



**S.Y.B.Sc. (C. S.)**  
**SEMESTER - III (CBCS)**

**OPERATING SYSTEM**

**SUBJECT CODE: USCS303**

**Dr. Suhas Pednekar**

Vice Chancellor

University of Mumbai, Mumbai

**Prof. Ravindra D. Kulkarni**

Pro Vice-Chancellor,

University of Mumbai

**Prof. Prakash Mahanwar**

Director,

IDOL, University of Mumbai

**Programme Co-ordinator**

**: Shri Mandar Bhanushe**

Head, Faculty of Science and Technology IDOL,  
University of Mumbai – 400098

**Course Co-ordinator**

**: Mr Sumedh Shejole**

Assistant Professor,  
IDOL, University of Mumbai- 400098

**Editor**

**: Ms. Priya Jadhav,**

Assistant Professor,  
N. G. Acharya & D. K. Marathe College,  
Chembur, Mumbai, Maharashtra 400071

**Course Writers**

**: Dr. Ninad More**

Chhatrapati Shivaji Maharaj University,  
Panvel, Navi Mumbai

**: Mr. Santosh Patilshirke**

SJRS College  
Kalwa, Thane, Maharashtra 400605

**: Ms. Rashmi Gera**

Pt. Jawahar Lal Nehru Govt. College,  
Faridabad

August 2022, Print - 1

**Published by : Director,**

Institute of Distance and Open Learning,

University of Mumbai,

Vidyanagari, Mumbai - 400 098.

**DTP composed and Printed by: Mumbai University Press**

# CONTENTS

| Unit No. | Title                         | Page No. |
|----------|-------------------------------|----------|
| 1.       | Operating System.....         | 1        |
| 2.       | Process Synchronization ..... | 55       |
| 3.       | Main Memory .....             | 64       |
| 4.       | Raspberry PI Interfaces.....  | 89       |
| 5        | File System.....              | 105      |

**S.Y.B.Sc. (C. S.)**  
**SEMESTER - III (CBCS)**  
**OPERATING SYSTEM**  
**SYLLABUS**

|                                                                                                                                                                                                                                                                                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |            |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|
| <b>Course:</b><br><b>USCS303</b>                                                                                                                                                                                                                                                          | <b>TOPICS (Credits : 02 Lectures/Week:03)</b><br><b>Operating System</b>                                                                                                                                                                                                                                                                                                                                                                                                               |            |
| <b>Objectives:</b><br>Learners must understand proper working of operating system. To provide a sound understanding of Computer operating system, its structures, functioning and algorithms.                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |            |
| <b>Expected Learning Outcomes:</b><br><div><div>1.</div><div>To provide a understanding of operating system, its structures and functioning</div></div> <div><div>2.</div><div>Develop and master understanding of algorithms used by operating systems for various purposes.</div></div> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |            |
| <b>Unit I</b>                                                                                                                                                                                                                                                                             | <b>Introduction and Operating-Systems Structures:</b> Definition of Operating system, Operating System’s role, Operating-System Operations, Functions of Operating System, Computing Environments<br><b>Operating-System Structures:</b> Operating-System Services, User and Operating-System Interface, System Calls, Types of System Calls, Operating-System Structure<br><b>Processes:</b> Process Concept, Process Scheduling, Operations on Processes, Interprocess Communication | <b>15L</b> |

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |            |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | <b>Threads:</b> Overview, Multicore Programming, Multithreading Models                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |            |
| <b>Unit II</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | <p><b>Process Synchronization:</b> General structure of a typical process, race condition, The Critical-Section Problem, Peterson's Solution, Synchronization Hardware, Mutex Locks, Semaphores, Classic Problems of Synchronization, Monitors</p> <p><b>CPU Scheduling:</b> Basic Concepts, Scheduling Criteria, Scheduling Algorithms (FCFS, SJF, SRTF, Priority, RR, Multilevel Queue Scheduling, Multilevel Feedback Queue Scheduling), Thread Scheduling</p> <p><b>Deadlocks:</b> System Model, Deadlock Characterization, Methods for Handling Deadlocks, Deadlock Prevention, Deadlock Avoidance, Deadlock Detection, Recovery from Deadlock</p>                                                              | <b>15L</b> |
| <b>Unit III</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | <p><b>Main Memory:</b> Background, Logical address space, Physical address space, MMU, Swapping, Contiguous Memory Allocation, Segmentation, Paging, Structure of the Page Table</p> <p><b>Virtual Memory:</b> Background, Demand Paging, Copy-on-Write, Page Replacement, Allocation of Frames, Thrashing</p> <p><b>Mass-Storage Structure:</b> Overview, Disk Structure, Disk Scheduling, Disk Management</p> <p><b>File-System Interface:</b> File Concept, Access Methods, Directory and Disk Structure, File-System Mounting, File Sharing</p> <p><b>File-System Implementation:</b> File-System Structure, File-System Implementation, Directory Implementation, Allocation Methods, Free-Space Management</p> | <b>15L</b> |
| <p><b>Textbook(s):</b></p> <ol style="list-style-type: none"> <li>1. Abraham Silberschatz, Peter Galvin, Greg Gagne, Operating System Concepts, Wiley, 8<sup>th</sup> Edition</li> </ol> <p><b>Additional Reference(s):</b></p> <ol style="list-style-type: none"> <li>1. Achyut S. Godbole, Atul Kahate, Operating Systems, Tata McGraw Hill</li> <li>2. Naresh Chauhan, Principles of Operating Systems, Oxford Press</li> <li>3. Andrew S Tanenbaum, Herbert Bos, Modern Operating Systems, 4e Fourth Edition, Pearson Education, 2016</li> </ol> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |            |



# OPERATING SYSTEM

## Unit Structure

- 1.0 Objectives of Operating System
- 1.1 Introduction to Operating Systems
- 1.2 Introduction and Operating-Systems Structures
  - 1.2.1 Definition of Operating System
  - 1.2.2 Operating System Role's
  - 1.2.3 Operating system operations
  - 1.2.4 Function of Operating System
  - 1.2.5 Computing Environments
  - 1.2.6 Types of operating system
- 1.3 Operating-System Structures
  - 1.3.1 Operating System Services
  - 1.3.2 User and Operating system interface
  - 1.3.3 System Calls
  - 1.3.4 Types of system calls
  - 1.3.5 Operating-System Structure
- 1.4 Processes
  - 1.4.1 Open-Source Operating Systems
  - 1.4.2 Mobile OS (Operating System
  - 1.4.3 Booting Process of Operating System
  - 1.4.4 Components of Operating System
  - 1.4.5 Processes
  - 1.4.6 Process State
  - 1.4.7 Process Control Block
  - 1.4.8 Process Scheduling
  - 1.4.9 Operations on processes
  - 1.4.9 Inter-process communication
- 1.5 Threads
  - 1.5.1 Types of Thread
  - 1.5.2 Multithreading Models
  - 1.5.3 Multicore Programming
- 1.6 Conclusion
- 1.7 Summary
- 1.8 Exercise
- 1.9 References:

---

## 1.0 Objectives of Operating System

---

- The objectives of the operating system are –
- To make the computer system convenient to use in an efficient manner.
- To hide the details of the hardware resources from the users.
- To provide users a convenient interface to use the computer system.
- To act as an intermediary between the hardware and its users, making it easier for the users to access and use other resources.
- To manage the resources of a computer system.
- To keep track of who is using which resource, granting resource requests, and mediating conflicting requests from different programs and users.
- To provide efficient and fair sharing of resources among users and programs.

---

## 1.1 Introduction to Operating Systems

---

A computer system has many resources (hardware and software), which may be expected to do a task. The for the most part required resources are I/O gadget, memory, archive additional room, CPU, etc. The functioning structure goes probably as a director of the above resources and assigns them to express tasks and customers, whenever imperative to play out a particular endeavour. Along these lines the operating system is the resource manager for example it can deal with the asset of a PC framework inside. The assets are processor, memory, documents, and I/O device. In straightforward terms, a working framework is an interface between the PC client and the machine.

It is very important for you that each PC should have a working framework to run different projects. The operating system mainly coordinates the use of the hardware among the various system programs and application programs for various users.

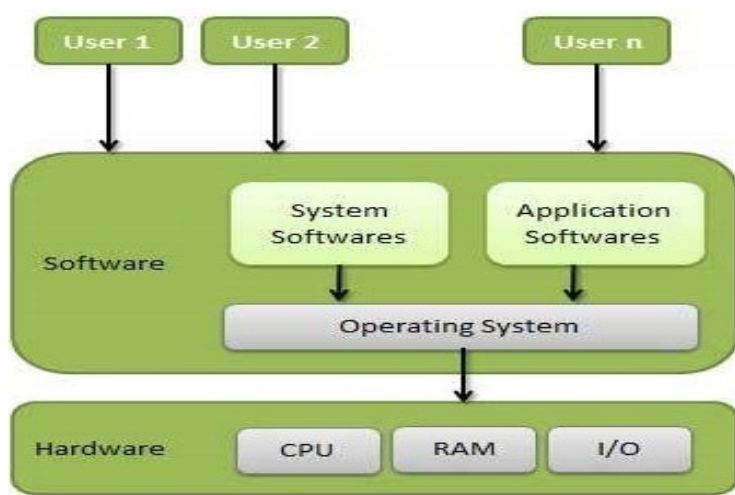
An operating system acts additionally like government means an operating system a working framework plays out no valuable capacity without help from anyone else; however it gives a climate inside which different projects can accomplish valuable work.

### 1.2.1 Definition of Operating System

Operating system is a collection of set of programs which manages all the services of the computer system. It is a program that acts as an intermediary between users of a computer network.

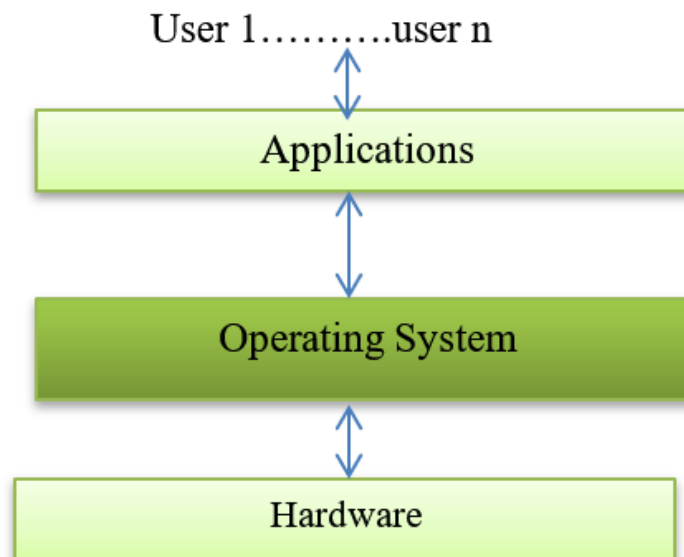
An operating system is a program which manages all the computer hardware's.

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



Operating System

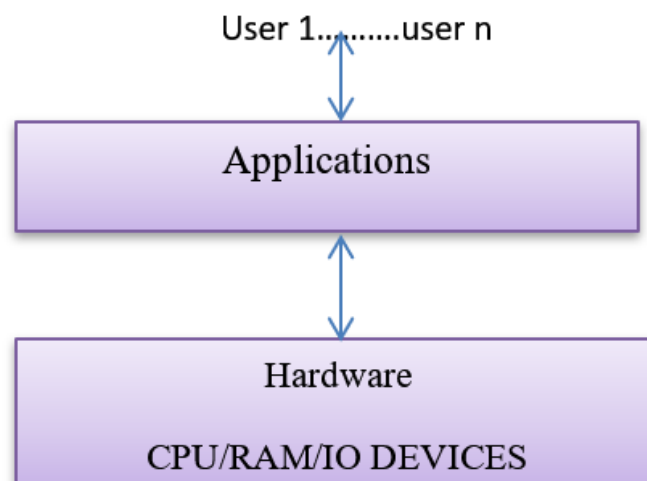
- It is a program that controls the execution of application program.
- Act as an interface between application and hardware.
- Execute user programs and make solving user problems easier
- Make the computer system convenient to use
- Use the computer hardware in an efficient manner



With operating system

Here, User can easily manage the hardware by using operating system.

User 1.....user n



Without Operating System

Here, User cannot easily manage the hardware by using without operating system.

Main objective of an operating system:

- i. Convenience
  - ii. Efficiency
  - iii. Ability to solve the problems
- 1) Convenience: It provides users the services (Processor/Memory/Files and I/O etc.) to execute the programs in a convenient manner.

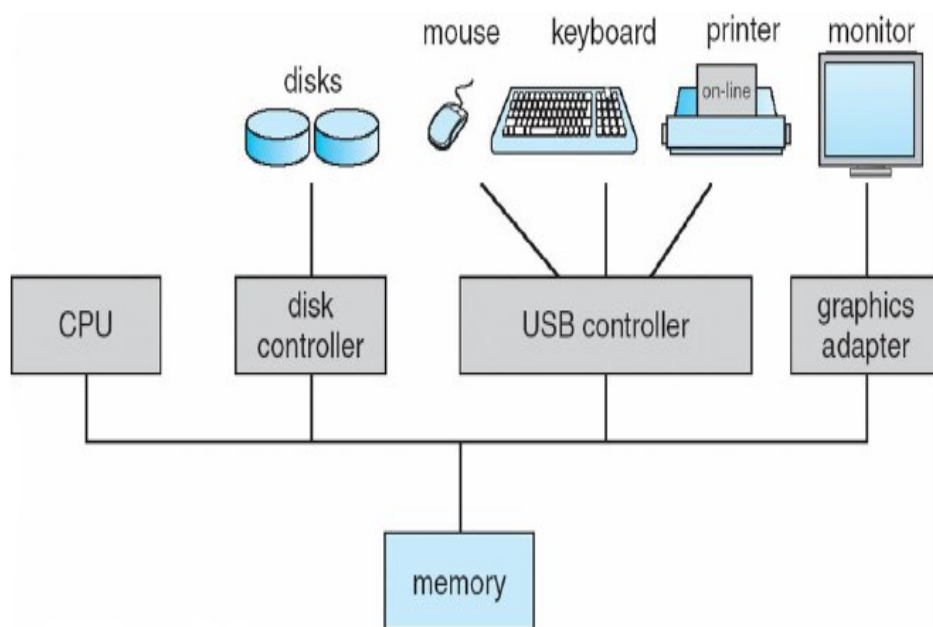
- 2) Efficiency: Operating system allows the computer system resources (Hardware/Software/Data/Network etc.) to be used efficiently.
- 3) Ability to solve the problems: Operating system should be performed in such a way to permit the effective deployment, testing and introduction of new system functions.

### Computer Start-up

- bootstrap program is loaded at power-up or reboot
- Typically stored in ROM or EPROM, generally known as firmware
- Initializes all aspects of system
- Loads operating system kernel and starts execution

### Computer System Organization

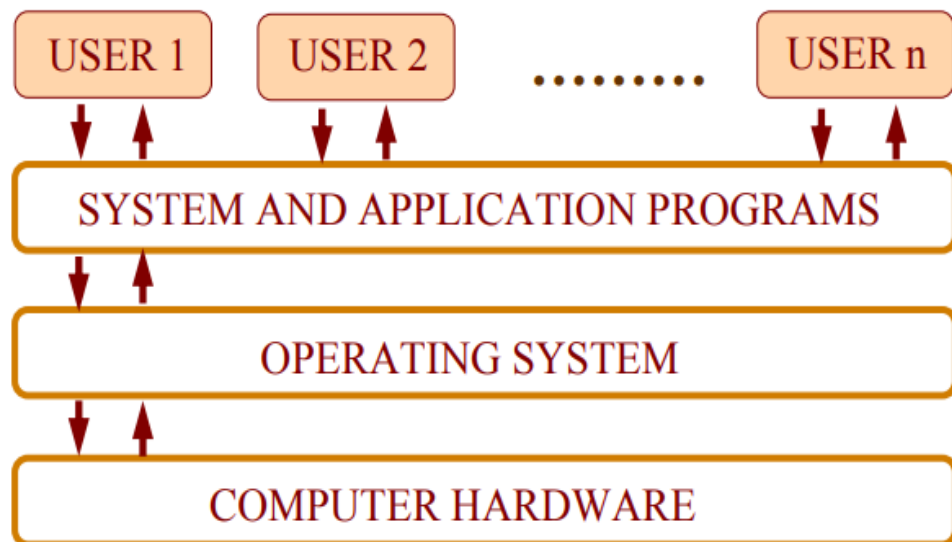
- Computer-system operation
- One or more CPUs, device controllers connect through common bus providing access to shared memory
- Concurrent execution of CPUs and devices competing for memory cycles



Computer System Organization

## 1. View of operating system

- **User view:** The user view of the computer varies by the interface being used. The examples are -windows XP, vista, windows 7 etc. Most computer user sit in the in front of personal computer (pc) in this case the operating system is designed mostly for easy use with some attention paid to resource utilization. Some client sit at a terminal associated with a centralized server/minicomputer. For this situation different clients are getting to similar PC through different terminals. Their clients are share assets and may trade the data. The operating system in this case is designed to maximize resources utilization to assume that all available CPU time, memory and I/O are utilized proficiently and no singular client takes more than his/her reasonable and share. The different clients sit at workstations associated with network of other workstations and servers. These users have dedicated resources but they share resources such as networking and servers like file, compute and print server. Here the operating system is designed to compromise between individual usability and resource utilization.
- **System view:** From the computer point of view the operating system is the program which is most intermediate with the hardware. An operating system has assets as equipment and programming which might be needed to tackle an issue like CPU time, memory space, file storage space and I/O devices and so on. That's why the operating system acts as manager of these resources. Another view of the operating system is it is a control program. A control program manages the execution of user programs to present the errors in proper use of the computer. It is especially concerned of the user the operation and controls the I/O devices.



View of operating system

## **2. History of operating system:**

The First Generation (1940's early 1950's):

- 1) When electronic computers were first introduced in the 1940's.
- 2) Computer without any operating system.
- 3) During this generation computers were generally used to solve simple math calculation.

### **The Second Generation (1955 – 1965)**

- 1) The first OS was introduced in the early 1956's.
- 2) It was GM-NAA I/O was made by (General Motors for IBM'S machine 704)
- 3) OS in the 1956's were called single stream batch processing system.

### **The Third Generation (1966 -1980)**

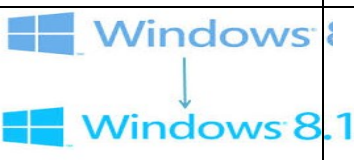
- 1) By the late 1960's OS designers were able to develop the system of multiprogramming in which computer able to perform multiple tasks at the same time.
- 2) Mini PCs beginning with DEC PDP-1(Digital Equipment Corporation's - Programmed Data Processor-1) in 1961.The PDP-1 had just 4K of 18 bit words yet at \$120,000 per machine.
- 3) These PDP'S helped lead to the making of PCs which are made in the fourth era/ generation.

### **The Fourth Generation (1980- Present Day)**

- 1) The fourth generation of operating system saw the creation of personal computing.
- 2) These PCs very similar to the mini computer.
- 3) Windows operating system was created in 1975.
- 4) They introduced the MS-DOS in 1981.
- 5) Windows went on to become the largest operating system used in technology today.
- 6) Along with Microsoft, Apple is the other operating system created in 1980's.

## Different versions of Microsoft Windows

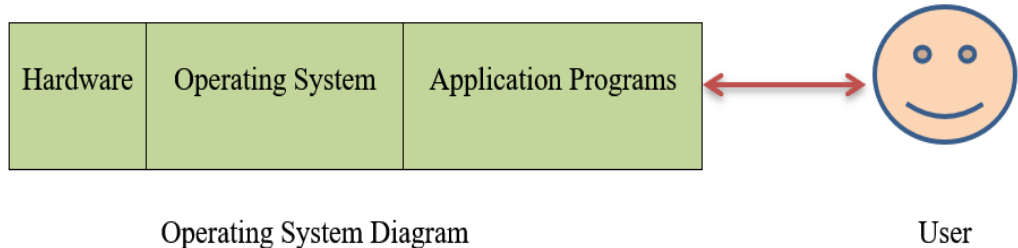
|                                    |                                                                                      |                |
|------------------------------------|--------------------------------------------------------------------------------------|----------------|
| Windows 95                         |    | August 1995    |
| Windows 98                         |    | June 1998      |
| Windows ME -<br>Millennium Edition |    | September 2000 |
| Windows NT<br>31 - 4.0             |  | 1993-1996      |
| Windows 2000                       |  | February 2000  |

|                   |                                                                                     |               |
|-------------------|-------------------------------------------------------------------------------------|---------------|
| Windows XP        |    | October 2001  |
| Windows Vista     |    | November 2006 |
| Windows 7         |    | October, 2009 |
| Windows 8 and 8.1 |   | October, 2012 |
| Windows 10        |  | July 2015     |
| Windows 11        |  | 2021          |

### 1.2.2 Operating System Role's:

An operating system provides an environment in which application software's such as( word processors, Database S/W, Graphic S/W, Spread sheet S/W, Presentation S/W, Web browsers, Microsoft access, Photoshop etc.) can be installed.

So an operating system (Like Windows, Ubuntu, and Linux, CentOS, Debian and so on) provides an environment in which a user can perform a task. Example formulate my information in suitable format i.e. say a word file using an application software like Microsoft word and save it onto my hardware (Hard Disk) in a convenient and efficient manner.



#### Computer-System Operation

- 1) Input /Output devices, and the CPU can execute concurrently
- 2) Every device controller is in charge of a particular device type
- 3) All the device controller has a local buffer
- 4) CPU moves data from/to main memory to/from local buffers
- 5) The Input/Out is from the device to the local buffer of the controller
- 6) The Device controller informs the CPU that it has completed its activity by causing an *interrupt*

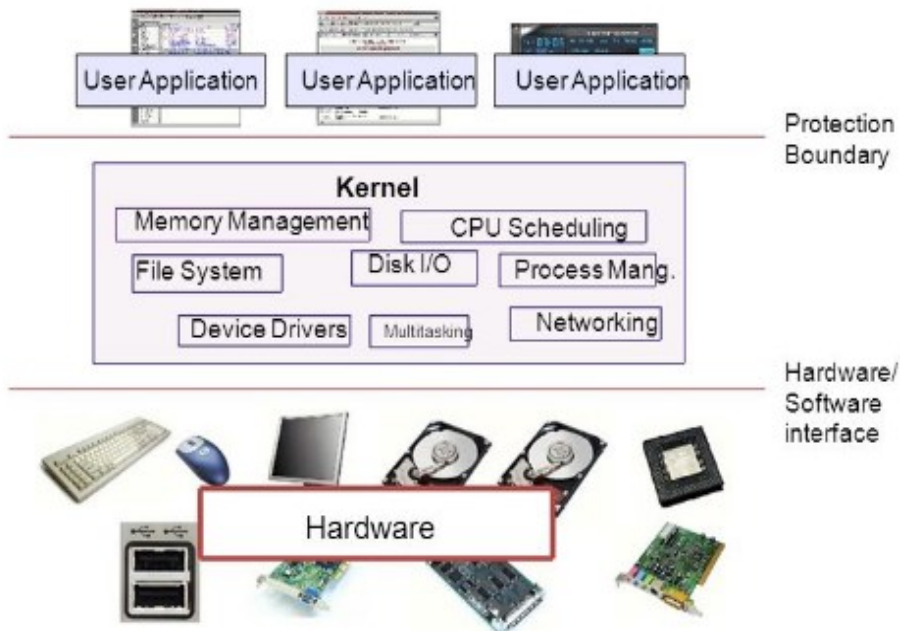
#### Common Functions of Interrupts

- 1) The Interrupt transfers handle to the interrupt service routine generally, through the interrupt vector, which carries the addresses of all the service routines
- 2) The Interrupt architecture must save the address of the interrupted instruction
- 3) The Incoming interrupts are *disabled* while another interrupt is being- processed to prevent a *lost interrupt*. A *trap* is a software-generated interrupt caused either by an error or a user request an OS is interrupt-driven

### 1.2.3 Operating system operations

- 1) Interrupt driven by hardware
- 2) Software error or request creates exception or trap
- 3) Division by zero, request for operating system service
- 4) Other process problems include infinite loop, processes modifying each Other or the operating system

- 5) In the Dual-mode operation allows operating System(OS) to protect itself and to the others system components
- 6) In the User mode and kernel mode
- 7) Mode bit provided by hardware



### Operating system operations

#### Dual-Mode Operation:

In the order to ensure the proper execution of the OS (operating system), we must be able to distinguish between the execution of operating-system code and user-defined code. The methodology taken by most PC frameworks is to give equipment support that permits us to separate among different methods of execution. At any rate, we need two separate methods of activity: client mode and piece mode (likewise called director mode, framework mode, or favoured mode). A bit, called the mode bit, is added to the equipment of the PC to show the current mode: kernel (0) or client (1). With the mode bit, we can recognize an errand that is executed for the benefit of the working framework and one that is executed for the client.

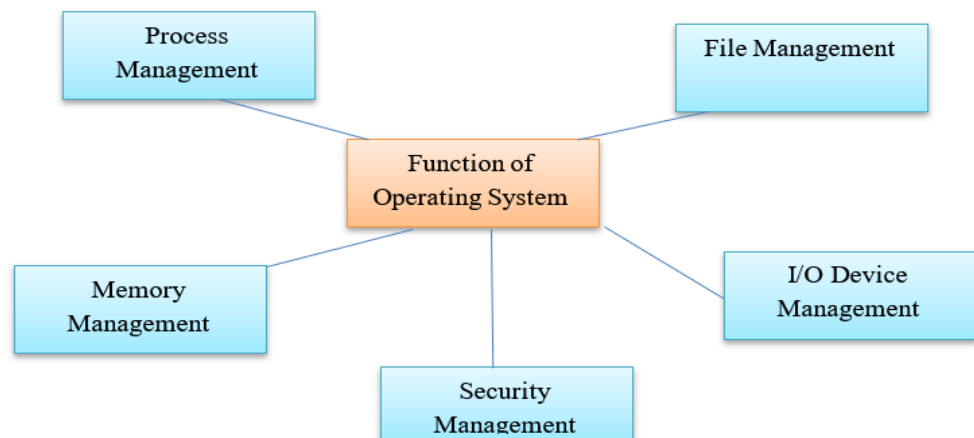
- When the computer system is executing on behalf of a client application, the system is in user mode.
- However, when a user application requests a service from the operating system (via a system call), it must transition from user to kernel mode to full-fill the request. As we shall see, this architectural

enhancement is useful for many other aspects of system operation as well.

- At system boot time, the hardware starts in kernel mode.
- The operating system (OS) is then loaded and starts the user applications in the user mode.
- Whenever a trap or interrupt occurs, the hardware switches from the user mode to kernel mode (that is, changes the state of the mode bit to 0).
- Thus, whenever the operating system gains control of the computer, it is in kernel mode.
- The system always switches to user mode (by setting the mode bit to 1) before passing control to a user program.

The dual mode of operation provides us with the means for protecting the operating system (OS) from deviant clients and wayward clients from one another. We achieve this insurance by assigning a portion of the machine directions that might cause hurt as favoured guidelines. The equipment permits advantaged directions to be executed uniquely in bit mode. In the event that an endeavour is made to execute a favoured guidance in client mode, the equipment doesn't execute the guidance yet rather regards it as unlawful and traps it to the operating system. The instruction to switch to user mode is an example of a privileged instruction. Some other examples include I/O (Input/ Output) control, timer management, and interrupt management. As we shall see throughout the text, there are many additional privileged instructions.

#### 1.2.4 Function of Operating System:



Function of Operating System

**1) Process Management:**

- In process management, operating systems do the management of CPU.
- The operating system takes care of the allotment of CPU to different processes.
- When CPU is free, operating system selects a process from job queue and allocates the CPU to the process.
- When process execution will complete, operation system free the processor and again select another process for execution.
- Selection of process form job queue is done by using various CPU scheduling techniques like FCFS, SJ, ROUND ROBIN etc.

**2) File Management:**

File: A file is a logical related collection of information. File is stored in secondary storage (Magnetic Disk, Tape, Optical Disk etc.). The operating system (OS) manages the files, folders and directory systems on a computer. An operating system does the following activities for file management.

- Creating and deleting files
- Creating and deleting directories
- Operating systems (OS) keep information of filers using file allocation table (FAT)
- Operating system takes care that files are opened with proper access rights (Read /RW).
- File manager of operating system (OS) helps to create, edit, copy, allocate memory to the files and also update the file allocation table FAT.

**3) Memory Management:**

- Memory is the large array of words or bytes each with its own address.
- Main memory is directly accessed by CPU.
- For a program to be executed it must be in the main memory.
- An operating system does the following activities for memory management.
  - 1) Keeps track of primary memory using free space management (Which partition is used and which is free).
  - 2) Allocates the memory to process when it requests it.
  - 3) De-allocates the memory when a process terminated and the same area is allocated to another process.

**4. I/O Device Management:**

- Operating system (OS) manages I/O devices and makes the I/O process effective.

- Operating system (OS) accepts the input from the I/P device (Keyboard) stores it in main memory, ask the CPU to process it and finally provides the result to the devices (Screen /Printer) for output.
- Operating systems (OS) controls all I/O devices attached to computer with the help of small software called device drivers.
- Operating system (OS) track of all devices with the help of I/O controller.
- Operating decides which process gets the device when and for how much time.
- Allocate and de-allocate device in the efficient way.

#### **5. Security Management:**

There are various virus threats which an interface with the normal operations of computer and can be very harmful and result in loss of data or system crashes.

Operating system of a computer has a number of built in tools to protect against security threats including the use of virus scanning utilities and setting up a firewall to block suspicious network activity. Operating system (OS) also helps to prevent unauthorized access to programs and data by use of password in user login etc.

### **1.2.5 Computing Environments**

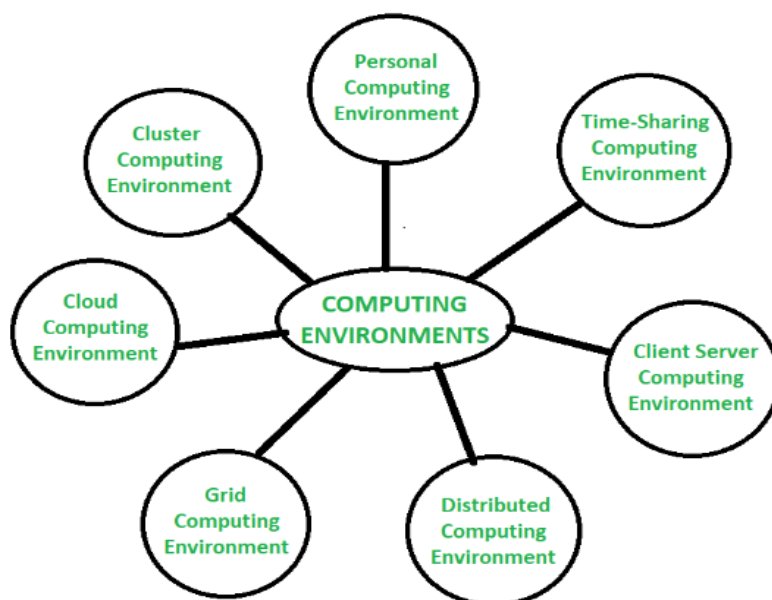
In the world of technology where every task is performed with help of computers, these computers have become one part of human life. Computing is nothing but process of completing a task by using this computer technology and it may involve computer hardware and/or software. But computing uses some form of computer system to manage, process, and communicate information. After getting some idea about computing now let's understand about computing environments.

When a problem is solved by the computer, during that computer uses many devices, arranged in different ways and which work together to solve problems. This constitutes a computing environment where various number of computer devices arranged in different ways to solve different types of problems in different ways. In different computing environments computer devices are arranged in different ways and they exchange information in between them to process and solve problem. One computing environment consists of many computers other computational devices, software and networks that to support processing and sharing information and solving task.

Based on the organization of different computer devices and communication processes there exists multiple types of computing environments. Now let's know about different types of computing environments.

## Types of Computing Environments:

They are the various types of computing environments. They are:



Types of Computing Environments

### 1) Personal Computing Environment :

In individualized computing climate there is an independent machine. Complete program lives on PC and executed there. Diverse independent machines that establish an individualized computing climate are PCs, mobiles, printers, PC frameworks, scanners and so forth that we use at our homes and workplaces.

### 2) Time-Sharing Computing Environment :

In Time Sharing Computing Environment multiple users share system simultaneously. Different users (different processes) are allotted different time slice and processor switches rapidly among users according to it, for example, student listening to music while coding something in an IDE. Windows 95 and later versions, UNIX, IOS, Linux operating systems (OS) are the examples of this time sharing computing environment.

### 3) Client Server Computing Environment :

In client server computing environment two machines are involved i.e., client machine and server machine, sometime same machine also serve as client and server. In this computing environment client requests resource/service and server provides that respective resource/service. A server can provide service to multiple clients at a time and here mainly communication happens through computer network.

#### 4) Distributed Computing Environment :

In a distributed computing environment multiple nodes are connected together using network but physically they are separated. A single task is performed by different functional units of different nodes of distributed unit. Here different programs of an application run simultaneously on different nodes, and communication happens in between different nodes of this system over network to solve task.

#### 5) Grid Computing Environment :

In grid computing environment, multiple computers from different locations work on single problem. In this system set of computer nodes running in cluster jointly perform a given task by applying resources of multiple computers/nodes. It is network of computing environment where several scattered resources provide running environment for single task.

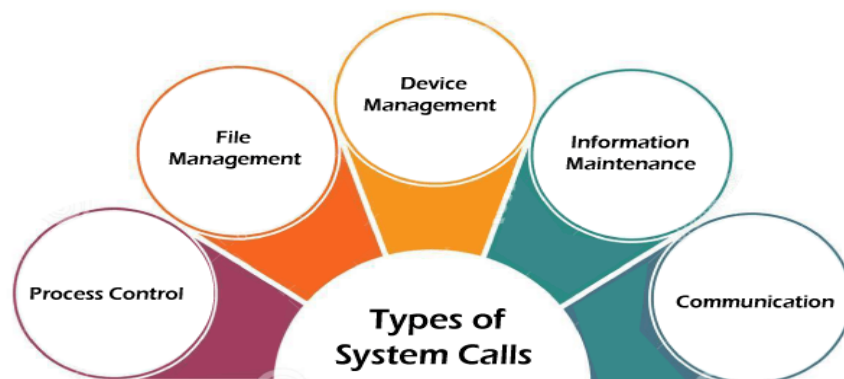
#### 6) Cloud Computing Environment :

In cloud computing environment on demand availability of computer system resources like processing and storage are availed. Here computing is not done in individual technology or computer rather it is computed in cloud of computers where all required resources are provided by cloud vendor. This environment primarily comprised of three services i.e software-as-a-service (SaaS), infrastructure-as-a-service (IaaS), and platform-as-a-service (PaaS).

#### 7) Cluster Computing Environment :

In cluster computing environment cluster performs task where cluster is a set of loosely or tightly connected computers that work together. It is viewed as single system and performs task parallelly that's why also it is similar to parallel computing environment. Cluster aware applications are especially used in cluster computing environment.

### 1.2.6 Types of operating system:

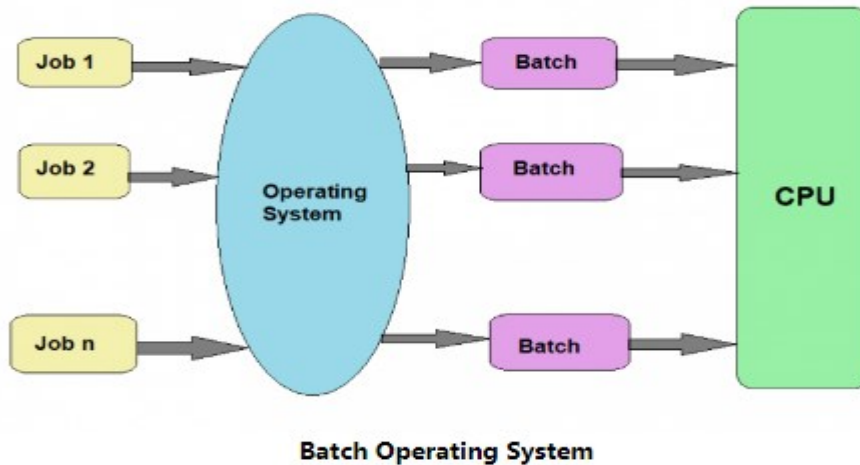


#### 1) Batch processing system

- 2) Time sharing operating system
- 3) Distributed operating system
- 4) Network operating system
- 5) Real time operating system
- 6) Multiprogramming Operating System
- 7) Multiprocessor Operating Systems

**1) Batch processing system:**

The batch operating system will work to submit similar kinds of job together. In this operating system user do not interact directly with our computer system.



Advantage:

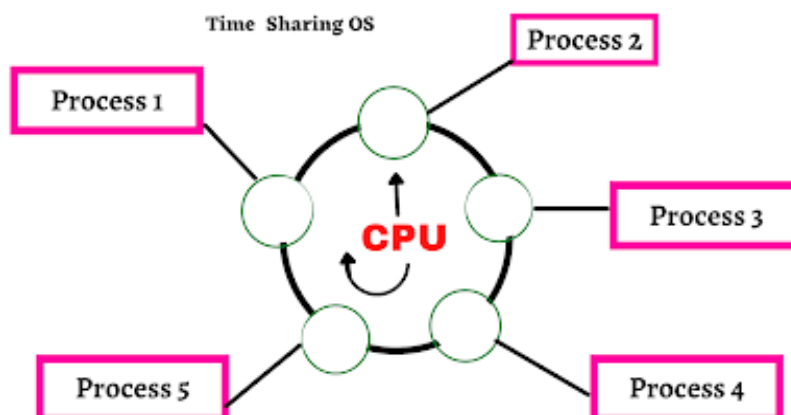
It is useful when we working with large files which can take more time to execute.

Disadvantage:

Once the job is submitted user did not have any interaction with it.

**2) Time sharing operating system:**

The time sharing operating system works with time sharing concept. Here CPU will provide a same time period to each and every process to complete its task as soon as possible whether it is a long process or short process.



Time sharing operating system

Advantage:

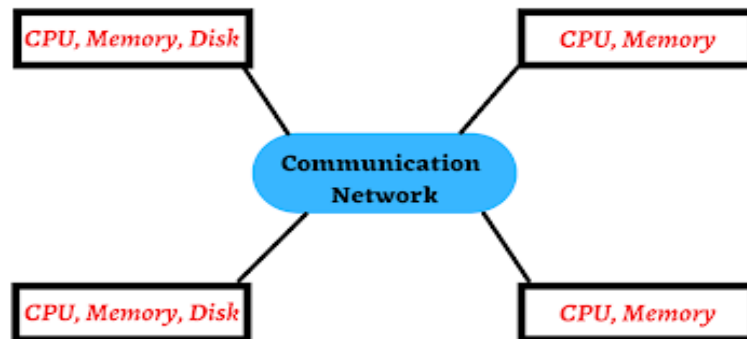
If the process of a task is completed then the time between the other task increases

Disadvantage:

In this operating system every process having higher priority will not get the chance to be executed first priority because in time sharing operating system given equal opportunity to each process.

### 3) Distributed operating system

When many computers are interconnected each other through a network for the purpose of sharing their task then it is called distributed operating system.



Advantage:

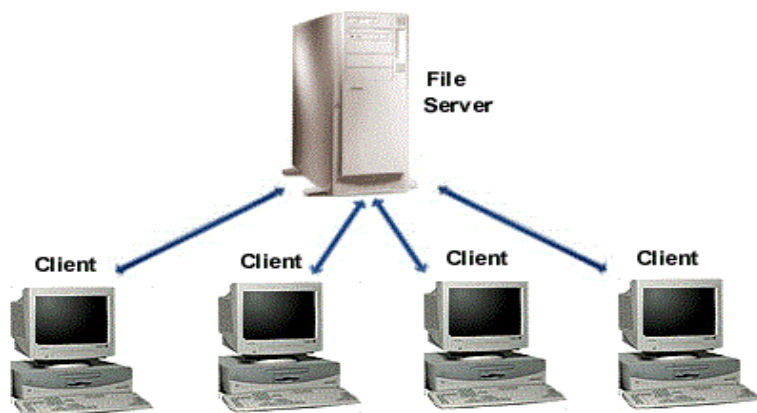
If a node is overused the distributed operating system shares that load to other node of the network.

Disadvantage:

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

### 4) Network operating system:

Network operating system has a server that connects many client computers. So we can easily share own files, resources and many more from the server machine to all machine which are connected through a server.



Network operating system

Advantage:

In network operating system, new technologies software Up gradation and hardware can be easily integrated into the computer system.

Disadvantage:

High cost of server and regular maintenance is required.

### 5) Real time operating system:

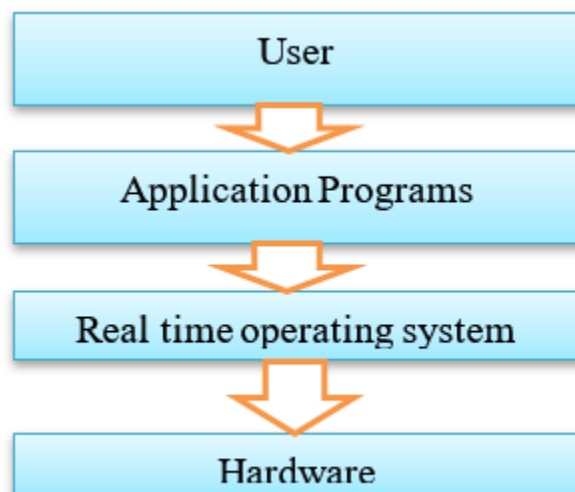
Real time operating system is very useful where we required a quick response. Example is missile system. In this operating system CPU provides maximum offers to its tasks with quick response.

Advantage:

The real time operating system provide a quick response hence, generally used in scientific engineering, NASA and many more organizations.

Disadvantage:

This operating system is very costly.



Real time operating system

### 6) Multiprogramming Operating System

This kind of OS is utilized to execute more than one position at the same time by a solitary processor. It expands CPU usage by getting

sorted out positions so the CPU consistently has one task to execute. The idea of multiprogramming is portrayed as follows:

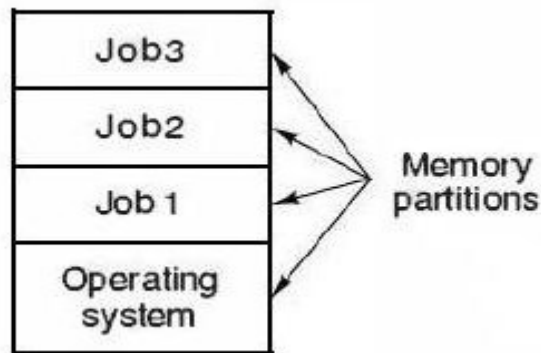
- All the positions that enter the framework are put away in the work pool (in plate). The working framework stacks a bunch of occupations from work pool into primary memory and starts to execute.
- during execution, the work might need to sit tight for some undertaking, like an I/O activity, to finish. In a multiprogramming framework, the working framework just changes to another work and executes. At the point when that work needs to stand by, the CPU is changed to another work, etc.
- When the primary occupation completes the process of pausing and it gets the CPU back.
- as long as no less than one occupation needs to execute, the CPU is rarely inactive. Multiprogramming working frameworks utilize the instrument of occupation planning and CPU booking.

### **Benefits of multiprogramming frameworks**

- 1) The CPU is utilized a large portion of time and never become inactive
- 2) The framework looks quick as every one of the errands runs in equal
- 3) Short time occupations are done quicker than long time tasks
- 4) Multiprogramming frameworks support duplicate clients
- 5) Resources are utilized pleasantly
- 6) Total read time taken to execute program/work diminishes
- 7) Response time is more limited
- 8) In a few applications numerous assignments are running and multiprogramming frameworks better handle these sort of utilizations

### **Disadvantages of multiprogramming systems**

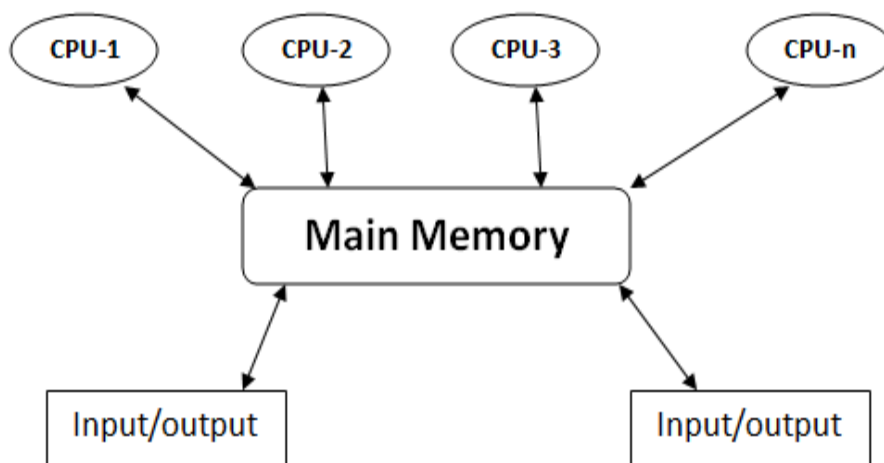
- 1) It is difficult to program a system because of complicated schedule handling
- 2) Tracking all tasks/processes is sometimes difficult to handle
- 3) Due to high load of tasks, long time jobs have to wait long



Multiprogramming Operating System

## 7) Multiprocessor Operating Systems

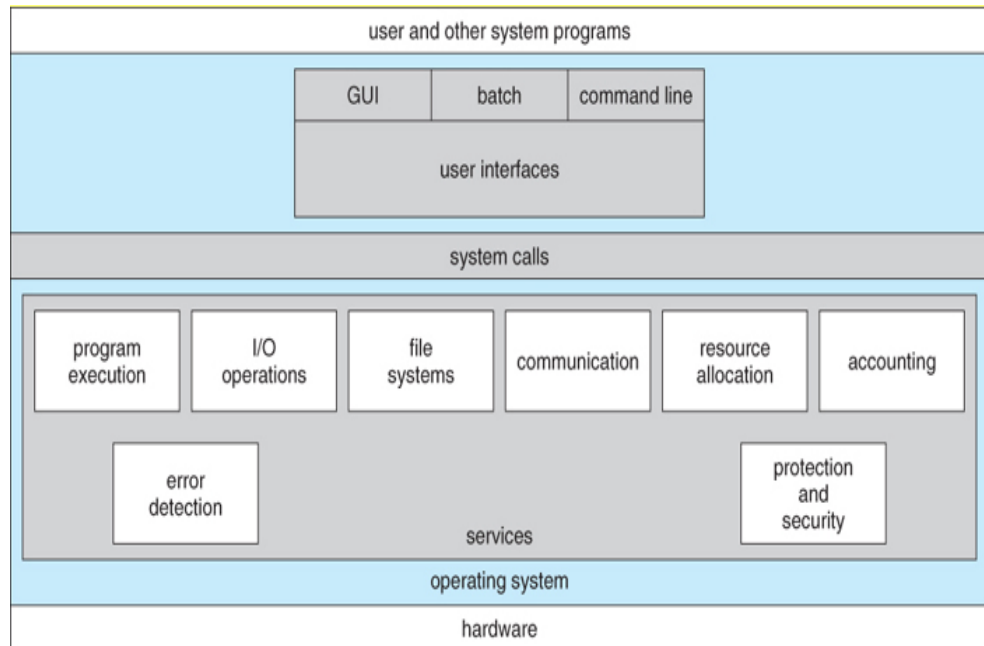
Multiprocessor operating systems are also known as parallel OS or tightly coupled OS. Such operating systems have more than one processor in close communication that sharing the computer bus, the clock and sometimes memory and peripheral devices. It executes multiple jobs at same time and makes the processing faster. Multiprocessor systems have three main advantages:  $\Leftarrow$  Increased throughput: By increasing the number of processors, the system performs more work in less time. The speed-up ratio with  $N$  processors is less than  $N$ .  $\Leftarrow$  Economy of scale: Multiprocessor systems can save more money than multiple single-processor systems, because they can share peripherals, mass storage, and power supplies.  $\Leftarrow$  Increased reliability: If one processor fails to done its task, then each of the remaining processors must pick up a share of the work of the failed processor. The failure of one processor will not halt the system, only slow it down. The ability to continue providing service proportional to the level of surviving hardware is called graceful degradation. Systems designed for graceful degradation are called fault tolerant.



Multiprocessor operating systems

## 1.3 Operating-System Structures

### 1.3.1 Operating System Services:



Operating-System Structures

Operating system services are:

- 1) Program execution
- 2) I/O Operation
- 3) Error Detection
- 4) Resource Allocation
- 5) File Management
- 6) Memory Management
- 7) Communication
- 1) **Program execution:**  
The system must be able to load a program into memory and to run that program.
- 2) **I/O Operation:**  
A running programs my required I/O. It allocates and interacts of various I/O devices when difficult programs are being executed.
- 3) **Error Detection:**  
The operating system needs to be aware of possible error. Error may be occurring in the CPU, memory, hardware, I/O devices and in user programs. It is duty of operating system that appropriate error message whenever an error occurs and takes the suitable action for this error.
- 4) **Resource Allocation:**  
When multiple users logged on the system or multiple jobs are running on the same time then resources must be allocated to each of them.

**5) File management:**

This is service or function of the operating system to keeping the record of files on various storage devices and move all these files from one device to another.

**6) Memory Management**

It includes the assignment of main memory and other storage areas to the system programs, user program and data.

**7) Communication:**

The principle objective of the working frameworks is to give collaboration among client and equipment. One more arrangement of OS capacities exists for guaranteeing the productive activity of the actual framework by means of asset sharing

**Bookkeeping** - To monitor which clients use how a lot and what sorts of PC assets

**Assurance and security** - The proprietors of data put away in a multiuser or organized PC framework might need to control utilization of that data, simultaneous cycles ought not to meddle with one another

Assurance includes guaranteeing that all admittance to framework assets is controlled

Security of the framework from outcasts requires client confirmation, reaches out to protecting outer I/O gadgets from invalid access endeavours

On the off chance that a framework is to be ensured and secure, safety measures should be organized all through it. A chain is just pretty much as solid as its most fragile connection.

**1.3.2 User and Operating system interface**

There are two fundamental approaches for users to interface with operating system.

- 1) Provide a command line interface (CLI) or command interpreter that allows the users directly enter commands that are to be performed by the operating system.
- 2) Allows the user to interface with operating system via a graphical user interface or GUI.
- 3) Some operating system includes the command interpreter in the kernel.
- 4) Others such as windows and UNIX treat the command interpreter as a special program (Like CMD in window and terminal in LINUX).
- 5) On the system with multiple command interpreters to choose from, the interpreter as known as shells.

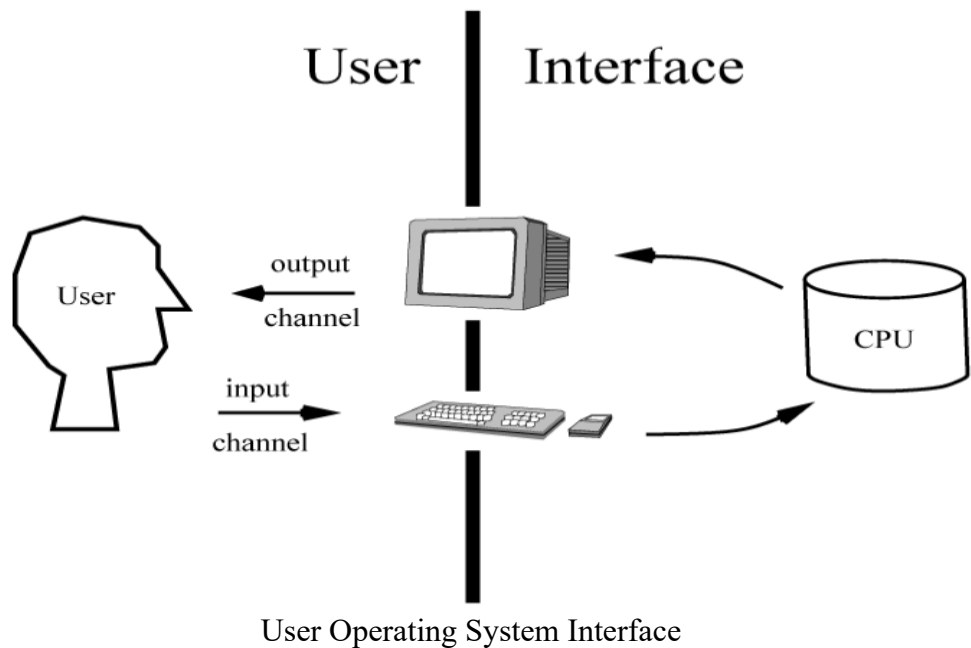
Example:

- 1) Bourne Shell
- 2) C Shell
- 3) Bourne Again shell (BASH)
- 4) Korn shell etc.

So there are two approaches in which actually command interpreter execute the task.

- 1) Code for perform the certain task is included command interpreter itself.
- 2) Command interpreter does not contain any code itself but the code are written in certain programs.

The choice of whether to use a command line or GUI interface is mostly one of the personal preference. System administrators who manages computers and power users who have deep knowledge of a system frequently use in command line interface.



### 1.3.3 System Calls:

**System Calls:** - Programming interface to the services provided by the operating system (OS). Typically written in a high-level language (C or C++)

For the most part got to by programs by means of an undeniable level Application Program Interface (API) as opposed to coordinate framework call useThree most normal APIs are Win32 API for Windows, POSIX API for POSIX-based frameworks (counting for all intents and purposes all adaptations of UNIX, Linux, and Mac OS X), and Java API for the Java virtual machine (JVM)

Why use APIs instead of framework calls?(Note that the framework call names utilized all through this text are conventional)

For performing any operation a user must have to request for a service call which is also known as the SYSTEM CALL or we can say

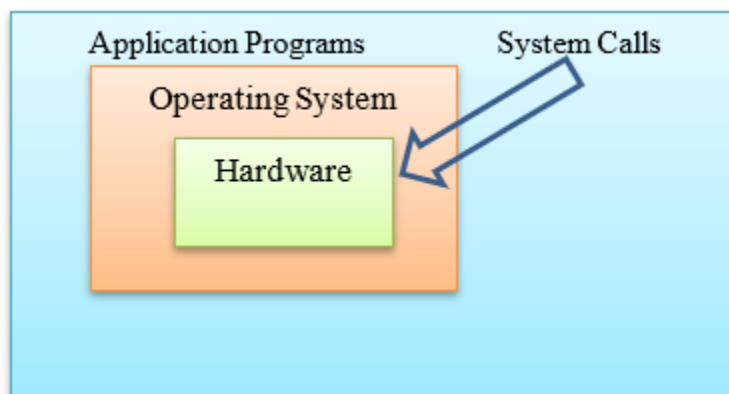
- Programming interface to the services provided by the operating system.
- They are typically written in a high level language (C or C++).
- There are two modes in the operation of system which user more or system mode.
  - 1) In user mode: All user processes are executed.
  - 2) In system mode: All privileged operations are executed.

The user program and kernel functions are being executed in their respective space allotted in the main memory partitions.

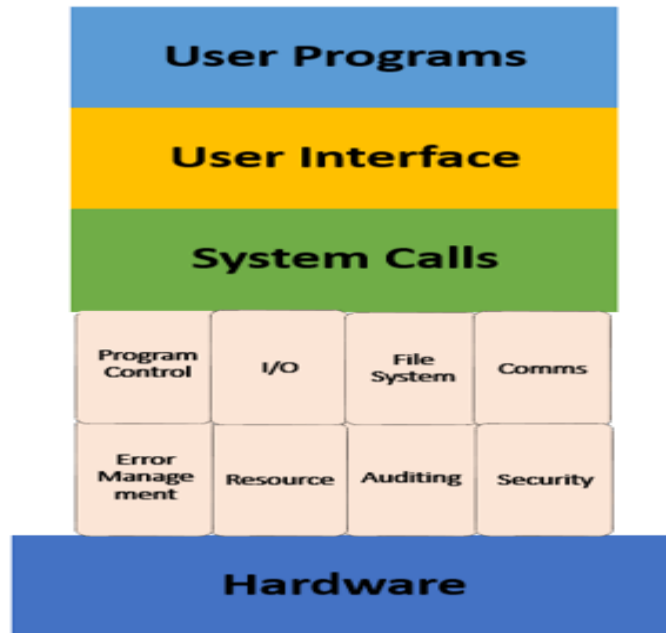
User mode programs need to execute some privileged operations, which are not permitted in user mode functions, user mode interface between user mode and kernel mode. This interface is called system calls. So system calls is an interface between the user programs and the operating system.

System calls expose all kernel functionalities that user mode program's require basically the system call is an instruction that request operating system to perform the desired operation that needs hardware access and other privileged operations.

System call generates an interrupt that causes the operating to gain control of the CPU. The operating system to find out the types of system call and corresponding interrupt handler routine is executed to perform the operation.



System calls are inherently used for security reasons. Due to the use of system calls user is not able to enter into operating system or any others user regions similarly I/O devices are also safe from any misuse by the user. Thus through the system calls kernel, other users programs and I/O devices are safe and secure malicious user programs.



### **Making a system calls:**

Now systems calls are directly available and used in high level languages like C and C++. So it has become easy to use system calls in programs. For a C programmer, system calls are same as calling procedure or function. The difference between a system calls and normal function calls is that system calls enters kernel but a normal function calls does not.

### **Executing the system calls:**

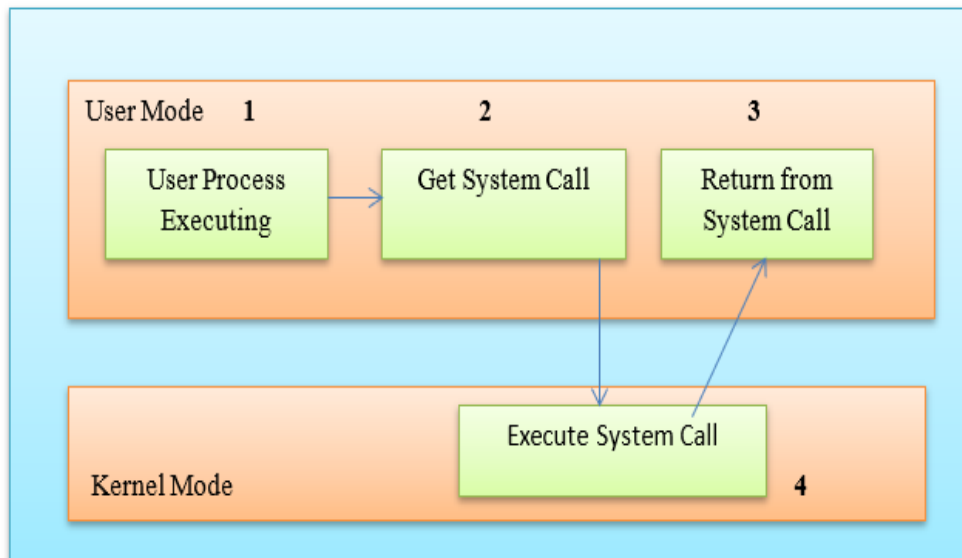
There is a sequence of steps to execute a system calls. For this, there is need to pass various parameters of system calls to operating system. For passing these parameters to operating system three methods are used as follows:

- 1) Register method where in the parameters are stored in the registers of the CPU.
- 2) If parameters are not in number, compared to size of registers, a block of memory is used and address of that block is stored in registers.
- 3) Stack method where in parameters are pushed onto the stack and popped off by the operating system.

### **Sequence in which system calls is executed:**

- 1) In the user program when a system a call is executed, first of all its parameters are pushed onto the stack and later on saved in processor registers.
- 2) The corresponding library procedure for the system is executed.
- 3) There is a partition code for every system call by which kernel identifies which system call function or handler need to be executed. Therefore library procedure places the system call number in the procedure registers.

- 4) Then library procedure traps to the kernel by executing interrupt registers. With this interrupt execution, the user mode switches to kernel mode by loading program status word (PSW register to operating system.



#### 1.3.4 Types of system calls:

##### 1) Process control system calls

A process is a basic entity in the system. The processes in the system need to be created, deleted and aborted. Beside these many operations are required on the processes for their management.

These are some example:

- FORK ( ): Create a process
- EXIT ( ): Terminate a process
- KILL ( ): Terminate a process abnormally
- NICE ( ): Increase a priority of a process

##### 2) File management system calls:

Creation, deletion, opening, closing, reading and writing are some general operation of files. Similarly for organizing files, there is a directory system and there by system calls managing them.

- CREATE ( ): To create a new file
- OPEN ( ): To open a file
- CLOSE ( ): To close a file
- READ ( ): To read a file
- WRITE ( ) : To write a file
- LINK( ): Give another name to a file
- UNLINK( ): Delete a file in a directory
- MKdir( ): Create a new directory

##### 3) Device management system calls:

### **The general commands are:**

- Request of device
- Release of device
- Read and Write operation and so on.

#### **4) Information maintenance system calls:**

Many system calls exist simply for the purpose of transferring information between the user program and the operating system.

- Return current date and time
- Number of current users
- Version number of operating system
- Amount of free memory

Get Time ( )                      Set Date ( )                      Get process ( )  
 Get Date ( )                      Get system data ( )                      Set process ( )  
 Set Time ( )                      Set system data ( )                      Get file ( ) and so on.

#### **5) Communication's system calls**

There is a need for communication among the processes in the system.

General operations are:

- Opening and closing the connection
- Sending and receiving messages
- Reading and writing messages and so on.

Msgsnd( ): Sending a message

Msgrec( ): Receiving a message

These system calls may be related to communication between processes either on the same machine or between processes on different nodes of a network. Thus inter process communication is provided by the operating system through these communication related system calls.

Examples of Windows and UNIX System Calls:

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

---

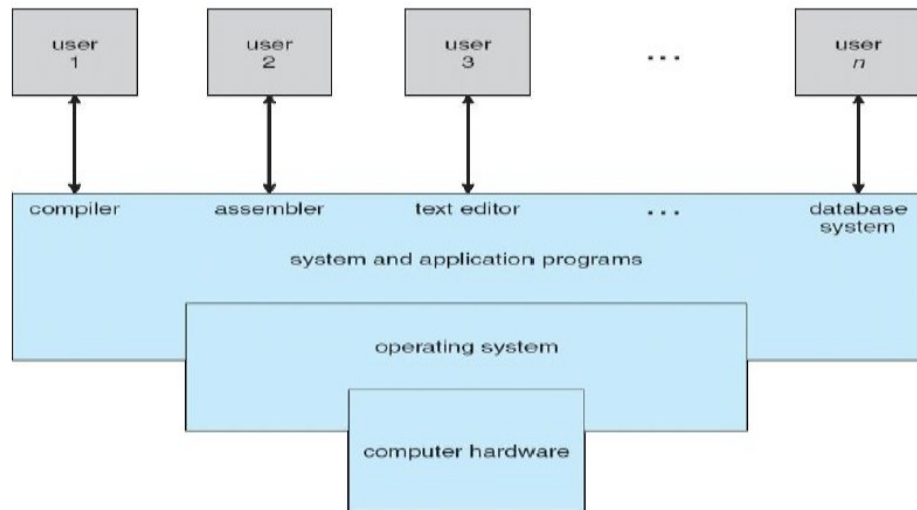
## 1.4 Processes:

---

### 1.4.1 Operating-System Structure

Computer system can be divided into four components

- 1) **Hardware :-** Provides the basic computing resources CPU, memory, Input / Output (I/O) devices
- 2) **Operating system (OS) :-** Controls and coordinates use of hardware among various applications & users
- 3) **Application programs –** define the ways in which the system resources are used to solve the computing problems of the users Word processors, compilers, web browsers, database systems, video games, etc.
- 4) **Users** People, machines, other computers the operating system can be implemented with the help of the various structures. The structure of the operating system (OS) depends mainly on how the various common components of the operating system are interconnected and melded into the kernel. Depending on this we have following structures of the operating system:

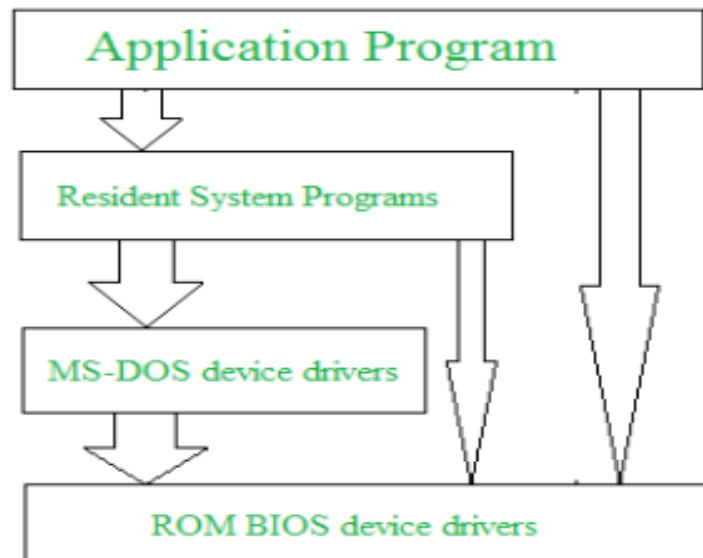


Operating-System Structure

### Simple structure:

Such operating systems do not have well defined structure and are small, simple and limited systems. The interfaces and levels of functionality are not well separated. The MS-DOS is an example of such operating system. In the MS-DOS application programs are able to access the basic input/output (I/O) devices. These types of operating system cause the entire system to crash if one of the user programs fails.

Diagram of the structure of the MS-DOS is shown below.



### Advantages of Simple structure:

- It delivers better application performance because of the few interfaces between the application program and the hardware.
- Easy for kernel developers to develop such an operating system.

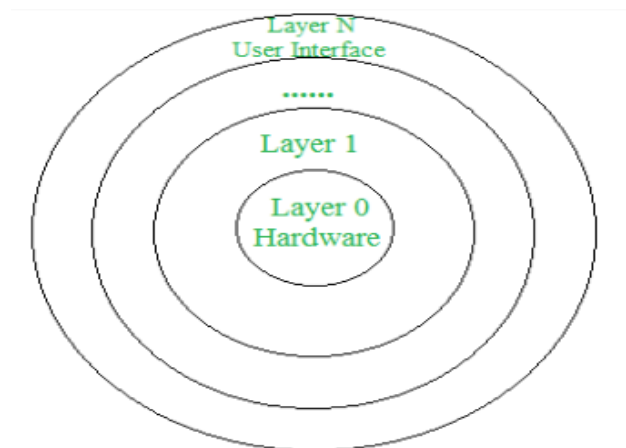
**Disadvantages of Simple structure:**

- The structure is very complicated as no clear boundaries exists between modules.
- It does not enforce data hiding in the operating system.

**Layered structure:**

An OS can be broken into pieces and retain much more control on system. In this structure the OS is broken into number of layers (levels). The bottom layer (layer 0) is the hardware and the topmost layer (layer N) is the user interface. These layers are so designed that each layer uses the functions of the lower level layers only. This simplifies the debugging process as if lower level layers are debugged and an error occurs during debugging then the error must be on that layer only as the lower level layers have already been debugged.

The main disadvantage of this structure is that at each layer, the data needs to be modified and passed on which adds overhead to the system. Moreover careful planning of the layers is necessary as a layer can use only lower level layers. UNIX is an example of this structure.

**Advantages of Layered structure:**

- Layering makes it easier to enhance the operating system as implementation of a layer can be changed easily without affecting the other layers.
- It is very easy to perform debugging and system verification.

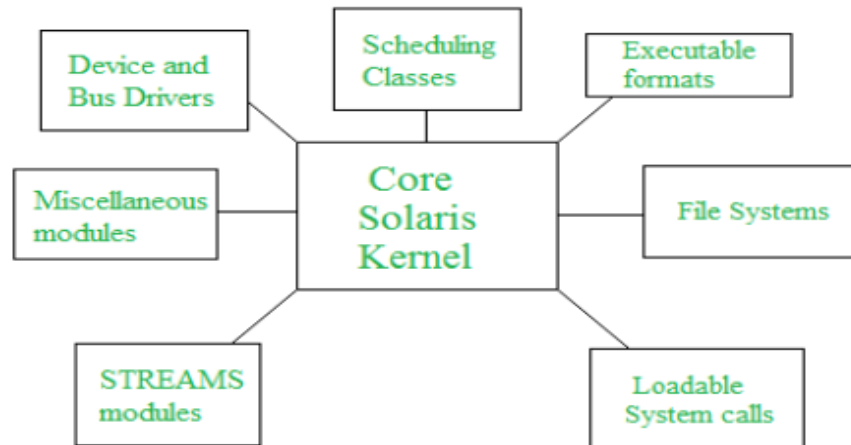
**Disadvantages of Layered structure:**

- In this structure the application performance is degraded as compared to simple structure.
- It requires careful planning for designing the layers as higher layers use the functionalities of only the lower layers.

**Modular structure or approach:**

It is considered as the best approach for an OS. It involves designing of a modular kernel. The kernel has only set of core components and other services are added as dynamically loadable modules to the kernel either during run time or boot time. It resembles layered structure due to the fact that each kernel has defined and protected interfaces but it is more flexible than the layered structure as a module can call any other module.

For example Solaris OS is organized as shown in the figure.



## Open-Source Operating Systems

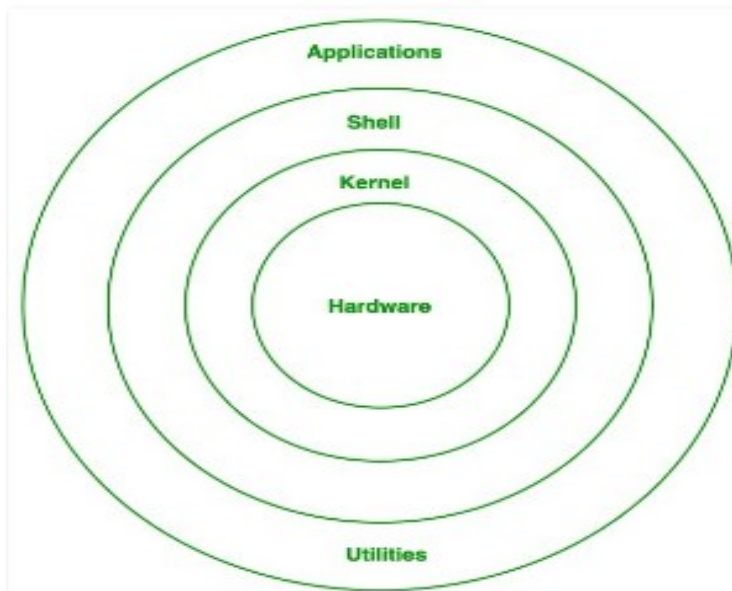
The first Open Source software is made available in 1997. Now there are Open Source alternatives for every Software application irrespective of the industry.

From the very beginning of the 21st-century, technical advancements and innovations lead to the creation of many Open Source Operating Systems. Here is everything you need to know about the Open Source Operating Systems. Open source refers to the computer software or applications where the owners or copyright holders allow the users or third party to see, use and provide the right to modify the source code of the product. An Open-source Operating System is the Operating System in which source code is visible publically and editable.

## Types of Open Source Operating System

Most of the Open Source Operating Systems are Linux based.

- 1) **Linux Kernel** is created by **Linus Torvalds**. It provides the core functions needed for an Operating System like Parcelling of data, processing of memory, and interactions with the computer hardware. Linux is open-source many developers studied the source code and created many supportive plug-ins and operating systems for their needs. Though Linux is the heart of the operating systems, there are also some Open Source Operating Systems that are **not based on Linux**.



- 1) **Linux Distributions** are available and some of the popular Linux distributions are:
- 2) MX Linux
- 3) Manjaro
- 4) Linux Mint
- 5) elementary
- 6) Ubuntu
- 7) Debian
- 8) Solus
- 9) Fedora
- 10) openSUSE
- 11) Deepin

#### **Advantages of Open Source Operating Systems:**

- 1) **Cost-efficient** – Most of the Open Source OS is free. And some of them are available at a very cheap rate than the commercial closed products.
- 2) **Reliable and efficient** – Most of them are monitored by thousands of eyes since the source code is public. So if there is any vulnerability or bugs, they are fixed by the best developers around the world
- 3) **Flexibility**- The great advantage is you can customize it as per your need. And there is creative freedom.

#### **Disadvantages of Open Source Operating Systems:**

- 1) **Security risk** – Though the bugs are identified, there is a risk of attacks as the source code is available to the attackers.

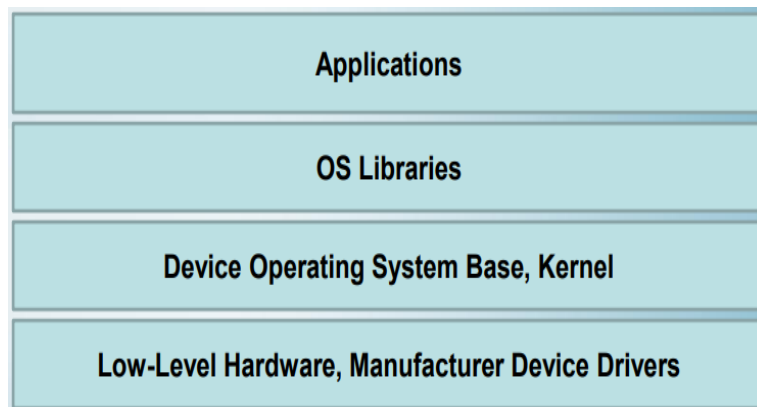
- 2) **Complicated** – It is not user-friendly as the closed ones. You need to have the minimum technical knowledge to use this software
- 3) **No support** – If you meet with the problem, then there is no customer support to help you out.

### 1.4.2 Mobile OS (Operating System)

Design and capabilities of a Mobile OS (Operating System) is very different than a general purpose OS running on desktop machines: – mobile devices have constraints and restrictions on their physical characteristic such as screen size, memory, processing power and etc. – Scarce availability of battery power – Limited amount of computing and communication capabilities.

Thus, they need different types of operating systems depending on the capabilities they support. e.g. a PDA OS is different from a Smartphone OS. • Operating System is a piece of software responsible for management of operations, control, coordinate the use of the hardware among the various application programs, and sharing the resources of a device.

A mobile OS is a software platform on top of which other programs called application programs, can run on mobile devices such as PDA, cellular phones, smartphone and etc.



There are many mobile operating systems. The followings demonstrate the most important ones: –

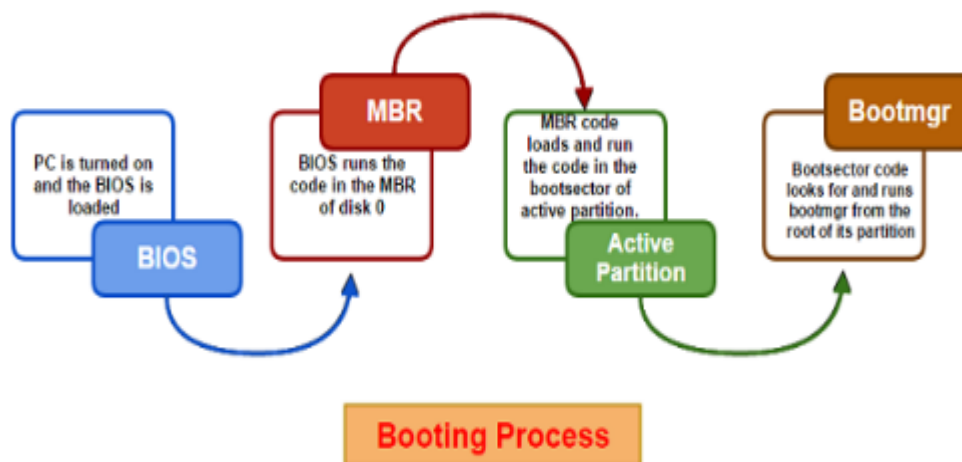
- 1) Java ME Platform –
- 2) Palm OS –
- 3) Symbian OS –
- 4) Linux OS –
- 5) Windows Mobile OS –
- 6) BlackBerry OS –
- 7) iPhone OS –
- 8) Google Android Platform

### 1.4.3 Booting Process of Operating System:

When you start the computer, it is observed that some initial text information is displayed on the screen. This is displayed by the firmware. The booting instructions are stored in ROM (read-only memory). Then the booting process starts. After booting, an operating system gets loaded in the main memory (RAM) of the computer.

Let us understand the complete booting process.

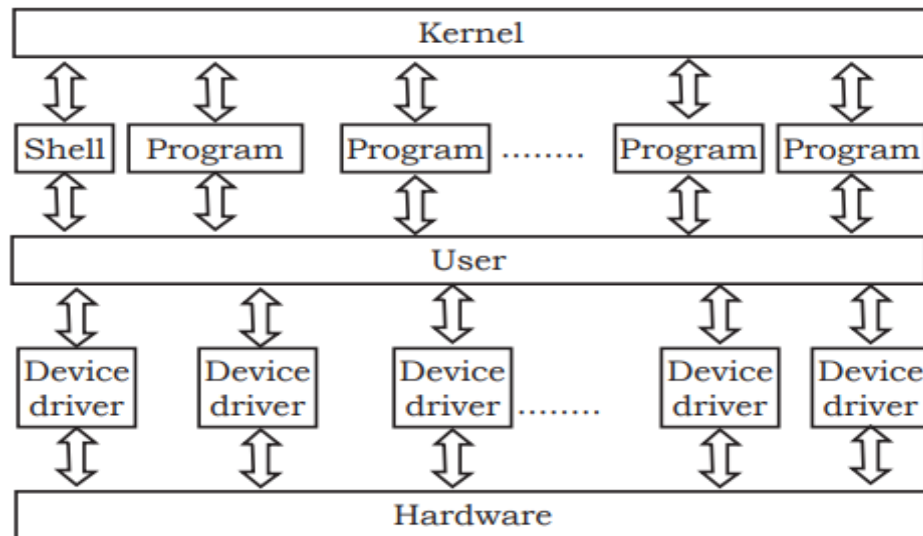
- 1) When you power on the computer, the CPU (central processing unit) activates the BIOS (basic input output system).
- 2) The first program activated is POST (power on selftest). Using the CMOS (complementary metal oxide semiconductor) memory it checks all the hardware and confirms that they are functioning properly.
- 3) After that it reads the MBR (master boot record) in boot drive in accordance with the firmware 'bootstrap loader' which is provided by the computer manufacturer.
- 4) Then the computer loads in the operating system in boot drive to the RAM.
- 5) Once this is performed, the operating system takes over the control of the computer and displays a user interface to the user.



### 1.4.4 Components of Operating System

We identify the operating system by its user interface. The look or initial screen of various operating systems looks different, but architectural view of the various operating systems remains the same. There are essentially three components of operating system as described below:

1. The device driver
2. The kernel
3. The shell



Components of Operating System

- 1) **The Device Driver:**  
This component is close to computer hardware. The device drivers are required for proper functioning of the devices attached to the computer system. These drivers can be installed or uninstalled as and when required. The kernel uses it for operating and controlling.
- 2) **The Kernel:**  
It is the core of the operating system. It performs all the major functions of the operating system. It manages resources, controls program execution, and schedules program execution. It is the main operating system. It detects the new hardware when attached and installs the device driver for it to function properly.
- 3) **The Shell:**  
We identify the operating system by how the shell looks. It provides the user interface to interact with the kernel and hardware. There are two types of user interface— command line interface (CLI) and graphical user interface (GUI) as explained in the Chapter earlier.

#### 1.4.5 Processes:

##### Process Concept:

- An operating system executes a variety of programs:
- Batch system – jobs
- Time-shared systems – user programs or tasks
- Textbook uses the terms *job* and *process* almost interchangeably

Process – a program in execution; process execution must progress in sequential fashion

A process includes:

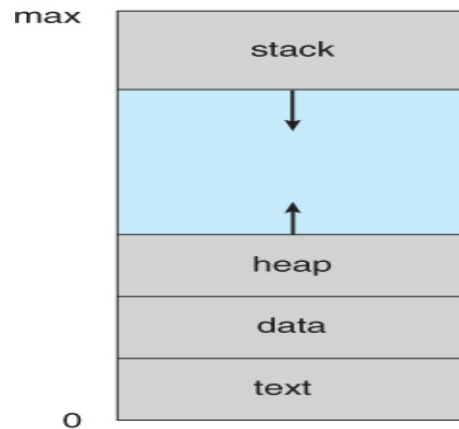
- program counter
- stack
- data section

**Process:** A process is an instance of a program in execution.

In batch operating systems work in terms of “jobs” used. In many advanced process methods are till expressed in terms of jobs, example is Job Scheduling.

The Process:

Process memory is mainly divided into four different categories as shown in figure.



A process in memory

- The text section comprises the compiled program code and read from non-volatile storage when the actual program is launched.
- The data section stores static and global variables.
- The heap is used for dynamic memory allocation and is managed via calls to delete, new, malloc, free, etc.
- The stack is used for mainly in local variables. Space on the stack is reserved for local variables only when they are declared.
- When processes are swapped out of memory and later restored, additional information must also be stored and restored.

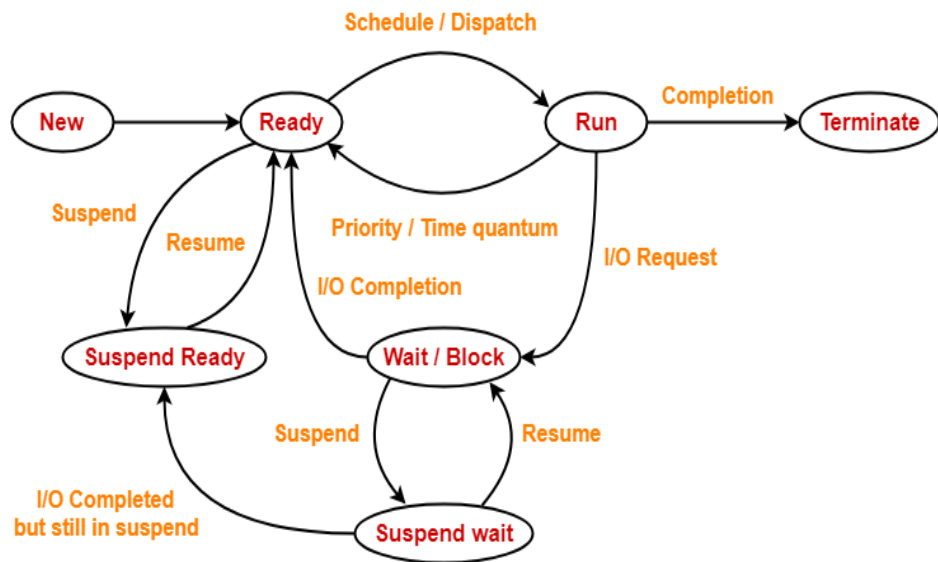
#### 1.4.6 Process State:

Processes may be in one of SIX states as shown in figure below:

- 1) New State
- 2) Ready State
- 3) Running State
- 4) Waiting or Block State
- 5) Suspended Ready State

## 6) Terminated

- New: The process is in the stage of being created.
- Ready State: In first stage, a process moves from new state to ready state after it is loaded into the main memory and this process is ready for execution.
- Running state: A process moves from ready state to run state after it is assigned the CPU for each process for execution.
- Waiting or Block State: The process cannot run at the time because process is waiting for some I/O resources or some event to occur.
- Suspended Ready State: A process moves from ready state to suspend ready state if a process with higher priority has to be executed but the main memory is full.
- Terminated: the process has completed.



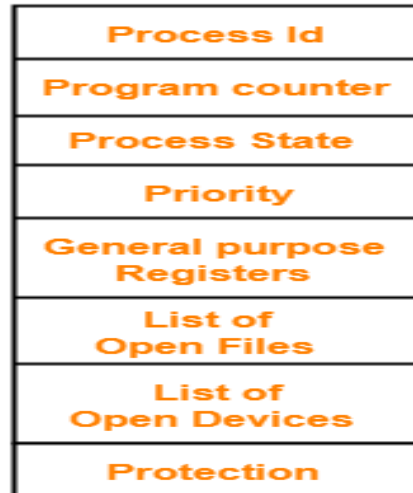
Process State Diagram

| State               | Present in Memory |
|---------------------|-------------------|
| New state           | Secondary Memory  |
| Ready state         | Main Memory       |
| Run state           | Main Memory       |
| Wait state          | Main Memory       |
| Suspend wait state  | Secondary Memory  |
| Suspend ready state | Secondary Memory  |
| Terminate state     | —                 |

### 1.4.7 Process Control Block-

Process Control Block (PCB) is a data structure that stores all information about a particular each process. This all information about a particular each process is required by the CPU while executing the process time.

The Process Control Block diagram of a process looks like-



**Process Control Block (PCB)**

#### 1. Process Id

- Process Id is a unique Id for each process that identifies process of the system uniquely.
- A process Id is assigned to each process during its creation of process.

#### 2. Program Counter

- Before process execution, program counter is initialized with the address of the first instruction of the program.
- After executing a first instruction, value of program counter is automatically incremented to point to the next instruction of process.
- This program counter process repeats till the end of the program.

#### 3. Process State

- Each process goes through different states (process state, running state, waiting state etc.) during its lifetime.
- Process state specifies the current state of the process.

#### 4. Priority

- Process with the very highest priority is allocated the CPU first among all the other processes.

#### 5. General Purpose Registers

- General purpose registers are mainly used to hold the data of process generated during its execution time.
- Each process has its own set of registers which are maintained by its process control block (PCB).

#### 6. List of Open Files

- Each process requires some important files which must be present in the main memory during its execution time.
- Process control block (PCB) maintains a list of files used by the process during its execution.

#### 7. List of Open Devices-

- Process control block (PCB) maintains a list of open devices used by the each process during its execution time.

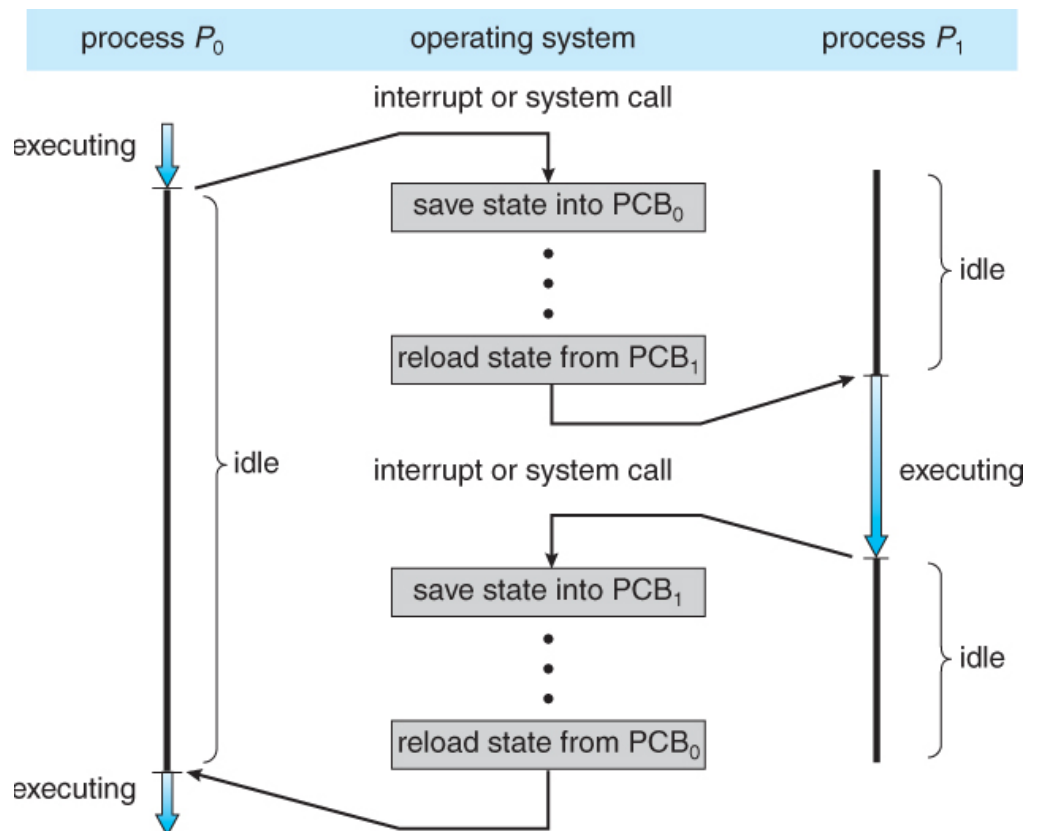


Diagram showing CPU switch from process to process

### 1.4.8 Process Scheduling:

- Schedulers in Operating System are special type of system software.
- They help in scheduling the processes in various ways.
- They are mainly responsible for selecting the jobs to be submitted into the system and deciding which process to run.

There are 3 kinds of schedulers-

- 1) Long-term scheduler
- 2) Short-term scheduler
- 3) Medium-term scheduler

#### 1. Long-term Scheduler-

- Long-term scheduler is also called as Job Scheduler.
- In long term scheduler, it selects a balanced mix of I/O bound and CPU bound processes from the secondary memory such as new state.
- Then, it loads the selected processes into the main memory (ready state) for time of execution.

#### 2. Short-term Scheduler-

- Short-term scheduler is also called as CPU Scheduler.
- It decides which individual process to execute next from the ready queue.
- After short-term scheduler decides the each process, Dispatcher assigns the each decided process to the CPU for execution purpose.

#### 3. Medium-term Scheduler-

- Medium-term scheduler processes swaps-out from main memory to secondary memory to free up the main memory when it required.
- Thus, medium-term scheduler mainly reduces the degree of multiprogramming.
- After some time when main memory becomes available to use, medium-term scheduler swaps-in and swapped-out process to the main memory and its execution is continued from where it left off.
- Swapping technique may also be required to improve the process mix.

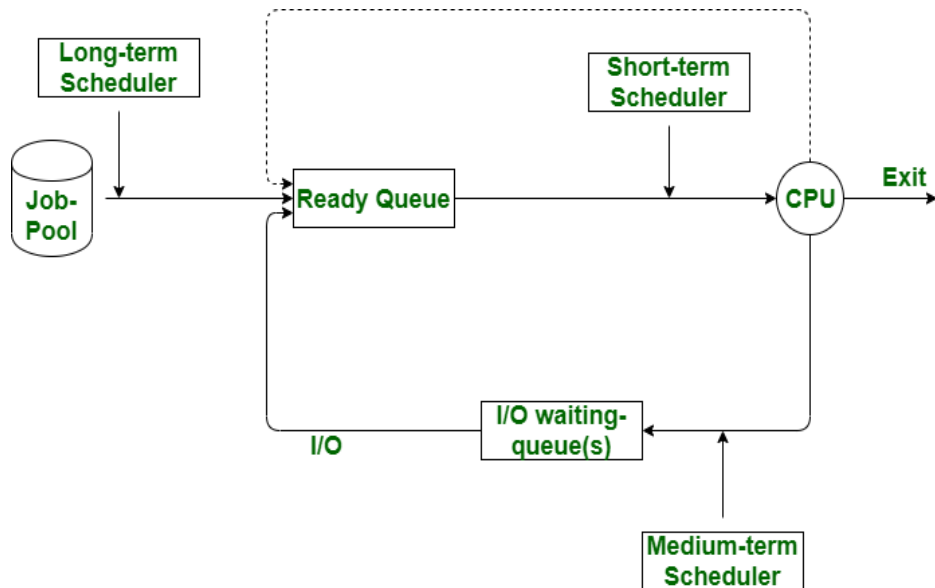


Diagram representation of process scheduling

### Comparison of Schedulers

| Long-term scheduler                                                  | Medium-term scheduler                                                                            | Short-term scheduler                                                                                      |
|----------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|
| It is a job scheduler                                                | It is a process swapping scheduler.                                                              | It is a CPU scheduler                                                                                     |
| It controls the degree of multiprogramming.                          | It reduces the degree of multiprogramming.                                                       | It provides lesser control over degree of multiprogramming.                                               |
| Speed is lesser than short-term scheduler                            | Speed is in between the long-term and short-term schedulers.                                     | Speed is fastest among the other two.                                                                     |
| It is minimal or almost absent in time sharing system.               | It is a part of time sharing system.                                                             | It is also minimal in time sharing system.                                                                |
| It selects processes from new state and loads them into ready state. | It swaps-out processes from main memory to secondary memory and later swaps in.                  | It selects processes from the ready state and assigns to the CPU.                                         |
| Operates less frequently since processes are not rapidly created.    | Operates more frequently than long-term scheduler but less frequently than short-term scheduler. | Operates very frequently to match the speed of CPU since CPU rapidly switches from one process to another |

#### 1.4.9 Operations on processes:

There are many types of operations that can be performed on processes. Some of these are process operation such as:

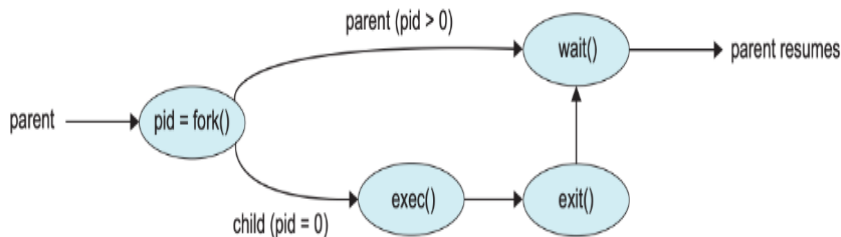
- 1) Creation
- 2) Process pre-emption
- 3) Process blocking
- 4) Process termination

## 1) Creation: Process Creation

Processes essential to be created in the system for different operations. This can be done by the following events –

- User request for new process creation
- Automatic system initialization
- Execution of a each process creation system call by a running process
- Batch job initialization

A process may be created by another new process using fork () system calls. The method creating process is called the parent process and the already created process is the child process. A child process can have only one parent but a parent process may have many children. Both the parent processes and child processes have the same memory image, open files, and environment strings.

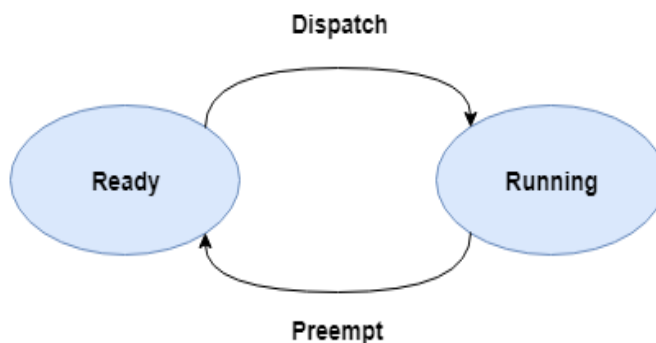


Process creation using FORK ( ) System Calls

## 2) Process Pre-emption

An interrupt instrument is used in process pre-emption that suspends the process executing presently and the next process to execute is determined by using a short-term scheduler. The main aims of process pre-emption is to makes sure that every processes get some CPU time for execution.

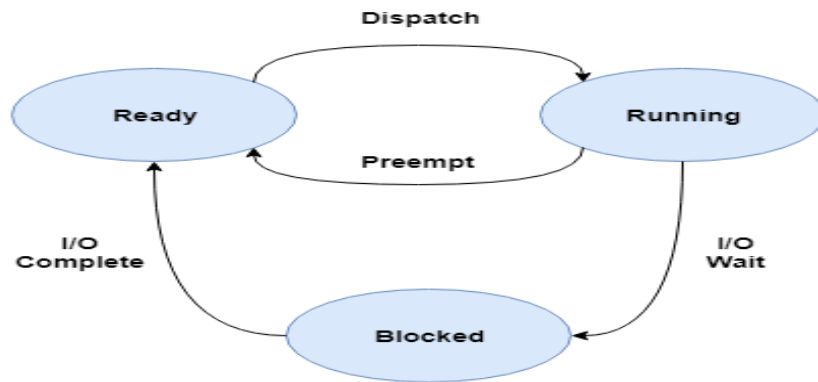
A diagram of process pre-emption is as follows:



### 3) Process Blocking

The process is blocked if it is process is waiting for some event to occur or process demanding is some I/O resources. This happening may be I/O as the I/O events are executed in the main memory and don't require the processor. After the event process is complete, the process again goes to the ready state.

A diagram that determines process blocking is as follows –

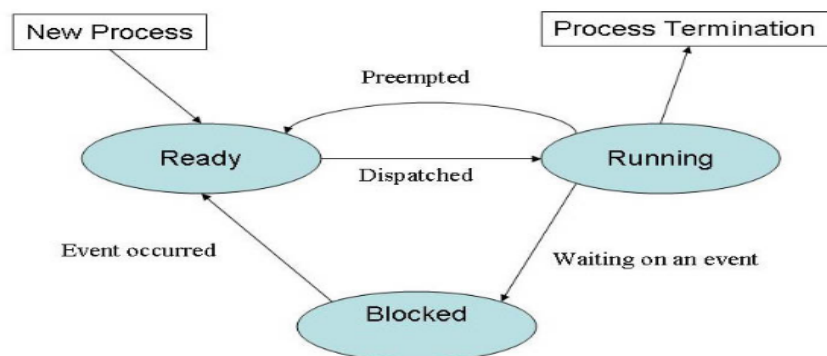


Process Blocking

### 4) Process Termination

After the process has successfully completed the execution task of its last instruction, it is terminated. The given resources held by a process are released after it is terminated.

A child process can be terminated by its parent process if its task is no longer used. The child process sends current status information to the parent process before it termination. Also, when a parent process is terminated, its child processes are terminated automatically as well as the child processes cannot run if the parent processes are already terminated.



Process Termination

### 1.4.10 Inter-process communication:

Working with multiple processes, require an inter process communication (IPC) method which will allow them to exchange data between multiple processes along with various useful information.

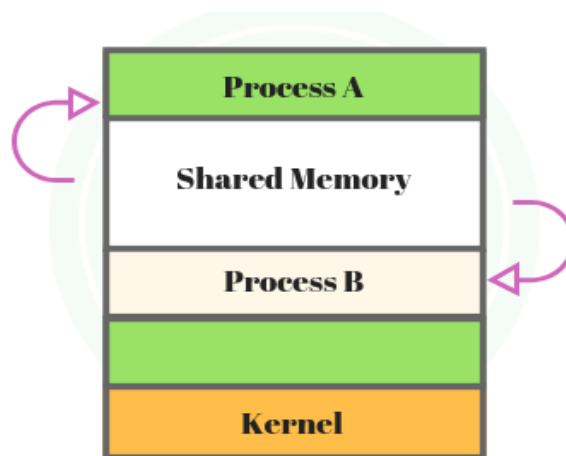
There are two primary models of inter-process communication:

- 1) Shared memory and
- 2) Message passing.

In the shared-memory model of inter-process communication, a region of memory which is shared by cooperating processes gets established. Processes can be then able to exchange all information by writing and reading all the data to the shared region. In the message-passing form, communication takes place by way of messages exchanged among the cooperating processes.

#### Shared Memory Process:

Inter-process communication (IPC) usually uses shared memory that requires communicating between different processes for establishing a region of shared memory. Typically, a shared-memory region resides within the address space of any process creating the shared memory segment. Other processes that want to communicate using this shared-memory segment must connect it to their address space.



Shared memory in operating system

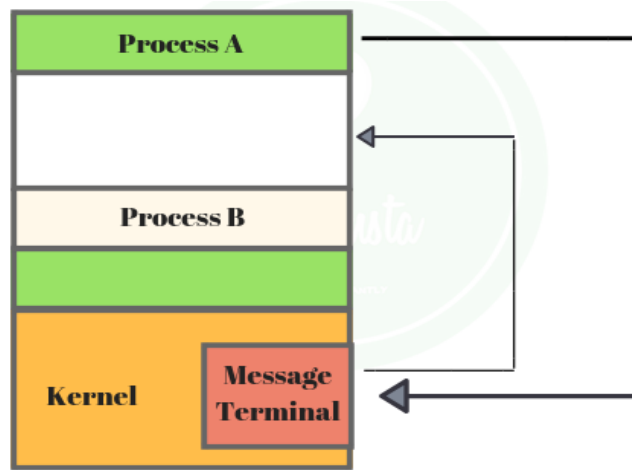
#### Message Passing Process:

Message passing systems are mainly support at a minimum system calls for purpose of send message and receive message.

A communication link must be developed between the cooperating processes before messages can be sent.

There are three key issues to be resolved in message passing process:

- 1) Direct or indirect communication
- 2) Synchronous or asynchronous communication
- 3) Automatic or explicit buffering



Message passing in operating system

---

## 1.5 Threads

---

### 4.1) Thread:

- To introduce the notion of a thread — a fundamental unit of CPU utilization that forms the basis of multithreaded computer systems
- To discuss the APIs for the Pthreads, Win32, and Java thread libraries
- To examine issues related to multithreaded programming

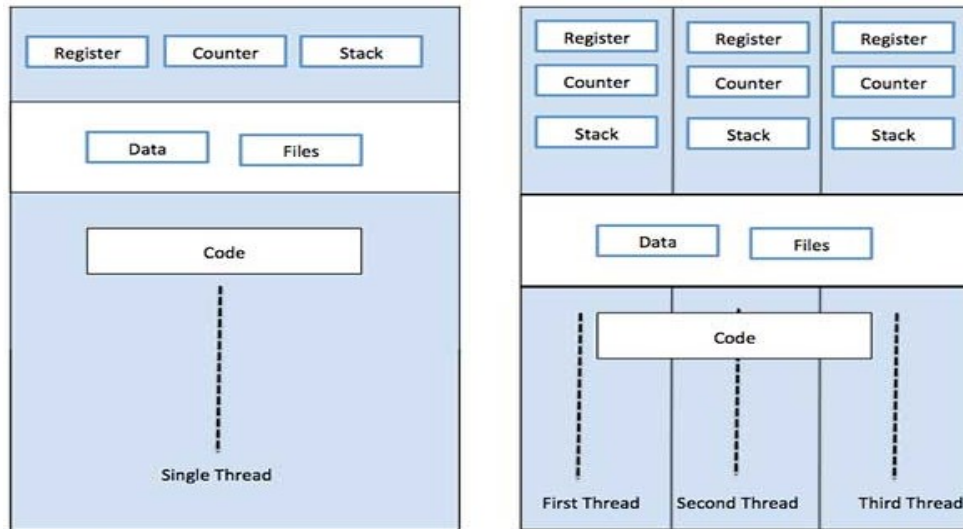
#### Benefits

- Responsiveness
- Resource Sharing
- Economy
- Scalability

**Thread** is a part of execution unit that are mainly consists of its own program counter, a stack, and a set of registers where the program counter mainly keeps track of which instruction to execute next, a stack mainly contains the history of execution and a set of registers mainly hold its current working variables. Threads are also called as Lightweight processes method. Threads are a very popular way to improve the performance of an application through parallelism. Threads are mainly used to represent a software approach in order to increase the performance of an operating system just by reducing the overhead thread that is mainly equivalent to a classical process.

It is significant to note here that each thread goes to exactly one process and outside a process no threads exist. Each thread basically represents the flow of control separately. In the implementation of network servers and web servers threads have been successfully used. Threads provide a suitable foundation for the parallel execution of applications on shared-memory multiprocessors.

The given below figure shows the working of a single-threaded process and a multithreaded process:



0

Single process P with Single Thread      Single Process P with three threads

#### Difference between Process VS Thread

| S.N. | Process                                                                                                             | Thread                                                                             |
|------|---------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------|
| 1    | Process is heavy weight or resource intensive.                                                                      | Thread is light weight, taking lesser resources than a process.                    |
| 2    | Process switching needs interaction with operating system.                                                          | Thread switching does not need to interact with operating system.                  |
| 3    | In multiple processing environments, each process executes the same code but has its own memory and file resources. | All threads can share same set of open files, child processes.                     |
| 4    | If one process is blocked, then no other process can execute until the first process is unblocked.                  | While one thread is blocked and waiting, a second thread in the same task can run. |
| 5    | Multiple processes without using threads use more resources.                                                        | Multiple threaded processes use fewer resources.                                   |
| 6    | In multiple processes each process operates independently of the others.                                            | One thread can read, write or change another thread's data.                        |

## Advantages of Thread:

- Threads require minimize the context switching time.
- Use of threads provides concurrency within a process.
- Efficient communication between processes.
- It is more efficient to create and context switch threads.
- Threads allow using of multiprocessor architectures to a better scale and efficiency.

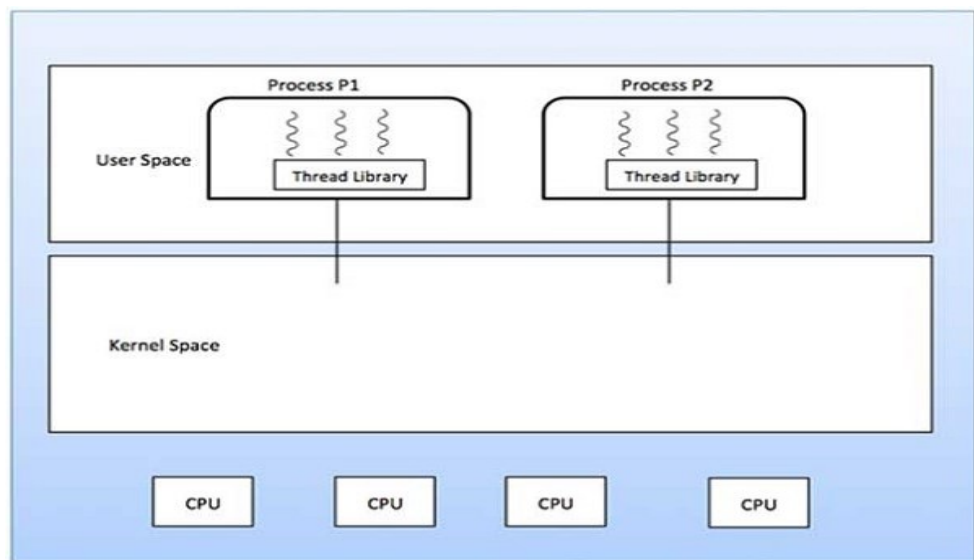
### 1.5.1 Types of Thread

Threads are implemented in following two ways –

- 1) User Level Threads
- 2) Kernel Level Threads

#### 1) User Level Threads

In the thread, the thread executive's kernel is not aware of the occurrence of threads. The string library covers code for making and obliterating strings, for the passing message and information b/w strings, for planning string execution, and for saving and re-establishing string settings. The application begins with a solitary string.



#### Advantages

- 1) Thread switching does not need Kernel mode privileges.
- 2) User level thread can run on any type of operating system.
- 3) Scheduling can be application specific in the user level thread.
- 4) User level threads are fast to the create and manage very easily.

- 1) In a typical operating system, most system calls are blocking.
- 2) Multithreaded application cannot take advantage of multiprocessing.

## 2) Kernel Level Threads

In this case, thread management is done by the Kernel only. There is no thread management code present in the application area. Kernel threads are supported directly by the operating system. Any application can be programmed to be multithreaded. All of the threads within an application are supported within a single process.

The Kernel maintains context type of information for the process as a whole and for individual's threads within the process. Scheduling by the Kernel is done on a thread basis. The Kernel is mainly performs three operations 1) thread creation, 2) scheduling and 3) management in Kernel space. Kernel threads are generally slower to create and manage than the user level threads.

### Advantages

- 1) If one thread in a process is blocked, the Kernel can schedule another thread of the same process.

### Disadvantages

- 1) Kernel threads are generally slower to create and manage than the user threads.

### Compare User Level Thread VS Kernel Level Thread

| S.N. | User-Level Threads                                                    | Kernel-Level Thread                                      |
|------|-----------------------------------------------------------------------|----------------------------------------------------------|
| 1    | User-level threads are faster to create and manage.                   | Kernel-level threads are slower to create and manage.    |
| 2    | Implementation is by a thread library at the user level.              | Operating system supports creation of Kernel threads.    |
| 3    | User-level thread is generic and can run on any operating system.     | Kernel-level thread is specific to the operating system. |
| 4    | Multi-threaded applications cannot take advantage of multiprocessing. | Kernel routines themselves can be multithreaded.         |

## 1.5.2 Multithreading Models

Multithreading models are three types

- 1) Many too many relationships.
- 2) Many to one relationship.
- 3) One to one relationship.

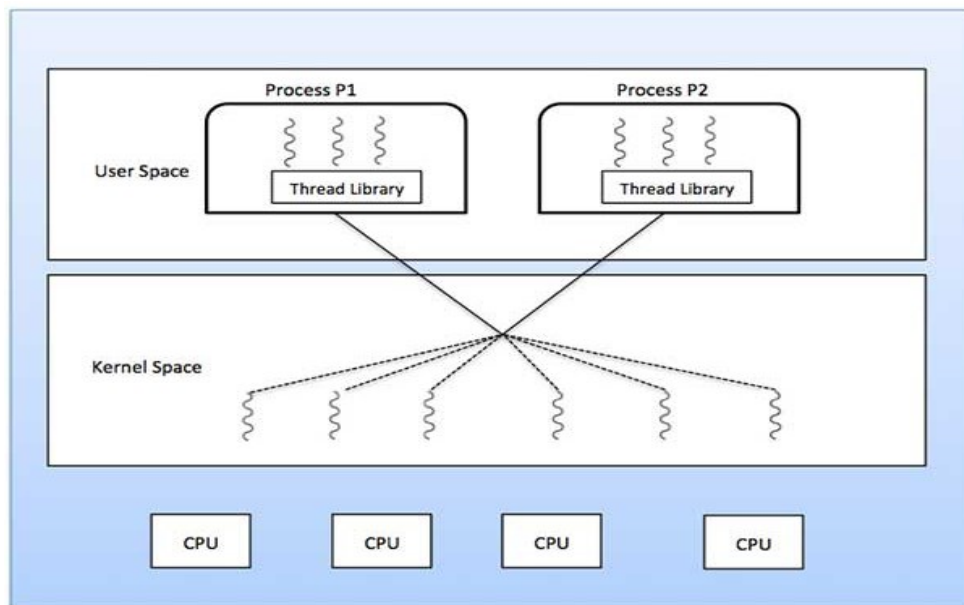
## 1) Many to Many Model

The many-to-many model multiplexes any number of the user threads onto an equal or smaller number of kernel threads.

The following diagram shows the many-to-many threading model where six user level threads are multiplexing with six kernel level threads. In this model, developers can create as many user threads as per necessary and the corresponding Kernel threads can run in parallel on a multiprocessor machine.

Example:

### 1) The Windows NT/2000



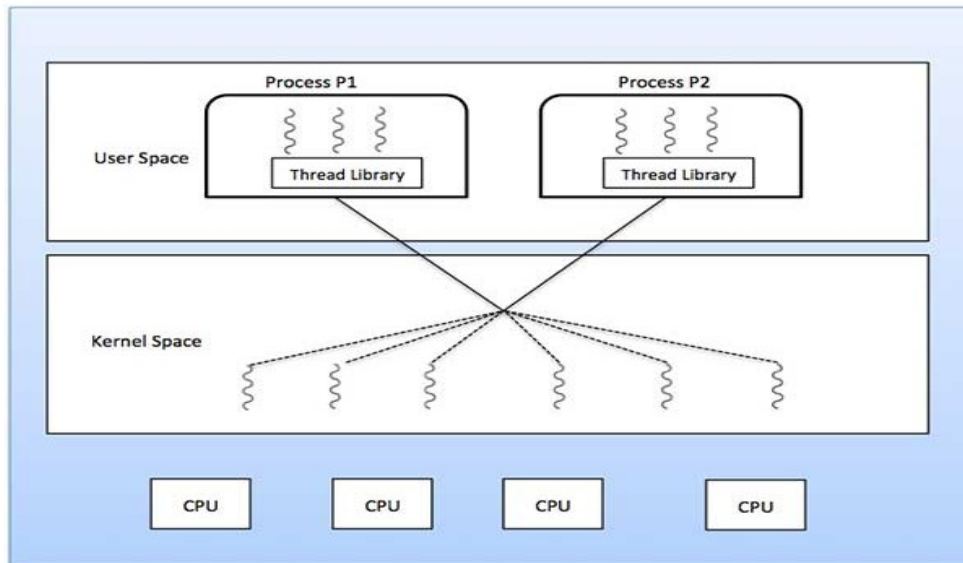
*Many to Many Model*

## 2) Many to One Model

When thread makes a blocking system call, the entire process will be blocked. When thread makes a blocking system call, the entire process will be blocked automatically. Only one thread can access the Kernel at a time, so multiple threads are unable to run in parallel on multiprocessors.

Examples:

- 1) Solaris Green Threads
- 2) GNU Portable Threads



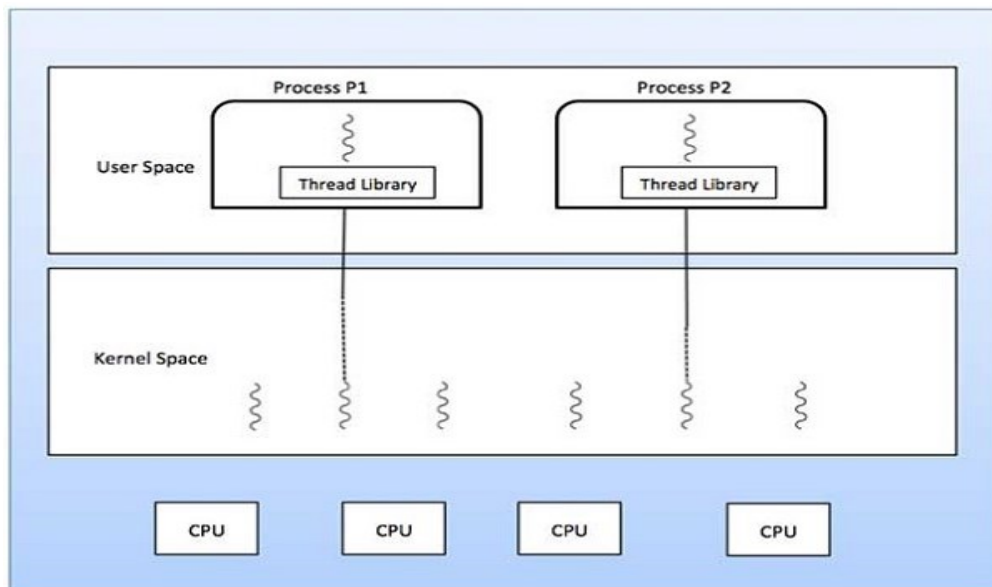
Many to One Model

### 3) One to One Model

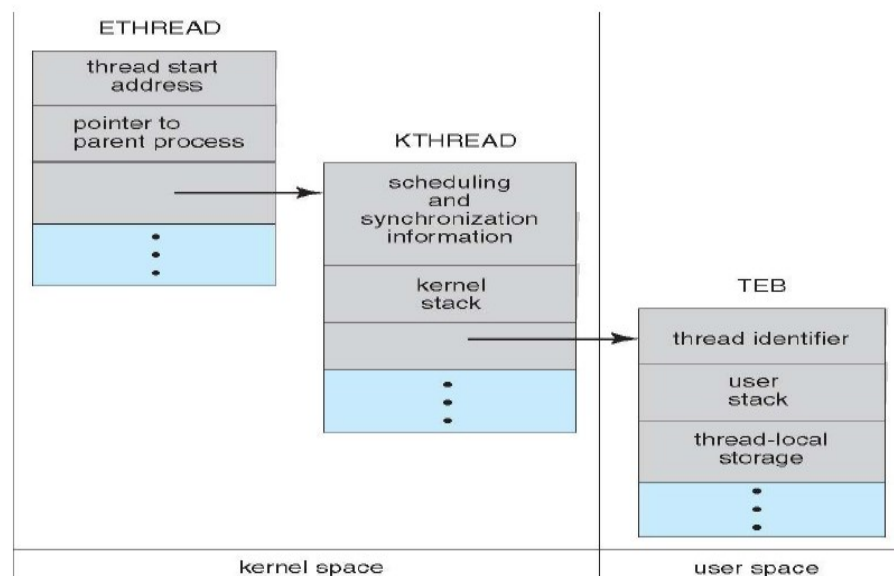
There is the one-to-one relationship of user-level thread to the kernel-level thread. This model provides more concurrency than the many-to-one model. It also allows another thread to run when a thread makes a blocking system call. It supports multiple threads to execute in parallel on microprocessors.

Examples

- 1) Windows NT/XP/2000
- 2) Linux



Windows XP Threads:



Implements the one-to-one mapping, kernel-level

Each thread contains

- 1) A thread id
- 2) Register set
- 3) Separate user and kernel stacks
- 4) Private data storage area
- 5) The register set, stacks, and private storage area are known as the context of the threads
- 6) The primary data structures of a thread include:
- 7) ETHREAD (executive thread block)
- 8) KTHREAD (kernel thread block)
- 9) TEB (thread environment block)

Linux Threads

| flag          | meaning                            |
|---------------|------------------------------------|
| CLONE_FS      | File-system information is shared. |
| CLONE_VM      | The same memory space is shared.   |
| CLONE_SIGHAND | Signal handlers are shared.        |
| CLONE_FILES   | The set of open files is shared.   |

- 1) Linux refers to them as *tasks* rather than *threads*
- 2) Thread creation is done through clone() system call
- 3) clone() allows a child task to share the address space of the parent task (process)

### 1.5.3 Multicore Programming:

Multicore programming benefits to the create concurrent systems for deployment on multicore processor as well as multiprocessor systems. A multicore processor system is basically a single processor with multiple execution cores in one chip. It has multiple numbers of processors on the motherboard or chip. A Field-Programmable Gate Array (FPGA) is power be included in a multiprocessor system. A FPGA is an integrated circuit containing an array of programmable logic blocks and a hierarchy of reconfigurable interconnects. Input data is processed by to produce outputs. It can be a processor in a multicore or multiprocessor system, or a FPGA.

The multicore programming approach has following advantages & minus;

- 1) The Multicore programming and FPGA processing helps to increase the performance of an embedded system.
- 2) To also help to complete scalability, so the system can consider the advantage of increasing the numbers of cores and FPGA processing power over time.

Concurrent systems that we create using multicore programming have various jobs to complete in parallel; this is known as concurrent execution. When multiple parallel tasks are executed by a processor, it is known as multitasking. A CPU scheduler handles the tasks that execute in parallel. The CPU implements tasks using operating system threads. So that tasks can execute autonomously but have some data transfer between them, such as data transfer between a data acquisition module and controller for the system. Data transfer occurs when there is a data dependency.

---

## 1.6 Conclusion:

---

In conclusion, an operating system is a software that manages computer hardware and software resources, and to provide public services for computer programs. The operating system is an important part of the system software in a computer system. ... In addition, there really is no such thing as a perfect operating system.

---

## 1.7 Summary:

---

In the introduction of operating system (OS), we saw that product can be generally separated into two gatherings: application programming and framework programming. Working frameworks are a sort of framework programming that permits applications to interface with PC equipment. Four significant classes of working frameworks are clump, timesharing, individualized computing, and committed. Assets are any items that can be assigned inside a framework, and the working framework is liable for overseeing them. A few assets, for example, essential memory can be space-multiplexed while different assets, for example, the CPU should be time-multiplexed

A cycle is an executing program. Since most PCs permit numerous cycles to execute at the same time, the working framework should deal with the request in which they execute. Three instances of interaction booking calculations are First Come First Serve, Round Robin, and Shortest Process Next.

---

### **1.8 EXERCISE:**

---

1. Write a short note on operating system.
2. Explain various operating system structures.
3. Explain various operating system services.
4. Write a short note on Inter-process communication.
5. State and explain various multi-threading models.

---

### **1.9 References:**

---

1. Operating Systems in Depth, Thomas W. Doeppner, Wiley.
2. Operating System Programming and Operating Systems, D M Dhamdhare, II nd Revised Edition, Tata McGraw.
3. Operating Systems, Achyut S. Godbole, 2nd edition, Tata McGraw Hill.
4. Application development using Android, Hello, Android, mobile development platform, Ed Burnette, 3rd Edition.
5. Linux Command Line & Shell Scripting, Richard Blum and Christine Bresnahan, 2nd edition, Wiley.

#### **Text Books:**

1. Modern Operating Systems, Tanenbaum, III rd Edition, PHI
2. Operating System-Internal & Design Principles, VI th Edition, William Stallings, Pearson
3. Operating Systems Concepts, Silberschatz A., Galvin P., Gagne G, VIII th Edition Wiley.
4. Principles of Operating Systems, Naresh Chauhan, First Edition , Oxford university press.

\*\*\*\*\*

## PROCESS SYNCHRONIZATION

### Unit Structure

- 2.1 General Structure of a Typical Process
- 2.2 Race Condition
- 2.3 The Critical-Section Problem
- 2.4 Peterson's Solution
- 2.5 Synchronization Hardware
- 2.6 Mutex Locks
- 2.7 Semaphores
- 2.8 Classic Problems of Synchronization
- 2.9 Monitors

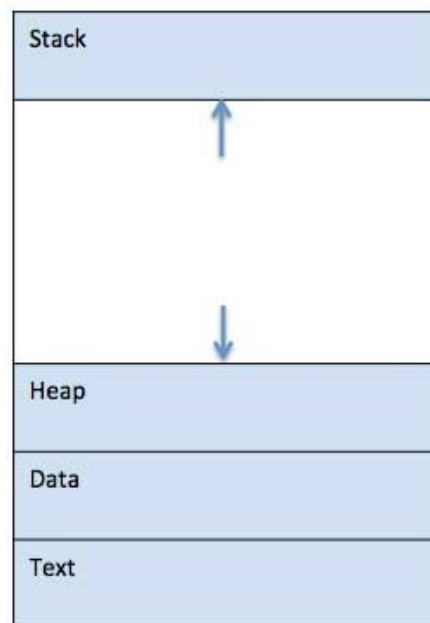
---

### 2.1 GENERAL STRUCTURE OF A TYPICAL PROCESS

---

Process is basically a program in execution. A process is defined as an entity which represents the basic unit of work to be implemented in the system

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



#### Stack

The process Stack contains the temporary data such as method/function parameters, return address and local variables.

## Heap

This is dynamically allocated memory to a process during its run time.

## Heap

This is dynamically allocated memory to a process during its run time.

## Data

This section contains the global and static variables.

---

## 2.2 RACE CONDITION

---

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

Several processes access and process the manipulations over the same data concurrently, then the outcome depends on the particular order in which the access takes place.

A race condition is a situation that may occur inside a critical section. This happens when the result of multiple thread execution in the critical section differs according to the order in which the threads execute.

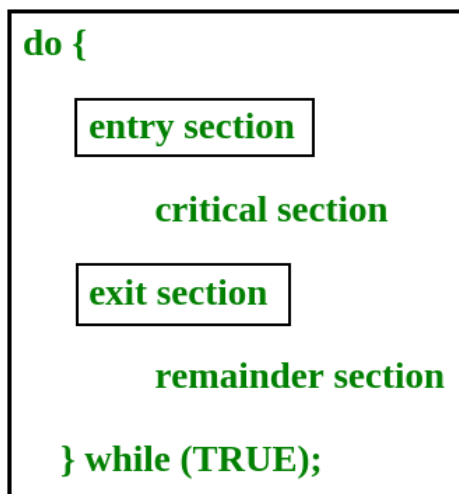
Race conditions in critical sections can be avoided if the critical section is treated as an atomic instruction. Also, proper thread synchronization using locks or atomic variables can prevent race conditions.

---

## 2.3 CRITICAL SECTION PROBLEM

---

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



In the entry section, the process requests for entry in the **Critical Section**.

Any solution to the critical section problem must satisfy three requirements:

- **Mutual Exclusion** : If a process is executing in its critical section, then no other process is allowed to execute in the critical section.
- **Progress** : If no process is executing in the critical section and other processes are waiting outside the critical section, then only those processes that are not executing in their remainder section can participate in deciding which will enter in the critical section next, and the selection can not be postponed indefinitely.
- **Bounded Waiting** : A bound must exist on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted.

---

## 2.1.4 PETERSON'S SOLUTION

---

Peterson's Solution is a classical software based solution to the critical section problem.

In Peterson's solution, we have two shared variables:

- boolean flag[i] : Initialized to FALSE, initially no one is interested in entering the critical section
- int turn : The process whose turn is to enter the critical section.

```
do {  
  
    flag[i] = TRUE ;  
    turn = j ;  
    while (flag[j] && turn == j) ;  
  
    critical section  
  
    flag[i] = FALSE ;  
  
    remainder section  
  
} while (TRUE) ;
```

Peterson's Solution preserves all three conditions :

- Mutual Exclusion is assured as only one process can access the critical section at any time.
- Progress is also assured, as a process outside the critical section does not block other processes from entering the critical section.
- Bounded Waiting is preserved as every process gets a fair chance.

Disadvantages of Peterson's Solution

- It involves Busy waiting
- It is limited to 2 processes.

---

## 2.1.5 SYNCHRONIZATION HARDWARE

---

TestAndSet

TestAndSet is a hardware solution to the synchronization problem. In TestAndSet, we have a shared lock variable which can take either of the two values, 0 or 1.

0 – Unlock

1 – Lock

Before entering into the critical section, a process inquires about the lock. If it is locked, it keeps on waiting until it becomes free and if it is not locked, it takes the lock and executes the critical section.

In TestAndSet, Mutual exclusion and progress are preserved but bounded waiting cannot be preserved.

---

## 2.1.6 MUTEX LOCKS

---

A **mutex** is a binary variable whose purpose is to provide locking mechanism. It is used to provide mutual exclusion to a section of code, means only one process can work on a particular code section at a time.

Mutex is a mutual exclusion object that synchronizes access to a resource. It is created with a unique name at the start of a program. The Mutex is a locking mechanism that makes sure only one thread can acquire the Mutex at a time and enter the critical section. This thread only releases the Mutex when it exits the critical section.

Eg :-

**Wait (mutex);**

... ..

**Critical Section**

... ..

**Signal (mutex);**

## 2.1.7 SEMAPHORE

A semaphore is a signalling mechanism and a thread that is waiting on a semaphore can be signaled by another thread. This is different than a mutex as the mutex can be signaled only by the thread that called the wait function.

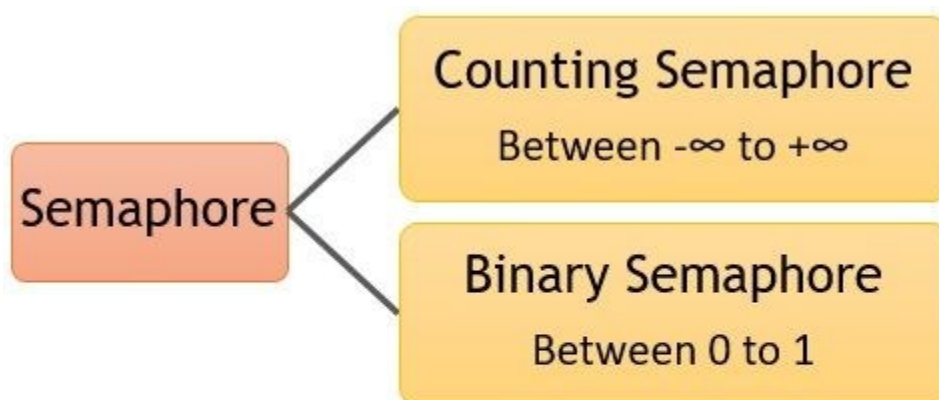
A semaphore uses two atomic operations, wait and signal for process synchronization.

The wait operation decrements the value of its argument S, if it is positive. If S is negative or zero, then no operation is performed.

```
wait(S)
{
    while (S<=0);
    S--;
}
```

The signal operation increments the value of its argument S.

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



**Types of Semaphore**

There are mainly two types of semaphores i.e.

1. counting semaphores
2. binary semaphores.

**The Counting Semaphores** are integer value semaphores and have an unrestricted value domain. These semaphores are used to coordinate the resource access, where the semaphore count is the number of available resources.

**The Binary Semaphores** are like counting semaphores but their value is restricted to 0 and 1. The wait operation only works when the semaphore is 1 and the signal operation succeeds when semaphore is 0.

---

## CLASSIC SYNCHRONIZATION PROBLEMS

---

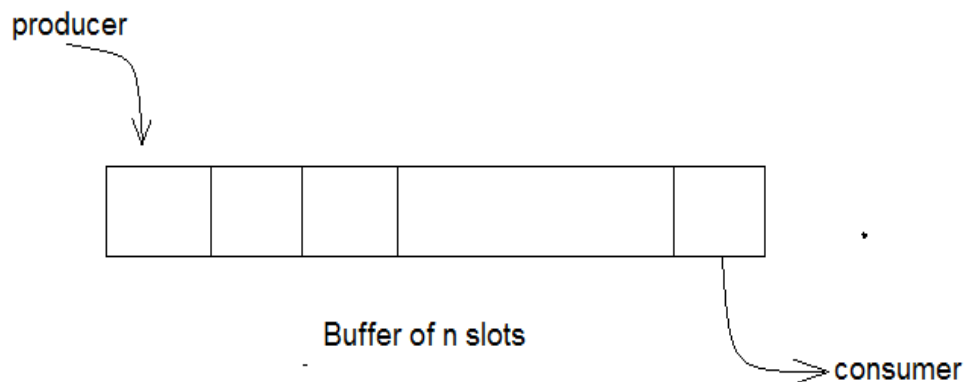
The classical problems of synchronization are as follows:

1. Bound-Buffer problem(producer-consumer problem)
2. Sleeping barber problem
3. Dining Philosophers problem
4. Readers and writers problem

### Bounded Buffer Problem

#### Bound-Buffer problem

Also known as the **Producer-Consumer problem**. In this problem, there is a buffer of  $n$  slots, and each buffer is capable of storing one unit of data. There are two processes that are operating on the buffer – Producer and Consumer. The producer tries to insert data and the consumer tries to remove data.

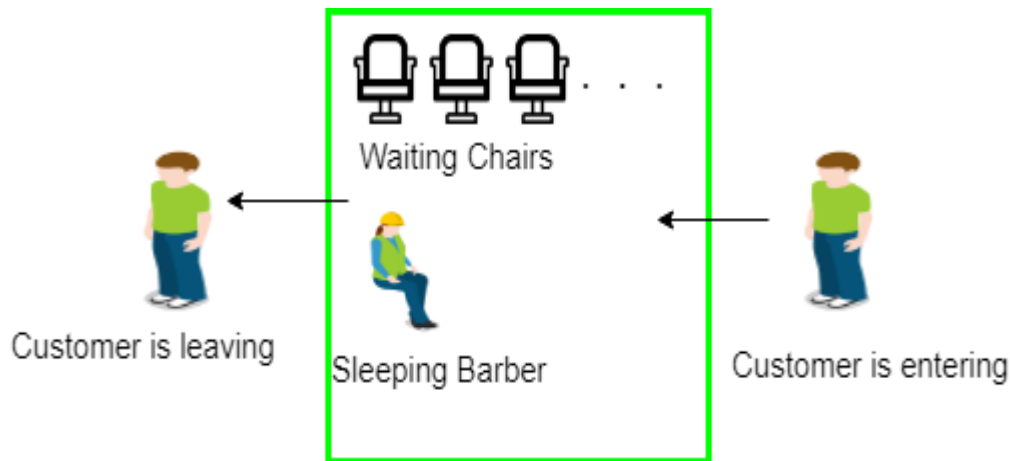


If the processes are run simultaneously, they will not yield the expected output.

The solution to this problem is creating two semaphores, one full and the other empty to keep a track of the concurrent processes.

### Sleeping Barber Problem

This problem is based on a hypothetical barbershop with one barber.

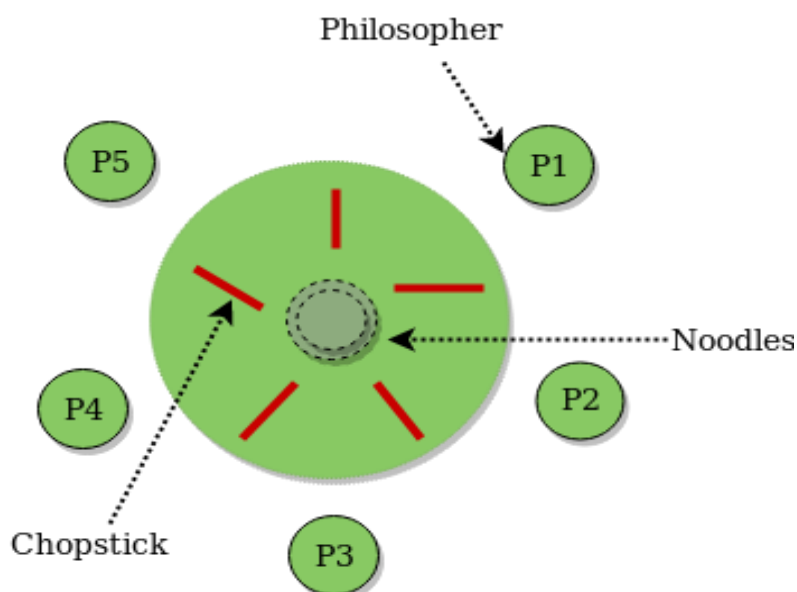


When there are no customers the barber sleeps in his chair. If any customer enters he will wake up the barber and sit in the customer chair. If there are no chairs empty they wait in the waiting queue.

### Dining Philosopher's problem

This problem states that there are K number of philosophers sitting around a circular table with one chopstick placed between each pair of philosophers. The philosopher will be able to eat if he can pick up two chopsticks that are adjacent to the philosopher.

This problem deals with the allocation of limited resources.



## Readers and Writers Problem

This problem occurs when many threads of execution try to access the same shared resources at a time. Some threads may read, and some may write. In this scenario, we may get faulty outputs.

### 2.1.8 Monitors in Process Synchronization

The monitor is one of the ways to achieve Process synchronization. The monitor is supported by programming languages to achieve mutual exclusion between processes. For example Java Synchronized methods. Java provides wait() and notify() constructs.

1. It is the collection of condition variables and procedures combined together in a special kind of module or a package.
2. The processes running outside the monitor can't access the internal variable of the monitor but can call procedures of the monitor.
3. Only one process at a time can execute code inside monitors.

Monitor syntax:

```

Monitor Demo //Name of Monitor
{
  variables;
  condition variables;

  procedure p1 {...}
  procedure p2 {...}

}

Syntax of Monitor

```

Condition Variables:

Two different operations are performed on the condition variables of the monitor.

1. **Wait**
2. **Signal**

**Example for 2 condition variables x,y**

condition x, y; // Declaring variable

#### **Wait operation**

x.wait() : Process performing wait operation on any condition variable are suspended. The suspended processes are placed in block queue of that condition variable.

**Note:** Each condition variable has its unique block queue.

### **Signal operation**

x.signal(): When a process performs signal operation on condition variable, one of the blocked processes is given chance.

If (x block queue empty)

    // Ignore signal

else

    // Resume a process from block queue.

### **Advantages of Monitor:**

Monitors have the advantage of making parallel programming easier and less error prone than using techniques such as semaphore.

### **Disadvantages of Monitor:**

Monitors have to be implemented as part of the programming language . The compiler must generate code for them. This gives the compiler the additional burden of having to know what operating system facilities are available to control access to critical sections in concurrent processes. Some languages that do support monitors are Java,C#,Visual Basic,Ada and concurrent Euclid.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

\*\*\*\*\*

## MAIN MEMORY

### Unit Structure

- 3.0 Objectives
- 3.1 Introduction
- 3.2 Background
  - 3.2.1 Memory Hierarchy
  - 3.2.2 Logical Versus Physical Address Space
- 3.3 Memory Management Unit
- 3.4 Swapping
- 3.5 Memory Allocation Techniques
- 3.6 Contiguous Storage Allocation
  - 3.6.1 Storage Placement Policies
  - 3.6.2 Fragmentation
  - 3.6.3 Compaction
- 3.7 Paging
  - 3.7.1 Address Translation
  - 3.7.2 Page Table Implementation
  - 3.7.3 Protection
  - 3.7.4 Shared Pages
- 3.8 Structure of the Page Table
  - 3.8.1 Hierarchical Paging
  - 3.8.2 Hashed Page Tables
  - 3.8.3 Inverted Page Tables
- 3.9 Segmentation
  - 3.9.1 Address Translation
  - 3.9.2 Segment Table Implementation
- 3.10 Virtual Memory
- 3.11 Demand Paging
  - 3.11.1 Steps to handling a page fault
- 3.12 Copy-on-write
- 3.13 Page Replacement
  - 3.13.1 FIFO Page Replacement Algorithm
  - 3.13.2 Optimal Algorithm
  - 3.13.3 LRU Page Replacement Algorithm
  - 3.13.4 Second Chance Page Replacement

## 3.14 Allocation of Frames

## 3.15 Thrashing

## 3.15.1 Cause of Thrashing

## 3.15.2 Methods to Handle Thrashing

## 3.16 Summary

## 3.17 Exercises

---

## 3.0 OBJECTIVES

---

After Completion of this chapter, you will be able to understand:

- Main Memory concepts;
- Different Memory Management techniques, including segmentation and paging;
- Detailed Structure of page table;
- Virtual Memory Management, Demand Paging and Page Replacement Algorithm;
- Causes of Thrashing and methods to handle it.

---

## 3.1 INTRODUCTION

---

The operating system manages the different processes of the computer, in addition to this, it also manages the primary memory of the computer. The primary memory or main memory functions as a central storage unit of the computer system. The main memory of the computer holds the instructions or programs which are currently running. It is volatile in nature which means when power is switched off, the data is lost. It is faster than secondary memory and slower than registers. The main memory is also called semi-conductor memories as the technology used in this memory is based on semiconductor integrated circuits. The **memory manager** which is the part of the operating system keeps track of the primary memory. The following are the main functions of a memory manager:

- In multiprogramming, it keeps track that which process will take how much memory and when.
- Allocates the memory to program or process when they request it.
- Deallocate the memory when the program or process has been terminated.
- It also makes sure that every process access different memory location, which are not being used by other processes.

---

## 3.2 BACKGROUND

---

Computer memory contains an array of bytes or words, each having their own address. The CPU first fetches an instruction from memory, execute

it and then results may be stored again in memory. The memory unit does not know how the memory addresses are generated; it only sees the sequence of memory addresses created by running processes. The CPU can directly access the main memory and registers. Any process or instruction that is currently running must be in the direct-access memory. If the instruction is not in memory, they must shift there before execution. The CPU can access the registers in one cycle of CPU clock, but in case of main memory it takes many cycles of the CPU clock because main memory can be accessed via memory bus. To overcome this problem a fast memory called *cache*, is added between main memory and CPU.

### 3.2.1 Memory Hierarchy

Memory and storage devices present in the computer system have different features like speed, capacity, performance and cost. The organization of memory hierarchy is based on program behaviour called as principal of locality or locality of references. The memory hierarchy can be divided into two parts: primary memory and secondary memory. The following diagram shows the different levels of memory hierarchy:

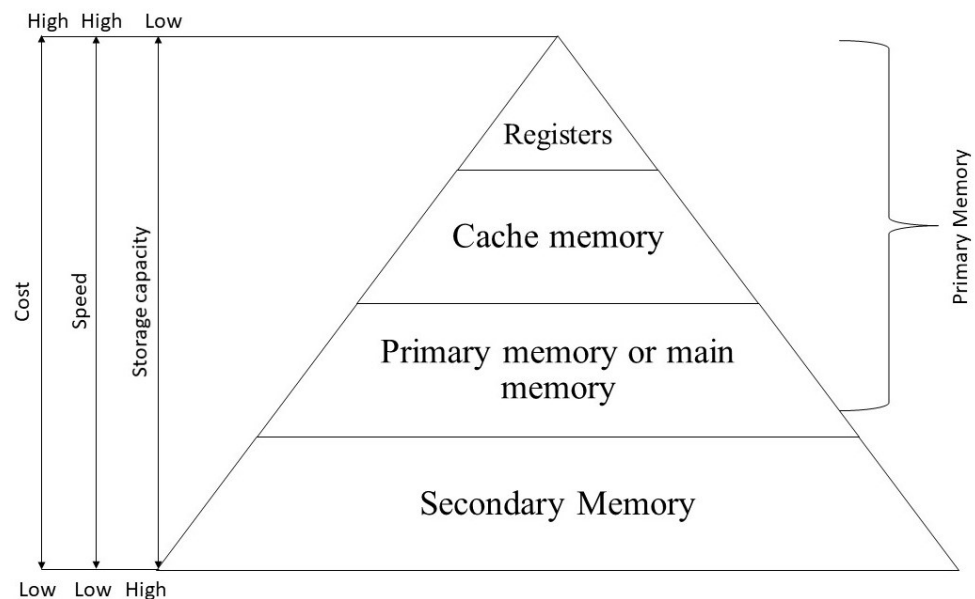


FIGURE 3.1

According to the memory hierarchy, the following occur:

- Storage capacity of secondary memory is higher than primary memory.
- Secondary memory has lower speed than primary memory.
- Cost of primary memory is higher than secondary memory.

### 3.2.2 Logical Versus Physical Address Space

Before discussing the memory allocation techniques, let us know about logical address space and physical address space and how they are differing from each other.

## Logical Address Space

- When an address is generated by the CPU, it is called as *logical address*.
- Logical address does not exist physically so it is also known as virtual address.
- Basically, logical addresses are used to refer the data of a program's address space. The user can view the logical address of a program.
- Logical address can be used to access the physical address.
- When a program generates the set of all logical addresses is called *Logical Address Space*.
- It can be referred as the size of the process. The size of the process should be within the range of main memory.

## Physical Address Space

- An address which is loaded into the memory address register is called as *physical address*.
- In other words, Physical addresses means the main memory addresses where the data is stored. User can not directly access the physical address but via logical address.
- The set of physical addresses communicating to logical addresses of program is called *Physical address space* of that program.
- It can be referred as the size of main memory. The size of the program must be less than physical address space.

Physical and Logical addresses are different at execution time but same at compile time and load-time. The figure 3.2 shows the concept of logical and physical address space.

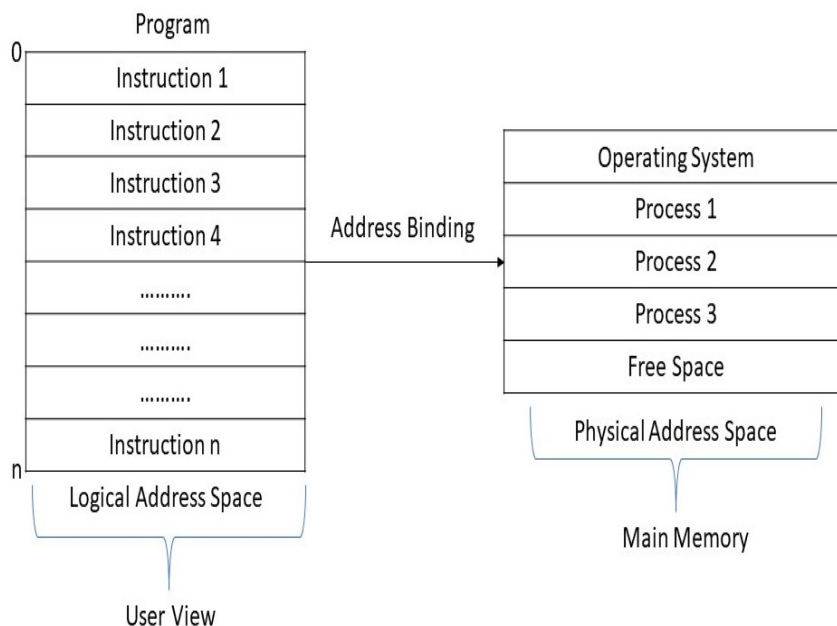


FIGURE 3.2

### 3.3 MEMORY MANAGEMENT UNIT

Memory Management Unit (MMU) is a hardware device that is located in the CPU. It maps the logical address to the physical address. The base-register scheme is used for mapping. The base-register is also called a relocation register, which is a special register in the CPU. The relocation register value is added to the program's address generated by the user at the time it is sent to memory. This method is called dynamic relocation or program relocation. Hence,

$$\text{Physical address} = \text{logical address} + \text{value of relocation register}$$

This mapping can be accomplished by different methods or mapping schemes. The user program directly deals with the logical address, it never sees the real physical addresses. The program or process must be in physical memory before executing. The figure 3.3 shows the dynamic relocation using base register or relocation register.

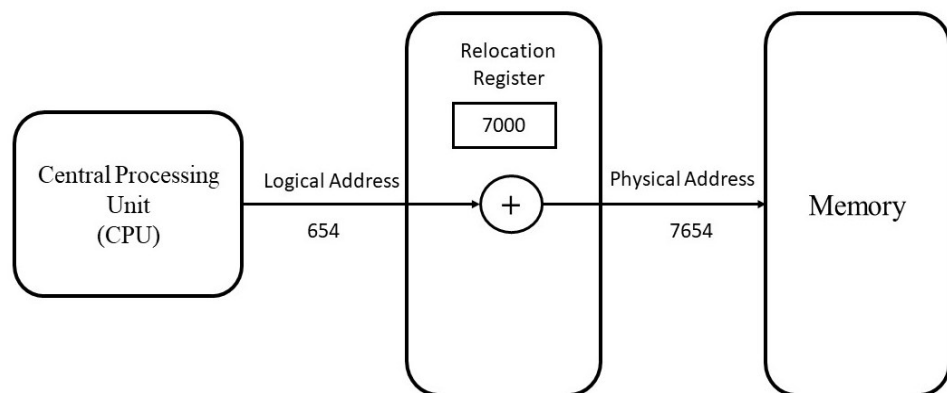


FIGURE 3.3

For example, the value in relocation register is 7000 and the value in the logical address is 654. Then the memory management unit mapped the logical address with the value in the relocation register and converts into physical address. After the mapping it dynamically relocated the location to 7654. The user only generates the logical address but they must be mapped to physical address before execution.

### 3.4 SWAPPING

A process that is currently executing must be in the main memory (RAM). The process has to be first loaded from secondary memory (hard disk) to primary memory or main memory before execution. This technique is called *process loading*. After completing the execution, it must load back to secondary storage as main memory has limited space and other processors need that space for execution. When the two processes swaps, it is called **Swapping**. Swapping is done by the operating system. The figure 3.4 shows the operation of a swapping. First there is only process

P1 in main memory, then process P2 is initiated and **swapped in** from secondary memory and P1 **swapped out** to secondary storage.

Main Memory

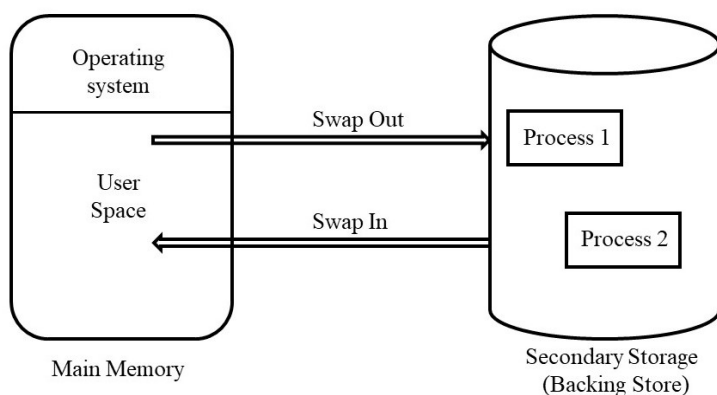


FIGURE 3.4

For example, in a multiprogramming environment, when the time expires of any particular process it starts to swap out the process and swap in another process for execution. If there is any process running and a higher priority process arrives then the memory manager swaps the current running process with high- priority process. When high priority process completes its execution the lower priority processor swapped in. this is also called roll-in, roll out.

### 3.5 MEMORY ALLOCATION TECHNIQUES

Memory allocation techniques or memory management techniques are of two types:

**(1) Contiguous Memory Allocation:** In contiguous memory allocation, every data and process occupy single contiguous storage area. The following are the schemes used in contiguous memory allocation:

- Single Partition Allocation
- Multiple Partition Allocation

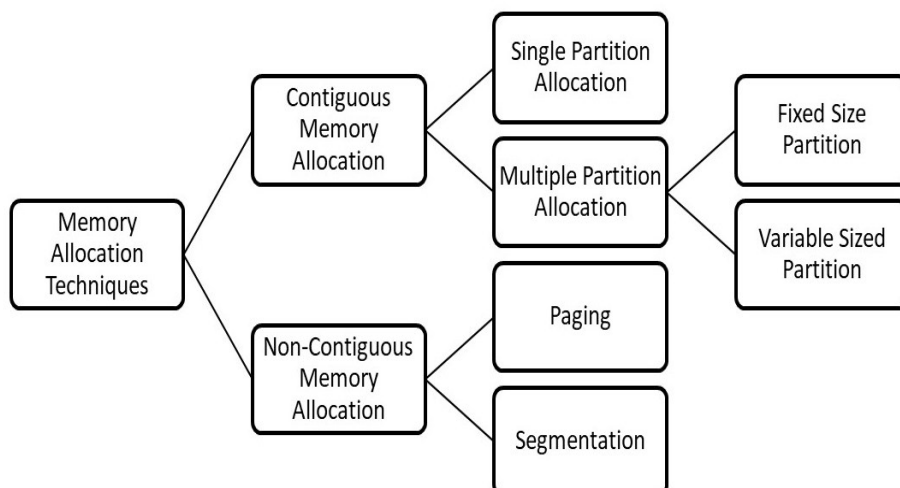


FIGURE 3.5 Summary of Memory Allocation Techniques

**(2) Non-Contiguous Memory Allocation:** In this allocation, the instructions and data may occupy non-continuous storage area. The schemes used in Non-Contiguous memory allocation are:

- Paging
- Segmentation

---

### 3.6 CONTIGUOUS STORAGE ALLOCATION

---

**(A) Single Partition Allocation:** Single allocation partition is also called single process or single user contiguous allocation. In this type of scheme just one program runs at a time. Memory is split up into two parts, first part contains the operating system and second part contains the user's processes which are loaded and executed according to the user's command. When the process completes its execution, it comes out from main memory and next waiting process is brought into main memory by the operating system.

#### Advantages

- This scheme is very simple.
- No expertise is required to use and understand this type of system.

#### Disadvantages

- Utilization of processes is poor as they have to wait for input/output resources.
- Size of the main memory is limited.
- No proper utilization of memory.

**(B) Multiple Partition Allocation:** This allocation scheme is divided into two parts as following:

**(a) Fixed Size Partition Memory Allocation:** - In this scheme, each process can hold a separate fixed area in the memory. This is also known as *partitioned memory allocation-static*. As shown in figure 3.6, the main memory is divided into three parts, of size 100k, 200k and 300k respectively. Each part holds a process. The operating system can support the execution of processes simultaneously, as all the processes can run at any time.

Every part of the memory contains some unused space. This wasted space is referred to as **internal fragmentation**. Early batch systems used this scheme where space requirement of a process is known beforehand.

#### Advantages

- This scheme is easy to implement.

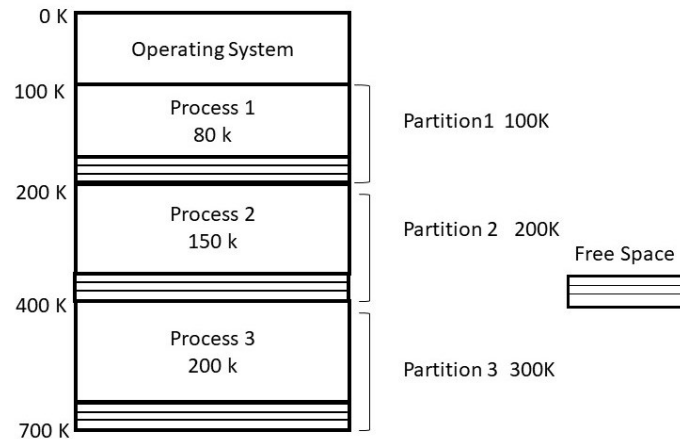


FIGURE 3.6 Fixed Size Partition Memory Allocation

### Disadvantages

- Due to unavailability of a partition of required size can prevent a process for execution.
- Unused space in each partition.

- (b) **Variable Sized Partition Memory Allocation:** - To overcome the problem of fixed size partition, variable sized partition scheme comes into picture. It allocates the memory to process according to its requirement at the load-time. Processes are loaded in continues manner till the memory is full and sometimes remaining memory size is very small to load any process. In this scheme partition is done at load-time so it is also called *partition-memory allocation-Dynamic*.

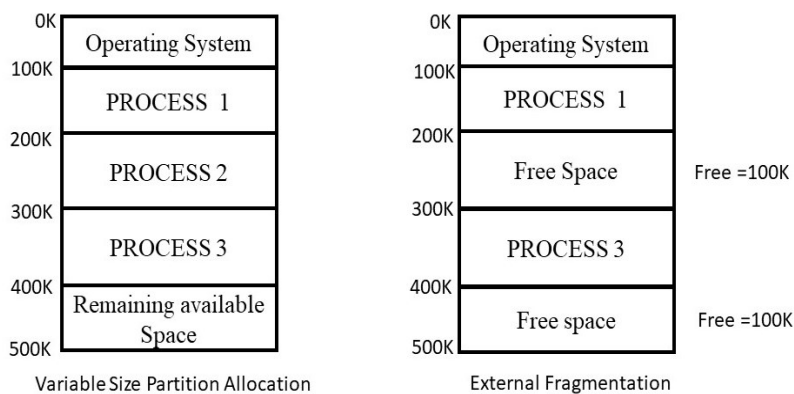


FIGURE 3.7

When a process completes its execution and free the occupied space, then free space appeared as the series of 'holes' between the memory areas. The operating system then loads another process in that area but sometimes there is not enough space for the next process as the size of the

process is more than that free space. So, this free memory space is called **external fragmentation**.

### 3.6.1 Storage Strategies or Placement Policies

The purpose of placement policy algorithm is to select all the free space which will give maximum throughput for the system. When any new process is loaded into memory using variable partition scheme, try to select best location for that process. The following are some placement policies.

- **Best Fit Policy:** - In this policy, the memory manager loads a process in the small area of memory which is unallocated and it fits. For example, in figure 3.8 there are unallocated blocks of 10KB, 14KB, 17KB, 12KB and 16KB. A process needs 11KB of memory, then the best fit policy will allocate 11KB of the 12KB block to the process. In this policy memory manager allocate the smallest hole which is big enough for the process.
- **First Fit Policy:** - In this policy, memory manager allocates the first available hole to the process that is big enough for the process. To load process into the memory first fit policy allocates 14KB block for 11KB process (see figure 3.8).

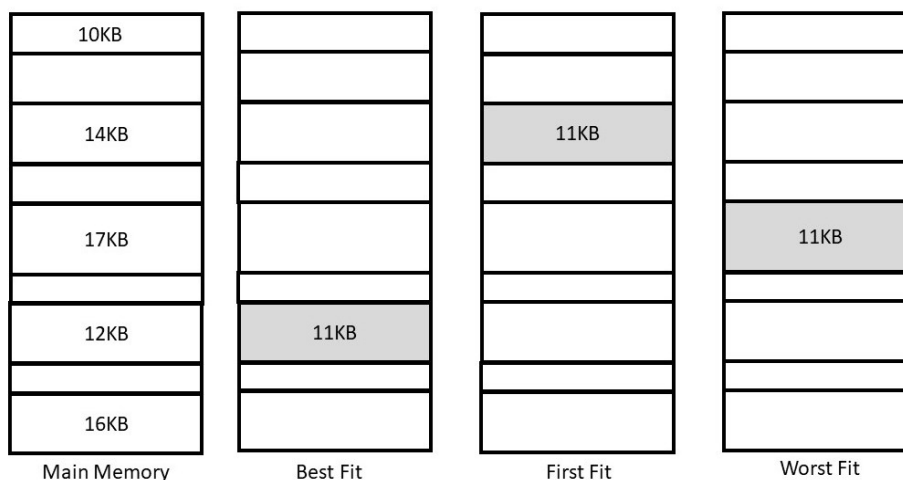


FIGURE 3.8

- **Worst Fit Policy:** - The worst fit policy allocates largest hole to the process from unallocated memory available. Using same example, memory manager allocates 17KB block to 11KB process.

### 3.6.2 Fragmentation

The processes free the memory space after the completion of execution and that memory space is not big enough for other processes, then there occurs the problem of **fragmentation**. The cause of fragmentation is that because of dynamic allocation of processes sometimes the free space is insufficient for the particular process. There are two types of fragmentation: -

## Internal Fragmentation

- The free space wasted internally in memory blocks.
- Process is larger than memory.
- Memory is partitioned into fixed size.
- Difference between memory allocation and required space is called internal fragmentation.
- Solution to internal fragmentation is best-fit.

## External Fragmentation

- The space wasted externally to allocated memory blocks.
- When a process is removed from memory.
- It occurs when there is variable size partition of memory.
- Non-contiguous memory blocks that are not big enough for any process is called external fragmentation.
- Solution to external fragmentation is segmentation, paging and compaction.

### 3.6.3 Memory Compaction

In the main memory when processes removed from memory then multiple holes occurs at different blocks. To combine free blocks (holes) and move downward all the processes, to create enough possible space for any new process is called *memory Compaction* or *relocatable partitioned memory management*. This technique can be performed at fixed interval of time, when fragmentation occurs or when any process removes from memory.

#### Advantages

- This technique removes external fragmentation.
- Allows multiprogramming as utilization of processes is increased.

#### Disadvantages

- The cost of this technique is high.
- It slows down the speed.
- This technique is complex.
- Sometimes there are chances of unused memory space.

---

## 3.7 PAGING

---

Paging is a technique of memory management, which can be the possible solution to external fragmentation. This is also called paged memory management. In this system, each process is divided into stable number of **pages** also called *chunks*. The main memory space is also divided into areas of same size which are called **frames**. The memory manager loads the process page to particular frame. The process pages remain logically continues but frames may not be continuous. The figure 3.9 shows the paging technique with example where two different processes have been loaded into memory (frames). Paging has been supported by hardware,

however in recent designs hardware and operating system are closely integrated.

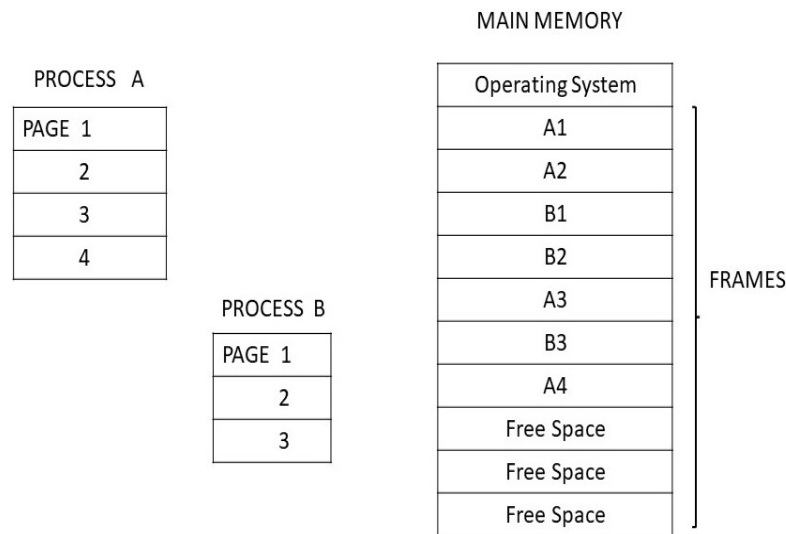


FIGURE 3.9 Paging Allocation Technique

### 3.7.1 Address Translation

As we know paging is supported by hardware, so every address is generated by CPU. Address is separated into two parts: a page number( $p$ ) and displacement or page offset( $d$ ). The displacement means the location from the start of the page. In a page table page number is used as an index. In the page table base address of every page in physical memory is there. Its together with page offset defines the address of physical memory. The page size is depended on the computer architecture. Usually, the page size is a power of 2. Let us take an example of a 16-bit address shown in figure 3.10.

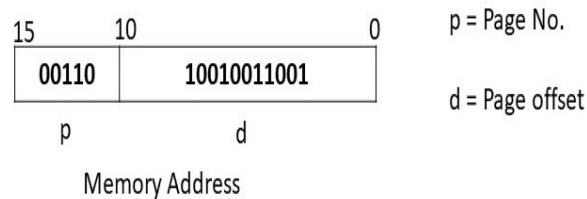


Figure 3.10

The displacement or offset uses 11 bits of memory address and the page number uses first 5 bits. The value of page number is from range 0 to 31 ( $2^5 - 1$ ) or 32 pages and the range of displacement is from 0 to 2023 ( $2^{11} - 1$ ). According to this it has 32 pages each of 2024 locations. The relocation problem is simply solved by page table because it contains each process's page number and its corresponding frame number. Figure 3.11 shows the overall paging model of address translation.

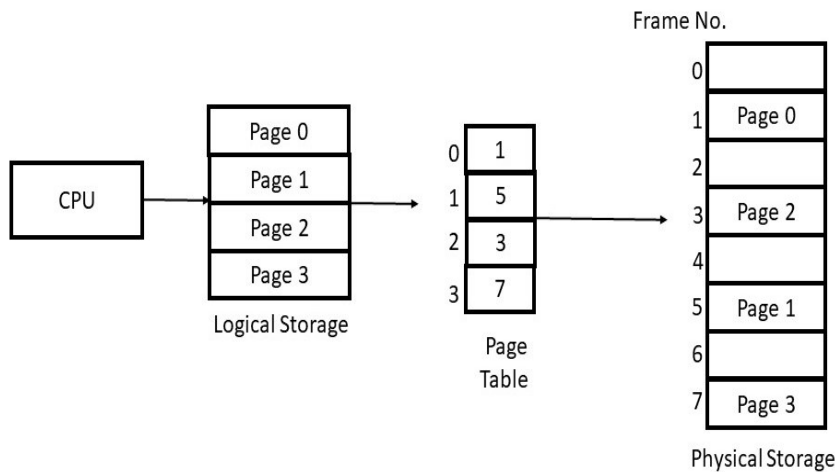


FIGURE 3.11 Address Translation

### 3.7.2 Page Table Implementation

Implementation of a page table or its hardware support can be done through many ways.

- First the simplest one is implemented as dedicated registers made from high-speed logic. It carried out the address translation very fast. But this implementation is not suitable when entries in page table are very large in numbers.
- The next method is using **page table base register (PTBR)**. The page table is store in the main memory. PTBR points to page table, so it requires changes only at one base register. But access time can be increased and it can slow the system.
- The next solution of this problem is to add cache memory. It consists of most frequently used pages and also known as **Translation Look Ahead Buffer (TLAB)**. In this method when address is generated by CPU, the page number is first searched in cache memory if it is their then it combines with offset and provide physical address called as TLB hit, If the page number is not in cache, then it searches the page table called as TLB miss.

### 3.7.3 Protection

Protection bits are associate with each frame to accomplish memory protection. Page table contains the protection bits. These bits determine a page to read/write or read only. These bits are verified before translating logical address to physical address. These bits also allow the access to any particular page. Protection bit is also called valid/Invalid bit. The following figure shows the projection bit in page table.

| Page No. | Frame No. | Valid/Invalid bit |
|----------|-----------|-------------------|
| 0        | 5         | V                 |
| 1        | 3         | V                 |
| 2        | 8         | I                 |
| 3        | 5         | V                 |
| 4        | 1         | I                 |
| 5        | 7         | V                 |
| 6        | 4         | v                 |

FIGURE 3.12

### 3.7.4 Shared Pages

The paging method supports the possibility of sharing common program code. It can be helpful in time sharing environment. This code is also called re-entrant, which means it will never change during execution by any write operation. For example, if 15 users execute a text editor, only one copy of text editor needs to be stored in the main memory. So, to support 15 users only one copy of the editor is required.

---

## 3.8 STRUCTURE OF PAGE TABLE

---

The following are the techniques for structuring the page table.

### 3.8.1 Hierarchical Paging

- Hierarchical page table is also known as multilevel paging.
- Logical address space is divided into multiple page tables.
- It generates hierarchy with several levels for the page table that are too big to fit in a continuous memory area.
- Basically, it divides the page table into smaller blocks.
- A two level and three level schemes are simple to use in hierarchical paging.

### 3.8.2 Hashed Page Tables

- To handle larger than 32-bit address space hashed page table is used.
- It handles with the virtual page number. The table contains linked list of elements hashing to same location.
- Every element contains three fields:
  - The Virtual page numbers.
  - The Value of the mapped page frame.
  - The Pointer to the next element
- The virtual page numbers are compared with the linked list, if the match is found, then that frame is extracted.

### 3.8.3 Inverted Page Tables

- Inverted page table combines page table and a frame table.
- It has one entry for each virtual page number and real page.

- The paging information is represented by a single page table.
- It reduces the memory space.
- In the table there are different fields like page number, process id, control bits and change pointer.
- It takes more time to find the entry.
- Shared memory implementation is difficult in inverted page table.

---

### 3.9 SEGMENTATION

---

Segmentation supports the user's view of memory. The programmer's view their programs as collection of libraries, data and routines as shown in figure 3.16. The collection of logically related information in variable sized segments is called segmentation. The logical address is divided into segment number and segment offset.

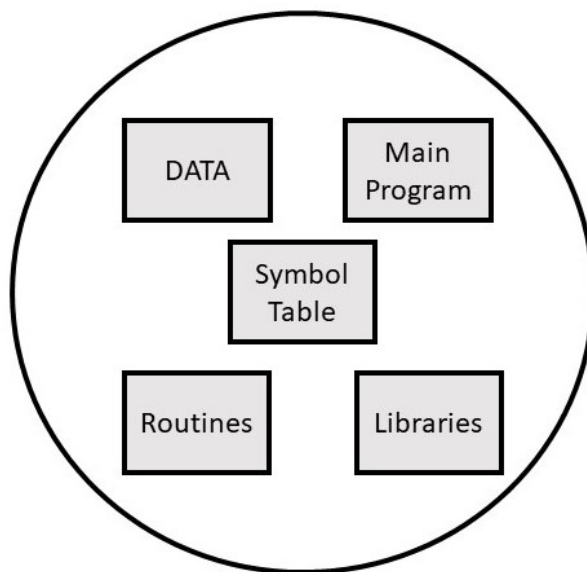


FIGURE 3.16

#### Characteristics of segmentation

- It is a variable sized partition technique.
- When secondary memory is divided into partitions is called segments.
- This technique helps primary and secondary memory to partition into unequal sized.
- The size of the partition depends upon the length of modules.

#### 3.9.1 Address Translation

To convert logical address into segment address is similar to paging scheme. The segment address has two parts, first is segment reference  $s$ , and second is displacement  $d$ . A process segment table that has entries of base table and segment size refers to the segment index as shown in figure 3.17

The following steps requires for segmented address reference:

- From the logical address take out the segment number and the displacement.
- To get the segment base address use the segment number to index the segment table.
- Offset should not greater than the length. If the offset is greater then an invalid address occurs.
- By adding the base address and offset, it generates the physical address.

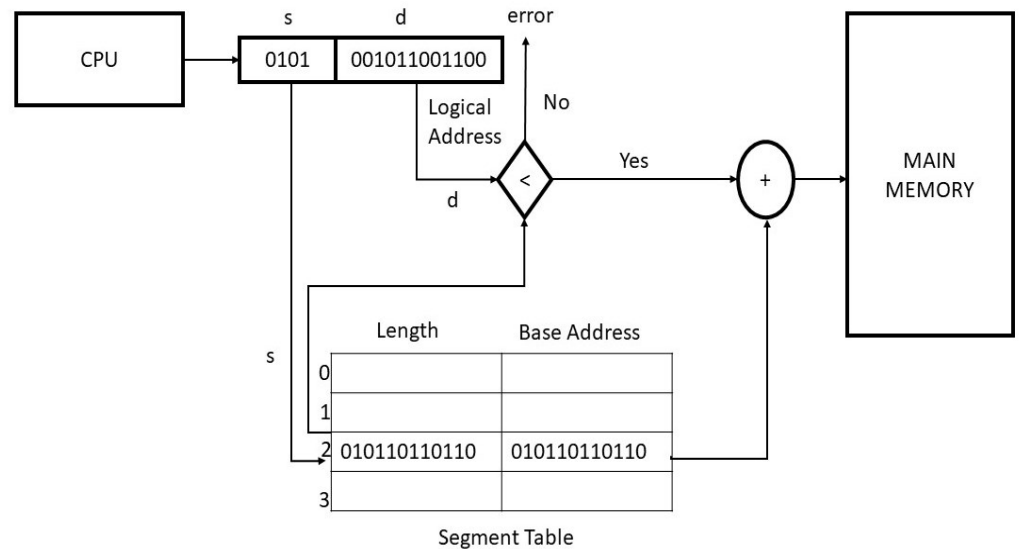


FIGURE 3.17

### Advantages

- It allows the shared segments.
- It allows to grow segments dynamically.
- Dynamic linking is possible.

### Disadvantages

- Difficult to manage variable size segments on secondary storage.
- The maximum size of segment depends on the size of main memory.
- Segmentation is expensive technique.
- Leads to external fragmentation.

### 3.9.2 Segment Table Implementation

- The segment table can be in system memory or in fast registers.
- If a segment table is in registers, it can reference quickly.
- There can be large number of segments of program so, segment table can be kept in system memory.
- Two tables are implemented. First is Segment Table Base Register (STBR), second is Segment Table Length Register (STLR).
- STBR is used to point the segment table in memory and STLR is used to restrict the possible manageable number of segments.

When a process with large size than physical memory can be executed by loading the parts of the process refers to the concept of virtual memory. It is not an existing memory but only a scheme. Sometimes the size of the data, programs and stacks may be bigger than physical memory, so the concept of virtual memory comes into picture. Only the part of the program that required execution kept into virtual memory and the rest part of the program is on the disk. The separation of user's logical memory from physical memory is the virtual memory. When only small amount of main memory is available then, then this separation allows large amount of virtual memory for users or programmers as shown in figure 3.18

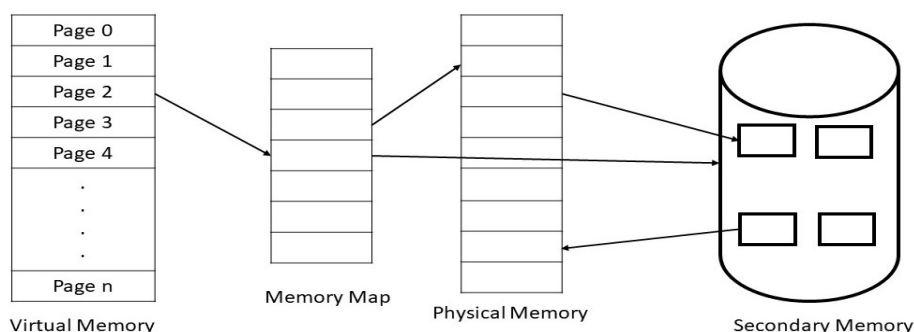


FIGURE 3.18

### Advantages

- The programmer does not need to care about the physical memory, as virtual memory makes the task much easier.
- It increases the throughput and utilization of CPU.
- It reduces external fragmentation, as process can be loaded into arbitrary size space.
- The processes with high priority can run faster.

### Disadvantages

- The systems with virtual memory are slow.
- Due to large number of paging, thrashing can be occurred.
- Additional system hardware support is required.

---

## 3.11 DEMAND PAGING

---

Demand paging concept is similar to swapping in paging system. In virtual memory concept demand paging is used. In the virtual memory, the process is in secondary memory. The process is divided into pages. When any process needs to be executed only one page of that process loads into memory, which required execution. Instead of loading whole process, the loader only loads required pages into memory as shown in figure 3.19.

During the translation it checks whether the page is in physical memory or not. Valid (IN) and invalid (OUT) bits are attached with the table. If the bit is set to IN, then the page is in the main memory and if bit is OUT the page is not in the primary memory rather it is in secondary memory. Due to the OUT bit, an interrupt occurs which is called **page fault**.

Page Frame Map Table (PFMT) (an additional data structure in virtual memory) is used to store secondary storage addresses of the pages. The operating system consults PFMT, when a page fault occurs.

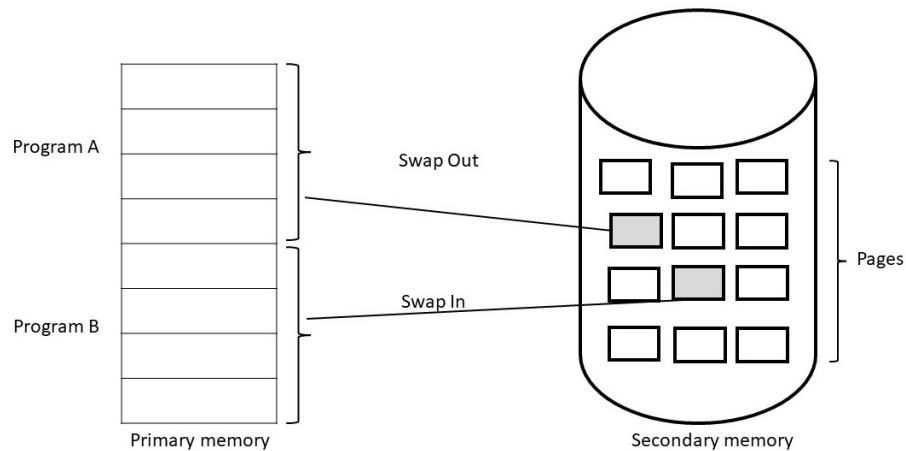


FIGURE 3.19

### 3.11.1 Steps to handling a page fault

When there is a page fault, the currently executing process must stop until page is loaded into main memory. The page fault handler which is a module of the operating system gets the page number and address of page table of the missing page to bring into the main memory from secondary storage. Here are the steps followed by operating system to handle page fault:

- An internal table is checked to verify whether a memory reference is valid or invalid, when a process refers to any page in the main memory.
- If it is valid but the page is not there then page is loaded to physical memory.
- To load a missing page a free space location is identified.
- After finding the free location the whole page is loaded into physical memory.
- When page is in main memory, the internal table and map table is updated to indicate the page.
- Now, the system restarts the instruction which was terminated due to page fault.

It is a resource management technique which allows the parent and child process to share pages. This technique minimizes the number of pages due to shared pages. If there is any modification in parent or child process only then the page is copied. The fork () system call is used to create the copy of process called child process.

- Copy-on- Write is used by many operating systems.
- Only modified pages are copied so this technique is efficient.
- Many operating-systems provide pool of free pages, can be used by copy-on-write technique while creating the duplicate pages.

## 3.13 PAGE REPLACEMENT

If there are multiple processes executing in main memory and the page fault occurs and if no free page frame is available then operating system has to delete or remove currently loaded page so that new page can be loaded. This technique is called **page replacement**. Which page should be replaced from memory? This can be decided by some of the page replacement algorithms.

### 3.13.1 FIFO Page Replacement Algorithm

First-in, First-out (FIFO), algorithms remove the page from the memory which is residing from longest time. Due to the probability its performance is poor as some of the processes are in constant use throughout life. FIFO queue is used to hold the page in the memory. A page is loaded at the rear and replaced at the front of the queue. In this algorithm sometime there is **belady anomaly**. This anomaly occurs when page fault increases due to number of allocated frame increases.

#### Example 3.1

Consider three frames with the following reference string:

2    1    2    3    4    2    5    1    6    3

The following figure shows how the FIFO algorithm works:

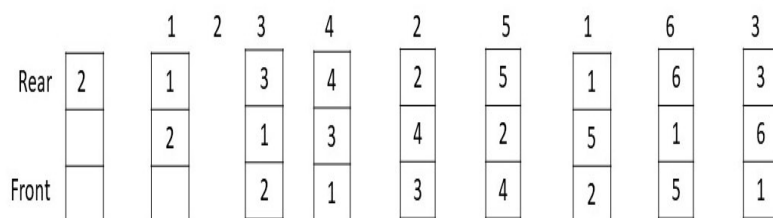


FIGURE 3.20

### 3.13.2 Optimal Algorithm

This algorithm replaces the page that will not be used for longest period of time. Page fault rate is lowest in this algorithm. Belady's anomaly never affects this algorithm but it needs to know the reference string in advance so it is difficult to implement.

#### Example 3.2

Consider three frames with the following reference string:

7    1    4    3    4    2    5    1    3    6

The following figure shows how the Optimal algorithm works:

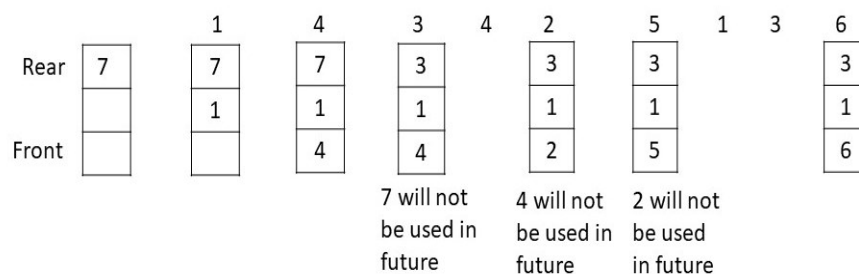


FIGURE 3.21

### 3.13.3 LRU Page Replacement Algorithm

Least recently used (LRU) algorithm replaces the page that has not been used from longest period of time. It is implemented by using stack. In the top of the stack there is always most recently used page and, in the bottom, least recently used page. It is best implemented with doubly linked list with a head and a tail.

#### Example 3.3

Consider three frames with the following reference string:

2    1    2    3    2    4    2    6

The following figure shows how the LRU algorithm works:

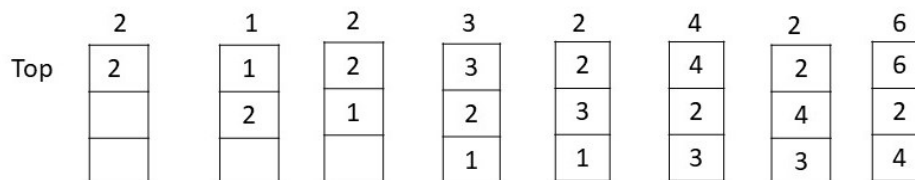


FIGURE 3.22

### 3.13.4 Second Chance Page Replacement

It is modified version of FIFO. In this algorithm a **page reference** bit is associated with each page frame. If the bit is 0 then the page is unused and old and if it is 1, it means it just had arrived in the memory. Then this algorithm finds the old page and replace them. This is implemented by linked list.

#### Example 3.4

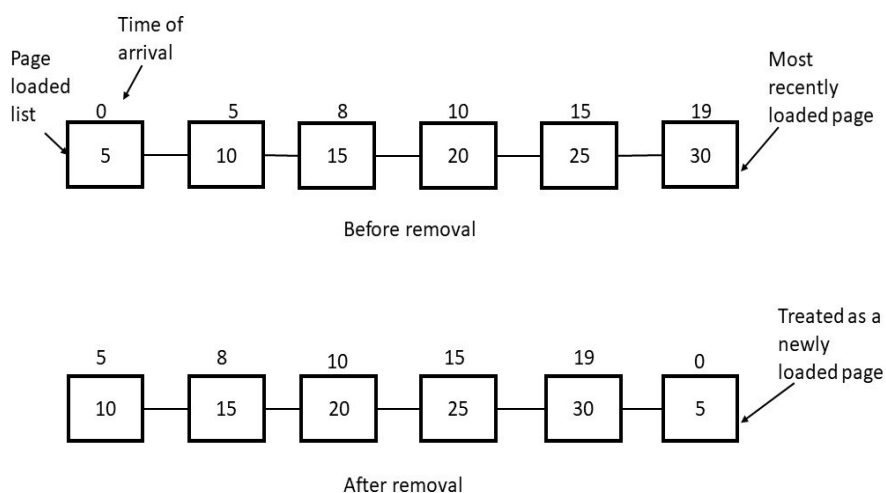


FIGURE 3.23

### 3.13.5 Clock Page Replacement Algorithm

The pages move around on its list in the second chance page replacement algorithm. To avoid this circular list is maintained with a pointer to the page in clock page replacement algorithm.

#### Example 3.4

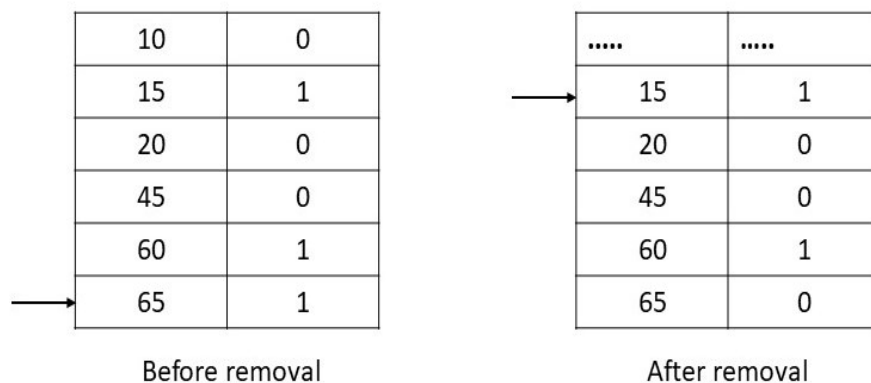


FIGURE 3.24

---

## 3.14 ALLOCATION OF FRAMES

---

The primary memory is divided into frames. The operating system provides many algorithms to allocate frames for every process. The following are some ways to allocate frames:

- **Proportional frame allocation:** - It allocates the frames according to the size which is required for execution.
- **Priority frame allocation:** - It allocates the frames according to the process's priority. And required number of frames.
- **Global replacement allocation:** - This method takes care the page fault in the operating system. The lower priority process gives the frames to higher priority processes.
- **Local replacement allocation:** - In this method, the frame of pages can be loaded on same page.
- **Equal frame allocation:** - In this method frames are equally distributed among processes. The problem can occur when any process required more frames.

---

## 3.15 THRASHING

---

Sometimes the memory is full with pages and no free space left for new required pages then the page fault occurs. Even after swapping the pages, page fault can be occurring continuously due to the swapped-out page required for the execution. This situation is called **thrashing** where system is continuously processing page faults and executing instructions.

Where many processes are running at the same time like in multitasking and multiprogramming operating systems, thrashing occurs. It lesser the performance of the system.

### 3.15.1 Cause of Thrashing

In multiprogramming, many processes are introduced to system so that CPU utilization can be increased. Now many processes need pages in the memory and sometime there is no free space in the memory then the system continuously swapped-in and swapped-out the pages from the memory. This leads to more page faults. When CPU scheduler tries to increase the degree of multiprogramming, then the utilization of CPU decreases as shown in figure. this situation leads to thrashing where system continuously working on page faults.

### 3.15.2 Methods to Handle Thrashing

The following are some methods to handle thrashing:

**Local replacement algorithm:** - By using local replacement algorithm, the process swaps the page which belongs to same process and it does not take frames form another process. So that the process can execute properly.

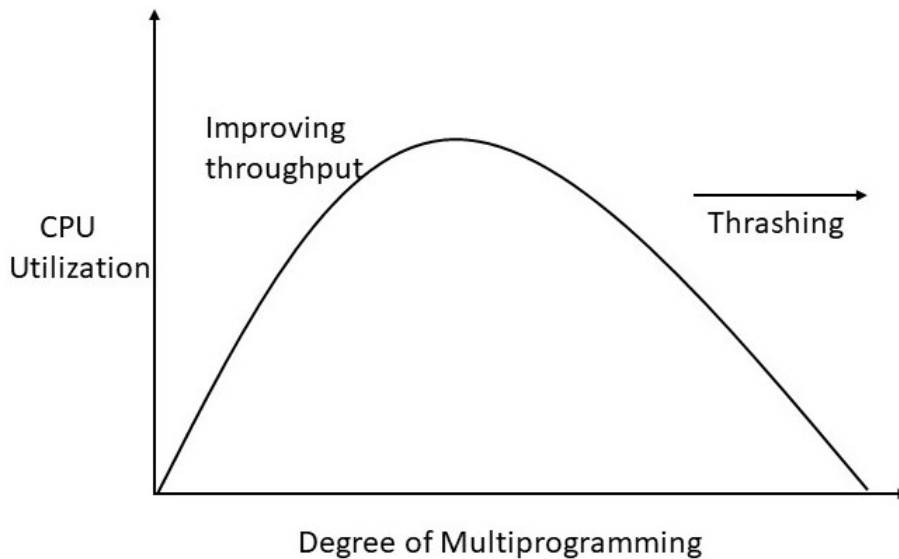


FIGURE 3.25

**Allocating frames to process as required:** - If a process have as many frames it requires, then the thrashing can be prevented. There are several techniques to do this. **Working-set model** is one of them. It is based on locality. It is a collection of pages referred by process at a recent interval of time.

**Controlling the page-fault frequency:** - Another method to handle the thrashing is by controlling the page-fault frequency. It can be controlled by establishing upper and lower bounds on the desired page faults as shown in figure. If the page fault exceeds upper limit, allocate more frames and if it is below than lower limit, remove the extra frames.

---

### 3.10 SUMMARY

---

**MMU:** - Memory Management Unit (MMU) is a hardware device that is located in the CPU. It maps the logical address to the physical address.

**Cache:** -The CPU can access the registers in one cycle of CPU clock, but in case of main memory it takes many cycles of the CPU clock because main memory can be accessed via memory bus. To overcome this problem a fast memory called *cache*, is added between main memory and CPU.

**Logical Address Space:** - When a program generates the set of all logical addresses is called *Logical Address Space*.

**Physical Address Space:** - The set of physical addresses corresponding to logical addresses of program is called *Physical address space* of that program.

**Process Loading:** - A process that is currently executing must be in the main memory (RAM). The process has to be first loaded from secondary memory (hard disk) to primary memory or main memory before execution. This technique is called *process loading*.

**Paging:** - Paging is a technique of memory management, which can be the possible solution to external fragmentation. This is also called paged memory management.

**Fragmentation:** - The processes free the memory space after the completion of execution and that memory space is not big enough for other processes, then there occurs the problem of *fragmentation*.

**Memory Compaction:** - To combine free blocks (holes) and move downward all the processes, to create enough possible space for any new process is called *memory Compaction* or *relocatable partitioned memory management*.

**Segmentation:** - The collection of logically related information in variable sized segments is called *segmentation*. The logical address is divided into segment number and segment offset.

**Page Fault:** - If the bit is set to IN, then the page is in the main memory and if bit is OUT the page is not in the primary memory rather it is in secondary memory. Due to the OUT bit, an interrupt occurs which is called *page fault*.

**Page Replacement:** - If there are multiple processes executing in main memory and the page fault occurs and if no free page frame is available then operating system has to delete or remove currently loaded page so that new page can be loaded. This technique is called *page replacement*.

**Thrashing:** - Sometimes the memory is full with pages and no free space left for new required pages then the page fault occurs. Even after swapping the pages, page fault can be occurring continuously due to the swapped-out page required for the execution. This situation is called *thrashing* where system is continuously processing page faults and executing instructions.

1. What is the difference between logical address space and physical address space?
2. What do you mean by MMU?
3. Explain Swapping.
4. Explain the types of memory allocation techniques.
5. What do you mean by contiguous memory allocation? Explain the schemes used in contiguous memory allocation.
6. Explain the difference between internal and external fragmentation.
7. What do you mean by storage policies?
8. What do you mean by page table implementation in paging?
9. Define memory compaction.
10. Explain the techniques used in structuring the page table.
11. What do you mean by protection and sharing in segmentation?
12. Write the advantages and disadvantages of segmentation.
13. What do you mean by best fit, first fit and worst fit?
14. What are the advantages and disadvantages of memory compaction?
15. What do you mean by swap in and swap out?
16. Explain all the page replacement algorithms with examples.
17. What do you mean by thrashing? Explain the methods to handle thrashing.
18. Explain virtual memory.
19. What do you mean by demand paging?
20. What is copy-on-write?

\*\*\*\*\*

## MASS-STORAGE STRUCTURE

### Unit Structure

- 4.0 Objectives
- 4.1 Overview of Mass Storage Structure
  - 4.1.1 Magnetic Disks
  - 4.1.2 Magnetic Tapes
- 4.2 Disk Structure
- 4.3 Disk Attachment
  - 4.3.1 Host-Attached Storage
  - 4.3.2 Network-Attached Storage
  - 4.3.3 Storage-Area Network
- 4.4 Disk Scheduling
- 4.5 Disk Scheduling Algorithms
  - 4.5.1 First-cum, First-Serve Scheduling
  - 4.5.2 Shortest-Seek Time First Scheduling
  - 4.5.3 SCAN Scheduling
  - 4.5.4 Circular SCAN Scheduling
  - 4.5.5 LOOK Scheduling
  - 4.5.6 Circular LOOK Scheduling
  - 4.5.7 Selection of Disk Scheduling Algorithms
- 4.6 Disk Management
  - 4.6.1 Disk Formatting
  - 4.6.2 Boot Block
  - 4.6.3 Bad Block
- 4.7 Summary
- 4.8 Exercises

---

### 4.0 OBJECTIVES

---

After Completion of this chapter, you will be able to understand:

- Performance of mass storage structure.
- All the disk scheduling algorithms.
- Examples of disk scheduling algorithms.
- Choosing a disk Scheduling Algorithms.
- Management of disk structure.

Main memory is the temporary storage in the computer but mass storage devices retain the data even when system is switched off. Mass storage devices are used to store large amount of data permanently. These devices are also called secondary storage or auxiliary storage. Let us discuss the physical structure of basic secondary storage devices.

### 4.1.1 Magnetic Disks

Magnetic disk is a secondary storage device. The shape of magnetic disk is like a CD. The flat area of disk is covered with magnetic material. Magnetic disks can store large amount of data and its price is low as compared to primary memory. The disk has many platters of circular shape. Average range of platter in diameters is from 1.8 to 5.25 inches. The information is stored magnetically on platters. Every platter has its read-write head. Platters are logically divided into tracks and sectors. In this type of storage data can be deleted or modified. Magnetic disk also allows to access data randomly but its access rate is slower than main memory.

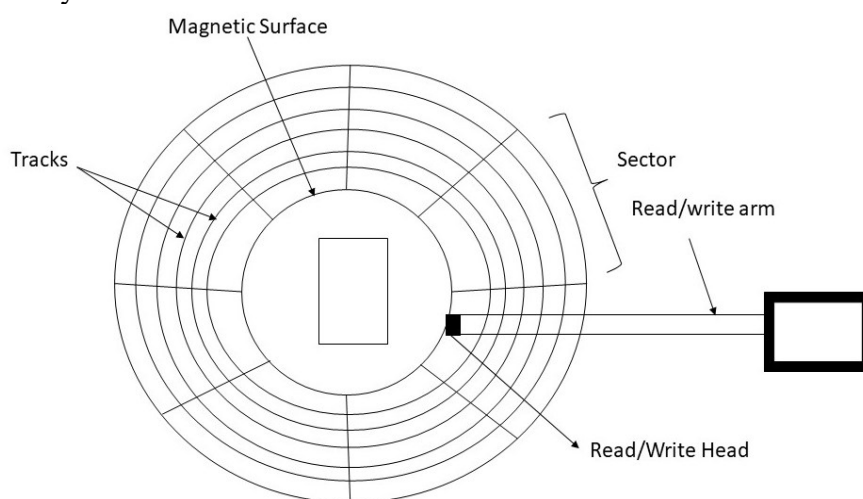


FIGURE 4.1 Magnetic Disk

#### Advantages

- Data can be accessed randomly.
- Magnetic disks are cheaper than primary memory (RAM).
- Data can be store in large number.
- Rate of data transfer is better as compared to magnetic tape.

#### Disadvantages

- To carry magnetic disk, environment should be dust free.
- This type of memory is not suitable for sequential access.
- More expensive than magnetic tape.

### 4.1.2 Magnetic Tapes

Magnetic tape is also a secondary storage device. This memory is useful to access sequential data. A ribbon that is coated with magnetic oxide is used to store the data. The speed of magnetic tape memory is slow due to its sequential access. Magnetic tapes can also store large amount of data. This memory has the storage capacity from range 100 MB to 200 GB and the size of ribbon varies from 4mm to 1 inch.

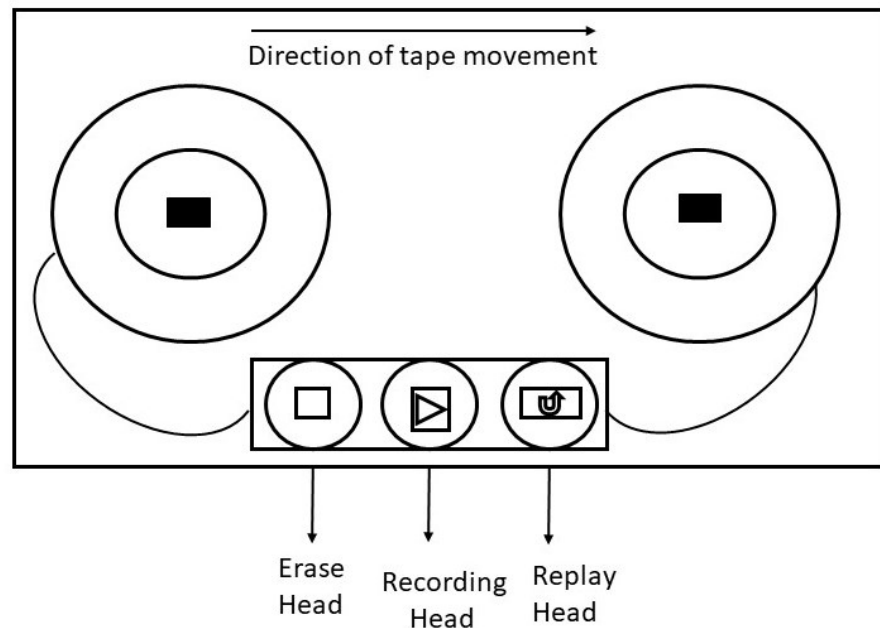


FIGURE 4.2 Magnetic Tape

#### Advantages

- Large number of files can be store.
- It can be reused.
- These memories are of low cost.
- Can be suitable for backup storage.
- Its size is compact and easy to store.

#### Disadvantages

- Does not allow the direct access.
- Difficult to modify and update.
- It requires clean environment to store.

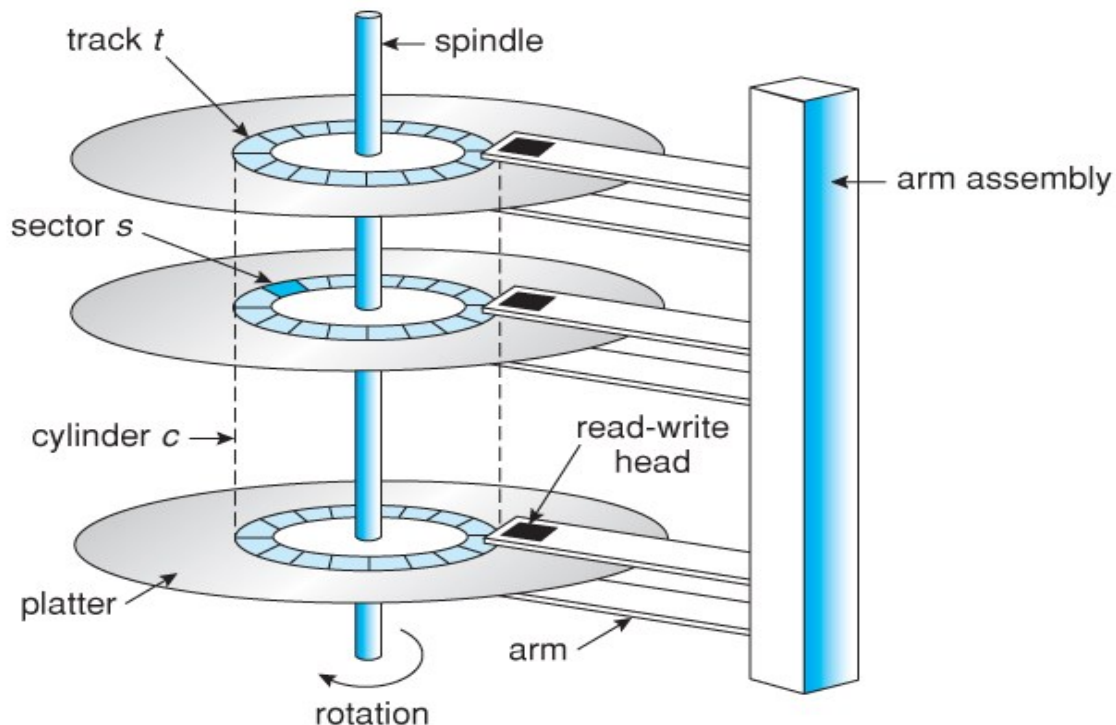
---

## 4.2 DISK STRUCTURE

---

The disk is made up from multiple platters and platters are divided into multiple tracks which are then divided into multiple sectors. The platters are accessed through head arms connected with head and can move only in two directions. All the heads move together on the platter and head is

always located at same track number or cylinder number. The set of all tracks where the head is located currently is referred as cylinder. The speed of the disk drive rotating on the disk is constant. The read/write operation is performed when head is positioned at a particular location.



**Figure 4.3 Disc structure**

If a process needs an input/output resource, it gives a system call to the operating system. The request can be served immediately if that desired controller is available otherwise the process will have to be queued.

The time needed to access the particular location and transfer the data is called **access time**.

The access time has the following three components: -

- **Seek time:** - The time needed to position the read or write head at a particular track is called the seek time. For example, if the read/write head is on track 3 and it has to be positioned at 7, then read/write head has to move from track 3 to 7.
- **Latency time:** - The time needed to position the read or write head at the desired sector, when the head is already at the particular track is called latency time or rotational delay.
- **Data Transfer Time:** - How much actual time is required to transfer the data is called the data transfer time.

---

## 4.3 DISK ATTACHMENT

---

Disk storage can be accessed in two ways by the system. One is through I/O port or host-attached storage and other is through remote host or network attached storage.

### 4.3.1 Host-Attached Storage

It is a simple way used in small systems. This storage can be accessed via local I/O ports. These ports have different technologies. The system uses I/O bus architecture called IDE. The architecture of this supports maximum two drives per bus. Some systems use more advanced architecture such as fiber channel. The devices like hard disk drives, CD, DVD and tape drives can be used as host-attached storage.

### 4.3.2 Network-Attached Storage (NAS)

This is a special purpose storage device that can be remotely accessed over a network. Different protocols like TCP or UDP are used for procedure call on same LAN. The computer that are connected on a LAN as shown in figure 4.3, network-attached storage gives a simpler way to share a space for storage with name and accessed through host-attached storage. It is less-efficient and its performance is also lower as compared to some directly attached storage options.

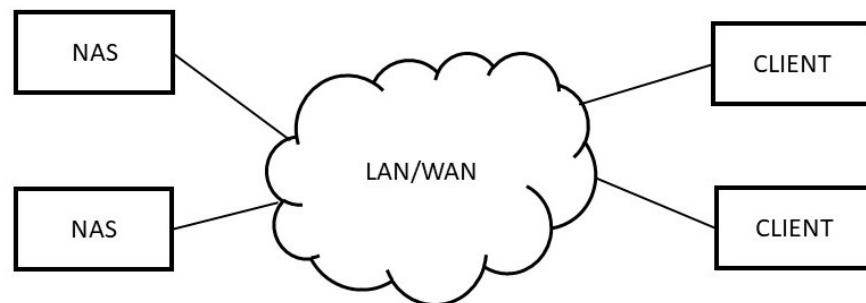


FIGURE 4.4 Network-Attached Storage

### 4.3.3 Storage-Area Network (SAN)

The disadvantage of network-attached storage is that its I/O operations consume lot of bandwidth on the network. A storage-area network connects the servers and storage units using storage protocols as shown in figure 4.4. It is a private network. The storage area network is flexible. Many hosts and storage arrays can connect to same SAN and storage allocated to hosts can be done dynamically. It can switch access between hosts and storage.

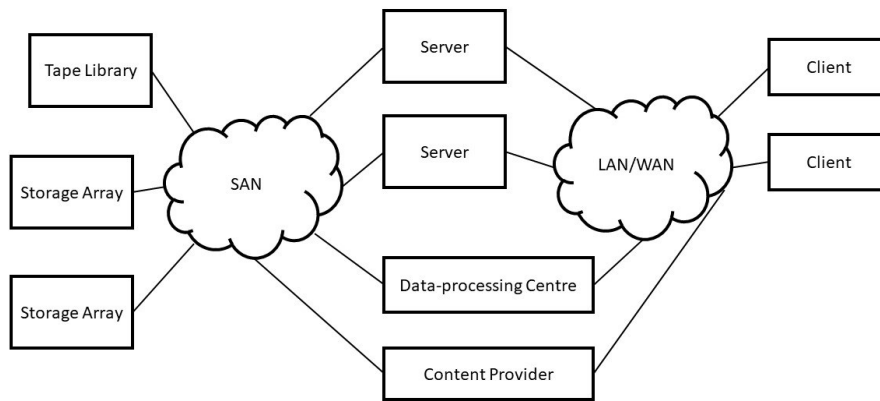


FIGURE 4.5 Storage-Area Network

---

## 4.4 DISK SCHEDULING

---

To use the hardware efficiently, disk scheduling plays the major role. This scheduling is also called I/O scheduling. The purpose of disk scheduling is to increase the access time. When there are many requests from processes for CPU or I/O. The process requests operating system to access the disk for I/O. Sometime it is tough for the operating system to satisfy the request of every process, then the technique disk scheduling is used. The several disk scheduling algorithms are used so that the CPU can make the choice from the pending processes.

---

## 4.5 DISK SCHEDULING ALGORITHMS

---

The objectives of disk scheduling algorithms are:

- The response time is minimized.
- Throughput is increased i.e., number of requests increased per unit of time.

The following are the disk scheduling algorithms.

1. FCFS (First-come-first-served) Scheduling
2. SSTF (Shortest-Seek Time First) Scheduling
3. SCAN Scheduling
4. Circular SCAN Scheduling
5. LOOK Scheduling
6. Circular LOOK Scheduling

### 4.5.1 FCFS Scheduling

FCFS (First-come-first-served) Scheduling is a simplest technique used in disk scheduling. In FCFS, which request comes first will be served first. This scheduling increases the response time as its schedule is fixed. This scheduling is fair but its service is slow. This technique involves lots of head and disk movements randomly.

**Example 4.1**

Disk Queue with requests for I/O on cylinders is in following order:

70, 125, 40, 150, 20, 118

- Disk head is at 50.
- Cylinders are 200 numbered from 0-199

Initially the head is at 50, and then it will move from 50 to 70, then 70 to 125, then to 40, 150, and 10 and at the end to 118.

In this scheme there are lot of head movements, see figure 4.2. So, this algorithm does not provide the fastest service.

Let us calculate the head movements as:

$$\begin{aligned}
 &|50 - 70| + |70 - 125| + |125 - 40| + |40 - 150| + |150 - 20| + |10 - 118| \\
 &= 20 + 55 + 85 + 110 + 130 + 108 \\
 &= 508
 \end{aligned}$$

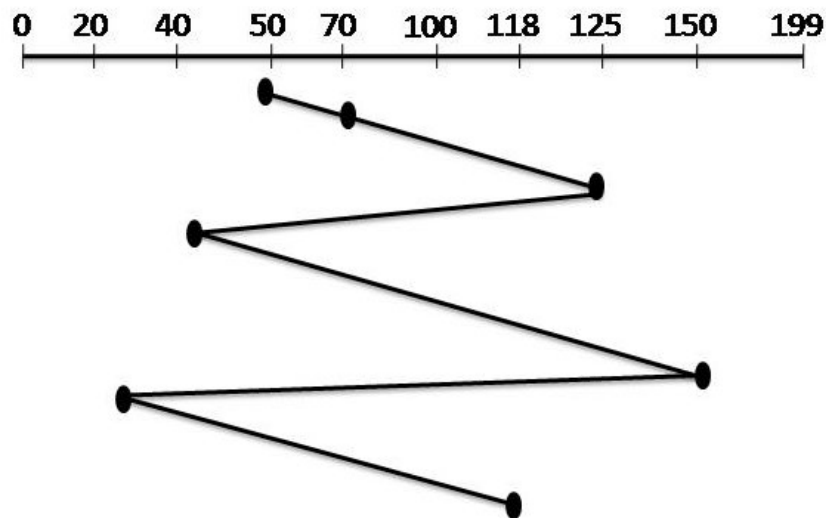


FIGURE 4.6

**4.5.2 SSTF (Shortest-Seek-Time-First) Scheduling**

In SSTF scheduling, the request with shortest seek distance is served first, even if it comes last in the queue. The disk controller selects the next request which is close to the current location of the head. In this algorithm, innermost and outermost tracks have poor service as compared to mid-range tracks. The advantage of this scheduling is increased throughput and mean-response time. The disadvantage is starvation as innermost and outermost tracks may have to wait for long time.

### Example 4.2

Disk Queue with requests for I/O on cylinders is in following order:

70, 125, 40, 150, 20, 118

- Disk head is at 50.
- Cylinders are 200 numbered from 0-199

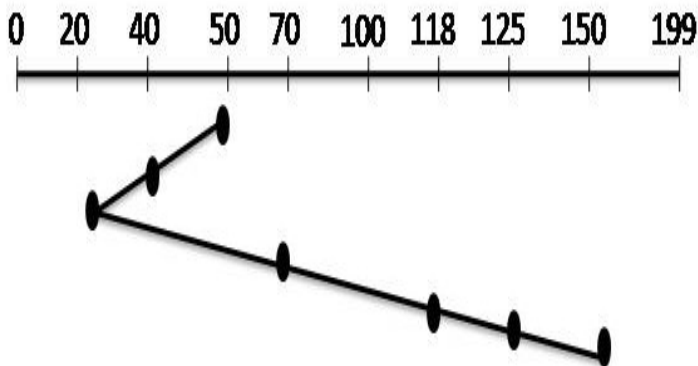


FIGURE 4.7

The closest request to head cylinder 50 is at cylinder 40. From cylinder 40 closest is 20 then next closest is 70, then 118, 125 and finally 150 as shown in figure 4.3

Let us calculate the head movements as:

$$|50 - 40| + |40 - 20| + |40 - 70| + |70 - 118| + |118 - 125| + |125 - 150|$$

$$= 10 + 20 + 30 + 48 + 7 + 25$$

$$= 140$$

#### 4.5.3 SCAN Scheduling

SCAN algorithm works like SSTF scheduling except it chooses the shortest seek distance in one direction. If the requested track is on another direction, then it does not change the direction until it reaches the outermost cylinder. This algorithm is also called elevator algorithm because it works like elevator as it continuous in one direction until there are no any requests left and then it changes its direction. The advantage of SCAN algorithm is improved mean response time and throughput. Due to oscillating motion of its head in SCAN algorithm, the outer tracks are less visited than mid-range tracks.

**Example 4.3**

Disk Queue with requests for I/O on cylinders is in following order:

70, 125, 40, 150, 20, 118

- Disk head is at 50.
- Cylinders are 200 numbered from 0-199

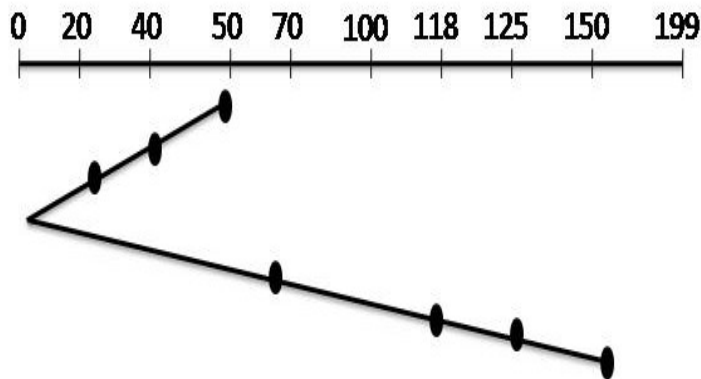


FIGURE 4.8

As in figure 4.4, first the head moves from 50 to 40, the closest one. Then head moves from 40 to 20. From cylinder 0, the head arm will reverse its position and moves to tracks 70, 118, 125 and 150.

Let us calculate the head movements as:

$$\begin{aligned}
 &|50 - 40| + |40 - 20| + |20 - 0| + |0 - 70| + |70 - 118| + |118 - 125| + |125 - 150| \\
 &= 10 + 20 + 20 + 70 + 48 + 7 + 25 \\
 &= 200
 \end{aligned}$$

**4.5.4 C-SCAN Scheduling**

C-SCAN or Circular SCAN is an improvement in SCAN scheduling algorithm. In C-SCAN, the head moves from current position to forward direction position, servicing the requests till it reaches to last track, then it immediately returns to the first track of the disk, without servicing any request at the time of return.

**Example 4.4**

Disk Queue with requests for I/O on cylinders is in following order:

70, 125, 40, 150, 20, 118

- Disk head is at 50.
- Cylinders are 200 numbered from 0-199

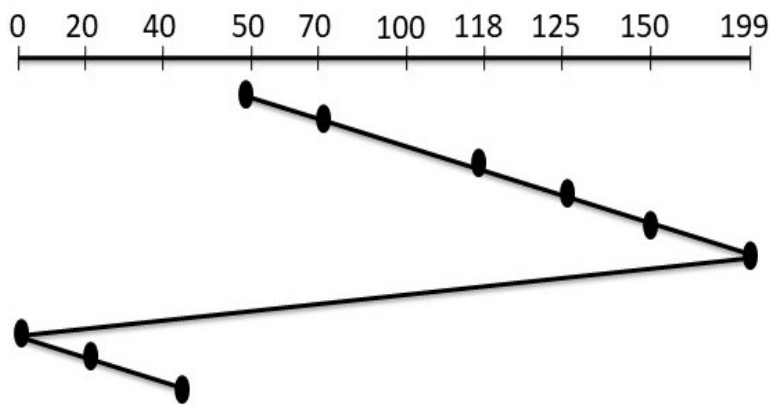


FIGURE 4.9

In this algorithm, the head will move from track 50 to 70, then to 118, 125, 150 and the outermost track 199. After reaching to the end of disk it will return to starting of the disk at track 0, then it will move from 0 to 20 and 20 to 40 as shown in figure 4.5.

Let us calculate the head movements as:

$$\begin{aligned}
 & |50 - 70| + |70 - 118| + |118 - 125| + |125 - 150| + |150 - 199| + |199 - 0| + |0 - 20| + \\
 & |20 - 40| \\
 &= 20 + 48 + 7 + 25 + 49 + 199 + 20 + 20 \\
 &= 388
 \end{aligned}$$

#### 4.5.5 LOOK Scheduling

In LOOK scheduling, the head arm moves in one direction servicing every request till no more request pending, then it reverses the direction. LOOK is the modified version of SCAN scheduling. The difference is that in SCAN head moves to innermost and outermost cylinders but in LOOK it only serves the requests in one direction than the other direction.

#### Example 4.5

Disk Queue with requests for I/O on cylinders is in following order:

70, 125, 40, 150, 20, 118

- Disk head is at 50.
- Cylinders are 200 numbered from 0-199

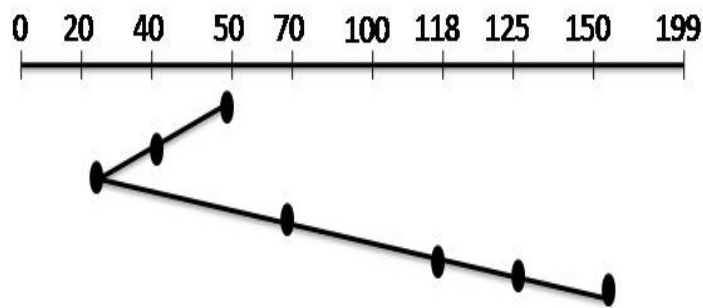


FIGURE 4.10

Let us calculate the head movements as:

$$\begin{aligned}
 & |50 - 40| + |40 - 20| + |40 - 70| + |70 - 118| + |118 - 125| + |125 - 150| \\
 &= 10 + 20 + 30 + 48 + 7 + 25 \\
 &= 140
 \end{aligned}$$

#### 4.5.6 C-LOOK Scheduling

Circular LOOK scheduling is a modified version of Circular SCAN scheduling. The difference between both the scheduling is that in C-SCAN scheduling head arm moves across the full disk from outermost to innermost cylinders. But in C-LOOK scheduling, head only serves request in each direction.

#### Example 4.6

Disk Queue with requests for I/O on cylinders is in following order:

70, 125, 40, 150, 20, 118

- Disk head is at 50.
- Cylinders are 200 numbered from 0-199

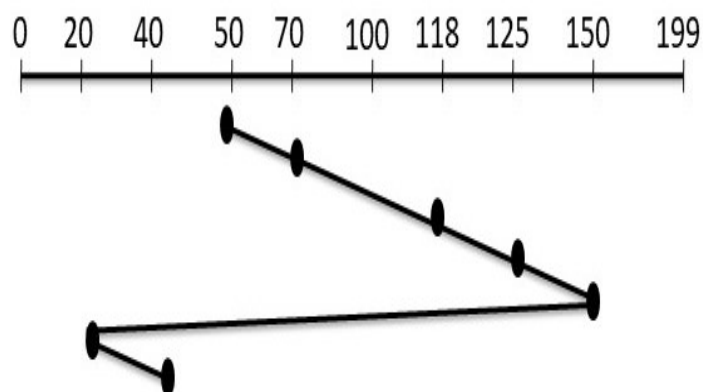


FIGURE 4.10

Let us calculate the head movements as:

$$|50 - 70| + |70 - 118| + |118 - 125| + |125 - 150| + |150 - 20| + |20 - 40|$$

$$= 20 + 48 + 7 + 25 + 130 + 20$$

$$= 250$$

#### 4.5.7 Disk Scheduling Algorithm Selection

The following are some criteria's to choosing disk scheduling algorithm.

Every scheduling algorithm has its own advantage and disadvantages.

- SSTF is common scheduling algorithm, its performance is better than FCFS.
- Performance of SCAN, C-SCAN, LOOK and C-LOOK algorithms are better than others as there are less chances of starvation problem.
- The performance of the algorithm depends upon the types and number of requests.
- Some algorithms focus on throughput, but some focus on the response time.
- The choice of algorithm also depends on the type of system and the types of I/O requests.
- If there is only one request in the queue, then all the scheduling algorithm behave like FCFS because there is no other choice.

---

## 4.6 DISK MANAGEMENT

---

Disk management is a utility program in an operating system which handles the many aspects of disk drive. An operating system is responsible for booting, initialization, recovery, partitions etc for the disk. Disk management involves some of the following responsibilities.

### 4.6.1 Disk Formatting

- A blank new disk coated with magnet is just a platter.
- The disk must be divided into tracks then tracks must be divided into sectors so that disk controller can read/write the data on the disk. This process referred as physical formatting or low-level formatting.
- When disk controller formats the disk, it asks for the space for header and trailer of all sectors.
- In the header and trailer, error-correcting codes and sector numbers are stored, which is used by disk controller.
- Many hard-disks are already low-level formatted.
- Before using the disk, first an operating system partitioned it into group of cylinders. In one partition, there is a copy of operating system's code and another partition holds user's files.
- Next is logical formatting in which file system is created.

- In this step, an operating system data structure of file-system on the disk.
- The data structure of file system includes the information about free space, allocated space or empty directory.

#### 4.6.2 Boot Block

- To start a computer, an initial program must run. The process of starting a computer is called booting process.
- The bootstrap program fetches all the aspects from main memory and then starts the operating system.
- First it initializes the kernel into main memory and from first address it begins the execution of operating system.
- The bootstrap is stored in ROM (Read-Only Memory), as it is read only memory so there are no chances of virus.
- If there is a requirement of changing the bootstrap code then ROM hardware chips needs to be changed. So, most system stores small bootstrap loader program in ROM and the full program is stored on disk.
- In the disk with full bootstrap programs is called “boot blocks”.
- Bootstrap program is stored in system disk.

#### 4.6.3 Bad Blocks

- Sometimes parts of the disk are prone to failure as they have rotatable parts and their tolerance is very small. So, these are called bad blocks.
- If there is full failure then the disk needs a back up to store the contents.
- Some disks come defective (bad block) from the factory.
- There are many ways to handle these bad blocks depends upon the disk controller.
- In some disks bad blocks are handled manually like some disk with IDE controllers.
- Some disks like SCSI, handled bad blocks using high-end workstations and are smart about recovery. In this the controller maintains the list of bad blocks.
- At the time of the formatting, the controller replaces the bad sector with spare sectors. This technique is called forwarding or sector sparing.
- Manual intervention is required in bad block recovery.

---

### 4.7 SUMMARY

---

**Mass-Storage Structure:** - Mass Storage devices are the secondary storage or memory. These can store large amount of data. These can store data permanently unlike primary storage. There are many secondary storage devices used to store data like magnetic disk, magnetic tape, CD, DVD etc.

**Disk Structure:** - The disks are divided into tracks and tracks are divided into sectors before using it. Read/write head is used to read and write data on the disk.

**Disk Scheduling Algorithm:** - Disk Scheduling algorithms are used to increase the throughput and access time. Disk scheduling is also called I/O scheduling. Following are the different types of scheduling algorithms:

- FCFS Scheduling
- SSJF Scheduling
- SCAN Scheduling
- C-SCAN Scheduling
- LOOK Scheduling
- C-LOOK Scheduling

**Selection of disk scheduling algorithm:** - There can be different parameters for selection of a disk scheduling algorithm. It depends on the system and number of requests arrived.

**Disk Management:** - It is the responsibility of the operating system to manage the disk. These includes boot block, bad blocks, Disk formatting. It manages the disk as how the data will store on disk.

**Access Time:** - The time needs to access the particular location and transfer the data.

**Seek time:** - The time need to position the read or write head at a particular track is called the seek time. For example, if the read/write head is on track 3 and it has to be positioned at 7, then read/write head has to move from track 3 to 7.

**Latency time:** - The time need to position the read or write head at the desired sector, when the head is already at the particular track is called latency time or rotational delay.

**Data Transfer Time:** - How much actual time is required to transfer the data is called the data transfer time.

---

## 4.8 EXERCISES

---

1. What do you mean by Mass-Storage Structure?
2. What are the advantages and disadvantages of Magnetic Disk?
3. What are the advantages and disadvantages of Magnetic Tape?
4. Explain the components of access time.
5. What do you mean by Disk Scheduling?
6. Explain the objectives of the disk scheduling algorithms?
7. What do you mean by disk structure?
4. A Disk cylinder has 3000 cylinders, from 0 to 2999. The head is currently at cylinder 150. The following queue is in FIFO order:  
85, 1370, 950, 1675, 870, 1410, 1020, 1680, 140

What is the total distance that the disk arm moves for all of the following Algorithms?

- FCFS
- SSJF
- SCAN
- C-SCAN
- LOOK
- C-LOOK

9. Explain the criteria for selecting a disk-scheduling algorithm?
10. What do you mean by disk management? Explain its responsibilities.
11. Explain network-attachment Storage?
12. A Disk cylinder has 2000 cylinders, from 0 to 1999. The head is currently at cylinder 1000. The following queue is in FIFO order:

75, 1170, 920, 1475, 870, 1210, 820, 1380, 120

What is the total distance that the disk arm moves for all of the following Algorithms?

- FCFS
- SSJF
- SCAN
- C-SCAN
- LOOK
- C-LOOK

13. What do you mean by disk formatting?
14. Explain the Disk Structure.
15. A Disk has 200 cylinders, from 0 to 199. The head is currently at cylinder 55. The following queue is in FIFO order:

70, 20, 130, 40, 155, 10, 185

What is the total distance that the disk arm moves for all of the following Algorithms?

- FCFS
- SSJF
- SCAN
- LOOK

\*\*\*\*\*

## FILE SYSTEM

### Unit Structure

- 5.0 Objectives
- 5.1 File Management- Introduction
- 5.2 File Concept
  - 5.2.1 File Naming
  - 5.2.2 File Attributes
  - 5.2.3 File Operations
  - 5.2.4 File Types
  - 5.2.5 File Structure
- 5.3 Access Methods
  - 5.3.1 Sequential Access
  - 5.3.2 Random Access
  - 5.3.3 Indexed Sequential Access
- 5.4 Directory and Disk Structure
  - 5.4.1 Single-level Directory
  - 5.4.2 Two-Level Directory
  - 5.4.3 Hierarchical Level Directory
  - 5.4.4 Acyclic Graph Directory Structure
  - 5.4.5 General Graph Directory
- 5.5 File-System Mounting
- 5.6 File Sharing
  - 5.6.1 I-Node Linking
  - 5.6.2 Symbolic Linking
- 5.7 File-System Structure
- 5.8 File System Implementation
  - 5.8.1 Overview
- 5.9 Directory Implementation
  - 5.9.1 Linear List
  - 5.9.2 Hash Table
- 5.10 Allocation Methods

- 5.10.1 Contiguous Allocation
- 5.10.2 Linked Allocation
- 5.10.3 Indexed or i-node Allocation
- 5.11 Free - Space Management
  - 5.11.1 Bit Vector
  - 5.11.2 Linked List
  - 5.11.3 Grouping
  - 5.11.4 Counting
- 5.12 Summary
- 5.13 Exercises

---

## 5.0 OBJECTIVES

---

After Completion of this chapter, you will be able to understand:

- Functions and interfaces of the file system.
- Access and Allocation methods of the file system.
- Directory structure and file sharing concepts.
- Structure of file system and their implementation.
- How to manage the free space.

---

## 5.1 FILE MANAGEMENT - INTRODUCTION

---

In the computer system, large amount of data is stored and retrieved. All the applications need to store and fetch the information in the computer. It is essential to store the data for long-term and make data sharable so that various processes can access the data at any time. The size of data is large so some appropriate storage devices must be used.

To match these requirements, the information needs to store on disk in units called **files**. The data stored in files must be persistent, so that it is not affected by the processes. Processes can read/write the files according to their need or requirement. Files are handled by the operating system. Operating system design deals with how the files are structured, used, accessed, named and implemented. That part of operating system which deals with files is known as **file system**.

---

## 5.2 FILE CONCEPT

---

A collection of related data and information stored on secondary memory such as optical disks, magnetic disks and magnetic tapes is known as a *file*. Many different types of information can be stored in a file like text, images, graphics, codes, numeric and so on. The meaning of the data in file is defined by its user or file's creator. A file is a series of bits, bytes, lines or records.

### 5.2.1 File Naming

The following are some rules for naming a file that varies from system to system:

- File names should be string of 8 letters. For example, 'name', 'age', 'class' etc.
- Some operating systems supports two parts of a file name. First is 'file name' and second is 'file extension'. Both parts are separated by period (.).

For example, **class.cpp**

- Keywords or reserved words of the operating system cannot be used as a file name.
- Many operating systems does not allow blank spaces in a file name.
- In some operating systems Upper- and lower-case letters are same, but in some like UNIX upper- and lower-case letters are different. So, they choose the file name accordingly.
- The following are some common file extensions:

| Extension of the file | Meaning                |
|-----------------------|------------------------|
| file.c                | C source program       |
| file.bak              | Backup file            |
| file.exe              | Executable file        |
| file.doc              | Document file          |
| file.dat              | Data file              |
| file.txt              | Text file              |
| file.lib              | Library of object file |

**Table 5.1**

### 5.2.2 File Attributes

When a file is created, a lot of information regarding the file is created like its name, date of creation, size and so on. These items are called **file's attributes**. The following are some of the file attributes which varies from system to system:

- a) **File name:** - A file name is referred by its name and created for the convenience of its users. It is a string of characters like “name.c”. The rules for the file name depend upon the operating system.
- b) **File Size:** - The size of the file can be in bytes, words or blocks.
- c) **File Type:** - This attribute is needed where system supports different file types.
- d) **Location:** - This attribute stores the location of the file where it is stored in secondary storage.
- e) **Access Rights:** - This attribute stores the information of how the file can be accessed and who can access the file.
- f) **Time, Date and identification:** - This attribute stores the information about the creation of file, last modification and last use. This can be useful for the protection and security of files.
- g) **Flags:** - Flags are set by operating system for some specific properties. The following are some flags attributes:

| Flag Attribute | Meaning                               |
|----------------|---------------------------------------|
| Lock flag      | 0 for unlock file and 1 for lock file |
| System flag    | 0 for normal and 1 for system file    |
| Hidden flag    | 0 for normal and 1 for hidden file    |
| Read-only flag | 0 for normal and 1 for read only      |
| Binary flag    | 0 for ASCII and 1 for binary          |

Table 5.2

### 5.2.3 File Operations

There are some operations that can be performed on files to defined them properly. Different operating systems provide different operations to store and fetch the data or information. The following are some common operations of files:

- **Creating a file:** - First find a space for the file and then new file is created in the directory.
- **Writing a file:** - The pointer must be at the location where next file is to be written.
- **Reading a file:** - System call is used to read a file; it identifies the name of the file and specifies the next block of the file should be place.
- **Appending a file:** - This operation can add the data to the end of an existing file.
- **Repositioning within a file:** - This is also called seek operation. In this current-file-position is given value and search for appropriate entry.
- **Renaming:** - This is used to rename the file.

- **Deleting a file:** - This system call is used to erase the directory and delete the files.
- **Truncating a file:** - This is used to remove the contents of the files without deleting its attributes.
- **Copying the file:** - This operation is used to create a copy of file and reading from old file and writing to the new file.
- **Open:** - The purpose of this operation is to fetch the lists and attributes of disk into main memory. A process must open a file before using it.
- **Close:** - To free up the internal space, the file must close after it finishes all its accesses.

### 5.2.4 File Types

When designing a operating system, it should be considered that operating system can recognize and support file types. So that it can operate on file in different ways. The extension is used to indicate the file type and operations that can be execute on that file. Following are some common file types:

- **Text file:** - A text file or a document file contains the sequence of characters. These sequences of characters are organized into lines. The extension of text file is .txt or .doc.
- **Executable file:** - An executable file has codes that can be loaded into memory for execution. The extension of this file is .com or .exe.
- **Source file:** - It contains the source code of a program or language. All programming language source codes have different extensions. For example, extension for pascal is. pas and for C language is .c.
- **Object file:** - Object file is generated by the compiler after the execution of the source file. .obj or .o are the extensions of object file.
- **Word processor file:** - This type of file contains the text and word processor formats. Its extensions are .tex, .wp, .rrf etc.
- **Library file:** - It contains built-in functions for the programming languages. The extension for this is .lib or .a.
- **Archive file:** - Same type of files are grouped into one file for storage. It can be stored in compressed format. The extension is .arc or .zip or .tar.

### 5.2.5 File Structure

The files can be structured in several ways. The following are the three possible internal structure of a file are:

- a) **Byte Sequence:** An operating system only cares about bytes; it does not know what is in the file. It is like a unstructured sequence of bytes as shown in figure 5.1. Files can be imposed by user-level programs. Windows and UNIX both the operating system use this way of file structure.

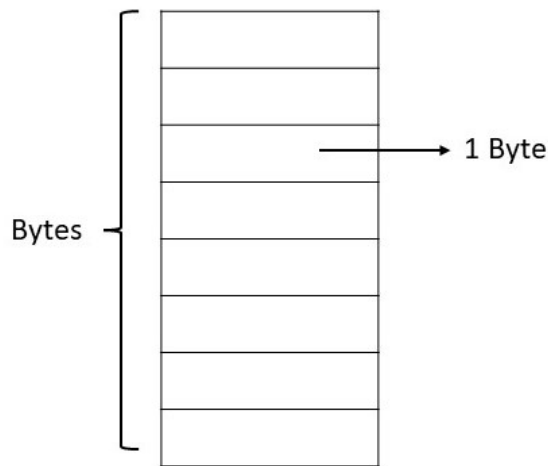


FIGURE 5.1

- b) **Record Sequence:** It is a sequence of fixed length records of a file with some internal structure as shown in figure 5.2. In this structure, the read operation returns the whole record and write operation appends one record. CP/M OS supports this structure.

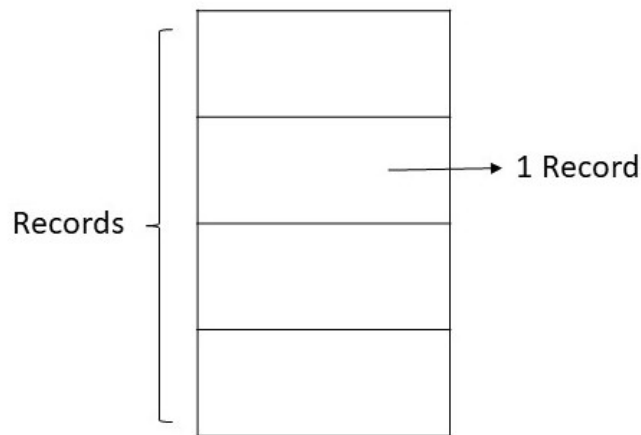


FIGURE 5.2

- c) **Tree Structure:** In this structure, a file consists of a hierarchy of records called tree as shown in figure 5.3. All the records may not necessarily of same size, each contain a key attribute in a fixed position in the record. The tree is in sorted way.

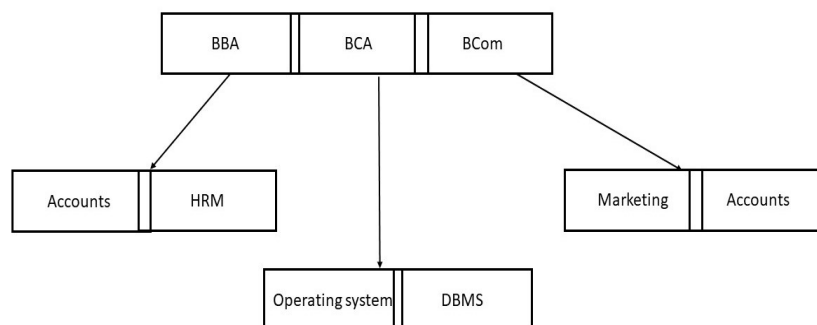


FIGURE 5.3

Information is stored in files. When required information can be accessed. There are several methods to access the information. But which one to choose is a major problem. Let us discuss various access methods.

### 5.3.1 Sequential Access

In the sequential access, the files are accessed in sequence manner or in order one after another. From starting to the end, process can read all the records in a file sequentially. Sequential access is more convenient when storage memory is magnetic tape rather than disk. These types of files can be rewound, so they can be read as required as shown in figure 5.4.

### 5.3.2 Random Access

Random Access is also called direct access. In this type of access, records of the files can be read in any order. It is a collection of records that are stored on a disk. These records have their numbers so can be referred by their numbers instead of position. These files must be stored on storage medium like disk.

Many applications use random access files. For example, database system of railway reservation. If a customer wants to books a particular seat on a train, then that customer can directly access the data of that particular train or seat without checking other trains.

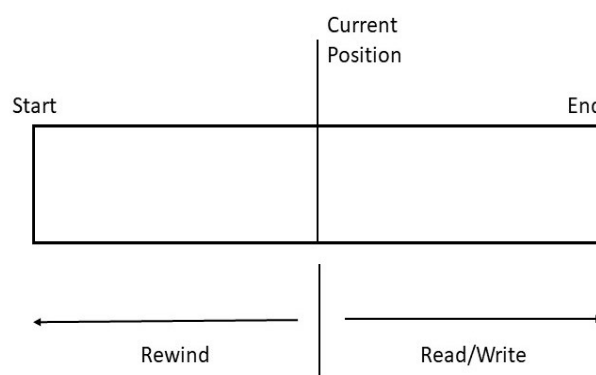


FIGURE 5.4

### 5.3.3 Indexed Sequential Access

This method combines the advantages of both direct access and sequential access methods. In this method records are in sequence manner but they can be accessed randomly through their index number. Index uses pointers to every record. To search a record, first it search the index then use the pointer to access that particular record.

The secondary storage devices that support the indexed sequential access method is Magnetic disk and magnetic drum.

---

## 5.4 DISK AND DIRECTORY STRUCTURE

---

To manage thousands of files stored on disk, we need to organize that disk. It can be done in two parts:

### First part

a) **Disk has only one partition.**

In a system, disk contains one partition for files and directories which is a low-level structure as shown in figure 5.5.

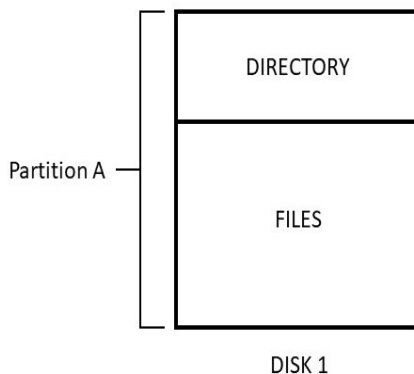


FIGURE 5.5

b) **Disk has several partitions.**

Sometimes there are several partitions within one disk. Each partition can be treated as individual storage device as shown in figure 5.6.

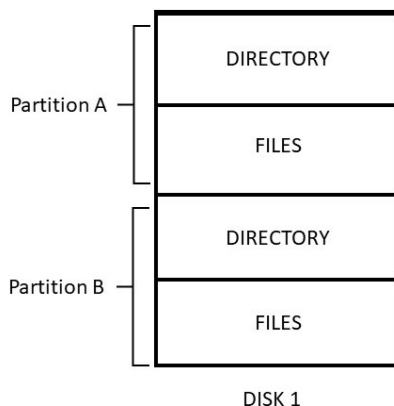


FIGURE 5.6

c) **Group the disks into logical structure.**

In this structure, more than one disks are grouped together into one logical structure as shown in figure 5.7. This allows the partitions to be larger. User only take care about the logical directory.

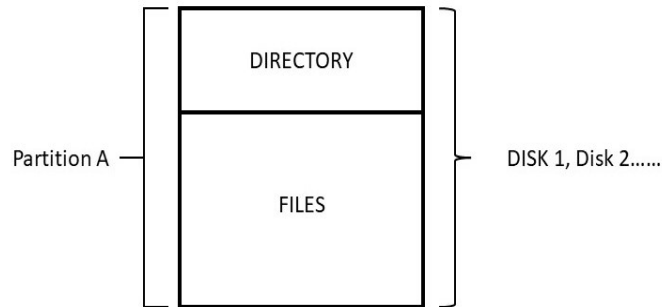


FIGURE 5.7

## Second Part

Directories contains the information of all the file like its name, size, location, type etc. It can be thought as a symbol table. Let us discuss the methods for identifying the logical structure of a directory.

**The following are the levels of Directory Structure:**

**5.4.1 Single-Level Directory:** It is the simplest structure of directory. All the files kept in same directory as shown in figure 5.8.

### Advantages

- Easy to understand.
- Easy to support.

### Disadvantages

- Same names of files can create confusion.
- Same name of different files can collide.

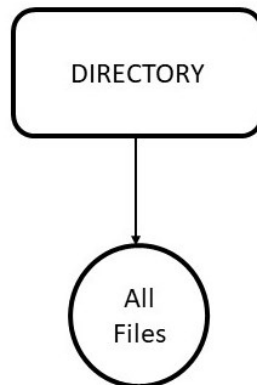


FIGURE 5.8

**5.4.2 Two-Level Directory:** In this type of structure, every user must have their own user file directory (UFD). This structure lists the files of a single user but each UFD has same structure. When the user log in, the Master file directory (MFD) is searched as shown in figure 5.9. The MFD has user name or account number and entry points to the UFD.

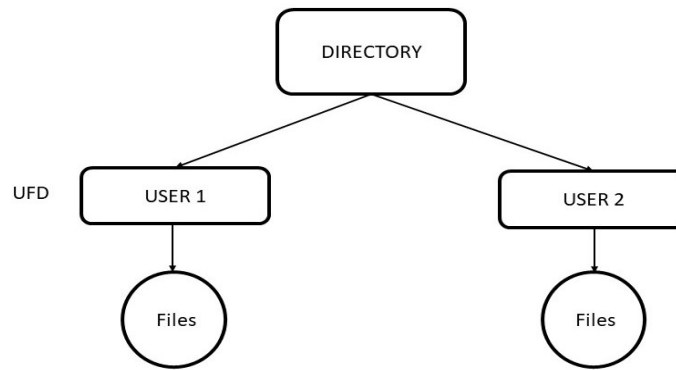


FIGURE 5.9

### Advantages

- The two-level directory structure is efficient.
- It removes the problem of name collision.
- Separate one user from other.

### Disadvantages

- When user's wants to access one another's file, they cannot as they are separate from each other.
- No logical grouping is there other than by user.

**5.4.3 Hierarchical structure:** Hierarchical structure directory is also called tree-structure directory. This directory structure allows the users to create their own sub directories and they can organize file accordingly. It has a root directory. All the files have unique names. A directory contains files and subdirectories as shown in figure 5.10. Internal format of all the directories is same. Bit 0 defines the file and 1 defines the subdirectory. Directories can be created and deleted through system calls. Example is UNIX file system.

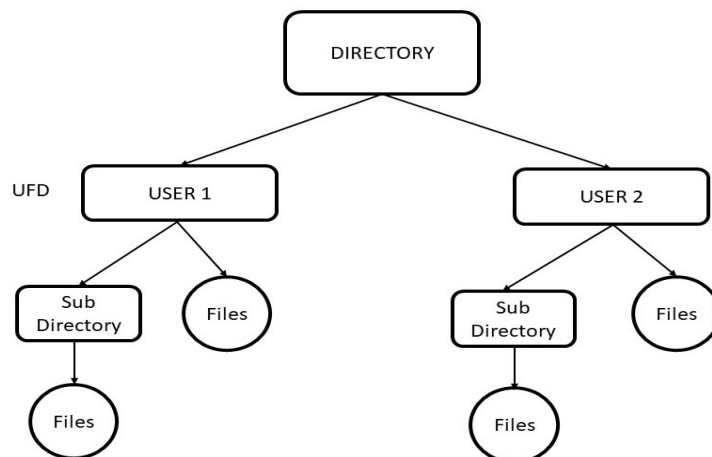


FIGURE 5.10

### Advantages

- It has grouping capability.
- Users can access other user's files through path name.

### Disadvantages

- Structure is very complicated.
- Directory deletion is difficult.

**5.4.4 Acyclic Graph Structure:** In tree-structure, sharing of files and directories not allowed. This graph structure allows to share files and subdirectories as shown in figure 5.11. Acyclic graph means a graph with no cycles. The same file name and subdirectory name can be in different directories.

### Advantages

- It is more flexible.
- Common subdirectory can be shared.

### Disadvantages

- This structure is more complex.
- It has traverse and deletion problem.

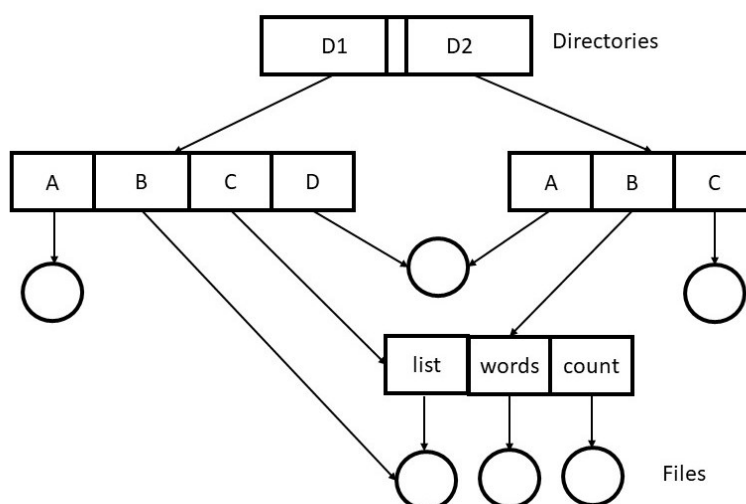


FIGURE 5.11

**5.4.5 General Graph Directory:** In general directory structure, a simple graph structure can be made by adding links to existing tree structure directory as shown in figure 5.12.

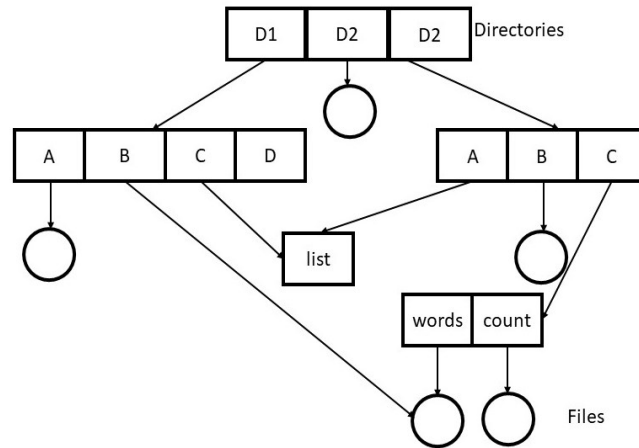


FIGURE 5.12

### Advantages

- It is more flexible.

### Disadvantages

- It has deletion problem.
- Infinite loop may exist.
- Garbage collection scheme is needed, which takes more time.

---

## 5.5 FILE-SYSTEM MOUNTING

---

- It is compulsory to mount the file system before its use.
- Operating system make directories and file on memory devices, so that users can access that data through file system. This process is called File-system mounting.
- A location where partition used as root system is called a **mount point**.
- Mounting is combined an extra file system to currently accessible file system.
- When operating system makes the memory device for safe removal and disconnect all user access to directories and file on the mount point is called **unmounted**.

---

## 5.6 FILE SHARING

---

When users working together, they need to share files. It is convenient for files to appear at the same time in different directories which belongs to different users as shown in figure 5.13

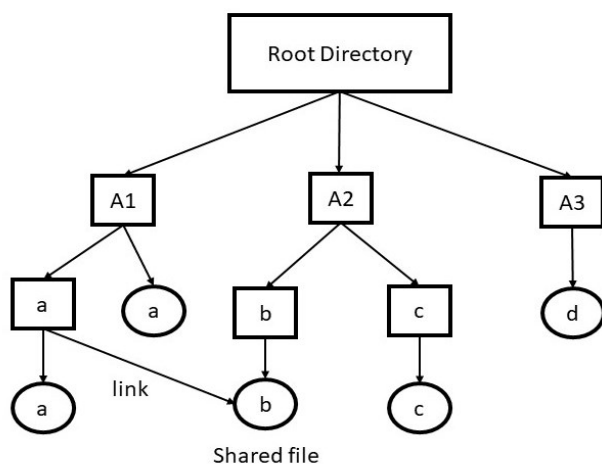


FIGURE 5.13

### Issues due to sharing files

User can create link between directory and the shared file by copying the address of records. Now if both the directories make changes to the records, that new records will be listed only that particular directory. The change will not show to other user, it creates the issue of sharing.

The following are two solutions to this problem.

#### 5.6.1 i-node Linking

In index-node (i-node) linking, in directories i-node is listed instead of disk blocks. The directories then point to i-node. This technique is used in UNIX.

#### Drawbacks

- If any directory links to the shared file, the i-node records the owner as its previous directory. Ownership is not changed after creating the link but in i-node link counts are increased as shown in figure 5.14

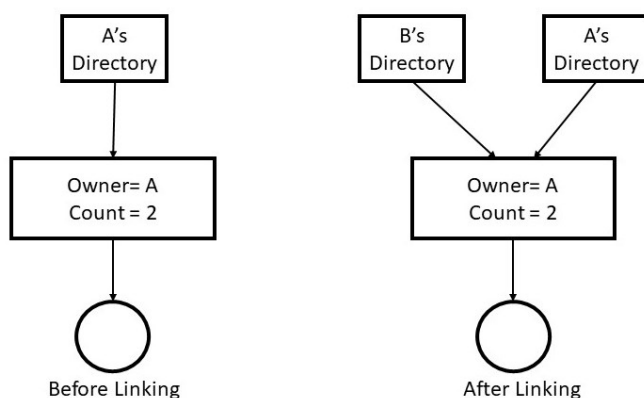


FIGURE 5.14

- If one directory removes the file and clear the i-node, then another directory will have point to an invalid i-node.

### 5.6.2 Symbolic Linking

In this approach, when one directory links to file of another directory. Then a new file is created of type LINK. The new file contains the path name of the file with whom it is linked. If any directory wants to read from the linked file, the OS looks the file is of type LINK and read that file.

#### Advantages

- This technique can be used to link the files on system anywhere in the world by providing the network address.

#### Disadvantages

- An extra i-node is required.
- Extra overhead is needed.

---

## 5.7 FILE SYSTEM STRUCTURE

---

The operating system uses layered file structure for some tasks in the file system. Every layer has their own responsibilities to do some particular tasks. For example, to store the data, to locate the data, to retrieve the data. The figure 5.15 shows the basic layered file structure.

- The application program is the first layer. When this layer asks for a file, first request is sent to logical file system.
- Logical file system has the meta data. If it does not give permission to application program, then this layer gives the error. This layer also verifies the path.

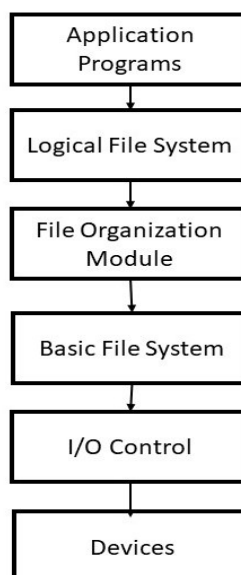


FIGURE 5.15

- In the file organization module mapping is done. Logical blocks are mapped into physical blocks.
- The basic file system takes the information from file organization module and issues the commands to I/O control to fetch the blocks.
- I/O control layer contains the coding (called device drivers) which can be used to access hard disk. This layer also handles interrupt.

---

## 5.8 FILE SYSTEM IMPLEMENTATION

---

### 5.8.1 Overview

The data structure store on disk by file system:

#### On disk Structure

- **Boot Control Block:** This block (per volume) contains information needed by the system to start or boot an operating system. This block can be empty if there is no operating system in the disk. Boot control is the first block of a volume.
- **Volume Control Block:** Its per volume contains partition details like free-blocks, pointers, size of blocks, number of partitions in blocks and FCB blocks.
- **File Control Block:** It contains details about dates, permission, ownership etc. It has unique number to association with directory entry as shown in figure 5.16.

#### In-memory Structure

- **Mount table:** In this table, information regarding each mounted volume is stored.
- **Directory structure cache:** Stored information regarding accessed directories.
- **System-wide open file table:** The copy of file block control of open file stored in this table.
- **Pre-process open file table:** Information regarding the process and maps it with open file.

| File Control Block (FCB)      |
|-------------------------------|
| Permissions for files         |
| Date of creation, write       |
| File owner                    |
| File Size                     |
| File data blocks and pointers |

FIGURE 5.16

---

## 5.9 DIRECTORY IMPLEMENTATION

---

The selection of directory algorithms affects the performance, reliability and efficiency of the file system. Directories should be fast and space should be minimum wasted.

### 5.9.1 Linear List

- It is the easiest and simplest directory structure.
- To find a file linear search algorithm is used.
- To search the data, sorting algorithms are used.
- Deletion can be easily done by moving the entries.
- If the list is sorted then insertion and deletion can be done easily in linear list.

### 5.9.2 Hash Table

- Hash table are used to speed up the searches.
- These are implemented with a linear and other structures.
- Hash table are of fixed size.

---

## 5.10 ALLOCATION METHODS

---

The allocation methods are used so that files can be accessed easily and disk space can be utilized effectively. The following are the three methods of allocating files on disk:

### 5.10.1 Contiguous Allocation

- In contiguous allocation method, all blocks of files are kept in continuous manner on the disk.
- Its speed is fast because disk head requires no movements.
- The file is defined by the length of first block and the disk address.
- The problem in continuous allocation occurs when data of any file is extended or size of the file is not known beforehand.
- So, there can be a problem of external and internal fragmentation.

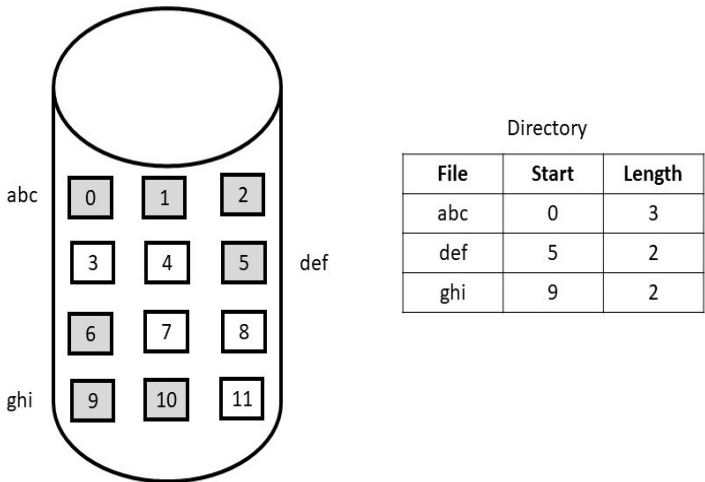


FIGURE 5.17 Contiguous Allocation

### 5.10.2 Linked Allocation

- In linked allocation method, files on disks can be stored as linked list.
- It does not involve external fragmentation; it allows the size of file to grow dynamically.
- Linked allocation is only effective with sequential access files.
- Internal fragmentation occurs in linked allocation, as the allocating blocks lessen the space wasted by pointers.
- Linked allocation is not reliable, if the link is damaged.

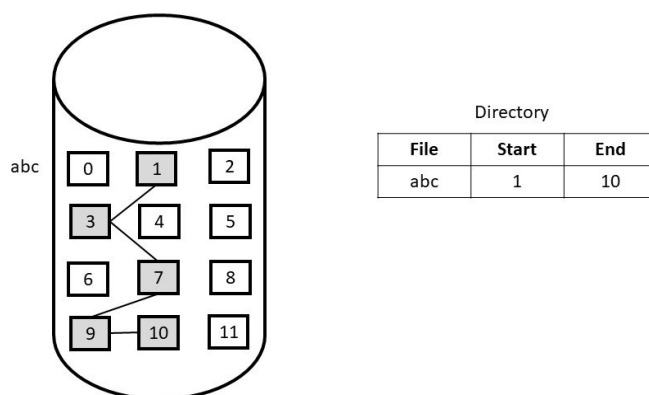


FIGURE 5.18 Linked Allocation

### 5.10.3 Indexed or i-node Allocation

- It combines all the indexes of files into a single block for accessing.
- The whole block is given to each file so, some space is wasted.
- **Linked scheme:** One disk block is an index block and single disk operation is enough for read and write. The first block contains block address, header information or may be a pointer.

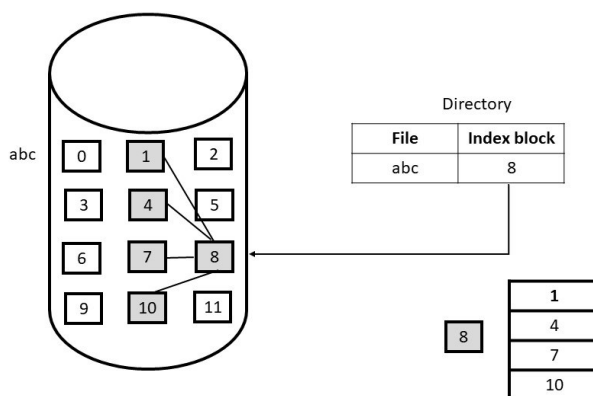


FIGURE 5.19 Indexed Allocation

- **Multi-level index:** The first index block contains pointers to next index blocks, which then refer to actual data blocks.
- **Combined scheme:** in this scheme some pointers are directly store in i-node and it provide access to more data blocks as required.

---

## 5.11 FREE - SPACE MANAGEMENT

---

Free-space management means to monitor the free disk space. Free-space list keeps records of free disk blocks so that the space can be allocated to required file or directory. When any file is removed or deleted, then that space is added to the free-space list. If any space is used then it is removed from the list. The following are some approaches how the list is implemented:

### 5.11.1 Bit Vector

- It is a simple approach to use, in this each bit shows a disk block, set to 0 if allocated and 1 if it is free.
- For quickly finding contiguous blocks fast algorithms are used.

### 5.11.2 Linked List

- Linked list is also used to manage free blocks.
- The File allocation table contains all the records of free blocks.

### 5.11.3 Grouping

- The free list stores the addresses of k (say) blocks in first free block.
- First k-1 blocks are free and last block contains data of another free k blocks.
- Addresses can be found easily.

### 5.11.4 Counting

- Sometimes there are contiguous free blocks, so to keep track of this just store the addresses of first block and the last block.
- These can be stored in a B-tree rather than linked lists.

---

## 5.12 SUMMARY

---

**Files:** - All the applications need to store and fetch the information in the computer. It is essential to store the data for long-term and make data sharable so that various processes can access the data at any time. The size of data is large so some appropriate storage devices must be used. To match these requirements, the information needs to be stored on disk in units called **files**.

**File System:** - Files are handled by the operating system. Operating system design deals with how the files are structured, used, accessed, named and implemented. That part of operating system which deals with files is known as **file system**.

**Mounting:** - Operating system makes directories and files on memory devices, so that users can access that data through file system. This process is called **File-system mounting**.

**Unmounted:** - When operating system makes the memory device for safe removal and disconnect all user access to directories and file on the mount point is called **unmounted**.

**File's Attributes:** - When a file is created, a lot of information regarding the file is created like its name, date of creation, size and so on. These items are called **file's attributes**.

---

## 5.13 EXERCISES

---

1. What do you mean by file?
2. Explain the rules for file naming.
3. Explain various file attributes and its types.
4. Describe file structure.
5. Explain the various access methods.
6. Write note on:
  - Contiguous allocation method
  - Linked allocation method
  - Indexed allocation method.
7. What do you by Free space management?
8. Explain the directory operations.
9. Explain Directory implementation.
10. Explain file system implementation.
11. What are the issues of sharing files.
12. What do you mean by indexed sequential access?

\*\*\*\*\*