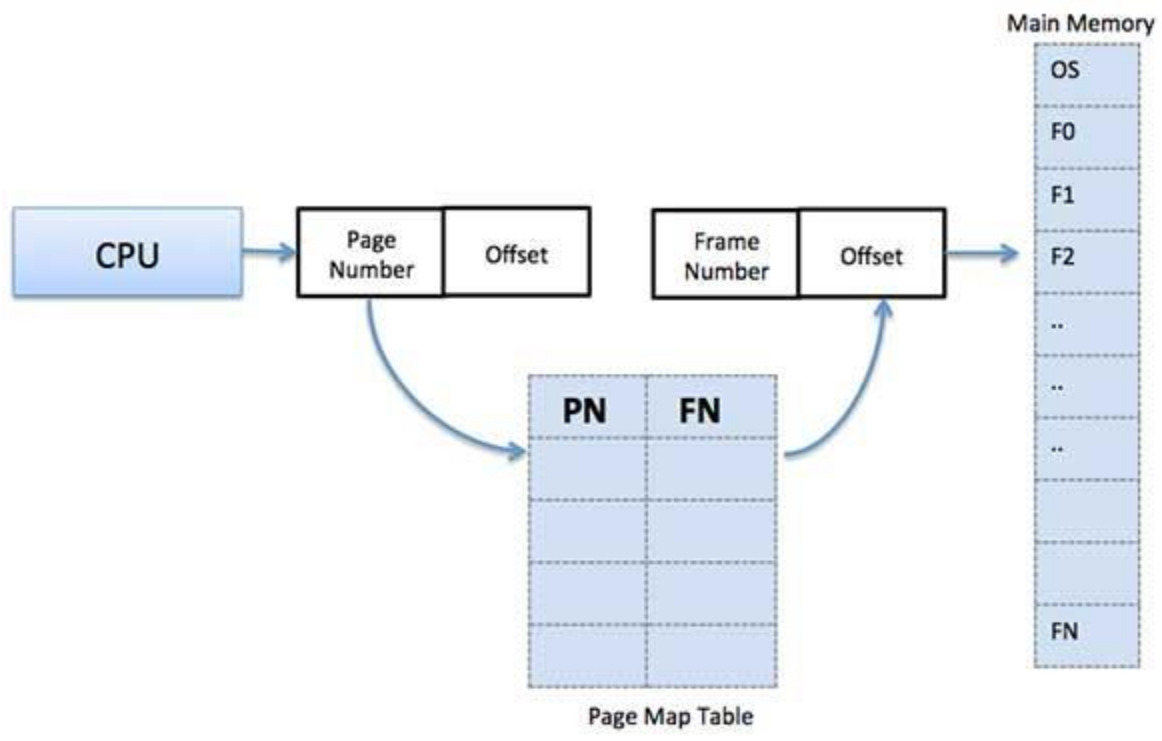
Page address is called **logical address** and represented by **page number** and the **offset**.

```
Logical Address = Page number + page offset
```

Frame address is called **physical address** and represented by a **frame number** and the **offset**.

```
Physical Address = Frame number + page offset
```

A data structure called **page map table** is 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. When a computer runs out of RAM, 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.

Advantages and Disadvantages of Paging

Here is a list of advantages and disadvantages of paging −

- Paging reduces external fragmentation, but still suffer from internal fragmentation.
- Paging is simple to implement and assumed as an 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.

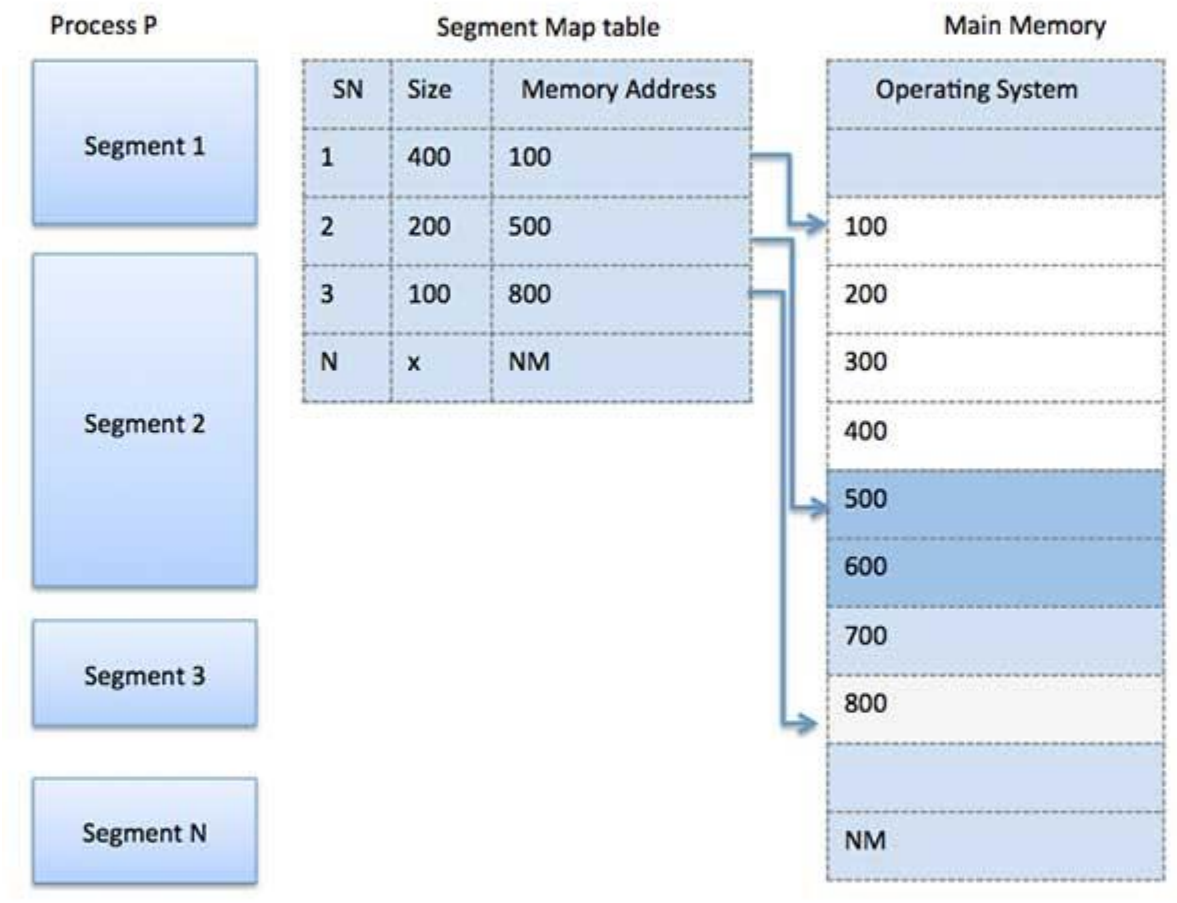
Segmentation

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

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

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

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



SN	Size	Memory Address
1	400	100
2	200	500
3	100	800
N	x	NM

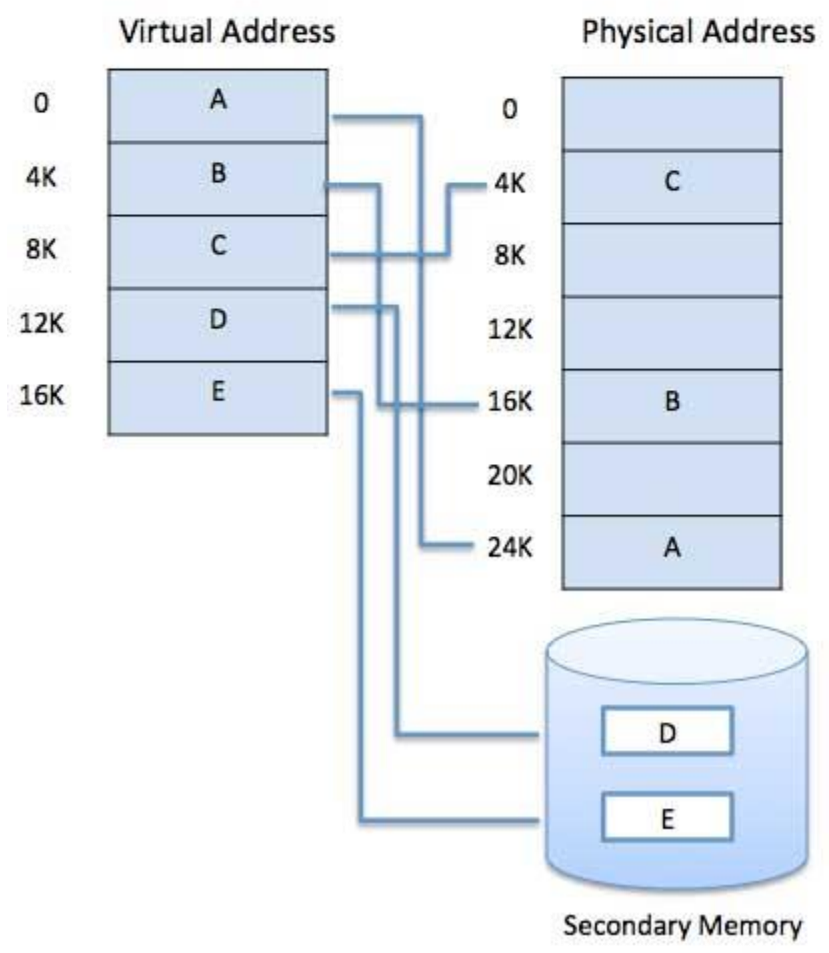
A computer can address more memory than the amount physically installed on the system. This extra memory is actually called **virtual memory** and it is a section of a hard disk that's set up to emulate the computer's RAM.

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

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

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

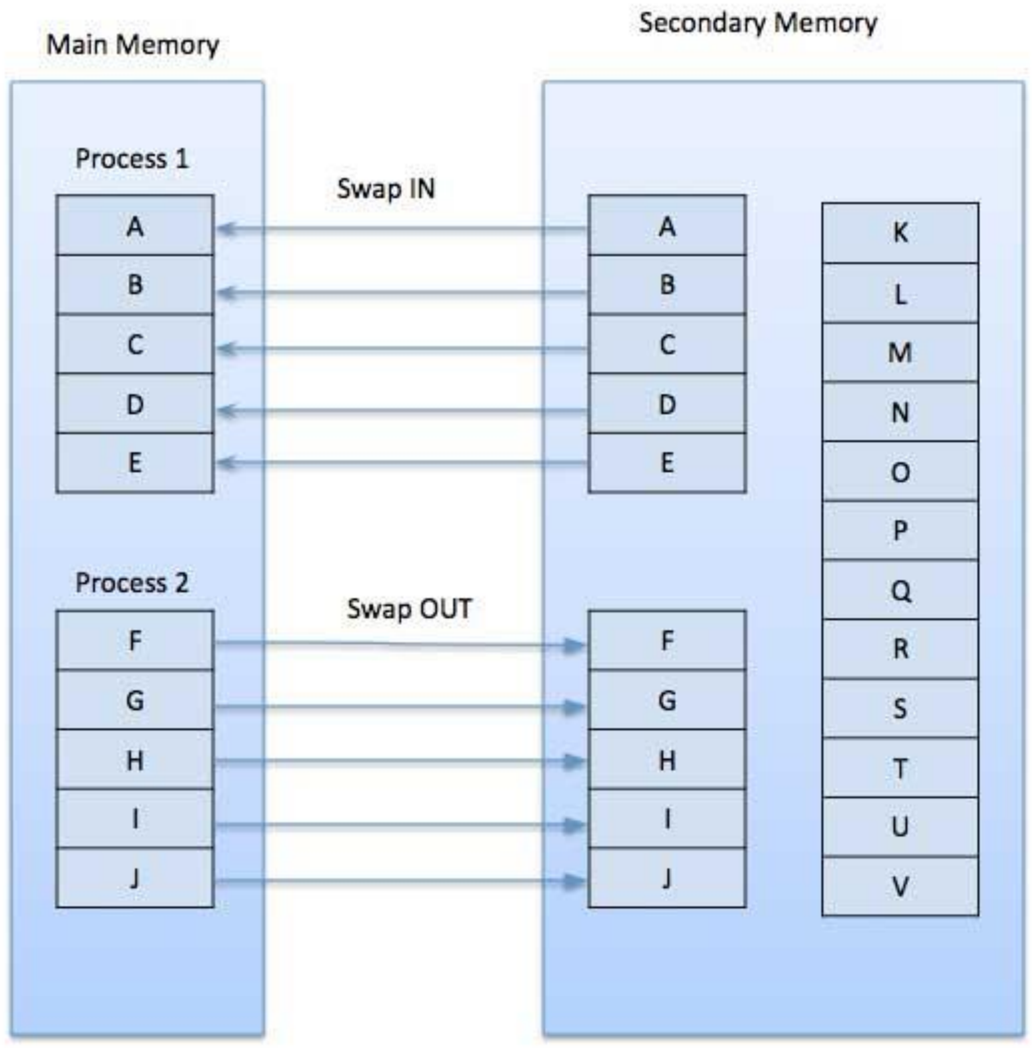
Modern microprocessors intended for general-purpose use, a memory management unit, or MMU, is built into the hardware. The MMU's job is to translate virtual addresses into physical addresses. A basic example is given below −



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

Demand Paging

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



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

Advantages

Following are the advantages of Demand Paging −

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

Disadvantages

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

Page Replacement Algorithm

Page replacement algorithms are the techniques using which an Operating System decides which memory pages to swap out, write to disk when a page of memory needs to be allocated. Paging happens whenever a page fault occurs and a free page cannot be used for allocation purpose accounting to reason that pages are not available or the number of free pages is lower than required pages.

When the page that was selected for replacement and was paged out, is referenced again, it has to read in from disk, and this requires for I/O completion. This process determines the quality of the page replacement algorithm: the lesser the time waiting for page-ins, the better is the algorithm.

A page replacement algorithm looks at the limited information about accessing the pages provided by hardware, and tries to select which pages should be replaced to minimize the total number of page misses, while balancing it with the costs of primary storage and processor time of the algorithm itself. There are many different page replacement algorithms. We evaluate an algorithm by running it on a particular string of memory reference and computing the number of page faults,

Reference String

The string of memory references is called reference string. Reference strings are generated artificially or by tracing a given system and recording the address of each memory reference. The latter choice produces a large number of data, where we note two things.

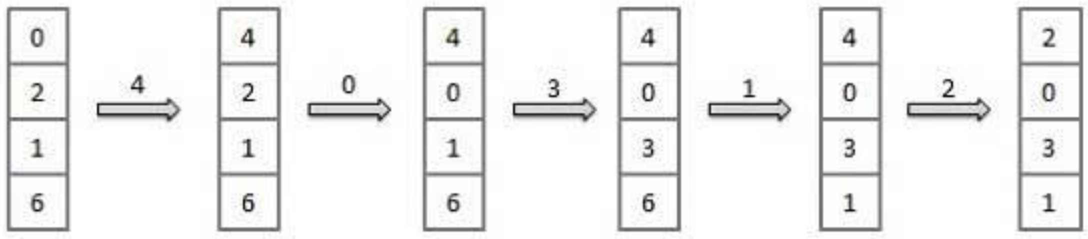
- For a given page size, we need to consider only the page number, not the entire address.
- If we have a reference to a page **p**, then any immediately following references to page **p** will never cause a page fault. Page p will be in memory after the first reference; the immediately following references will not fault.
- For example, consider the following sequence of addresses − 123,215,600,1234,76,96
- If page size is 100, then the reference string is 1,2,6,12,0,0

First In First Out (FIFO) algorithm

- Oldest page in main memory is the one which will be selected for replacement.
- Easy to implement, keep a list, replace pages from the tail and add new pages at the head.

Reference String : 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Misses   : x x   x x   x x   x x x



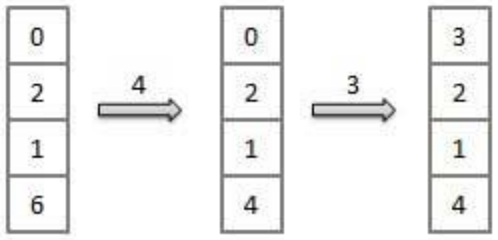
Fault Rate = 9 / 12 = 0.75

Optimal Page algorithm

- An optimal page-replacement algorithm has the lowest page-fault rate of all algorithms. An optimal page-replacement algorithm exists, and has been called OPT or MIN.
- Replace the page that will not be used for the longest period of time. Use the time when a page is to be used.

Reference String : 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Misses   : x x   x x   x   x



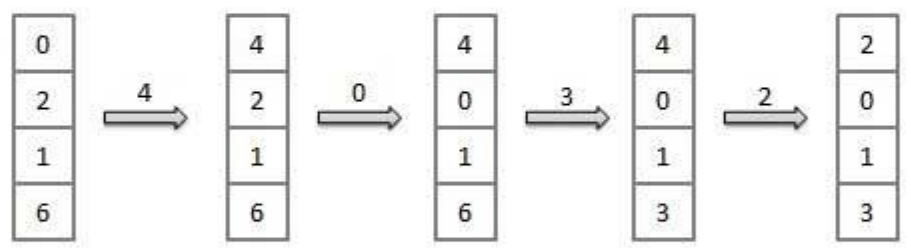
Fault Rate = 6 / 12 = 0.50

Least Recently Used (LRU) algorithm

- Page which has not been used for the longest time in main memory is the one which will be selected for replacement.
- Easy to implement, keep a list, replace pages by looking back into time.

Reference String : 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Misses :x x x x x x x x



Fault Rate = 8 / 12 = 0.67

Page Buffering algorithm

- To get a process start quickly, keep a pool of free frames.
- On page fault, select a page to be replaced.
- Write the new page in the frame of free pool, mark the page table and restart the process.
- Now write the dirty page out of disk and place the frame holding replaced page in free pool.

Least frequently Used(LFU) algorithm

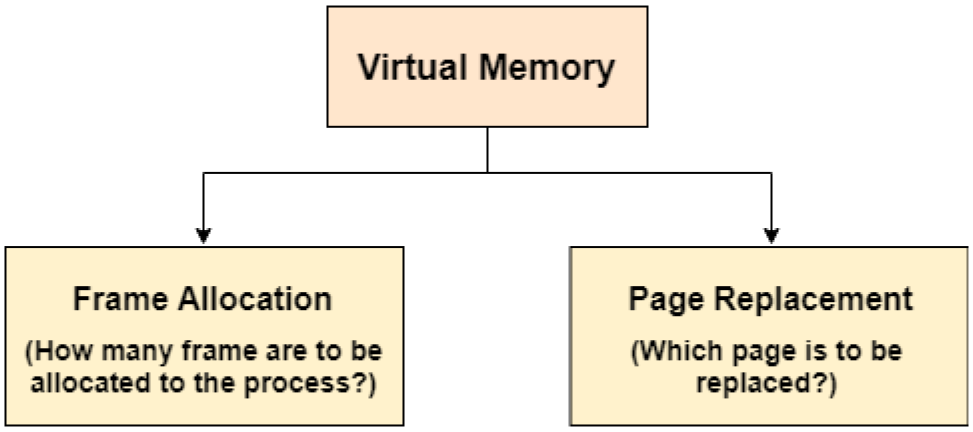
- The page with the smallest count is the one which will be selected for replacement.
- This algorithm suffers from the situation in which a page is used heavily during the initial phase of a process, but then is never used again.

Most frequently Used(MFU) algorithm

- This algorithm is based on the argument that the page with the smallest count was probably just brought in and has yet to be used.

Page Replacement Algorithms

The page replacement algorithm decides which memory page is to be replaced. The process of replacement is sometimes called swap out or write to disk. Page replacement is done when the requested page is not found in the main memory (page fault).


Virtual Memory
Frame Allocation
(How many frame are to be allocated to the process?)
Page Replacement
(Which page is to be replaced?)

There are two main aspects of virtual memory, Frame allocation and Page Replacement. It is very important to have the optimal frame allocation and page replacement algorithm. Frame allocation is all about how many frames are to be allocated to the process while the page replacement is all about determining the page number which needs to be replaced in order to make space for the requested page.

What If the algorithm is not optimal?

1. if the number of frames which are allocated to a process is not sufficient or accurate then there can be a problem of thrashing. Due to the lack of frames, most of the pages will be residing in the main memory and therefore more page faults will occur.

However, if OS allocates more frames to the process then there can be internal fragmentation.

2. If the page replacement algorithm is not optimal then there will also be the problem of thrashing. If the number of pages that are replaced by the requested pages will be referred in the near future then there will be more number of swap-in and swap-out and therefore the OS has to perform more replacements then usual which causes performance deficiency.

Therefore, the task of an optimal page replacement algorithm is to choose the page which can limit the thrashing.

Types of Page Replacement Algorithms

There are various page replacement algorithms. Each algorithm has a different method by which the pages can be replaced.

1. **Optimal Page Replacement algorithm** → this algorithms replaces the page which will not be referred for so long in future. Although it can not be practically implementable but it can be used as a benchmark. Other algorithms are compared to this in terms of optimality.
2. **Least recent used (LRU) page replacement algorithm** → this algorithm replaces the page which has not been referred for a long time. This algorithm is just opposite to the optimal page replacement algorithm. In this, we look at the past instead of staring at future.
3. **FIFO** → in this algorithm, a queue is maintained. The page which is assigned the frame first will be replaced first. In other words, the page which resides at the rare end of the queue will be replaced on the every page fault.

Unit 4:

information management function of an operating system.

Ans. A computer system works with 'Information'. It stores information, processes information, provides information etc. Thus managing this information is also an important and necessary task performed by OS. To support this function, OS's have one component called **'Information Management Component'**. This information management component of OS is structured as follows:

- **Physical IOCS (Input-Output Control System)** is responsible for ***device management*** and for ensuring ***device independence***. It provides a basic capability for the programs to perform their own IO, without involving themselves with the intricacies of device handling.
- **Logical IOCS** is responsible for efficient organization and access of data on IO devices. It provides basic capabilities for ***file definition***, choice of **data organization** and ***access methods***.
- **File System** is responsible for protection and controlled sharing of files.

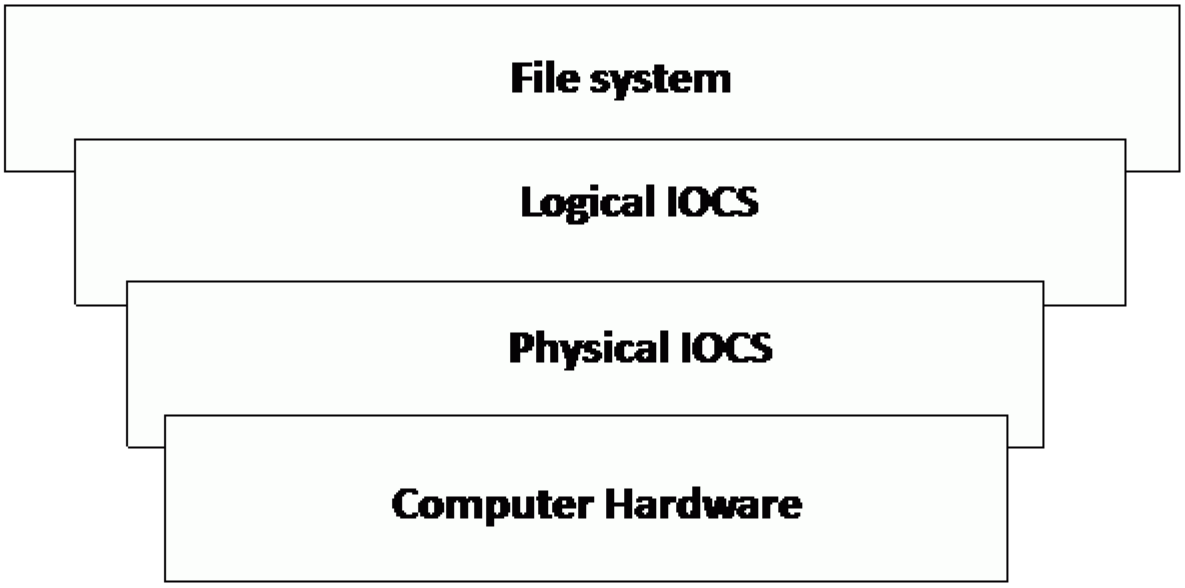


Figure: Hierarchy Of Information Management Modules.

1. **The Physical IOCS and IO Organization**

We have already become acquainted with
I/O *Channel* which makes CPU free for the duration of an I/O channel. One or more input or output devices are connected to an I/O channel. An I/O channel provides a data path between the main storage and an I/O device, and monitors the entire execution of I/O operation.

Each I/O channel, its control units and devices on them in the system are given unique codes recognized by the hardware. Thus a device is uniquely identified by a *device address* which consists of three component addresses: for I/O channel, control unit and device, (Ch, CU, Dev). Steps in the execution of an I/O operation are as follows:

1. **I/O Initiation**

I/O initiation starts with issuing of a CPU instruction called ‘Start IO’. This “Start IO’ instruction provides the device address (*ch, cu, dev)* for the required device.

The CPU identifies the concerned channel, control unit of the device and the device through the component addresses: ch, cu, dev.

2. **Device Selection**

The concerned channel then checks the device to see whether it is available or not. This process is called *device selection*. If device selection is successful, ‘*success*‘ condition code is set in PSR (register that stores CPU state) and IO instruction’s execution starts.

If device selection fails (because of device’s non availability if it is busy doing some other operation, or hardware malfunction), ‘Start IO’ is terminated and *‘failure’* condition code is set in PSR. CPU now determines which action should be taken *viz.* retry the operation, cancel the program which wants I/O, or switch to execution of another program.

3. **I/O Instruction Execution**

If the required device is available, I/O channel reads the I/O instruction from the program in the main storage and then sets up the execution of I/O operations on the designated device.

4. **I/O termination**

At the end of the I/O operation(s), the designated device raises an interrupt for both type of ends: Successful and unsuccessful. IO termination interrupt provides following information to CPU:

- Device address of the device raising the interrupt condition,
- Status flags indicating
- IO termination type
- Error message if any.

The interrupt handling routine then takes action by analyzing status flags and by invoking error-recovery procedure (if required).

Role of the Physical IOCS

Physical IOCS integrates all the OS responsibilities regarding IO operations like making IO sequence easier for a problem by taking ***device management*** functions.

(Device management functions, (covered in IO orgranisation) handle the device for an I/O operation and for error recovery). Physical IOCS provides device-level error recovery routines which can be easily invoked when a recoverable error occurs. Thus, the problem program has only to issue an I/Os request and rest would be handled by Physical IOCS.

When an I/O operation is requested by a program (process), physical IOCS initiates IO in the above explained manner, marks the process state as ***blocked*** and frees the CPU for other computations, thus makes multiprogramming feasible. Physical IOCS also ensures ***protection*** to the programs from undue interference during I/O operation is being carried out.

Thus Physical IOCS makes the entire I/O sequence operations ‘transparent’ to the problem program. ***Transparency means that the only thing to be issued by the program is the request/order/command for it and rest all internal operations are doe by the control program (physical IOCS in this case) without making it visible to the problem program.***

Physical IOCS also makes ***device independence*** possible as it provides a name to the device (***device naming***). Thus the device becomes a ***logical device*** for the user/programmer. Now the programmer uses only the ***device name*** rather than a device address. Internally the device name is bound to a physical device address which can be changed if necessary (only by physical IOCS) without affecting the program performance.

2. Logical IOCS

The creation and access of files are facilitated by the logical IOCS through file level commands. Open a file, read a record, delete a record, etc. ***Three-wayed*** help is provided for the programmer:

- The programmer need not know the intricate details of I/O device. Channel, Control unit of the device.
- The programmer need not know physical IOCS interface, its details and handling.
- The programmer need not know how an I/O is to be performed.

The logical IOCS also facilitates file processing by providing generalized access methods for various file organizations. The programmer can decide access method suited to his requirements. Access methods refers to the mechanism used to retrieve records from a file. Details regarding location of file, device address, etc are made available to the access methods by the logical IOCS.

3. File System

'File System's function is to facilitate easy creations, storage and access of files in order to enable easy sharing of files between programs and their protection against illegal access. Here it differs from logical IOCS. The logical IOCS permits any program if it provides the file name and processes the file in any manner the program desires whereas the file system ensures protection by allowing only that program which has proper access privileges. ***The major functions of a file system are*** given below:

- It provides file naming freedom to the users and permits controlled sharing of file (protection).
- provides long and short term storage of files.
- provides security against loss of information due to system failure (security).

File systems also facilitates grouping of files in form of directories for the purpose of the access and protection.

File

A file is a named 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.

File Structure

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.
- 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.

File Type

File type refers to the ability of the operating system to distinguish different types of file such as text files source files and binary files etc. Many operating systems support many types of files. Operating system like MS-DOS and UNIX have the following types of files −

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.

Directory files

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

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** − data is handled character by character as in case of terminals or printers.
- **Block special files** − data is handled in blocks as in the case of disks and tapes.

File Access Mechanisms

File access mechanism refers to the manner in which the records of a file may be accessed. There are several ways to access files −

- Sequential access
- Direct/Random access
- Indexed sequential access

Sequential access

A sequential access is that in which the records are accessed in some sequence, i.e., the information in the file is processed in order, one record after the other. This access method is the most primitive one. Example: Compilers usually access files in this fashion.

Direct/Random access

- Random access file organization provides, accessing the records directly.
- Each record has its own address on the file with by the help of which it can be directly accessed for reading or writing.
- The records need not be in any sequence within the file and they need not be in adjacent locations on the storage medium.

Indexed sequential access

- This mechanism is built up on base of sequential access.
- An index is created for each file which contains pointers to various blocks.
- Index is searched sequentially and its pointer is used to access the file directly.

Space Allocation

Files are allocated disk spaces by operating system. Operating systems deploy following three main ways to allocate disk space to files.

- Contiguous Allocation
- Linked Allocation
- Indexed Allocation

Contiguous Allocation

- Each file occupies a contiguous address space on disk.
- Assigned disk address is in linear order.
- Easy to implement.
- External fragmentation is a major issue with this type of allocation technique.

Linked Allocation

- Each file carries a list of links to disk blocks.
- Directory contains link / pointer to first block of a file.
- No external fragmentation
- Effectively used in sequential access file.
- Inefficient in case of direct access file.

Indexed Allocation

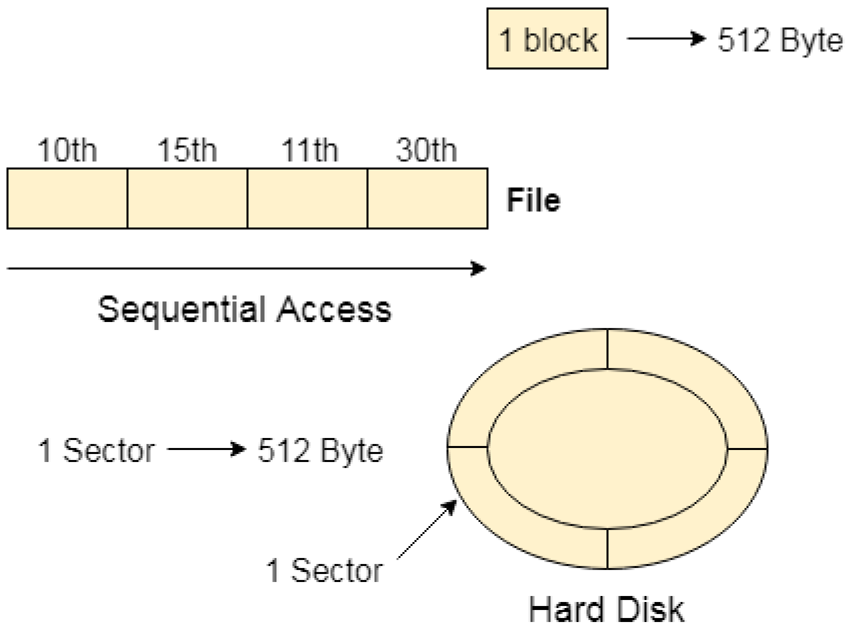
- Provides solutions to problems of contiguous and linked allocation.

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

File Access Methods

Let's look at various ways to access files stored in secondary memory.

Sequential Access



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

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

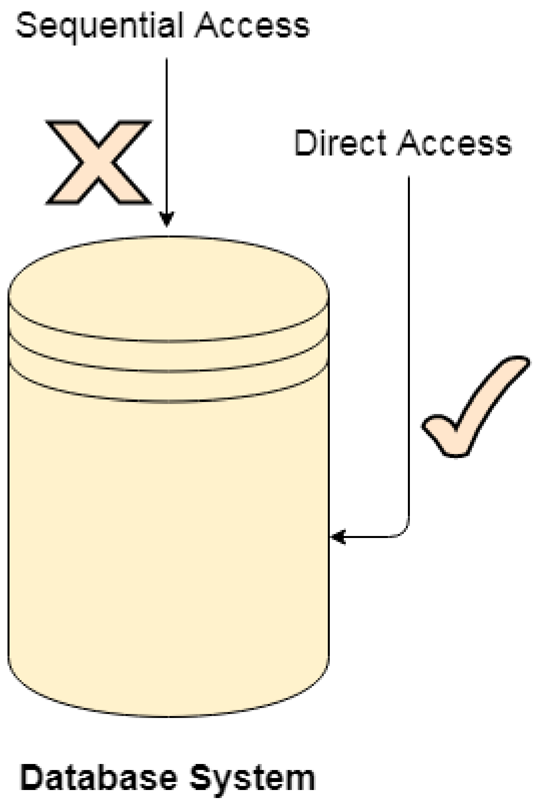
Modern word systems do provide the concept of direct access and indexed access but the most used method is sequential access due to the fact that most of the files such as text files, audio files, video files, etc need to be sequentially accessed.

Direct Access

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

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

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



Indexed Access

If a file can be sorted on any of the filed then an index can be assigned to a group of certain records. However, A particular record can be accessed by its index. The index is nothing but the address of a record in the file.

In index accessing, searching in a large database became very quick and easy but we need to have some extra space in the memory to store the index value.

Free space management in Operating System

- Difficulty Level : Easy
- Last Updated : 14 Aug, 2019

The system keeps tracks of the free disk blocks for allocating space to files when they are created. Also, to reuse the space released from deleting the files, free space management becomes crucial. The 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 mainly as:

1. **Bitmap or Bit vector –**
   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.
   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**.
   Attention reader! Don't stop learning now. Get hold of all the important CS Theory concepts for SDE interviews with the **CS Theory Course** at a student-friendly price and become industry ready.

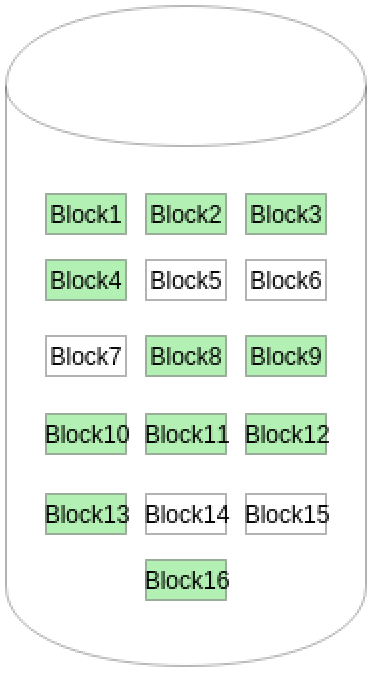


Figure - 1

Advantages –

- Simple to understand.
- Finding the first free block is efficient. It requires scanning the words (a group of 8 bits) in a bitmap for a non-zero word. (A 0-valued word has all bits 0). The first free block is then found by scanning for the first 1 bit in the non-zero word.

The block number can be calculated as:
*(number of bits per word) *(number of 0-values words) + offset of bit first bit 1 in the non-zero word* .
For the *Figure-1*, we scan the bitmap sequentially for the first non-zero word.
The first group of 8 bits (00001110) constitute a non-zero word since all bits are not 0. After the non-0 word is found, we look for the first 1 bit. This is the 5th bit of the non-zero word. So, offset = 5.
Therefore, the first free block number = 8*0+5 = 5.

2. **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.

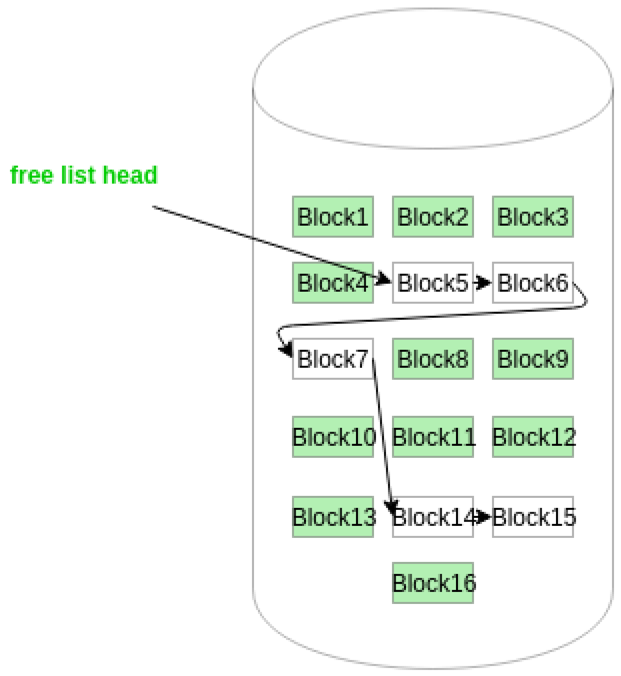


Figure - 2

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.

3. **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.
4. **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:
   1. Address of first free disk block
   2. 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.

Disk Scheduling Algorithms

- Difficulty Level : Easy
- Last Updated : 28 Jun, 2021

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.

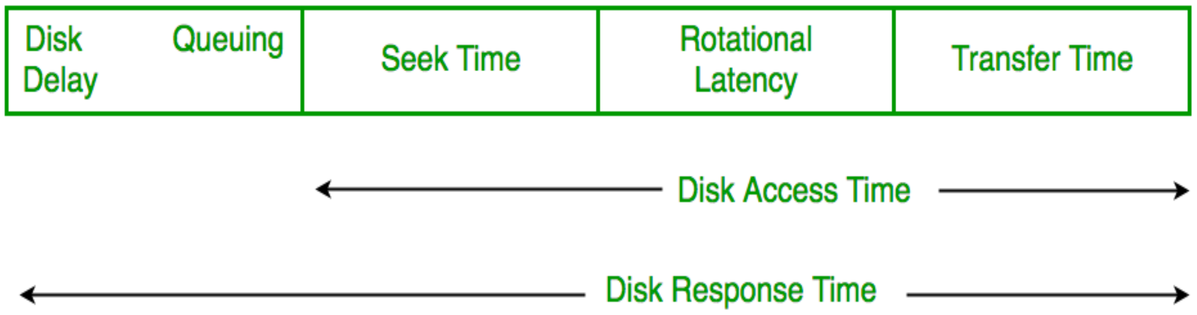
There are many Disk Scheduling Algorithms but before discussing them let's have a quick look at some of the important terms:

- **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:** Disk Access Time is:

```
Disk Access Time = Seek Time +

                   Rotational Latency +

                   Transfer Time
```

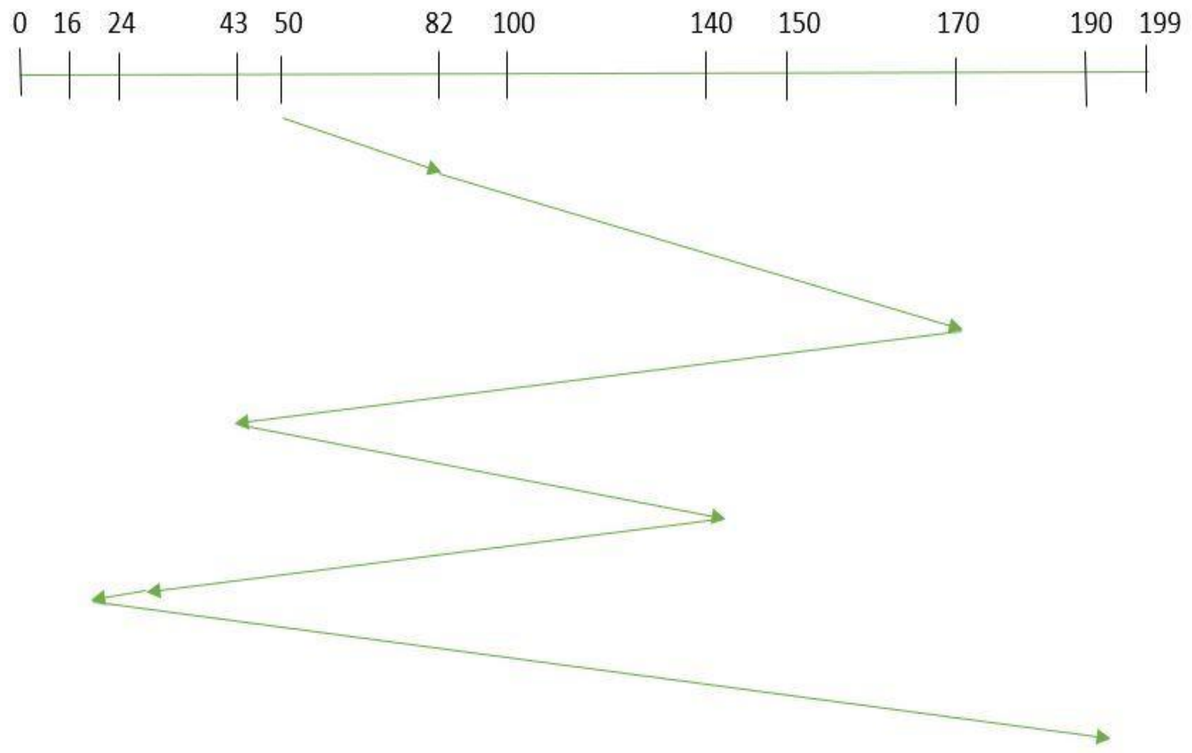


- **Disk Response Time:** Response Time is the average of time spent by a request waiting to perform its I/O operation. *Average Response time* is the response time of the all requests. *Variance Response Time* is measure of how individual request are serviced with respect to average response time. So the disk scheduling algorithm that gives minimum variance response time is better.

Disk Scheduling Algorithms

1. **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

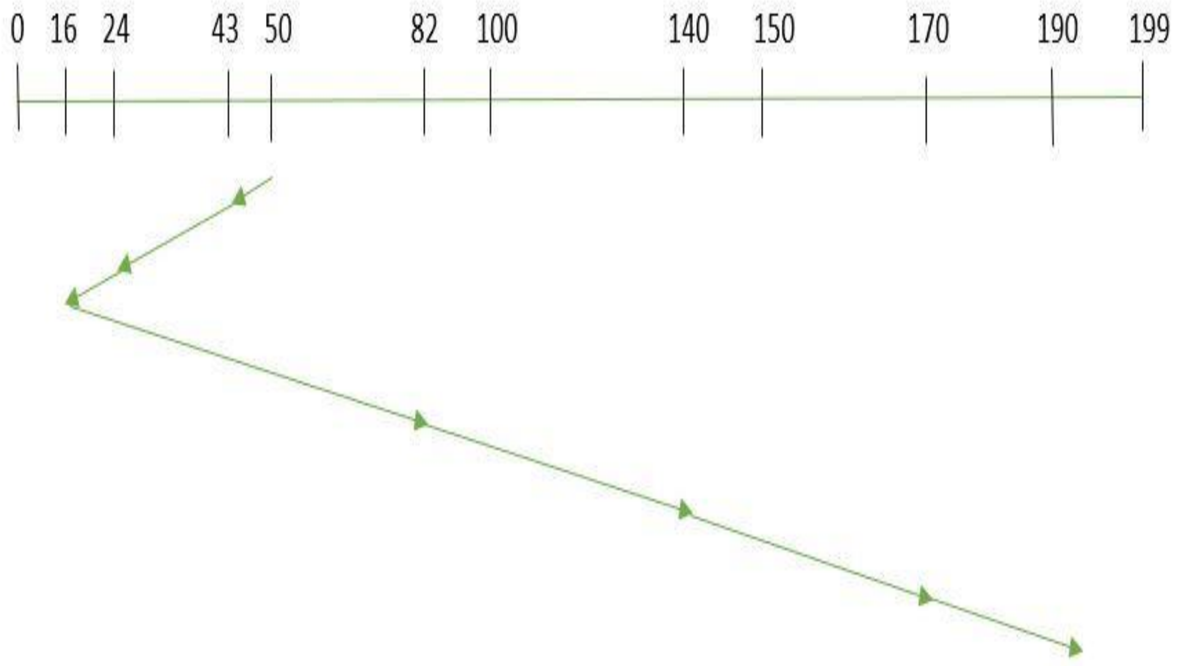
Disadvantages:

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

2. **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.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:

=(50-43)+(43-24)+(24-16)+(82-16)+(140-82)+(170-40)+(190-170)
=208

Advantages:

- Average Response Time decreases
- Throughput increases

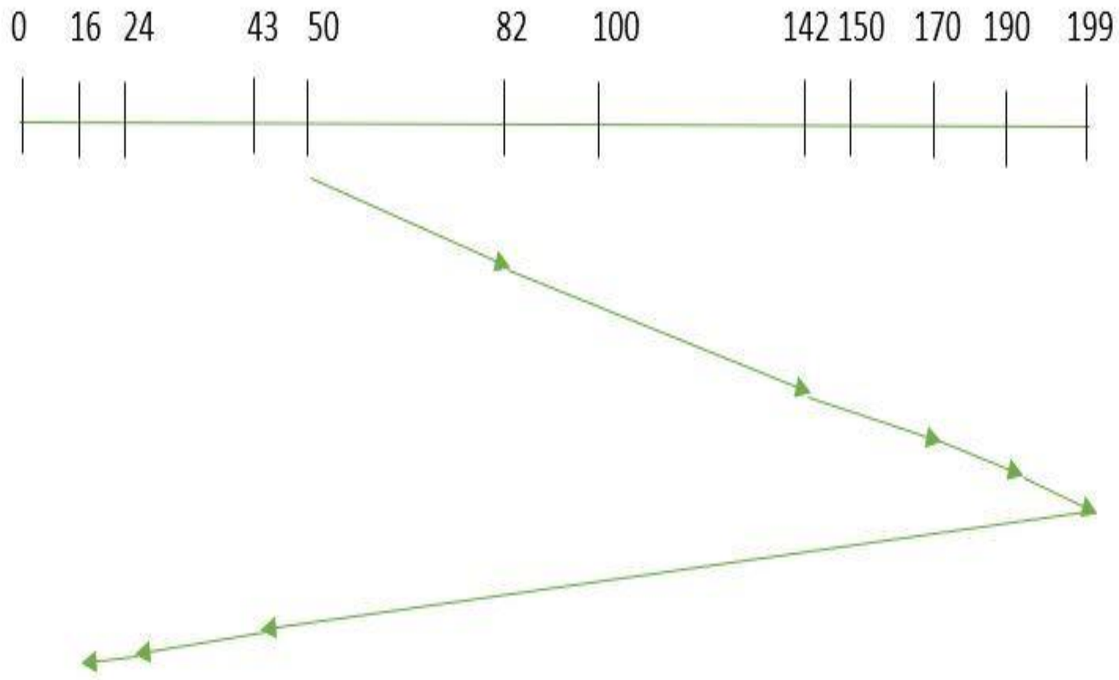
Disadvantages:

- 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

3. **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.

   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".**



   Therefore, the seek time is calculated as:

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

Advantages:

- High throughput
- Low variance of response time
- Average response time

Disadvantages:

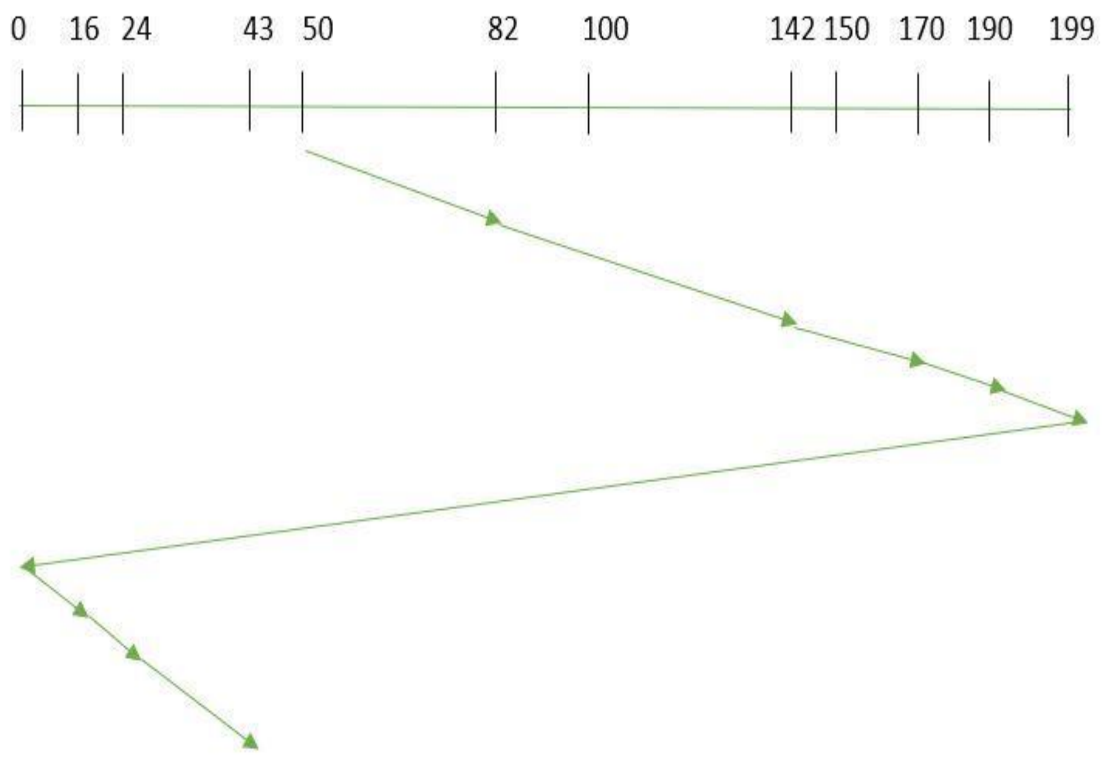
- Long waiting time for requests for locations just visited by disk arm

4. **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).

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:

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

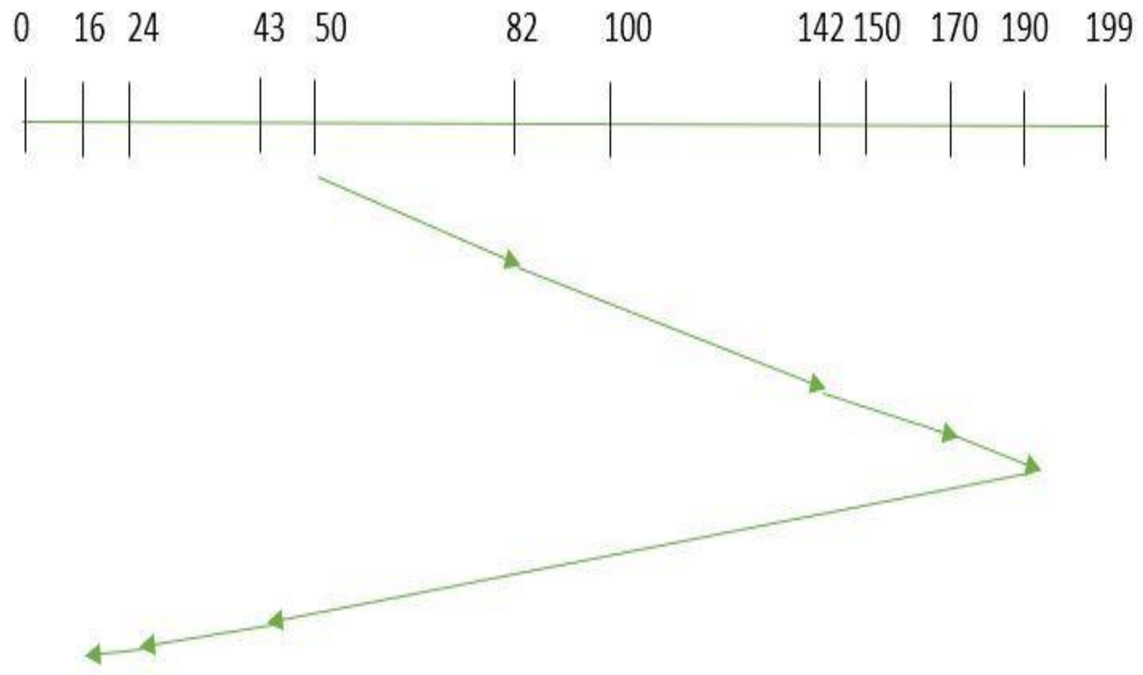
Advantages:

- Provides more uniform wait time compared to SCAN

5. **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".**

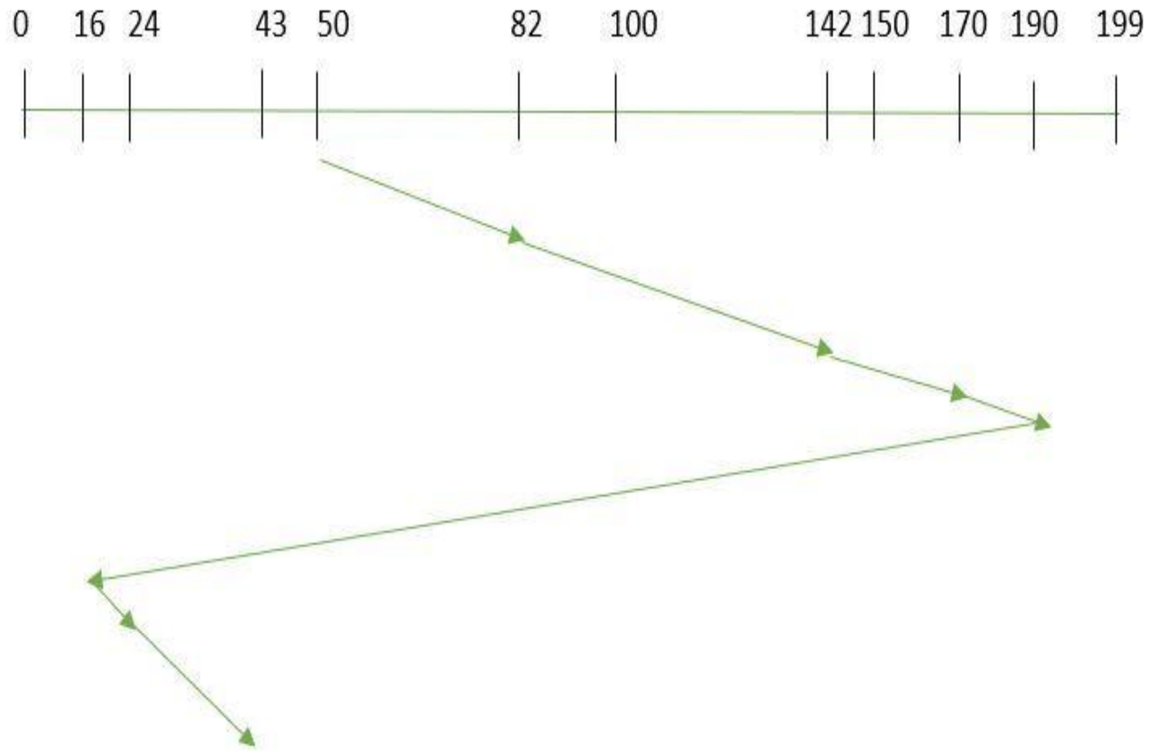


   So, the seek time is calculated as:

   =(190-50)+(190-16)
   =314

6. **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:

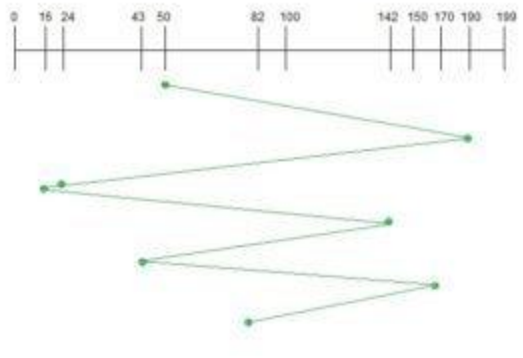
   =(190-50)+(190-16)+(43-16)
   =341

7. **RSS**– It stands for random scheduling and just like its name it is nature. It is used in situations where scheduling involves random attributes such as random processing time, random due dates, random weights, and stochastic machine breakdowns this algorithm sits perfect. Which is why it is usually used for and analysis and simulation.

8. **LIFO**– In LIFO (Last In, First Out) algorithm, newest jobs are serviced before the existing ones i.e. in order of requests that get serviced the job that is newest or last entered is serviced first and then the rest in the same order.
   Advantages
      - Maximizes locality and resource utilization

   Disadvantages
      - Can seem a little unfair to other requests and if new requests keep coming in, it cause starvation to the old and existing ones.

   Example
   Suppose the order of request is- (82,170,43,142,24,16,190)
   And current position of Read/Write head is : 50



9. **N-STEP SCAN** – It is also known as N-STEP LOOK algorithm. In this a buffer is created for N requests. All requests belonging to a buffer will be serviced in one go. Also once the buffer is full no new requests are kept in this buffer and are sent to another one. Now, when these N requests are serviced, the time comes for another top N requests and this way all get requests get a guaranteed service
   Advantages
      - It eliminates starvation of requests completely
10. **FSCAN**– This algorithm uses two sub-queues. During the scan all requests in the first queue are serviced and the new incoming requests are added to the second queue. All new requests are kept on halt until the existing requests in the first queue are serviced.
   Advantages
      - FSCAN along with N-Step-SCAN prevents “arm stickiness” (phenomena in I/O scheduling where the scheduling algorithm

continues to service requests at or near the current sector and thus prevents any seeking)

Each algorithm is unique in its own way. Overall Performance depends on the number and type of requests.

Unit 5:

UNIX Introduction

What is UNIX?

UNIX is an operating system which was first developed in the 1960s, and has been under constant development ever since. By operating system, we mean the suite of programs which make the computer work. It is a stable, multi-user, multi-tasking system for servers, desktops and laptops.

UNIX systems also have a graphical user interface (GUI) similar to Microsoft Windows which provides an easy to use environment. However, knowledge of UNIX is required for operations which aren't covered by a graphical program, or for when there is no windows interface available, for example, in a telnet session.

Types of UNIX

There are many different versions of UNIX, although they share common similarities. The most popular varieties of UNIX are Sun Solaris, GNU/Linux, and MacOS X.

Here in the School, we use Solaris on our servers and workstations, and Fedora Core Linux on the servers and desktop PCs.

The UNIX operating system

The UNIX operating system is made up of three parts; the kernel, the shell and the programs.

The kernel

The kernel of UNIX is the hub of the operating system: it allocates time and memory to programs and handles the filestore and communications in response to system calls.

As an illustration of the way that the shell and the kernel work together, suppose a user types `rm myfile` (which has the effect of removing the file **myfile**). The shell searches the filestore for the file containing the program `rm`, and then requests the kernel, through system calls, to execute the program `rm` on **myfile**. When the process `rm myfile` has finished running, the shell then returns the UNIX prompt % to the user, indicating that it is waiting for further commands.

The shell

The shell acts as an interface between the user and the kernel. When a user logs in, the login program checks the username and password, and then starts another program called the shell. The shell is a command line interpreter (CLI). It interprets the commands the user types in and arranges for them to be carried out. The commands are themselves programs: when they terminate, the shell gives the user another prompt (% on our systems).

The adept user can customise his/her own shell, and users can use different shells on the same machine. Staff and students in the school have the **tcsh shell** by default.

The tcsh shell has certain features to help the user inputting commands.

Filename Completion - By typing part of the name of a command, filename or directory and pressing the [**Tab**] key, the tcsh shell will complete the rest of the name automatically. If the shell finds more than one name beginning with those letters you have typed, it will beep, prompting you to type a few more letters before pressing the tab key again.

History - The shell keeps a list of the commands you have typed in. If you need to repeat a command, use the cursor keys to scroll up and down the list or type history for a list of previous commands.

Files and processes

Everything in UNIX is either a file or a process.

A process is an executing program identified by a unique PID (process identifier).

A file is a collection of data. They are created by users using text editors, running compilers etc.

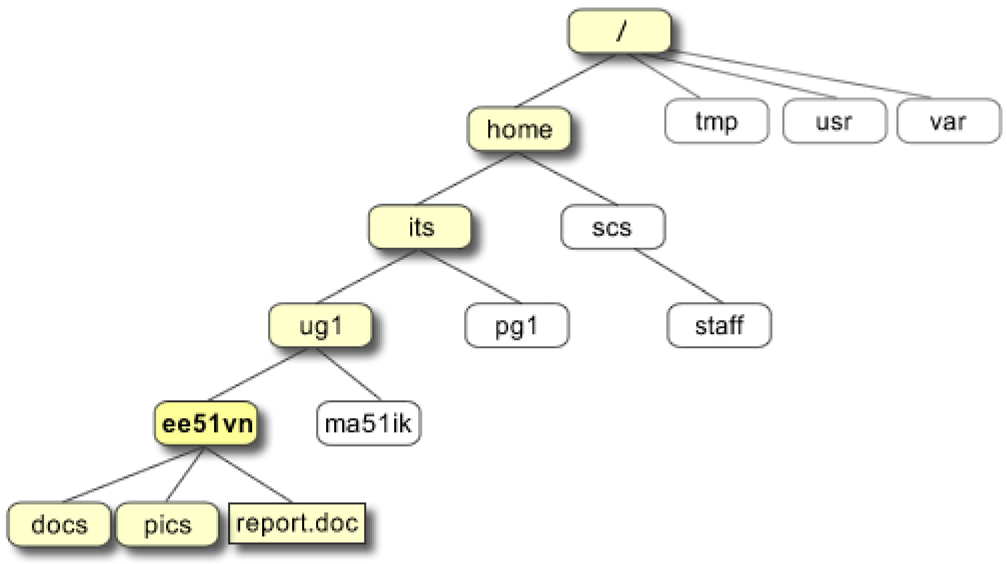
Examples of files:

- a document (report, essay etc.)
- the text of a program written in some high-level programming language

- instructions comprehensible directly to the machine and incomprehensible to a casual user, for example, a collection of binary digits (an executable or binary file);
- a directory, containing information about its contents, which may be a mixture of other directories (subdirectories) and ordinary files.

The Directory Structure

All the files are grouped together in the directory structure. The file-system is arranged in a hierarchical structure, like an inverted tree. The top of the hierarchy is traditionally called **root** (written as a slash /)

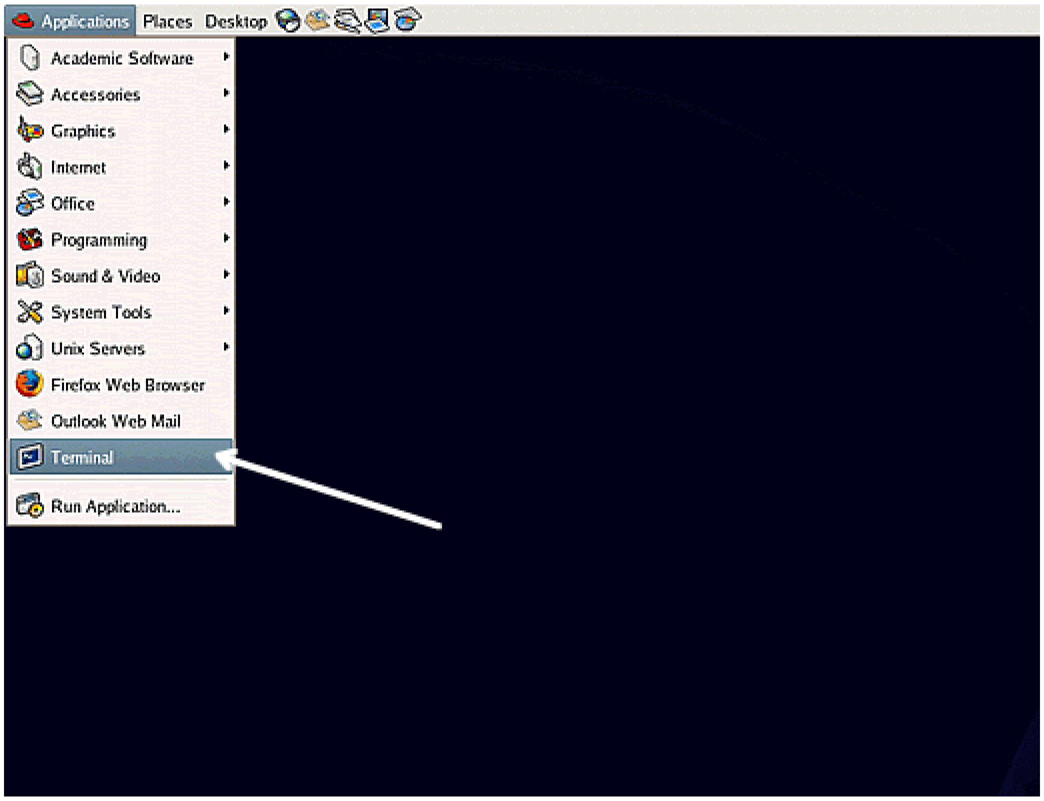


In the diagram above, we see that the home directory of the undergraduate student **"ee51vn"** contains two sub-directories (**docs** and **pics**) and a file called **report.doc**.

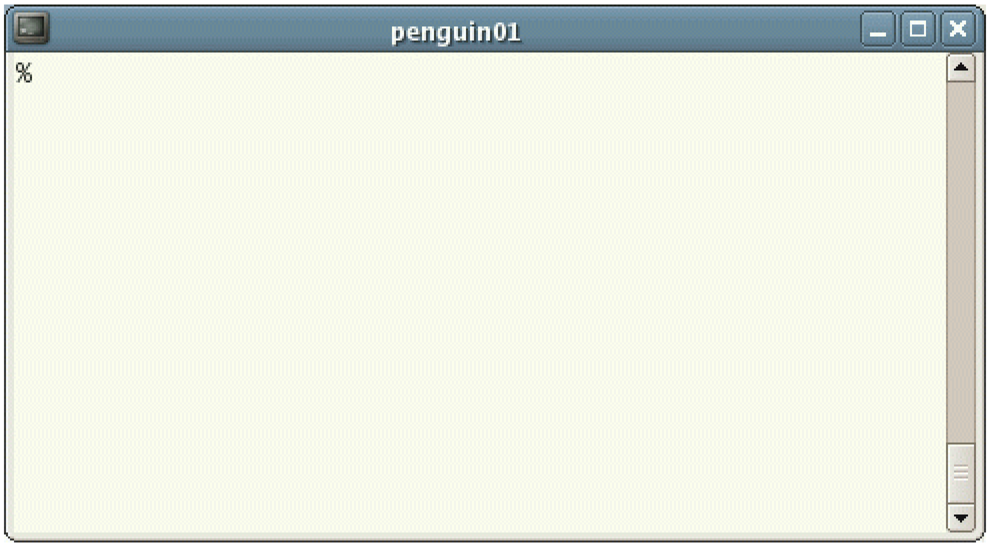
The full path to the file **report.doc** is **"/home/its/ug1/ee51vn/report.doc"**

Starting an UNIX terminal

To open an UNIX terminal window, click on the "Terminal" icon from the drop-down menus.



An UNIX Terminal window will then appear with a % prompt, waiting for you to start entering commands.



```
%
```

Types, functions of User Interfaces of Operating Systems

In this article, we are going to discuss about the **Introduction to User interfaces, functions of user interfaces and the classification/types of user interfaces** in operating system.
Submitted by Prerana Jain, on June 26, 2018

User Interface

A **User interface (UI)** facilitates communication between an application and its user by acting as an intermediary between them. Each application including the operating system is provided with a specific UI for effective communication. The two basic function of a user interface of an application is to take the inputs from the user and to provide the output to the users. However, the types of inputs taken by the UI and the types of output provided by the UI may vary from one application to another.

A user interface of any operating system can be classified into one of the following types:

1. Graphical user interface (GUI)
2. Command line user interface (CLI)

1) Graphical user interface (GUI)

The graphical user interface is a type of GUI that enables the users to interact with the operating system by means of point-and-click operations. GUI contains several icons representing pictorial representation of the variables such as a file, directory, and device. The graphical icon provided in the UI can be manipulated by the users using a suitable pointing device such as a mouse, trackball, touch screen and light pen. The other input devices like keyboard can also be used to manipulate these graphical icons. GUIs are considered to be very user- friendly interface because each object is represented with a corresponding icon. Unlike the other UIs the users need not provide text command for executing tasks.

Some advantages of GUI based operating system

- The GUI interface is easy to understand and even the new users can operate on them on their own.
- The GUI interface visually acknowledges and confirms each type of activities performed by the users. For example when the user deletes a file in the Windows operating system, then the operating system asks for the confirmation before deleting it.
- The GUI interface enables the users to perform a number of tasks at the same time. This features of the operating system are also known as multitasking.

2) Command line Interface (CLI)

Command line interface is a type of UI that enables the users to interact with the operating system by issuing some specific commands. In order to perform a task in this interface, the user needs to type a command at the command line. When the user enters the key, the command line interpreter received a command. The software program that is responsible for receiving and processing the commands issued by the user. After processing the command are called command line interpreter, the command line interpreter displays the command prompt again along with the output of the previous command issued by the user. The disadvantages of the CLI is that the user needs to remember a lot to interact with the operating system. Therefore these types of interface are not considered very friendly from the users perspective.

Example: In order to perform a task, we need to type a command at the command prompt denoted by `C:\>` to copy a text file, say al.text, from the C drive of our computer system. To the D drive, we need to type the `copy` command at the command prompt.

File Systems in Operating System

A file is a collection of related information that is recorded on secondary storage. Or file is a collection of logically related entities. From user's perspective a file is the smallest allotment of logical secondary storage.

The name of the file is divided into two parts as shown below:

- name
- extension, separated by a period.

Files attributes and its operations:

Attributes	Types	Operations
Name	Doc	Create
Type	Exe	Open
Size	Jpg	Read
Creation Data	Xis	Write
Author	C	Append
Last Modified	Java	Truncate
protection	class	Delete
		Close

File type	Usual extension	Function
Executable	exe, com, bin	Read to run machine language program

File type	Usual extension	Function
Object	obj, o	Compiled, machine language not linked
Source Code	C, java, pas, asm, a	Source code in various languages
Batch	bat, sh	Commands to the command interpreter
Text	txt, doc	Textual data, documents
Word Processor	wp, tex, rrf, doc	Various word processor formats
Archive	arc, zip, tar	Related files grouped into one compressed file
Multimedia	mpeg, mov, rm	For containing audio/video information
Markup	xml, html, tex	It is the textual data and documents
Library	lib, a ,so, dll	It contains libraries of routines for programmers
Print or View	gif, pdf, jpg	It is a format for printing or viewing a ASCII or binary file.

FILE DIRECTORIES:

Collection of files is a file directory. The directory contains information about the files, including attributes, location and ownership. Much of this information, especially that is concerned with storage, is managed by the operating system. The directory is itself a file, accessible by various file management routines.

Information contained in a device directory are:

- Name
- Type
- Address
- Current length
- Maximum length
- Date last accessed
- Date last updated
- Owner id
- Protection information

Operation performed on directory are:

- Search for a file
- Create a file
- Delete a file
- List a directory
- Rename a file
- Traverse the file system

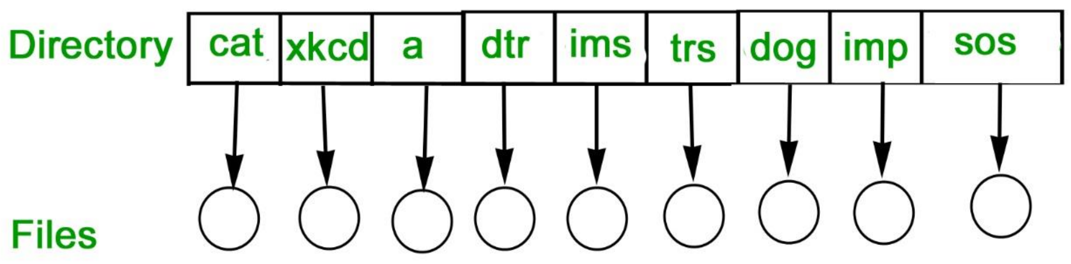
Advantages of maintaining directories are:

- **Efficiency:** A file can be located more quickly.
- **Naming:** It becomes convenient for users as two users can have same name for different files or may have different name for same file.
- **Grouping:** Logical grouping of files can be done by properties e.g. all java programs, all games etc.

SINGLE-LEVEL DIRECTORY

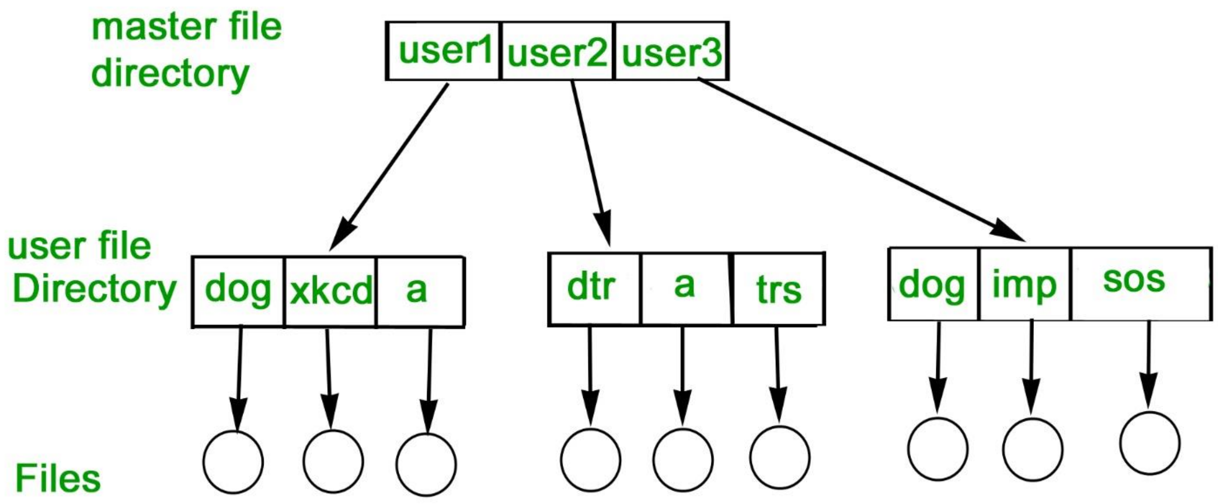
In this a single directory is maintained for all the users.

- **Naming problem:** Users cannot have same name for two files.
- **Grouping problem:** Users cannot group files according to their need.

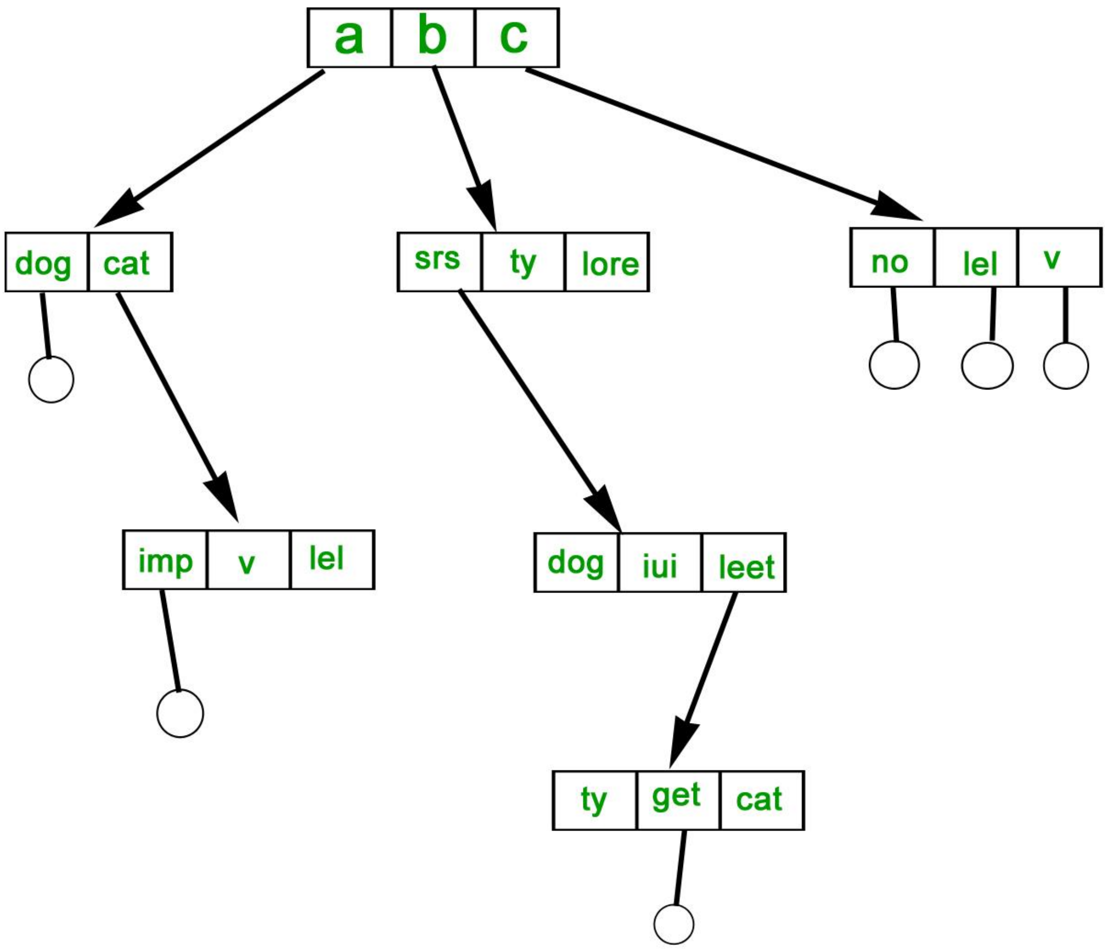


TWO-LEVEL DIRECTORY
In this separate directories for each user is maintained.

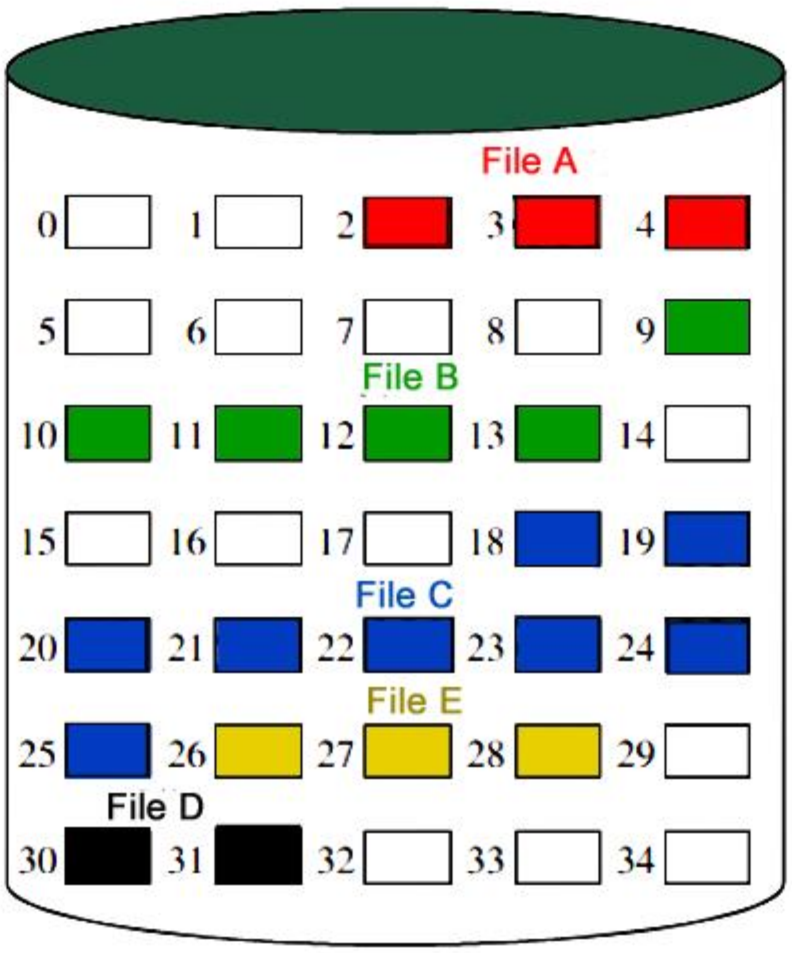
- Path name:Due to two levels there is a path name for every file to locate that file.
- Now,we can have same file name for different user.
- Searching is efficient in this method.



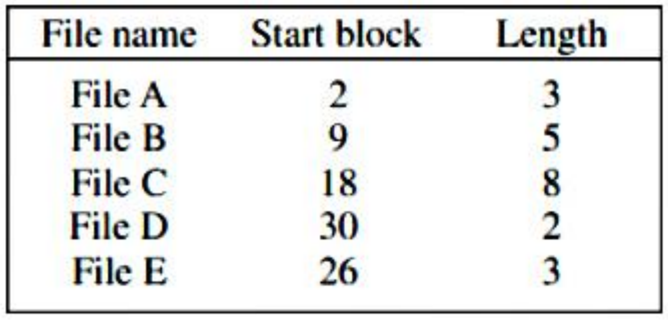
TREE-STRUCTURED DIRECTORY :
Directory is maintained in the form of a tree. Searching is efficient and also there is grouping capability. We have absolute or relative path name for a file.



FILE ALLOCATION METHODS :
1. Continuous Allocation –
A single continuous set of blocks is allocated to a file at the time of file creation. Thus, this is a pre-allocation strategy, using variable size portions. The file allocation table needs just a single entry for each file, showing the starting block and the length of the file. This method is best from the point of view of the individual sequential file. Multiple blocks can be read in at a time to improve I/O performance for sequential processing. It is also easy to retrieve a single block. For example, if a file starts at block b, and the ith block of the file is wanted, its location on secondary storage is simply b+i-1.



File allocation table



File name	Start block	Length
File A	2	3
File B	9	5
File C	18	8
File D	30	2
File E	26	3

Disadvantage –

- External fragmentation will occur, making it difficult to find contiguous blocks of space of sufficient length. Compaction algorithm will be necessary to free up additional space on disk.
- Also, with pre-allocation, it is necessary to declare the size of the file at the time of creation.

2. Linked Allocation(Non-contiguous allocation) –

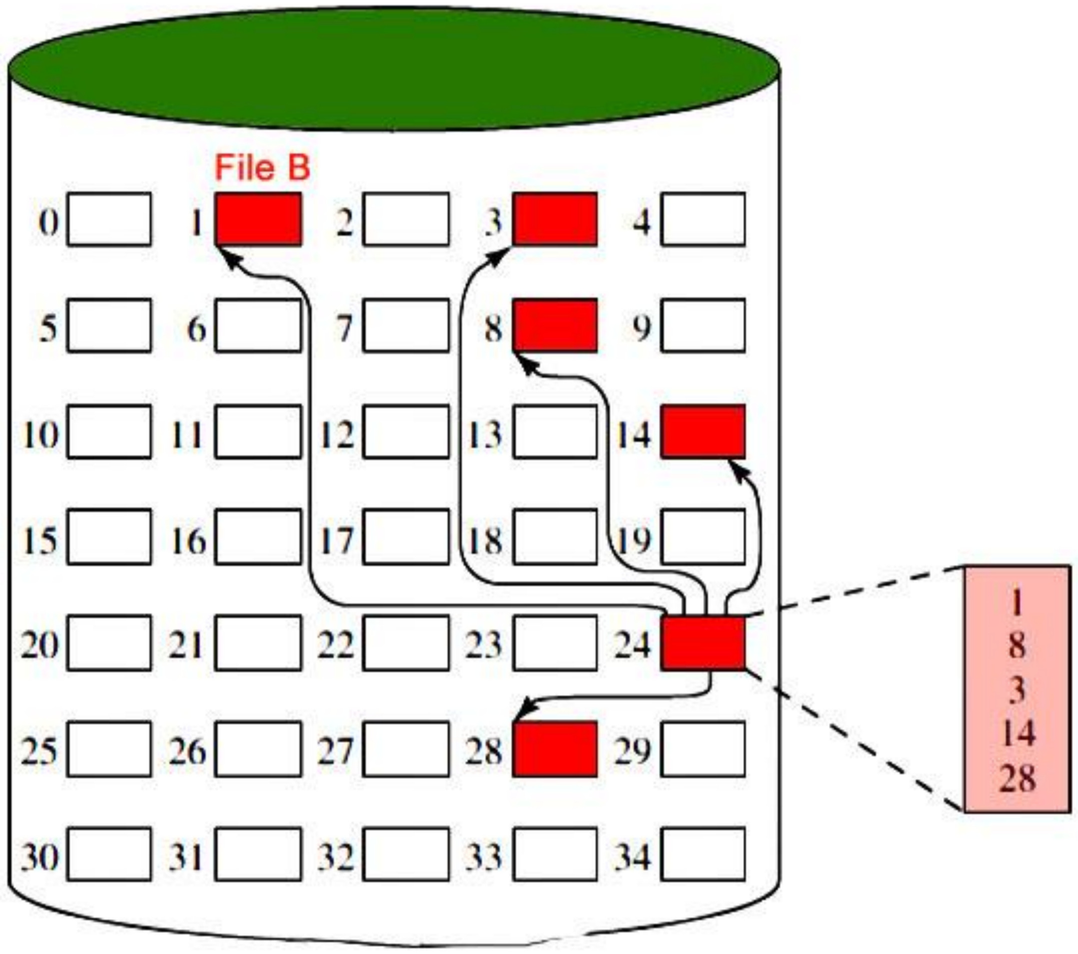
Allocation is on an individual block basis. Each block contains a pointer to the next block in the chain. Again the file table needs just a single entry for each file, showing the starting block and the length of the file. Although pre-allocation is possible, it is more common simply to allocate blocks as needed. Any free block can be added to the chain. The blocks need not be continuous. Increase in file size is always possible if free disk block is available. 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.

Disadvantage –

- 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 supports only the sequential access of files.

3. Indexed Allocation –

It addresses many of the problems of contiguous and chained allocation. 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. Allocation may be on the basis of fixed-size blocks or variable-sized blocks. Allocation by blocks eliminates external fragmentation, whereas allocation by variable-size blocks improves locality. This allocation technique supports both sequential and direct access to the file and thus is the most popular form of file allocation.



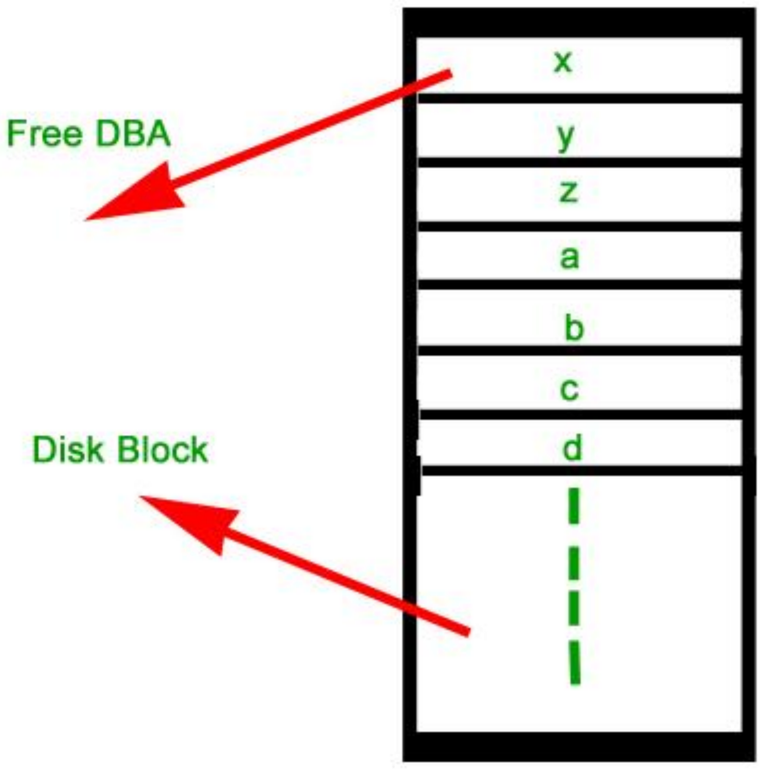
File allocation table

File name	Index block
●●●	●●●
File B	24
●●●	●●●

Disk Free Space Management :

Just as the space that is allocated to files must be managed ,so the space that is not currently allocated to any file must be managed. To perform any of the file allocation techniques,it is necessary to know what blocks on the disk are available. Thus we need a disk allocation table in addition to a file allocation table.The following are the approaches used for free space management.

1. **Bit Tables** : This method uses a vector containing one bit for each block on the disk. Each entry for a 0 corresponds to a free block and each 1 corresponds to a block in use.
   For example: 00011010111100110001
   In this vector every bit correspond to a particular block and 0 implies that, that particular block is free and 1 implies that the block is already occupied. A bit table has the advantage that it is relatively easy to find one or a contiguous group of free blocks. Thus, a bit table works well with any of the file allocation methods. Another advantage is that it is as small as possible.
2. **Free Block List** : In this method, each block is assigned a number sequentially and the list of the numbers of all free blocks is maintained in a reserved block of the disk.

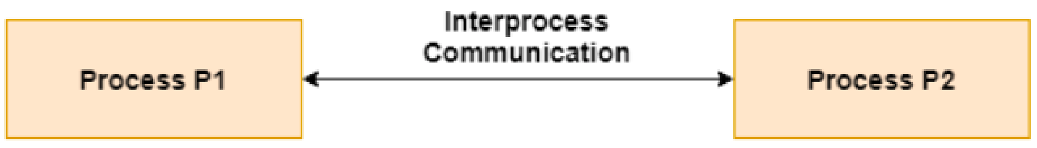


What is Interprocess Communication?

Computer EngineeringMCAOperating System

Interprocess communication is the mechanism provided by the operating system that allows processes to communicate with each other. This communication could involve a process letting another process know that some event has occurred or the transferring of data from one process to another.

A diagram that illustrates interprocess communication is as follows −



Synchronization in Interprocess Communication

Synchronization is a necessary part of interprocess communication. It is either provided by the interprocess control mechanism or handled by the communicating processes. Some of the methods to provide synchronization are as follows −

- **Semaphore**

  A semaphore is a variable that controls the access to a common resource by multiple processes. The two types of semaphores are binary semaphores and counting semaphores.

- **Mutual Exclusion**

  Mutual exclusion requires that only one process thread can enter the critical section at a time. This is useful for synchronization and also prevents race conditions.

- **Barrier**

  A barrier does not allow individual processes to proceed until all the processes reach it. Many parallel languages and collective routines impose barriers.

- **Spinlock**

  This is a type of lock. The processes trying to acquire this lock wait in a loop while checking if the lock is available or not. This is known as busy waiting because the process is not doing any useful operation even though it is active.

Approaches to Interprocess Communication

The different approaches to implement interprocess communication are given as follows −

- **Pipe**

  A pipe is a data channel that is unidirectional. Two pipes can be used to create a two-way data channel between two processes. This uses standard input and output methods. Pipes are used in all POSIX systems as well as Windows operating systems.

- **Socket**

  The socket is the endpoint for sending or receiving data in a network. This is true for data sent between processes on the same computer or data sent between different computers on the same network. Most of the operating systems use sockets for interprocess communication.

File

A file is a data record that may be stored on a disk or acquired on demand by a file server. Multiple processes can access a file as required. All operating systems use files for data storage.

- **Signal**

  Signals are useful in interprocess communication in a limited way. They are system messages that are sent from one process to another. Normally, signals are not used to transfer data but are used for remote commands between processes.

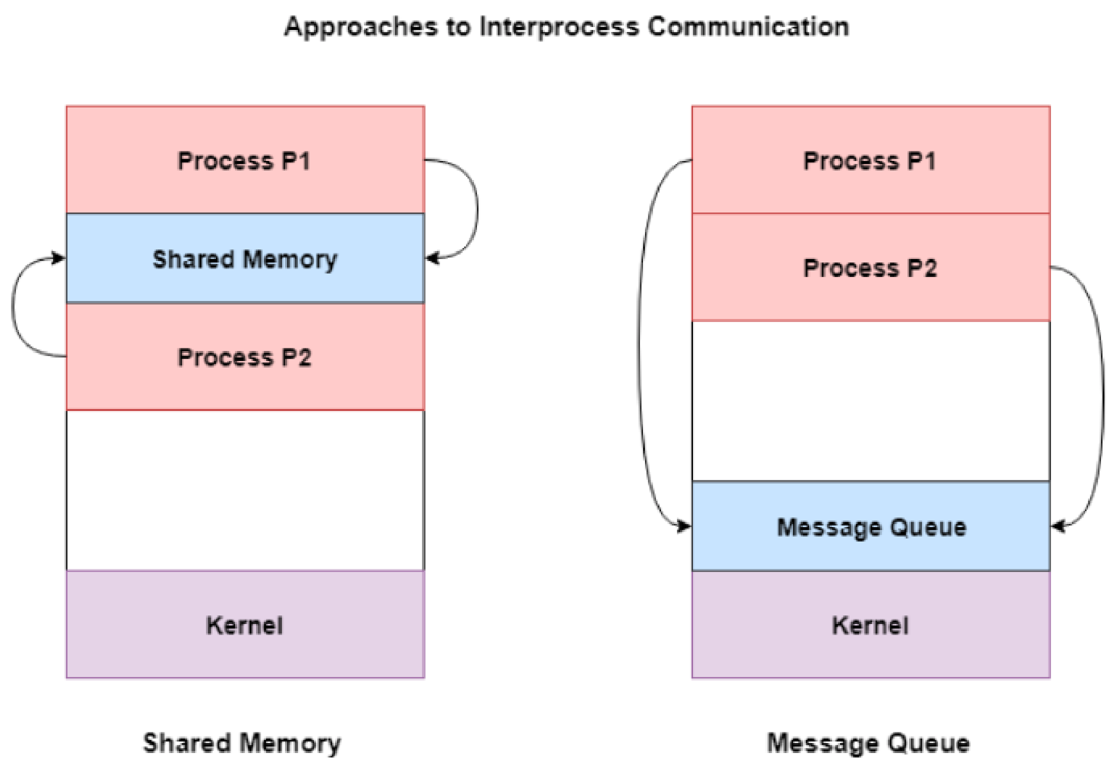
- **Shared Memory**

  Shared memory is the memory that can be simultaneously accessed by multiple processes. This is done so that the processes can communicate with each other. All POSIX systems, as well as Windows operating systems use shared memory.

- **Message Queue**

  Multiple processes can read and write data to the message queue without being connected to each other. Messages are stored in the queue until their recipient retrieves them. Message queues are quite useful for interprocess communication and are used by most operating systems.

A diagram that demonstrates message queue and shared memory methods of interprocess communication is as follows −


Approaches to Interprocess Communication
Process P1
Shared Memory
Process P2
Kernel
Shared Memory
Process P1
Process P2
Message Queue
Kernel
Message Queue

Case study: