OSY - Notes


V2V EDTECH LLP
Online Coaching at an Affordable Price.
OUR SERVICES:
• Diploma in All Branches, All Subjects
• Degree in All Branches, All Subjects
• BSCIT / CS
• Professional Courses
+91 93260 50669
V2V EdTech LLP
v2vedtech.com
v2vedtech

Index

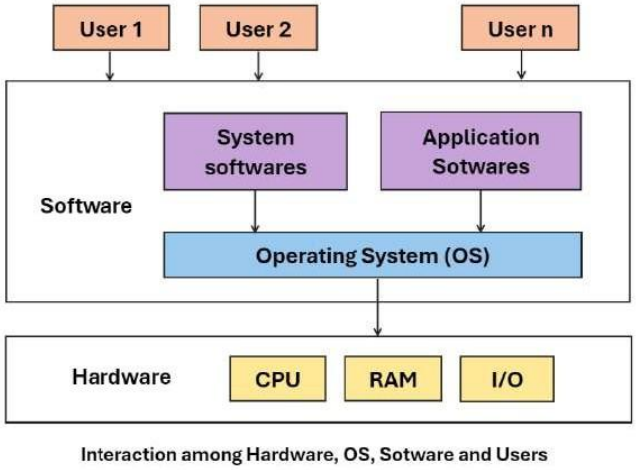
UNIT 1 - Operating System services and components

1.1 Operating System :

Concept of Operating System :

Q. Explain the concept of an Operating System .

An operating system (OS) is the most essential system software that manages the operations of a Computer. Without on operating operating system, it is not possible to use the computer. An operating system (OS) containing instructions (set of programs) that work together to co-ordinate all the activities among computer hardware resources. An operating system is an intermediary between user and the computer hardware. The purpose of an operating System is to provide on environment in which a user may execute program.



Interaction among Hardware, OS, Sotware and Users

- **User:**

  This is the human interacting with the computer system. The user provides input and receives output. They are at the top of the hierarchy.

- **Application Software:**

  These are programs designed to perform specific tasks for the user. They are what the user directly interacts with to achieve a goal.

- **System Software:**

  This acts as an intermediary between the application software and the operating system. It includes utilities, language translators (like compilers), and device drivers that help run the computer and its applications.

- **Operating System (OS):**

  This is the core software that manages all the computer's hardware and software resources. It provides a platform for application software to run and handles fundamental tasks like memory management, process scheduling, and input/output operations.

- **Hardware:**

  These are the physical components of the computer system. They are the actual electronic circuits and devices that perform computations and store data.

  Example:

  - **CPU (Central Processing Unit):** The "brain" of the computer that executes instructions.
  - **RAM (Random Access Memory):** Volatile memory used for temporary storage of data and programs currently being used.
  - **I/O (Input/Output Devices):** Devices that allow the computer to interact with the outside world (e.g., keyboard, mouse, monitor, printer, speakers).

Following are the objectives of OS:

Q. What are the different objectives of an operating system ?

i] Convenience

ii] Efficiency

iii] Ability to Evolve

i] Convenience:

An Operating System makes a computer more convenient to use.

ii] Efficiency:

An operating system allows the computer system resources to be used in an efficient manner.

iii] Ability to Evolve:

An operating system should be constructed in such a way as to permit the effective development, testing, executing and introduction of new system functions without interfacing with service.

- **Functions of Operating System :**

Q. Explain different functions of operating system.

1) Memory Management:

An operating system deals with the allocation of main memory and other storage areas to the system programs as well as user programs and data.

2) Processor Management:

An operating system deals with the assignment of processors to different tasks being performed by the computer system

3) Device Management:

An operating system deals with the co-ordination and assignment of the different output and input device white one or more programs are being executed

4) File Management

An operating system deals with the storage of file of various storage devices to another. It also allows all files to be easily changed and modified through the use of text editors or some other files manipulation routines.

5) Error Detection Aids:

Production of dumps, traces, error messages and other debugging and error detecting aids.

6) security:

By means of password and similar other techniques, preventing unauthorized access to programs and data.

7) Control over system performance:

An OS performs recording delays between request for a service and response from the system.

8) co-ordination between other software and Users:

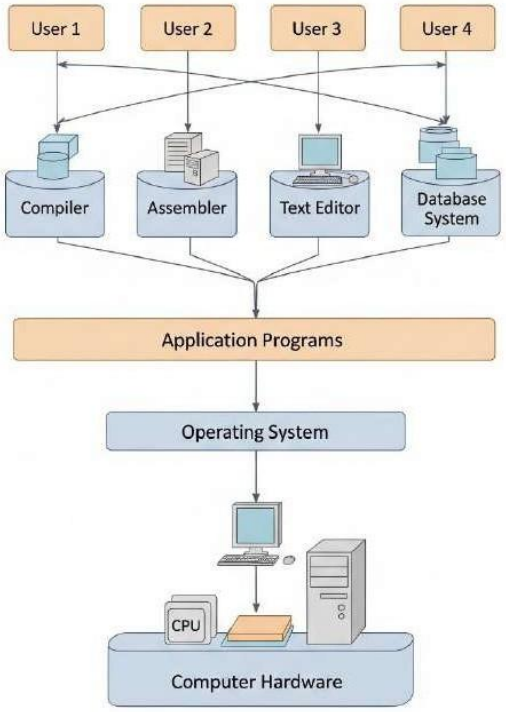
Co-ordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems.

9) Job Accounting

An OS keeping the track of time resources used by various jobs and users.

- **Computer system :**

Q. Explain Computer operating system and it's components .



A computer system is a collection of hardware and software components. Hardware refers to the physical computing equipment. Software refers to the programs written to provide services to the users of the system.

A computer system can be divided roughly into four component namely :

i. Hardware
ii. Operating system
iii. Application programs
iv. Users

Following are the components of computer system :

1. Hardware:

Hardware is physical parts of machine which provides basic computing resources. The hardware devices are central processing unit (CPU), memory, input devices, networking devices and devices like motherboard, power supply, DVD writer etc.

2. Operating System:

An operating system is a software program that enables the computer hardware to communicate and operate with the computer software. Without a computer operating system, a computer and software programs would be useless. Examples of computer operating systems are Microsoft Windows 7/8/10, Apple Mac-os, Ubuntu Linux, Google Android, iOS and so on.

3. Application Programs:

Word processors, spreadsheets, compilers, web browsers, database systems, video games etc., are examples of application programs. Application programs are the programs that define the way in which these resources are used to solve the computing problem of the users.

4. Users:

On the basis of the role of the users, they can be categorized as:

- Programmers
- Operational Users
- End Users

Generations of operating system :

Q. What are the different generations of operating system ?

* The 1940's - First Generations
* The 1950's - Second Generation
* The 1960's - Third Generation
* The 1980's - Fourth Generation

1. First generation 1945 – 1955:

The earliest electronic digital computers has no operating System. Machines of that time were so primitive that programs were often entered one bit at time on rows of mechanical switches (plug boards). Programming languages were unknown (not even assembly languages).

2. Second generation 1955 - 1965:

In second generation there is the introduction of punch card. The General Motors Research laboratories implemented the first operating systems in early 1950's for their IBM 701. The system of the 80's generally run one job at a time. These were called single-stream batch processing systems because program and data were submitted in group of batches.

3. Third generation 1960-1980:

The system of the 1960's were also batch processing system, but they were able to take better advantage of the computer's resources by running several jobs at once. So operating systems designers developed the concept of multiprogramming in which several jobs are in main memory at once; a processor is switched from job to job as needed to keep several jobs advancing while keeping the peripheral devices in use.

4. Fourth generation 1980:

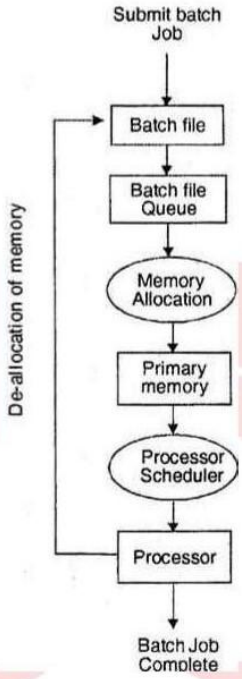
With the development of LSI (Large Scale Integration) circuits, chips, operating system entered in the system entered in the personal computer and the workstations. Microprocessor technology evolved to the point that it becomes possible to build desktop computers as powerful as the mainframes of the 1970's.

➢ 1.2 Different types of Operating System :

Q. Enlist / List types of operating system . (W-19, S-24)

1. Batch
2. Multi-programmed
3. Time Shared
4. Multiprocessor System
5. Distributed System
6. Real Time System
7. Mobile OS (Android OS)
8. Multitasking
9. Serial processing

- **Batch Operating System :**



Batch Operating System

In batch operating system, jobs with similar need batched together by operator and run as a computer system. A job is predefined sequence of commands, programs and data that are combined in to a single unit. Memory management in batch system is very simple. Memory is usually divided into two areas namely, operating system and user program area. With the use of this type of operating system, the user no longer has direct access to the machine, rather the user submits the job on cards or tape to a computer operator, who batches the jobs together sequentially and places the entire batch on an input device for use by the monitor.

Advantages of Batch OS:

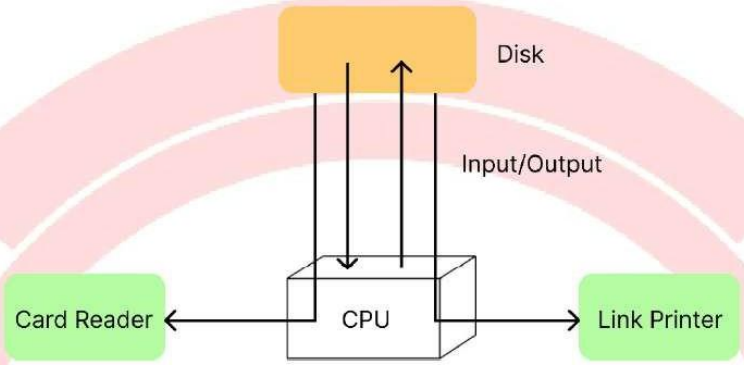
- Increase performance since it was possible for job to start as soon as the previous job finished.
- In batch OS huge amount of a data can be processed efficiently.

Disadvantages of Batch OS:

- As monitor needs to be in main memory, it results in certain wastage of memory.
- Program having large I/O operation wasted huge amount of CPU time.
- Difficult to debug program.
- No interaction is possible with the user while the program is being executed.

Concept of SPOOLing :

Spooling (Simultaneous Peripheral Operations On-Line) is a technique used by operating systems to handle the differences in speed between the CPU and I/O devices. When a job needs to use a peripheral device (like a printer or card reader), the data for that job is first stored on a faster device, typically a hard disk, acting as a large buffer. This process allows the CPU to continue working on other tasks without waiting for the slower I/O device to complete its operation.



How Spooling Works:

- **Buffering:** When a job requires an output, such as printing, the data is written into a system buffer on the disk. This buffer is essentially a temporary storage area.
- **Decoupling:** Once the data is in the buffer, the job is considered complete from the CPU's perspective, and the CPU is freed up to process other tasks.
- **Background Processing:** A separate process, often referred to as a "spooler," then handles the actual transfer of data from the buffer to the slow output device (e.g., the printer) in the background. Similarly, for input, data from a card reader is first read into a disk buffer, and then the CPU can access it from there.

Advantages of SPOOLing :

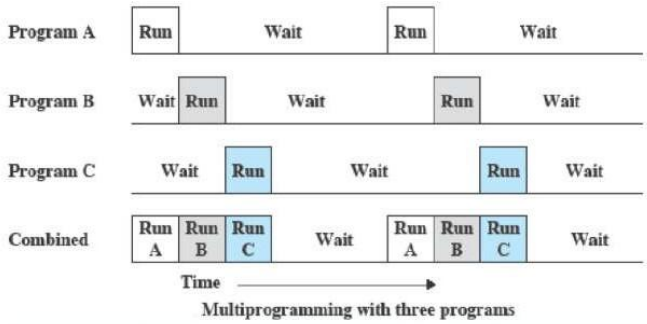
- Uses disk as a very large buffer.
- Improves I/O operation for jobs with processor operations.

Disadvantages of SPOOLing:

- Extra overhead due to maintaining tables of card images.
- Risk of wear on the magnetic tape and the tape itself if the disk and CPU access it frequently.

- **Muti-programmed Operating System :**

Q. Explain multiprogramming operating system in detail. (W-19)



Multiprogramming with three programs

In multiprogramming, more than one program lies in the memory. The shedular selects the job to be placed in ready queue from a number of programs. The ready queue is placed in memory and the existence of more than one program in main memory is known as multiprogramming. Since there is only one processor, there multiple programs cannot be executed at a time. Instead, the operating system executes part of one program, then the part of another and so on.

Advantages of Multiprogramming OS:

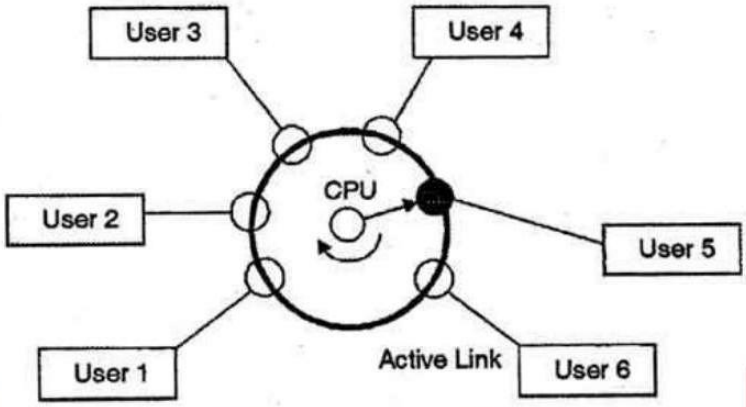
- CPU is used efficiently without sitting idle.
- System handles more jobs in less time.
- Memory is shared and used effectively.

Disadvantages of Multiprogramming OS:

- Programs may face security issues.
- Some tasks get delayed due to sharing.
- Debugging becomes difficult.

- **Time Shared Operating System :**

Q. Describe working of time sharing system with neat diagram. (S-22, W-22)



The main idea of time sharing Systems is to allow a large number of users to interact with a single computer (System) concurrently. A time sharing System is one that allows a number of different users to share the single computer System for a variety of applications at the time. In other words a time - sharing System is basically a multiprogramming, multitasking and multi-user environment of a large computer System. In time sharing a small time slots are available for each user. This short period of time during that a user gets attention of the CPU (known as a time slice or a quantum). The time allowed is extremely small and the users are given the impression that each of them has their own CPU and they are the sole owner of the CPU.

Advantages of time sharing System:

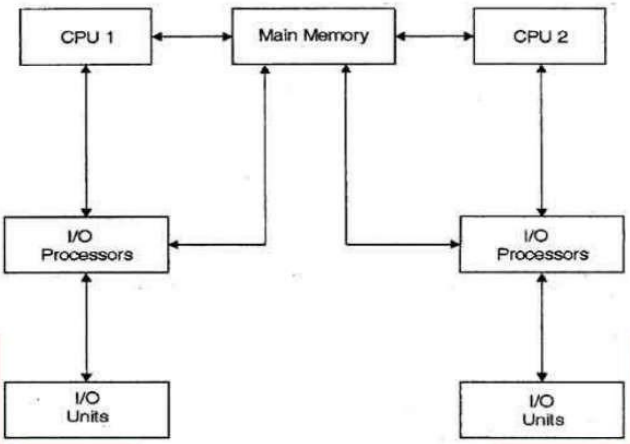
- Efficient CPU utilization.
- A time shared operating system uses CPU scheduling and multi programming to provide each user with a small portion of a time shared computer.
- Time-sharing OS may be more cost-effective for businesses because they permit multiple users to use they System without needing to purchase individual license.

Disadvantages of time sharing System.

- The time-shared Systems are more complex than the multi-programming Systems.
- In time-shared systems multiple processes are managed simultaneously which requires an adequate management of main memory so that the processes can be swapped in or swapped out within a short time.
- Time-sharing operating systems may have data security and integrity issues with user programs and data.

- **Multiprocessor Operating System :**

Q. Describe multiprocessor OS with it's advantages. (any two) (S-22, S-23, W-23)



Multiprocessing is the use of two or more Central Processing Units (CPUs) within a single computer system. The term multiprocessing also refers to the ability of a system to support more than one processor and/or the ability to allocate tasks between them. Multiprocessor System means more than one processor in close communication. All the processors share common bus, clock, memory and peripheral devices. Multiprocessor system is also called Parallel Systems. These types of systems are used when very high speed is required to process a large volume of data. These systems are generally used in environment like satellite control, weather forecasting etc.

Advantages of Multiprocessor Systems:

- It increased throughput by increasing the number of processors, more work done in a shorter period of time.
- Multiprocessors can also save money (cost) compared to multiple single systems
- These systems provides higher performance due to parallel processing.
- It increases reliability, as failure of one processor will not halt the system.

Disadvantages of Multiprocessor Systems:

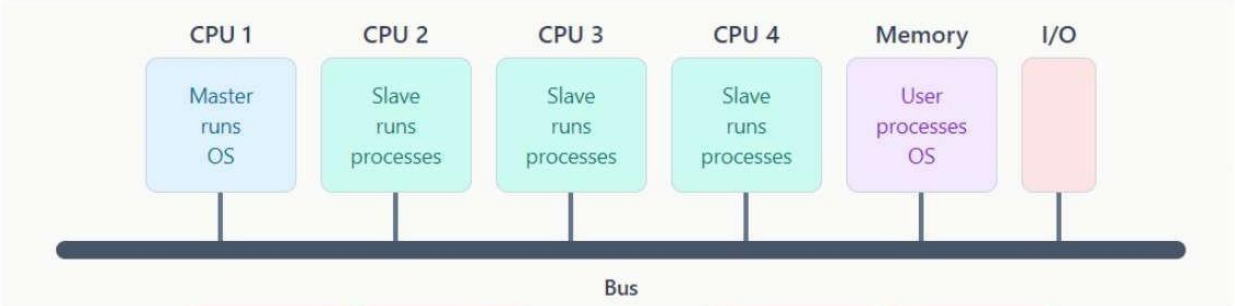
- If one processor fails then it will affect in the speed.
- Multiprocessor systems are expensive.
- Complex OS is required.
- Large main memory required.

Types of Multiprocessor Systems:

a. Asymmetric Multiprocessing

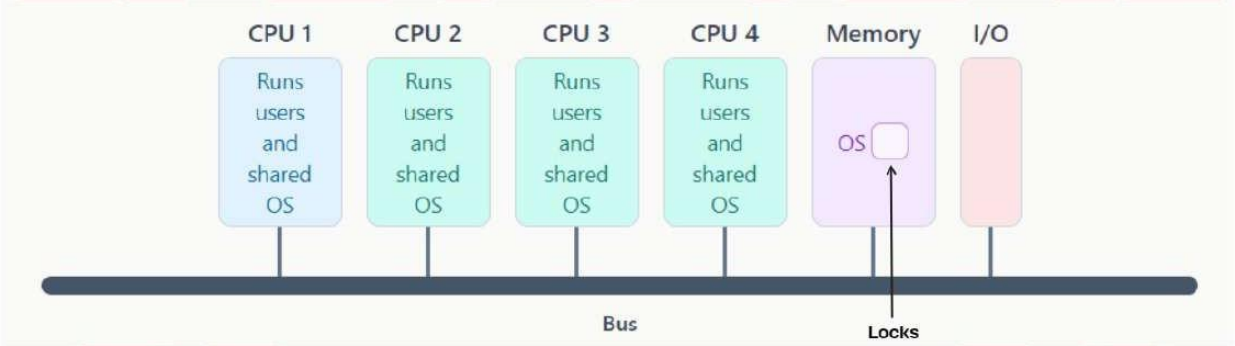
b. Symmetric Multiprocessing

a. Asymmetric Multiprocessing



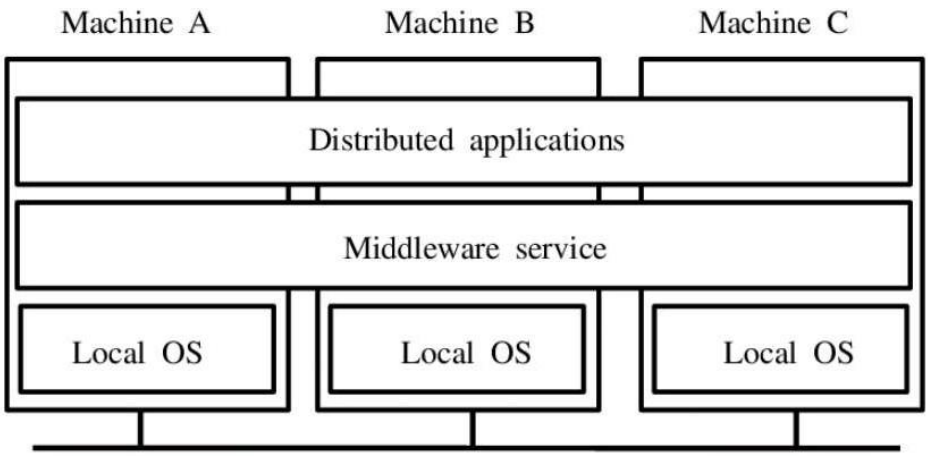
In this system, a specific task is assigned to each processor. The system has one master processor and others are slave processors. A master processor controls the System and slave processors follow the instructions of master or perform their predesigned task.

b. Symmetric Multiprocessing



In symmetric multiprocessing, there is no master-slave concept used. All the processors are peer processors. They perform all tasks within the operating system. The benefit of symmetric multiprocessing model is that many processes can run simultaneously.

- **Distributed Operating System :**



An operating system that manages a group of independent computers & makes them appear to be single computer is known as a distributed system. In this system the processors do not share memory or a clock; instead each processor has its own local memory, and clock. In such system, if one machine or site fails the remaining sites can continue operation, so these types of systems are the reliable systems. The processors communicate with one another through various communication lines, such as high speed buses or telephone lines. Distributed systems are also known as loosely coupled systems. In distributed system the middleware enable computers to coordinate their activities and to share the resources of the system.

Advantages of Distributed Systems:

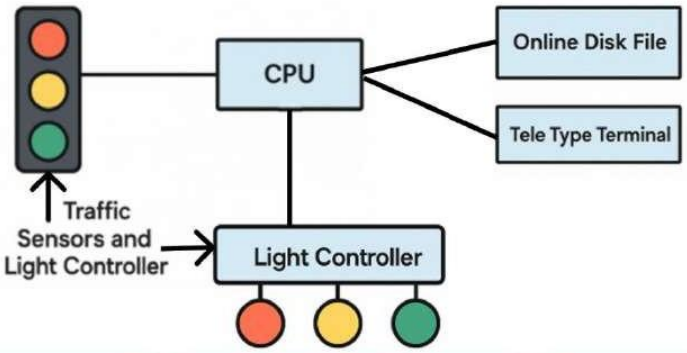
- It is more reliable than a single system. If one machine from system crashes the rest of the computer's remain unaffected.
- In distributed computing the computer power can be added in small increments.
- Distributed system is very flexible as it is very easy to install, implement and debug new services.

Disadvantages of Distributed System:

- In distributed system it is difficult to troubleshooting and diagnosing problems.
- Distributed operating system is less secure. The sharing of data creates the problem of data security.
- Distributed systems are more complex than centralized systems.
- In distributed system more effort required for system management.

- **Real Time Operating System (RTOS) :**

Q. Define real time operating system and its types. List its any four applications of it. (W-19, S-23, W-23)



A real time operating system has well defined fixed time constraints. Processing should be done within the defined constraints. The real time operating system used for a real-time application means for those applications where data processing should be done in the fixed and small quantum of time.

Types of Real Time Operating System:

- Hard real-time operating system
- Soft real-time operating system

Hard Real-time operating Systems

Hard real-time system guarantees that critical task complete on time. In hard real-time systems, secondary storage is limited or missing and the data is started in ROM. In these systems, virtual memory is almost never found.

Examples of hard real-time operating system are:

- Flight control system
- Missile guidance system
- weapons defence System
- Autopilot system in plane.

Soft real-time operating System

Soft real-time systems are less restrictive. A critical real-time task gets priority over other tasks and retain the priority until it completes. Soft real-time system have limited utility than hard real-time systems.

Examples of soft-real time operating system are :

- Personal Computer
- Multimedia system
- Web browsing

- Mobile communication

Advantages of real time operating system:

- A real-time OS gives a quick response.
- It schedules tasks on time.
- It uses system resources well.
- It can run many tasks at once.

Disadvantages of real time operating system:

- It is hard to design.
- It can't handle too many tasks.
- It may need special hardware.

Applications:

- Flight Control System
- Simulations
- Industrial Control
- Military applications.
- Defence application system like RADAR
- Missile guidance system
- Network and Multimedia System
- Medical critical care System.

- **Mobile (Android) Operating System :**

A **mobile operating system** is software that manages all functions of a mobile device, such as a smartphone, tablet, or PDA (Personal Digital Assistant). It controls the hardware and enables other apps to run, just like Windows or macOS does on a computer. Mobile OS also handles features like touch screen gestures, camera access, Wi-Fi, Bluetooth, GPS, and more. These systems are designed to be lightweight and power-efficient compared to desktop operating systems. As mobile devices became more popular, many companies started developing their own mobile OS to stay competitive in the tech market.

Types of Mobile Operating Systems:

1. Android (by Google)

Android is developed by Google and is based on the Linux kernel. It is the most widely used mobile OS and supports a wide range of devices. Android apps are mainly built using Java or Kotlin and are available through stores like Google Play. It allows a high level of customization and supports widgets, multitasking, and voice commands.

Advantages:

- Open source and free.
- Easy to customize.
- Supports multitasking.
- Good stability and security.

Disadvantages:

- Some apps are of low quality.
- Can slow down with too many apps.
- Less secure if apps are installed from unknown sources.
- More complex and time-consuming to develop apps.

2. iOS (by Apple)

iOS is Apple's mobile OS for iPhone, iPad, and iPod Touch. It is known for its smooth performance, strong security, and clean design. Apps are built using Swift or Objective-C and are only available through the Apple App Store. iOS supports gestures like swipe, tap, and pinch.

Advantages:

- High security.
- Great performance.
- Good customer support.
- Many apps available.

Disadvantages:

- Only works on Apple devices.
- Not open source.
- App development is expensive.
- Less customizable.

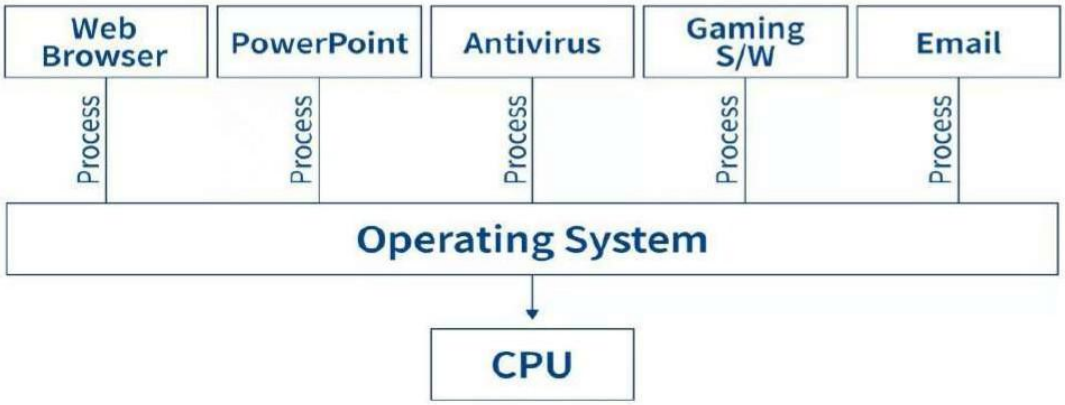
3. Windows Mobile (by Microsoft)

Windows Mobile was developed by Microsoft for smartphones. It included apps like Internet Explorer and Office Mobile. It was used mainly in business devices but is now discontinued and replaced by Windows Phone, which also ended support.

4. Symbian (Originally by Symbian Ltd., later Nokia)

Symbian was once a leading mobile OS used mainly in Nokia phones. It supported many programming languages and had good multitasking. It was the first to use a WebKit browser. However, it lost popularity with the rise of Android and iOS and is no longer used.

- **Multitasking Operating System :**



A CPU handling multiple tasks at a time is known as multitasking. A multitasking operating system can handle multiple tasks together by applying multiprogramming technique. Multitasking operating system is also known as multi-process operating system. Multitasking is the ability to support two or more active processes. In multitasking, only one CPU is involved, but it switches from one program to another so quickly that it gives the appearance of executing all of the programs at the same time. Multitasking is, on single processor machine, implemented by letting the running process own the CPU for a while (a time slice) and when required gets replaced with another process, which then owns the CPU.

Advantages of Multitasking OS:

- Multitasking helps in increasing the overall productivity of the user by performing a number of tasks at the same time.
- It helps in increasing the overall performance of the computer system.

Disadvantages of Multitasking OS:

- It requires more system resources. For example, a large amount of memory is required to execute several programs at the same time.
- For performing multiple tasks at a single time in multitasking, the CPU speed must be very high.

- **Serial processing Operating System :**

The serial processing operating systems are those which performs all the instructions into a sequence manners or the instructions those are given by the user will be executed by using the FIFO manner means first

in first out. For running the instructions the program counter is used which is used for executing all the instructions. In this the program counter will determines which instruction is going to execute and the which instruction will be execute after this. In this all the jobs are firstly prepared and stored on the card and after that card will be entered in the system and after that all the instructions will be executed one by one. But the main problem is that a user does not interact with the system while he is working on the system, means the user can't be able to enter the data for execution.

Advantages of Serial Processing OS:

- Very simple in operation.
- Although, machine dependent, but still gave rise to development of new interfacing software so as to make the programmer's task easier.

Disadvantages of Serial Processing OS:

- The process of development and preparation of a program in such an environment is slow.
- Poor utilization of resources.

Classify Operating System.

Q. Enlist types of operating system / classify operating system. (W-19, S-24)

The operating system is classified in following different types.

1. Single-user, Single-tasking OS
2. Single-user, Multi-tasking OS
3. Multi-user, Multi-tasking OS
4. Real Time Operating System (RTOS)
5. Distributed Operating System
6. Time sharing OS
7. Multi-programming OS
8. Multi-processing OS
9. Batch-processing OS
10. Multi-tasking OS
11. Multi-threading OS
12. Network Operating System (NOS)
13. Embedded OS
14. Real Time OS
15. Mobile Operating System

1) Single - user, Single - tasking OS :

As the name implies, this operating system is designed to manage the computer so that one user can effectively do one thing at a time. The palm OS for palm handheld computers is a good example of a modern Single-user, Single-task operating System. Another examples include MS-Dos and windows OS.

2) Single-user, Multi-tasking OS.

This type of OS which allows a single user to execute two or more tasks at a time. Single-user, multi-tasking is the type of operating system most people use on their desktop and laptop computers today. Both windows 98 and the macintosh OS are the examples of single-user, multi-tasking OS.

3) Multi-user, Multi-tasking OS:

A multi-user operating system allows many different users to take advantage of the computer's resources simultaneously. UNIX, Linux, VMS (Virtual Memory System) are examples of Multi-user, Multi-tasking OS.

3) Real Time Operating System:

A real time operating system has well defined fixed time constraints. The real time operating system used for a real-time application means for those applications where data processing should be done in the fixed and small quantum of time. Examples of real time operating system are flight control system, simulations, industrial control and military applications etc.

Difference Between Multiprogramming and Multitasking

Q. Differentiate between Multi programmed and Multitasking operating system (Any two points) (W-22)

Multiprogramming	Multitasking
Multiprogramming is the capacity to run or handle several programs at the same time.	Multitasking is the ability of a computer to handle a number of tasks or jobs simultaneously.
It is impossible for CPU to run more than one program at the same time.	It is possible for the CPU to run more than one task at the same time.
Based on context switching mechanism.	Based on the time-sharing mechanism and context sharing mechanism.
In multiprogramming a user cannot interact with the system.	In multitasking a user can interact with the system.

It takes maximum time to execute the process.	It takes minimum time to execute the process.
For example, Let's say there are two programs waiting in the pool to be executed by the CPU. The OS picks the first program. If it has I/O operations, it puts this program in the queue and picks the second program and executes it. Meanwhile, the first program receives its input. Since the working of the OS and CPU will be fast, it looks as if both programs are executed simultaneously.	For example, Let's say you are printing a document of 100 pages, while your computer is performing that. You can still do other jobs like typing a new document. So, more than one task is performed.
In multiprogramming there is more CPU idle time as compared to multitasking.	In multitasking there is less CPU idle time as compared to multiprogramming.

Difference Between Time Sharing System and Real Time System

Q. Difference between Time sharing system and Real time system (any 2 points) (S-23)

Time Sharing System	Real Time System
In a time-sharing system, fixed time is given to each process, and all processes are arranged in a queue.	In a real-time system, a job has to be completed within a fixed deadline (time allowed).
It requires a more complicated CPU scheduling algorithm.	A real-time system has well-defined, fixed time constraints.
If a job is not completed within the given time, it jumps to the next job, leaving the previous job unfinished. After processing each job, it again gives the same time for the unfinished job.	If a job is not completed within the given time, the system may extend time for doing the operations.
Response time is not important.	Response time is important.
Process deals with more than one application simultaneously.	Process deals with a single application at a time.
Emphasis is on providing a quick response to a request.	It focuses on accomplishing a computational task before its specified deadline.

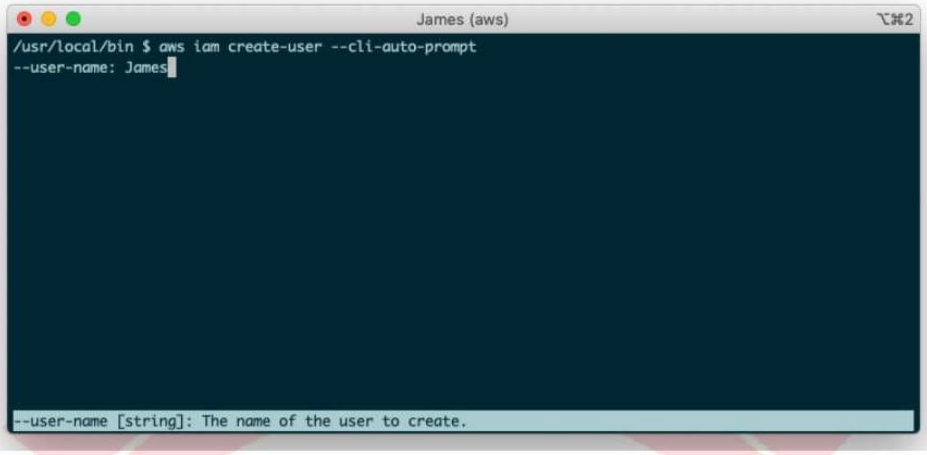
Difference between Time Sharing OS and Multiprogramming OS

Q. Compare between Time sharing operating system and multiprogramming operative system. (S-24)

Time Sharing OS	Multiprogramming OS
Time sharing OS system is more complex than multiprogramming OS.	Multiprogramming OS is less complex than Time Sharing OS.
In Time Sharing OS users can interact with system.	In Multiprogramming OS uses central interact with system.
It takes maximum time to execute a process.	It takes minimum time to execute a process.
Multiple processes can be executed simultaneously by using time slot.	Only a single process can be executed at a time.
Time sharing OS has fixed time slice.	Multiprogramming OS has no fixed time slice.
Time sharing system minimizes response time.	Multiprogramming OS has no maximize response time.
Example : Windows NT	Example : Mac OS

➢ 1.3 Command line, GUI based Operating System :

- **Command line Operating System :**



In CLI the user interacts with the operating system by typing commands/instructions on a command line. Various commands need to be typed for carrying out various jobs like creating, deleting, editing, renaming or printing a file. Usually in CLI each command represents an executable program, which is run when the command is typed with the proper parameters.

Advantages Command Line Interface (CLI):

1. If the user knows the correct commands then this type of interface can be much faster than any other type of interface.
2. This type of interface needs much less memory (Random Access Memory (RAM)) in order to use compared to other types of user interfaces.
3. CLI does not use as much CPU processing time as other UIs.

Disadvantages Command Line Interface (CLI):

1. CLI are not user-friendly because they require the user to remember a lot of commands.
2. Commands have to be typed precisely/correctly. If there is a spelling mistake then the command will not respond or fail or causes error.

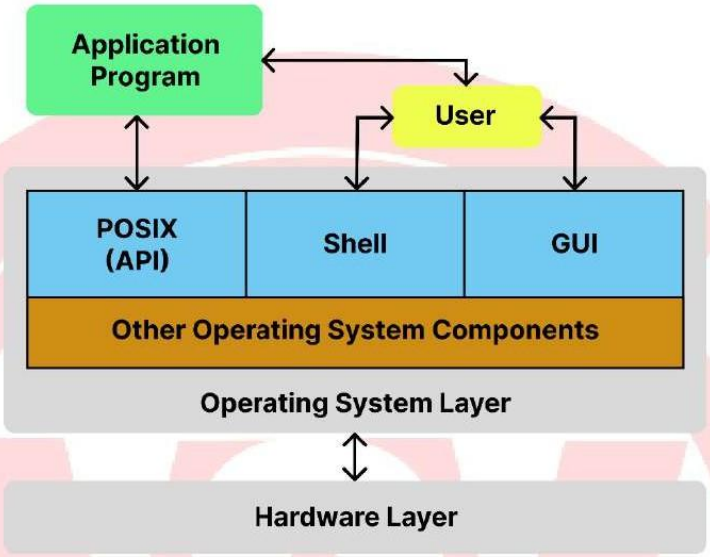
Types of CLI :

1. MS – DOS
2. UNIX

1. MS – DOS :

MS-DOS stands for Microsoft disk operating system. MS-DOS OS does not take much space for installation (about 8MB). MS-DOS OS give more control of the processes as it has a command line user interface. MS-DOS OS is a single user, single tasking operating system and not support graphics. It is not compatible with current browsers and the internet.

2. UNIX :



A Unix User has Several Options for Communicating With the Operating System

Unix is a multi-tasking, multi-processing, multi-user, and protected with built-in support for networking but not supported graphics. The Unix operating system was developed in 1969 by Ken Thompson, Dennis Ritchie and others at the AT&T Bell Laboratories. Unix is very flexible and can be installed on micro, mini, mainframe and Supercomputers. It is a very reliable, secured and robust operating system. It is a CLI based OS. The Unix shell is a command-Line Interface, similar in some ways to the old Dos prompt on the PC platform.

- **GUI (Graphical User Interface) Operating System :**

A Graphical User Interface (GUI) offers a user-friendly way to interact with an operating system by presenting information visually through windows and icons. This allows users to perform actions like opening files or running programs by simply clicking a mouse, eliminating the need to remember and type complex commands. The GUI streamlines user interaction, making computing more accessible and intuitive.

Advantages of GUI :

- GUIs are ease of use, operate and provides better accessibility.
- User can switch quickly between tasks on the GUI interface.
- GUI allows multiple programs and/or instances to be displayed simultaneously.
- GUI is convenient and user friendly. It also improved efficiency due to getting faster results.

Disadvantages of GUI :

- It uses more computer memory as the aim is to make it for user friendly.
- GUI becomes more complex if user needs to communicate with the computer directly.
- Difficulty of displaying all necessary controls because of limited Window space.
- GUIs has ambiguity of pictorial/graphical symbols.
- Slow speed because of long pointer operations.

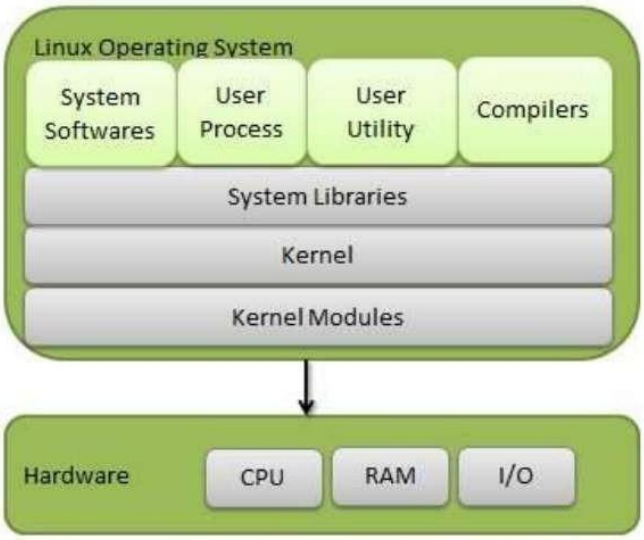
1. Windows :

The first version of windows was developed by Microsoft corporation in 1985. Windows OS has a Graphical User Interface (GUI) and is thus user-friendly. Windows OS is a multi-tasking operating system Some of the popular Windows OS are Windows 3.1, Windows 95, Windows 98, Windows 2000, Windows NT, Windows ME, Windows XP, Windows Vista and Windows 7, Windows 8, Windows 10, Windows 11 is the latest Windows operating system from Microsoft.

2. LINUX :

Linux is one of popular version of UNIX operating system. It is open source, portable, multi-user, multiprogramming operating system. Linux was developed by Linus Torvalds in 1992. Linux has both command line interface (CLI) and Graphical user Interface (GUI). Linux is a reliable and secure operating system.

Components of Linux Operating System :



- **Kernel :** Kernel is the core part of Linux. It is responsible for all major activities of this operating system. It consists of various modules and it interacts directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.
- **System Libraries :** System Libraries are special functions or programs using which application programs or system utilities accesses Kernel's features. These libraries implement most of the functionalities of the operating system and do not require kernel module's code access rights.
- **System Utility :** System Utility programs are responsible to do specialized, individual level tasks.

Features of Linux.

1) Open Source : Linux is free to use and its source code is open to everyone. Developers can study, modify, and share it.

2) Multitasking : Linux can run multiple tasks or applications at the same time without slowing down down.

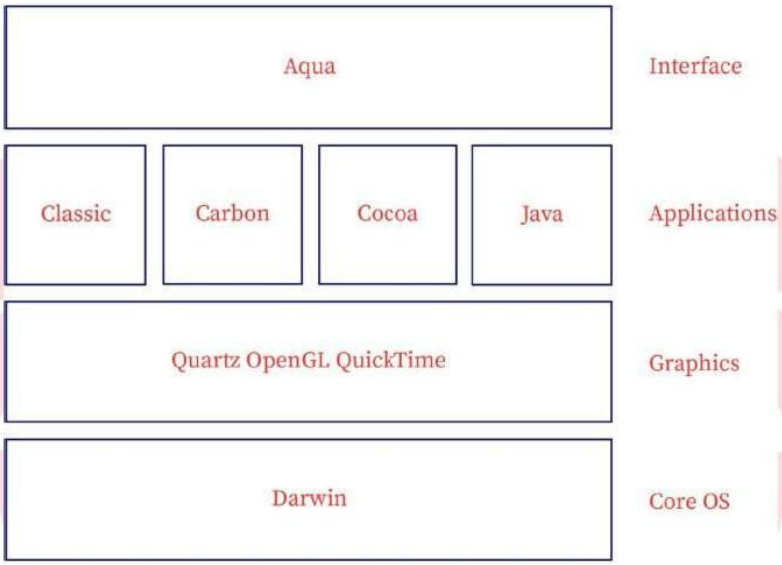
3) Multiuser : Many users can access the system simultaneously, each with their own accounts and permissions.

4) Portability : Linux can run on many types of devices, from phones to supercomputers, because it is highly adaptable.

5) Security : Linux is very secure, with features like file permissions, user authentication, and a strong firewall to protect against threats.

6) Lightweight : Linux can run on older or less powerful hardware, making it efficient and fast.

3. MaC OS :



Architecture of MAC OS

MacOS

The Macintosh operating system (Mac OS) is an operating system designed by Apple Inc. to be installed and operated on Apple Macintosh series of computers. It is a multi-user, multi-tasking operating system. Mac OS X was originally presented as the 10th major version of Apple's operating system for Macintosh computers and it is extremely secure, compatible, and easy to use.

Difference Between Command Line Interface (CLI) and Graphical User Interface (GUI)

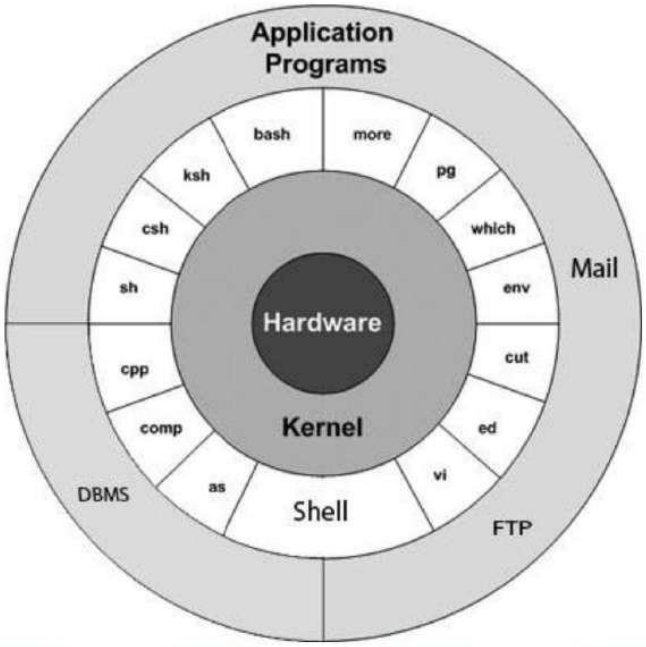
Q. Differentiate between command line based operating system and GUI based operating system (Any four points) (S-22, W-22, W-23)

Command Line Interface (CLI)	Graphical User Interface (GUI)

Interaction is by typing commands.	Interaction with devices is by graphics and visual components and icons.
Commands need to be memorized.	Visual indicators and icons are easy to understand.
Less memory is required for storage.	More memory is required as visual components are involved.
Use of keyboard for commands makes CLI quicker.	Use of mouse for interaction makes it slow.
Only keyboard is a resource used.	Mouse and keyboard both resources can be used.
Accuracy is high.	Accuracy is comparatively low.
Command line interface does not change, remains same over time.	Structure and design of Graphical user interface can change with updates.
Not much flexible.	GUI is more flexible.
Drag and drop features are usually absent. This makes the execution of certain commands lengthy or difficult.	Drag and drop features make certain command execution easier.
Examples: DOS, Unix etc.	Examples: Windows, Linux etc.

Structure of UNIX :

Q. Draw structure of UNIX operating system. (S-25)



Components of UNIX operating system :

- Kernel
- Shell
- Commands and utilities
- File and directories

1) Kernel:

The kernel is the heart of the operating system. It interacts with the hardware and performs most of the tasks like memory management, task scheduling and file management.

2) Shell:

The shell is the utility that processes your requests. When you type a command at your terminal, the shell interprets the command and calls the program that you want. The shell uses standard syntax for all commands. C Shell, Bourne Shell and Korn Shell are the most famous shells which are available with most of the Unix variant.

3) Commands and Utilities:

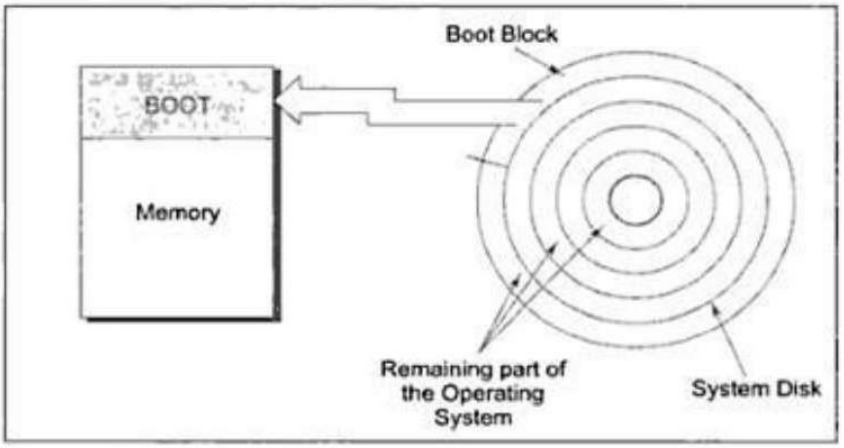
There are various commands and utilities which you can make use of in your day-to-day activities. CP, mv, cat and grep, etc. are few examples of commands and utilities. There are over 250 standard commands plus numerous others provided through 3rd party software. All the commands come along with various options.

4) Files and Directories:

All the data of Unix is organized into files. All files are then organized into directories. These directories are further organized into a tree-like structure called the file system.

Booting Process in UNIX :

Q. Explain Booting in UNIX.

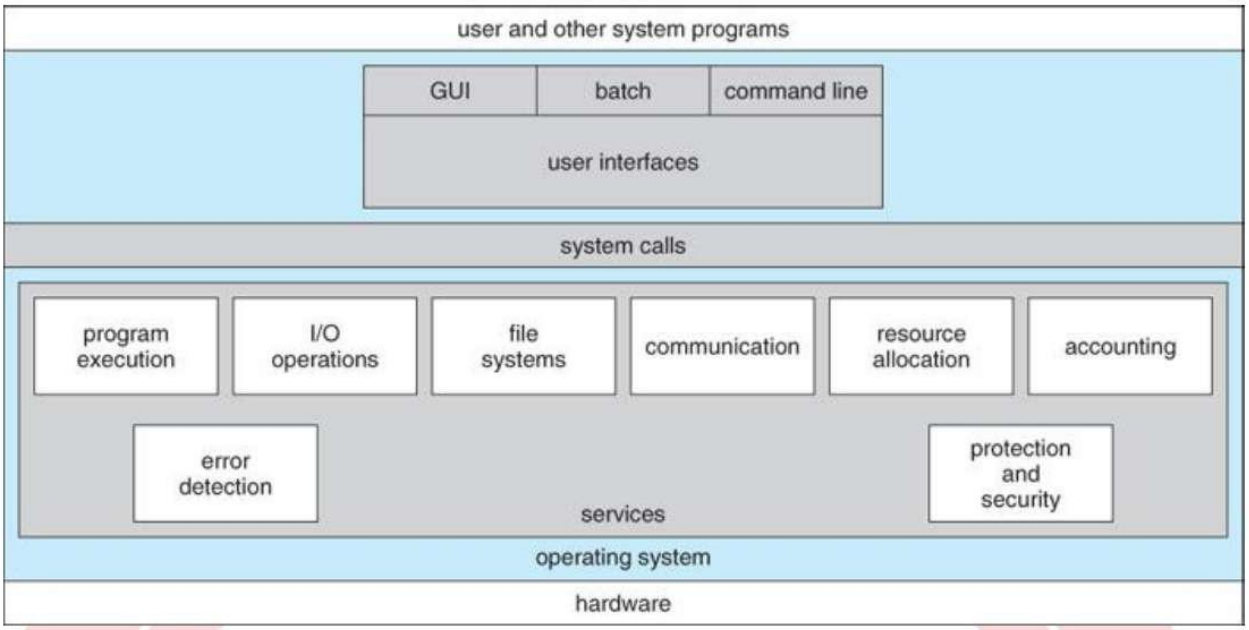


The loading of the operating system is achieved by a special program called BOOT. Generally, this program is stored in one (or two) sectors on the disk with a predetermined address. This portion is normally called "BOOT Block" as shown in dig. The ROM normally contains a minimum program. When one turns the computer "ON", the control is transferred to this program automatically by the hardware itself. This program in ROM loads the BOOT program in pre-determined memory locations. The beauty is to keep BOOT program as small as possible, so that the hardware can manage to load it easily and in a very few instructions. This BOOT program in turn contains to read the rest of the operating system into the memory. This is depicted in figures. The mechanism gives an impression of pulling oneself up. Therefore, its short form booting.

➢ 1.4 Different Services, System calls of Operating System :

- **Services of operating system :**

  Q. Explain services provided by OS. (W-19, S-22, W-22, S-23, W-23, S-24)



a. User Interface:

All operating system have a user interface that allows users to communicate with the system. Three types of user interfaces are available:

i) Command Line Interface (CLI)

ii) Batch interface

iii) Graphical user interface (GUI).

b. Program execution:

The operating system provides an environment where the user can conveniently run programs. It also performs other important task like allocation and deallocation of memory, CPU scheduling etc. It also provides services to end process execution either normally or abnormally by indicating error.

c. I/O operations:

When a program is running, it may require input/output resources such as a file or devices such as printer. So the operating system provides a service to do I/O.

d. File system manipulation:

Programs may need to read and write data from and to the files and directories. Operating System makes it easier for user programs to accomplish their task such as: opening a file, saving a file and deleting a file from the Storage disk.

e. Communication:

In the system, one process may need to exchange information with another process. Communication can be implemented via shared memory or through message passing in which packets of information are moved between processes by the Operating System.

f. Error detection:

Operating Systems detects CPU and memory hardware such as a memory error or power failure, a connection failure on a network or lack of paper in the printer etc.

g. Resource allocation:

Operating System manages resource allocation to the processes. These resources are CPU, main memory, File Storage and I/O devices.

h. Accounting:

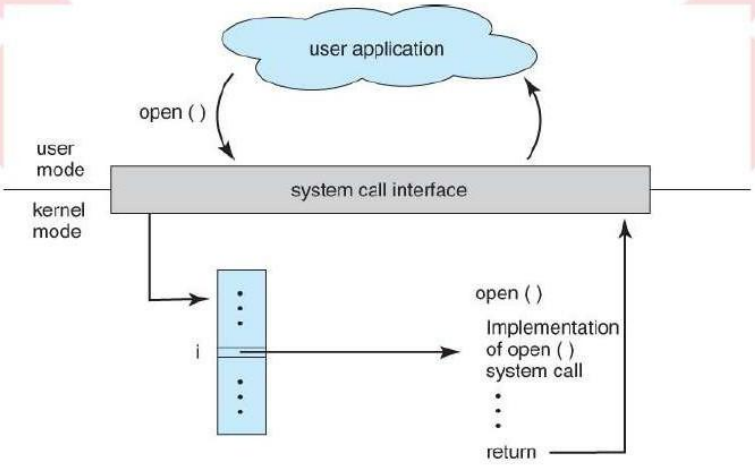
Operating System keeps track of usages of various computer resources allocated to users.

i. Protection & Security:

When several separate processes execute concurrently, one process should not interface with the other processes or operating system itself. Protection provides controlled access to system resources. Security is provided by user authentication such as password for accessing information.

- **System calls :**

Q. What is purpose of system call? State system calls with their functions. (W-19, S-22. W-22, S-23, W-23, S-24)



A system call is the programmatic way in which a computer system calls program requests a service from the kernel of the operating system it is executed on. System call provides an interface between a running program and operating system. It allows user to access services provided by operating system. This system calls are procedures written using C, C++ and assembly language instructions. Each system call is associated with a number that identifies itself.

- **Types of System calls :**

Process Control:

Program in execution is a process. A process to be executed must be loaded in main memory. While executing it may need to wait, terminate or create & terminate child processes.

Following are the functions of Process Control:

i. Create / terminate process
ii. Load / Execute process
iii. End / Abort process
iv. Ready / Dispatch process
v. Suspend / Resume process
vi. Get / Set process attributes
vii. Wait / Signal event
viii. Allocate and deallocate memory

File Management:

Q. Write any four systems call related to file management. (W-22)

System allows us to create and delete files. For create and delete operation System call requires the name of the file and other attributes of the file. File attributes include file type, file size, protection codes, accounting information and so on. System access these attributes for performing operations on file and directories.

Following are the functions of File management:

i. Create new file, delete existing file
ii. Open, close file
iii. Create and delete directories
iv. Read, write, reposition
v. Get/set file attribute

Device Management:

When a process is in running state, it requires several resources to execute. If the resource is available, it is assigned to the process. Once the resource is allocated to the process can read, write and reposition the device.

Following are the functions of Device management.

i. request device, release device
ii. read, write, reposition
iii. get/set device attributes
iv. logically attach or detach devices

Information Maintenance:

Transferring information between the user program and the operating system requires System call. Operating System keeps information about all its processes that can be accessed with System calls such as get process attributes and set process attributes.

Following are the functions of information maintenance.

i. get/set time or date
ii. get/set System data
iii. get/set process, file, or device attributes

Communication:

Processes in the system communicate with each other.
Communication is done by using two models:

i. message passing
ii. shared memory

For transferring messages, sender process connects itself to receiving process by specifying receiving process name or identity. Once the communication is over System close the connection between communicating processes.

Following are the functions of communication:

i. create, delete, communication connection.
ii. Send, receive messages.
iii. transfer status information.
iv. attach or detach remote device.

➢ 1.5 Operating System Components :

Q. Describe / List any components of O.S. (W-19, S-22, W-22, S-23, W-23, S-24)

Process management:

Q. List components of OS. Explain process management in detail. (W-19)

A program in execution is a process. The services provided through Process management are very important, if the Operating System supports multiple users as in the case of UNIX as MVS. The execution of a process must be sequential. A process needs various system resource including CPU time, memory, files and I/O devices to complete the job execution. These resources can be given to the process when it is created or allocated to it while it is running. The operating system responsible for the following activities in connection with process management.

a. Creation and deletion of user and system processes.

b. Suspension and resumption of processes

c. A mechanism for process synchronization

d. A mechanism for process communication

e. A mechanism for deadlock handling

Main memory management:

Q. Describe various activities performed by following operating system components. (S-22, W-23)

i) Main memory management ii) File management

Main memory is a large array of words or bytes, ranging in size from hundreds of thousands to billions. Each word or byte has its own address. Main memory is a repository of quickly accessible data shared by the CPU and I/O devices. The central processor reads instructions from main memory during the instruction fetch cycle and both reads and writes data from main memory during the data fetch cycle. The main memory is generally the only large storage device that the CPU is able to address and access directly. The operating system responsible for the following activities in connection with main memory management:

a. Keeping track of which parts of memory are currently being used and by whom.

b. Deciding which processes and data to move into and out of memory.

c. Allocating and deallocating memory space as needed.

File Management:

File management import component of all the operating, which deals with the management and organisation of various files in the system. Some examples of storage media are magnetic tape, magnetic disk and optical disk. Each of these media has its own properties like speed, capacity, and data transfer rate and access methods. A file system normally organized into directories to ease their use. These directories may contain files and other directories. The operating system responsible for the following activities in connection with file management:

a. The creation and deletion of files.

b. The creation and deletion of directories.

c. The support of primitives for manipulating files and directories.

d. The mapping of files onto secondary storage.

e. The backup of files on stable storage media.

I/O device management:

Input / output device management provides an environment for the better interaction between system and the I/O devices (such as printers, scanners, tape drives etc.) To interact with I/O devices in an effective manner, the operating system uses some special programs known as device drivers. The device drivers take the data that operating system has defined as a file and then translate them into streams of bits or a series of laser pulses (in regard with laser printer). The I/O subsystem consists of several components:

a. A memory management component that includes buffering, caching, spooling.

b. A general device driver interface.

c. Drivers for specific hardware device.

Secondary Storage Management:

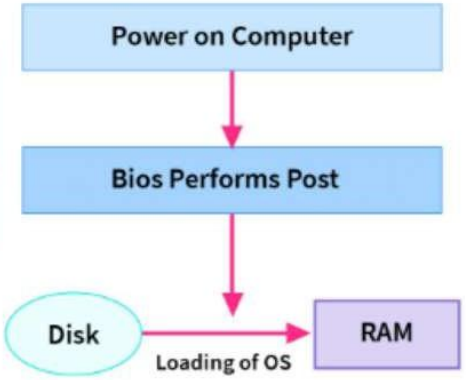
The computer system provides secondary storage to back up main memory. Secondary storage is required because main memory is too small to accommodate all data and programs. The data that it holds is lost when power is lost. Most of the programs including compilers, assemblers, word processors, editors, and formatters are stored on a disk until loaded into memory. Secondary storage consists of tapes drives, disk drives, and other media. The operating system is responsible for the following activities in connection with disk management.

(a) free space management

(b) storage allocation.

(c) disk scheduling.

- **Booting process :**



The procedure of starting a computer by loading the kernel is known as booting the System. On most computer systems, a small piece of code known as the bootstrap program as bootstrap loader locates the Kernel. This program is in the form of read-only memory (ROM), because the RAM is in an unknown state at system startup. ROM is convenient because it needs no initialization and cannot be infected by a computer virus. When the full bootstrap program has been loaded, it can traverse the file system to find the operating system kernel, load it into memory, and start its execution. It is only at this point that the system is said to be running.

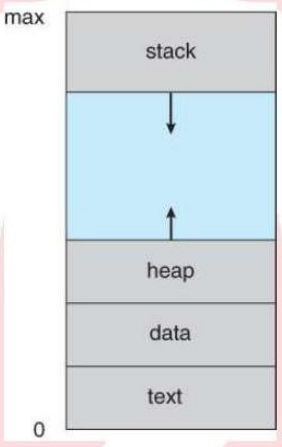
UNIT 2: Process Management

➢ 2.1 Processes

Define : Process, PCB (W – 22)

- **Process :**

Process is a program in execution. Process is also called as job, task and unit of work. The execution of a process must progress in a sequential fashion. A process is an instance of an executing program, including the current values of the program counter, registers and variables. Logically each process has its separate virtual CPU. A process is an activity and it has a program, input, output and a state.



Process in memory

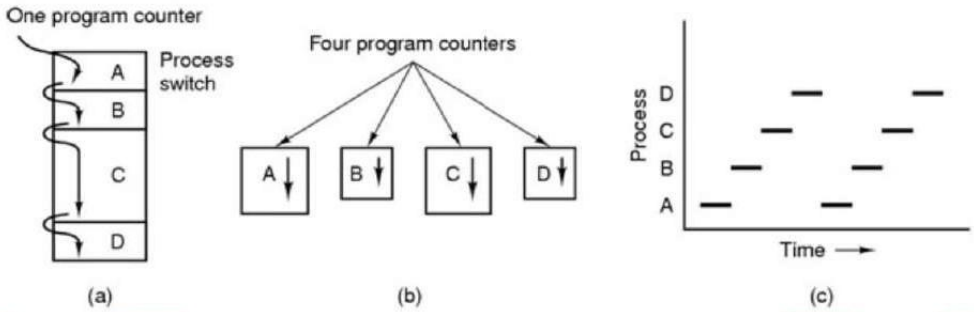
Each process has following sections:

- **Text section :** A Text section contains the program code.
- **Data section :** A Data section contains global and static variables.
- **Heap :** The heap is used for dynamic memory allocation, and is managed via calls to new, delete, malloc, free, etc.
- **Stack :** The stack is used for local variables. A process stack which contains the temporary data. Space on the stack is reserved for local variables when they are declared and the space is freed up when the variables go out of scope.
- **Program counter :** Program counter contains the contents of processor's registers.

- **Process Model:**

Explain in detail Process Model

In the process model, an operating system organizes operations into a number of sequential processes. A process is defined as an executing program, which includes its current state (program counter, register values, variables). Conceptually, each process has its own "virtual CPU." However, in reality, there's only one physical CPU. The illusion of multiple CPUs is achieved by rapidly switching the CPU back and forth between processes. This rapid switching is known as multiprogramming.



(a) illustrates four programs (A, B, C, D) and their respective program counters. In a multiprogramming environment, these programs appear to be running in parallel.

(b) depicts how the CPU switches between these four processes, each with its own independent flow of control. This means each program executes independently of the others, even though they share the same CPU.

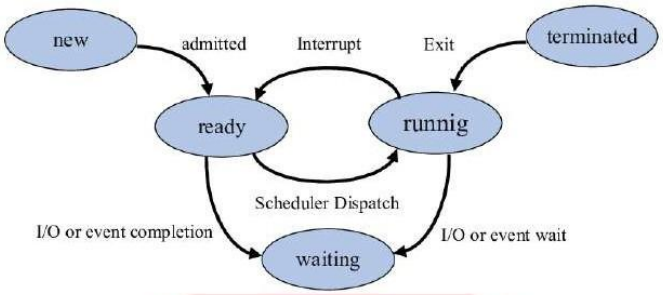
(c) shows the actual execution of processes over time. While it might appear that all processes are running simultaneously, in reality, only one process is executing at any given instant.

The CPU switches back and forth between processes, and the rate of these switches affects the perceived performance. Due to the overhead of CPU switching, the execution of the same processes might not be uniform or perfectly reproducible.

- **Process State :**

Draw and explain process state diagram (W – 19, S – 22, W – 22, S – 23, W – 23, S - 24)

The current activity of a process is known as its state. As a process executes, it changes state.



Process state diagram

A state diagram represents the different states in which a process can be at different times, along with the transitions from one state to another that are possible in the operating system.

Different process states are as follows :

1. **New:**

When a process enters into the system, it is in new state. In this state process is created. In new state the process is in job pool.

2. **Ready:**

When the process is loaded into the main memory, it is ready for execution. In this state the process is waiting for processor allocation.

3. **Running:**

When CPU is available, system selects one process from main memory and executes all the instructions from the process. So, when a process is in execution, it is in running state. In single user system, only one process can be in the running state. In multi-user system, there can be multiple processes which are in the running state.

4. **Waiting State:**

When a process is in execution, it may request for I/O resources. If the resource is not available, process goes into the waiting state. When the resource is available, the process goes back to ready state.

5. **Terminated state:**

When the process completes its execution, it goes into the terminated state. In this state the memory occupied by the process is released.

- Process Control Block (PCB) :

Explain PCB with diagram (W – 19, S – 22, W – 22, S – 24, W – 24, S - 25)

Process Control Block (PCB) is a data structure that contains information of the process related to it. It is also known as Task Control Block (TCB). The operating system groups all information that needs about a particular process into a data structure called as PCB. When a process is created, operating system creates a corresponding PCB and released whenever the process terminates. The basic purpose of PCB is to indicate the so far progress of a process.

Process Control Block

- **Pointer:** It is a stack pointer that is required to be saved when the process is switch from one state to another to retain the current position of the process.
- **Process State:** It indicates current state of a process. Process state can be new, ready, running, waiting and terminated.
- **Process Number:** Each process is associated with a unique number which is known process identification number.
- **Program Counter:** It indicates the address of the next instruction to be executed for the process.
- **Registers:** Registers includes accumulators, index registers, stack pointers and general purpose registers plus any condition code information.
- **Memory limits:** This field contains the information about memory management system used by the operating system. This may include page table, segment tables. etc,
- **List of open files:** This information includes the list of files opened for a process.

Following are the functions of PCB :

- **Context switching:**
  Allows the operating system to save the current state of a process and load the state of another process to enable multitasking.

- **Process scheduling:**
  provides the necessary information for the scheduler to determine which process to run next.

- **Memory management:**
  Facilitates the allocation and deallocation of memory to processes and track their memory usage.

- **Resource Management:**
  Tracks and manages the resources allocated to a process, such as file descriptors and I/O devices.

- **Process synchronization and communication:**
  Help manage process synchronization, including handling inter-process communication and synchronization primitives.

- **Process Control:**
  Assists in starting, pausing, resuming, and terminating processes.

- **Security and Access Control:**
  Ensures processes have the necessary permissions to process certain resources and data.

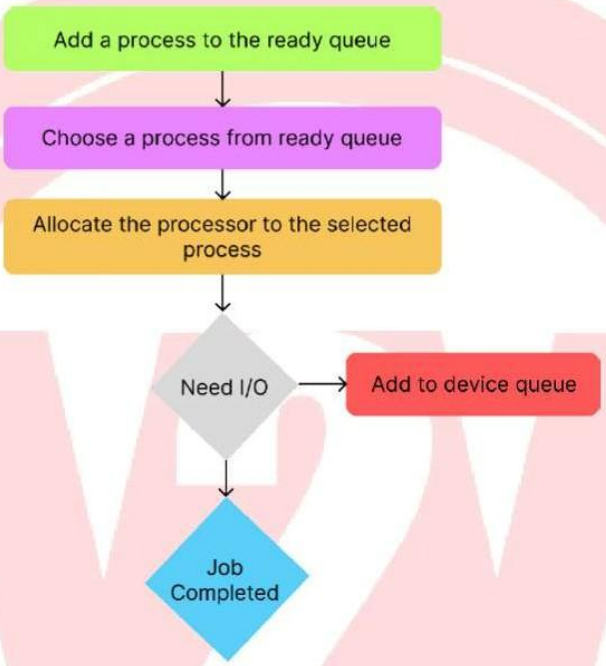
Difference between Program and Process : (W - 23)

Program	Process
It is a set of instructions that has been designed to complete a certain task.	It is an instance of a program that is currently being executed.
It is a passive entity.	It is an active entity.
It resides in the secondary memory of the system.	It is created when a program is in execution and is being loaded into the main memory.
It exists in a single place and continues to exist until it has been explicitly deleted.	It exist for a limited amount of time and it gets terminated once the task has been completed.
It is considered as a static entity.	It is considered as a dynamic entity.
It doesn't have a resource requirement.	It has a high resource requirement.
It requires memory space to store instructions.	It requires resources such as CPU, memory address, I/O during its working.
It doesn't have a control block.	It has its own control block, which is known as Process Control Block.

➢ 2.2 Process Scheduling

Explain the working of Process Scheduling

Procedure of determining the next process to be executed on the CPU is called process scheduling and the module of operating system that makes this decision is called the scheduler. Scheduling is that in which each process have some amount of time of CPU.



Process Scheduling

Working of process scheduling:

1. **Add to Ready queue:**
   New or ready-to-resume processes are added to the ready queue, which holds all processes waiting for CPU time.

2. **Select Process:**
   The short-term scheduler selects a process from the ready queue based on a scheduling algorithm.

3. **Allocate CPU:**
   The selected process is given access to the CPU for execution.

4. **Check for I/O Requirements:**
   If the process needs to perform an I/O operation, it is moved to the device queue and waits until the I/O is complete.

5. **Job Completion check:**

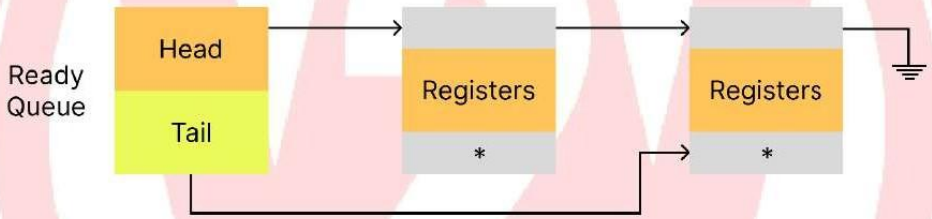
If no I/O is needed, the system checks whether the process has finished its execution.

6. **Process completion or Requeuing:**
   If completed, the process is terminated. If not completed and requires more CPU time, it is placed back in the ready queue.

- **Scheduling queues :**

With suitable diagram, describe use of scheduling queues in process scheduling. (S - 22)

For a uniprocessor system, there will never be more than one running process. If there are more than one process, the rest will have to wait until the CPU is free and can be rescheduled. The processes, which are ready and waiting to execute, are kept on a list called the ready queue. The list is generally a linked list. A ready queue header will contain pointers to the first and last PCBs in the list. Each PCB has a pointer field which points to the next process in the ready queue. There are also other queues in the system. When a process is allocated the CPU, it executes for a while and eventually quits, is interrupted or waits for the occurrence of a particular event, such as the completion of an I/O request.



Scheduling Queue

Example :

As new process is initially put in the ready queue. It waits there until it is selected for execution or is dispatched. Once, the process is assigned the CPU and is executing, one of several events could occur :

1. The process could create a new sub-process and wait for the termination of the sub-process.
2. The process could issue an I/O request and then be placed in an I/O queue.
3. The process could be removed forcibly from the CPU, as a result of an interrupt, and again put in the ready queue.

In the first two cases, the process transition from the waiting state to ready state occurs. A process continues this cycle until it terminates. When the process terminates, it is removed from all queues and its PCB and resources are de-allocated.

- **Scheduler :**

State and describe types of scheduler (W – 19, S - 23)

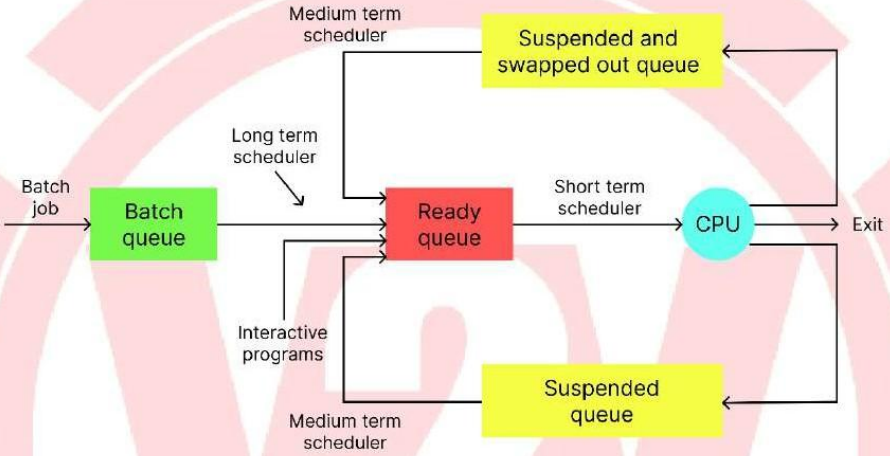
Schedulers are special system software's which handles process scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run.

Schedulers are of three types

a. Long Term scheduler

b. Short Term scheduler

c. Medium Term scheduler



Types of Schedulers:

1. **Long term scheduler:**
   Long term scheduler is also called Job scheduler. It determines which process are admitted to the system for processing. Processes are selected from the queue and loads into the main memory for execution.

2. **Medium term scheduler:**
   Medium term scheduler is part of swapping function. Sometimes it removes the process from memory. It also reduces the degree of multiprogramming.

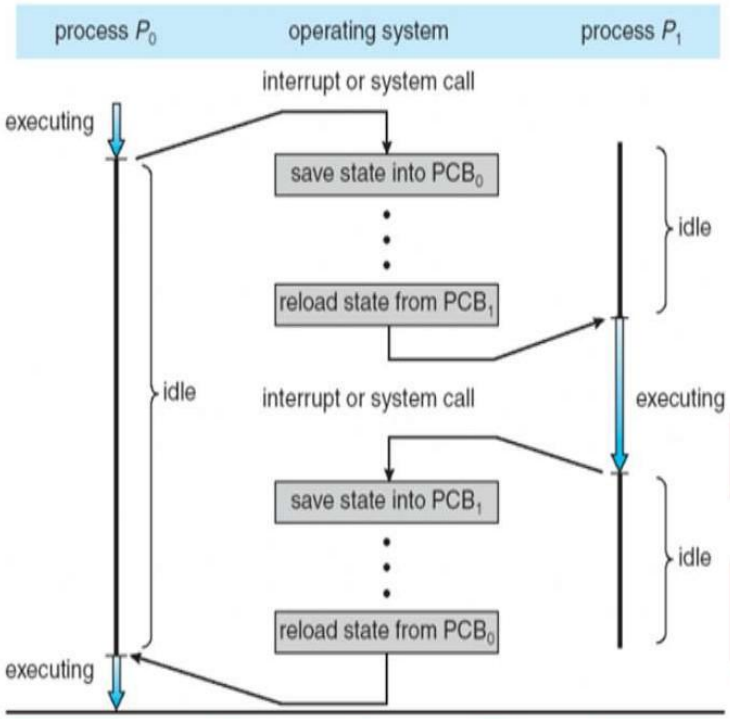
3. **Short term scheduler:**
   Short term scheduler is also called CPU scheduler. It selects the process from queue which are ready to execute and allocate the CPU for execution. Short term scheduler is faster than long term scheduler.

Difference between Long Term Scheduler and Short Term Scheduler (W - 22)

Long Term Scheduler	Short Term Scheduler
Job scheduler	CPU scheduler
Selects processes from job pool and loads them into memory for execution.	Selects processes from ready queue which are ready to execute and allocates CPU to one of them.
Job pool and ready queue	Ready queue and CPU
Executes much less frequently. It executes when memory has space to accommodate new processes.	Executes frequently. It executes when CPU is available for allocation.
Speed is less than short-term scheduler.	Speed is fast.
It controls the degree of multiprogramming.	It provides lesser control over degree of multiprogramming.
It chooses a good process that is a mix-up of input/output bound and CPU bound.	It chooses a new process for a processor quite frequently.
Used in batch processing systems.	Used in time-sharing systems.

- **Context switch :**

Explain working of CPU switch from process to process with neat labelled diagram(W - 23)
OR
Describe how context switch is executed by operating system (S - 24)



A context switching is a mechanism that store and restore the state or context of a CPU in process control block so that a process execution can be resumed from the same point at a late time. When the scheduler switches the CPU from one process to another process, the context switch save the contents of all process registers for the process being removed from the CPU in its process control block. Context switch includes two operations such as state save and state restore. State save operation stores the current information of running process into its PCB. State restore operation restores the information of process to be executed from its PCB.

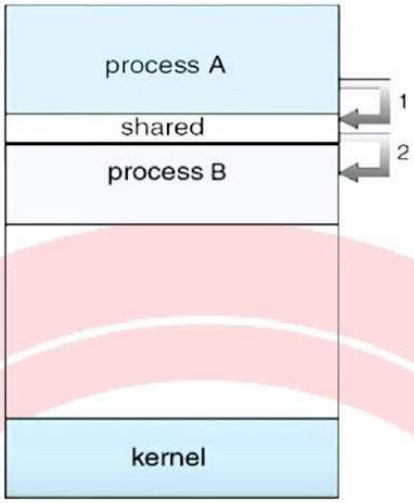
➢ 2.3 Inter Process Communication :

With neat diagram explain inter process communication model. (W – 19, S – 22, W – 22, S – 23, W – 23, S – 24, S - 25)

Cooperating processes require an inter-process communication (IPC) mechanism that will allow them to exchange data and information.

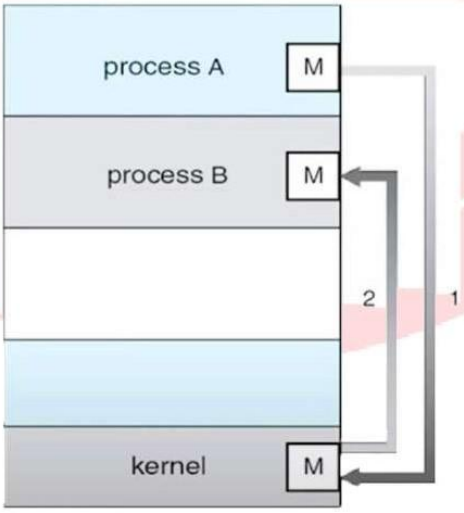
There are two models of IPC.
(a) Shared memory
(b) Message passing

(a) Shared memory :



IPC using shared memory requires a region of shared memory among the communicating process. A shared-memory region resides in the address space of the process creating the shared memory segment. Other processes that wish to communicate using this shared memory segment must attach it to their address space. Normally, the operating system does not allow one process to access the memory region of another process. Shared memory requires that two or more processes agree to remove this restriction. They can then exchange information by reading and writing data in the shared areas. Shared memory allow maximum speed and convenience of communication. Shared memory is faster than message passing.

(b) Message passing:



In this model, communication takes place by exchanging messages between co-operating processes. It allows processes to communicate and synchronize their action without sharing the same address space. It is particularly useful in a distributed environment when communication process may reside on a different computer connected by a network. Communication requires sending and receiving messages through the kernel. The processes that want to communicate with each other must have a communication link between them.

Comparison between Shared Memory Model and Message Passing Model:

Shared Memory Model	Message Passing Model
The shared memory region is used for communication.	A message-passing facility is used for communication.
It is used for communication between processes on a single processor or multiprocessor system.	It is typically used in a distributed environment.
The code for reading and writing the data from the shared memory should be written explicitly by the application programmer.	No such code is required here as the message-passing facility provides a mechanism for communication and synchronization of actions performed by the communicating processes.
Here the processes need to ensure that they are not writing to the same location simultaneously.	It is useful for sharing small amounts of data as conflicts need not be resolved.
Faster communication strategy.	Relatively slower communication strategy.
No kernel intervention.	It involves kernel intervention.
It can be used in exchanging larger amounts of data.	It can be used in exchanging small amounts of data.
Data from a client process may need to be transferred to a server process for modification before being returned to the client.	Web browsers, Web servers.

➢ 2.4 Threads:

- **Benefits of Thread :**

Describe benefits of Thread .

1. **Responsiveness:**
   Multithreading allows an application to remain responsive to the user even when parts of it are blocked or performing long operations. For example, a multithreaded web browser can still allow user interaction in one thread while an image is loading in another. This increases the overall responsiveness experienced by the user.

2. **Resource Sharing:**
   Threads, by default, share the memory and resources of the process they belong to. This shared memory space allows for efficient code sharing, enabling an application to have multiple threads of activity all operating within the same address space.

3. **Economy:**
   Allocating memory and resources for process creation is costly. Multithreading offers an economical alternative because threads share the resources of the process to which they belong. It is more efficient to create and manage threads than processes. For instance, in Solaris 2, creating a process is about 30 times slower than creating a thread, and context switching between threads is approximately five times faster.

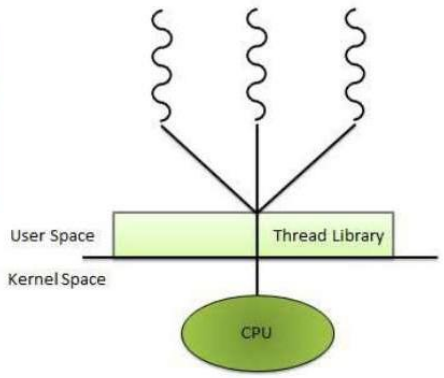
4. **Utilization of Multiprocessor Architectures:**
   Multithreading greatly enhances the benefits of multiprocessor architectures. In such systems, each thread can run in parallel on a different processor, leading to increased concurrency. While a single-threaded process can only utilize one CPU regardless of how many are available, multithreading on a multi-CPU machine can achieve true parallelism. Even in a single-processor architecture, the CPU can rapidly switch between threads, creating the illusion of parallelism.

- **User and Kernel level threads**

Explain user level thread and Kernel level thread with its advantages and disadvantages (S - 24)

User Level Thread :

The threads implemented at the user level are known as user level threads. In a user thread, all of the work of thread management is done by the application and the kernel is not aware of the existence of threads.



User Level Thread

The thread library contains code for creating and destroying threads, for passing message and data between threads, for scheduling thread execution and for saving and restoring thread contexts. User level threads are generally fast to create and manage.

Advantages of User Level Threads:

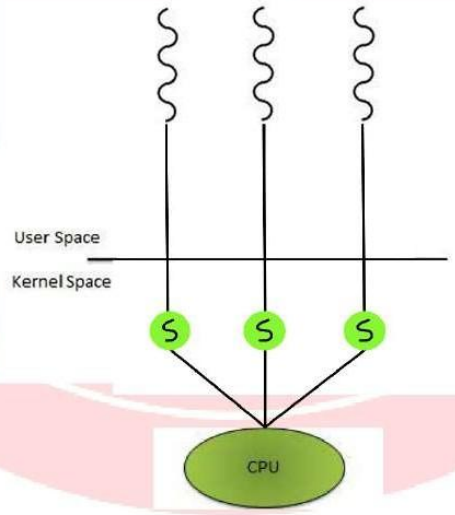
- User level thread can run on any operating system.
- A user thread does not require modification to opera operating systems.
- User level threads are fast to create and manage.
- User thread library easy to portable.

Disadvantages User Level Threads :

- At most, one user level thread can be in operation at one time, which limits the degree of parallelism.
- If a user thread is blocked in the kernel, the entire process (all threads of that process) are blocked.
- It is not appropriat for a multiprocessor system.

kernel level Threads :

The threads implemented at kernel level are known as kernel threads. Kernel threads are supported directly by the operating system.



Kernel Level Thread

The kernel performs thread creation, scheduling and management in kernel space. Because thread management is done by the operating system, kernel threads are generally slower to create and manage than are user threads. However, since the kernel is managing the threads, if a thread performs a

blocking system call, the kernel can schedule another thread in the application for execution. Also in a multiprocessor environment, the kernel can schedule threads on different processors.

Advantages of Kernel Level Threads :

- kernel can Simultaneously schedule multiple threads from the same process on multiple processes.
- If one thread in a process is blocked, the kernel can schedule another thread of the same process.

Disadvantages of kernel Level Threads :

- The kernel -level threads are slow and inefficient.
- Transfer of control from one thread to another within same process requires a mode switch to the kernel.

Comparison between User Level Threads and Kernel Level Threads :

User Level Thread	Kernel Level Thread
User level threads are implemented by user.	Kernel level threads are implemented by Operating System (OS).
User level threads are faster to create and manage.	Kernel level threads are slower to create and manage.
User level threads can run on any operating system.	Kernel level threads are specific to the operating system.
OS doesn't recognize user level threads.	Kernel level threads are recognized by OS.
Context switch time is less.	Context switch time is more.
No hardware support is required for context switching.	Hardware support is needed.
If one user-level thread performs a blocking operation then the entire process will be blocked.	If one kernel thread performs a blocking operation then another thread can continue execution.
User-level threads can be created and managed more quickly.	Kernel-level threads take more time to create and manage.
In user-level threads, each thread has its own stack, but they share the same address space.	In kernel-level threads have their own stacks and their own separate address spaces, so they are better isolated from each other.
Example: Java Thread, POSIX thread, Mach C-Threads.	Example: Window Solaris.

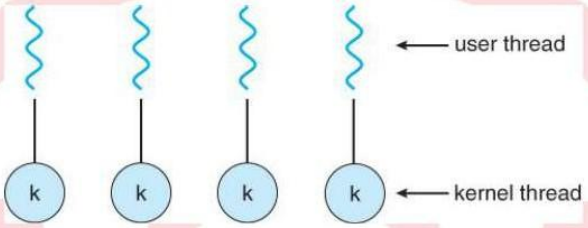
- **Multithreading Models :**

Explain multithreading model in detail. (W – 19, S - 22)

Many system provide support for both user and kernel threads, resulting in different multithreading models.

Following are three multithreading models:

1. One – to - One Model :



The one-to-one model maps each user thread to a kernel thread. It provides more concurrency than the many-to-one model by allowing another thread to run when a thread makes a blocking system call; it also allows multiple threads to run in parallel on multiprocessors. Windows NT, Windows 2000 and 08/12 implement the one-to-one model.

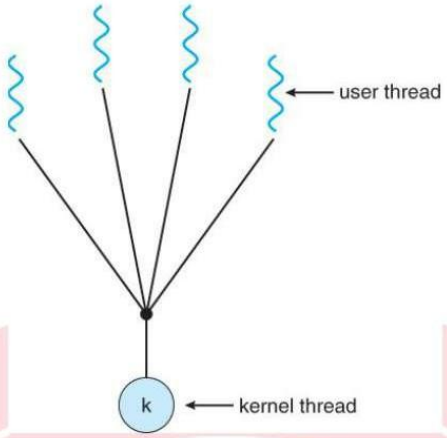
Advantages of one-to-one Model:

- More concurrency, because of multiple threads can run in parallel on multiple CPUs.
- Less complication in the processing.

Disadvantages of one-to-one Model:

- Every time with user's thread, kernel thread is created.
- Limiting the number of total threads.
- Kernel thread is an overhead.
- It reduces the performance of system.

2. Many – to – One Model :



The many-to-one model maps many user level threads to one kernel thread. Thread management is done in user space, so it is efficient, but the entire process will block if a thread makes a blocking system call. Only one method thread can access the kernel at a time. Multiple threads are unable to run in parallel on multiprocessors.

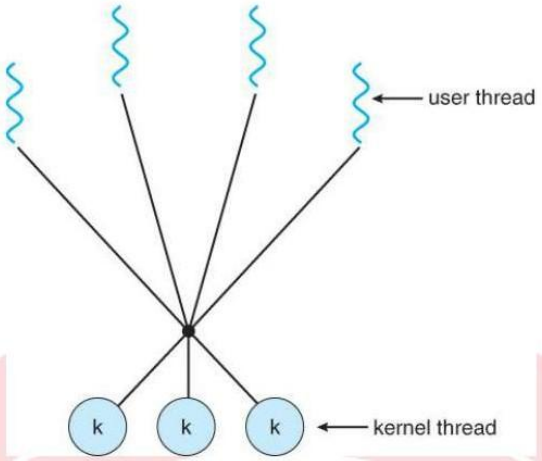
Advantages of many-to-one Model:

- Totally portable.
- Efficient system in terms of performance.
- One kernel thread controls multiple user threads.

Disadvantages of many-to-one Model:

- Cannot take advantage of parallelism.
- One block call blocks all user threads.

3. Many – to – Many Model :



The many-to-many model maps multiple user level threads to a smaller or equal number of kernel threads. The number of kernel threads may be specific to either a particular application or a particular machine. This model allows developer to create as many threads. Concurrency is not gained because the kernel can schedule only one thread at a time. Solaris 2, IRIX, HP-UX and Tru64 UNIX support this model.

Advantages of Many-to-Many model:

- Many threads can be created as per user's requirement.
- Multiple kernel or equal to user threads can be created.

Disadvantages of Many-to-Many Model:

- True concurrency cannot be achieved.
- Multiple threads of kernel is an overhead for operating system.
- Performance is less.

Differences between Process and Thread: (W - 23)

Process	Thread
A process is a program under execution i.e. an active program.	A thread is a subset of the process.
Process runs in separate memory space.	Thread runs within the process in a shared memory space.
Process is heavy weight	It is a lightweight
The process has its own Process Control Block, Stack, and Address space.	Thread has Parents PCB, its own Thread Control Block, and Stack and Common Address Space.

Context switching between the process is more expensive.	Context switching between threads of the same process is less expensive.
Processes are independent.	Threads are dependent.
Process is controlled by the operating system.	Threads are controlled by a programmer

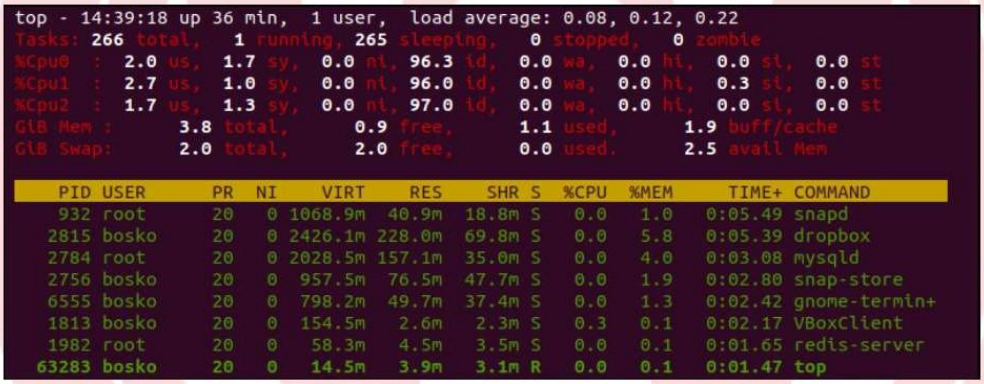
➢ 2.5 Execute process commands like :

Write syntax for following commands: i)Sleep ii)Kill (W – 19, W - 22)

Describe use of ps and wait commands with suitable example. (S - 22)

Explain following commands with their syntax i) kill ii) Sleep iii) Wait iv) Exit (W - 23)

- top



The top stands for table of processes. As the name suggests, it displays a list of processes in table form. The top command is used to show the Linux processes. It provides a dynamic real-time view of the running system. Usually, this command shows the summary information of the system and the list of processes or threads which are currently managed by the Linux Kernel.

Syntax: top

- ps

Explain 'PS' command with any four options. (W - 19)

What is the use of PS command? Write long forms of UID, PID in the output of this command. (W - 23)

It is used to display character slices of a process. This command when executed without options, it lists the process associated with a user at a particular terminal.

Syntax: $ ps [options]

Example: $ ps

Output:

PID	TTY	TIME	CMD
12330	pts/0	00:00:00	bash
21621	pts/0	00:00:00	ps

Each line in the output shows PID, the terminal with which the process is associated, the cumulative processor time that has been consumed since the process has been started and the process name.

UID - (Owner) User-id
PPID - Parent Process-id

Options :

1. **-f:**

It is used to display full listing of attributes of a process. It includes UID (User ID),PID (Process ID), PPID (Parent ID), C (Amount of CPU time consumed by the process) and STIME (chronological time that has elapsed since the process started). TTY (The terminal device from which the process was launched), TIME (The cumulative CPU time required to run the process, CMD (The name of the program that was started.

Example: $ ps -f

Output:

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	1	0	0	19:58	?	00:00:01	/sbin/init
root	2	0	0	19:58	?	00:00:00	[kthreadd]
root	3	2	2	19:58	?	00:00:00	

2. **-u:**

show the activities of any specified user at any time.

Example: $ ps -u abc

Output:

PID	TTY	TIME	CMD
1053	?	00:00:00	systemd
1062	?	00:00:00	(sd-pam)
1074	tty1	00:00:00	zsh

3. -a:

It shows the process of all users.

Example: $ ps -a

Output:

PID	TTY	TIME	CMD
27011	pts/0	00:00:00	man
27016	pts/0	00:00:00	less
27499	pts/1	00:00:00	ps

4. -e:

It displays processes including user and system processes.

Example: $ ps -e

Output:

PID	TTY	TIME	CMD
1	?	00:00:05	systemd
2	?	00:00:00	kthreadd
3	?	00:00:00	csmd
7	?	00:00:01	gjs
8	?	00:00:00	Gsd-sound

Below is a list of all ps commands:

① **-a:** Displays all processes except session headers and processes without a terminal.

② **-c:** Displays additional scheduler information about the process.

③ **-d:** Displays all processes except session header.

④ **-e:** Displays all processes.

⑤ **-f:** Displays a full format listing

⑥ **-j:** Displays job information.

⑦ **-l:** Displays a long listing

⑧ **-w:** Use wide output format, for unlimited width displays.

⑨ **-y:** Don't show process flags.

⑩ **-A:** Displays all processes.

⑪ **-F:** Use extra full output.

⑫ **-H:** Display processes in a hierarchical format (showing parent processes).

⑬ **-L:** Displays process threads.

⑭ **-M:** Display security information about the process.

⑮ **-N:** Display the opposite of the specified parameters.

⑯ **-U userlist:** Displays processes owned by a userid listed in userlist.

⑰ **-V:** Displays the version of ps.

⑱ **-Z:** Display the security context information.

- **kill**

Syntax: kill pid

Kill command is used to stop execution of particular process by sending an interrupt signal to the process.

- **wait:**

Syntax: wait [pid]

wait command waits for running process to complete and return the exit status. The process is identified by pid (process ID). If pid is not given, wait waits for all currently active child processes and the return status is zero.

- **sleep:**

Syntax: sleep NUMBER [SUFFIX]

The sleep command pauses the execution for specified time in command. It pauses execution for amount if time defined by NUMBER, and SUFFIX can be "s" for seconds (default), "h" for hours, "m" minutes, and "d" for days.

- **exit:**

Syntax: exit

used to quit the shell

OR

Syntax: exit [n]

The terminal window will close and return a status of n. It also returns value as which is available to script's parent.

- **nice**

The Linux nice command allows us to launch processes with altered priorities, which affects process scheduling. The "nice command is used to start a new process with a specific priority, known as the "nice value". A higher nice value lowers the process's priority, while a lower (negative) nice value increases it.

Syntax: nice -n [nice value] [command/process]

The -n flag is used to specify the Linux nice value ranging from -20 for most/highest favourable scheduling to +19 for least/lowest favourable scheduling.

Give Commands to perform following task:

i) To add delay in script

ii) To add terminate a process.

i) To add delay in script

To add delay in script sleep command is used.

sleep Command:

The sleep command is used to delay for a specified amount of time. The sleep command pauses for an amount of time defined by NUMBER. SUFFIX may be "s" for seconds (the default), "m" for minutes, "h" for hours, or "d" for days.

Syntax: sleep NUMBER [suffix]

Example: sleep 10

Delay for 10 seconds.

ii) To terminate a process

To terminate a process exit Command is used.

exit Command:

The exit Command terminates a script, just as in a c Program. It can also return a value, which is available to the script's parent process. Issuing the exit Command at the shell prompt will cause the shell to exit.

Syntax: exit

Example: exit

To exit from running State exit command is used

Write the output of following commands

i) wait 2385018

ii) sleep 9

iii) ps -u Asha

i) wait 2385018

Wait command waits until the termination of specified process ID 2385018.

ii) sleep 9

Sleep command is used to delay for 9 seconds during the execution of a process. i.e. it will pause the terminal for 9 seconds.

iii) ps -u Asha.

PS command with -u is used to display data / processes for specific user Asha.

Write Unix command for following:

i)create a folder OSY

ii) create a file FIRST in OSY folder

iii) List/display all files and directories.

iv) Write command to clear the screen

Ans

i) create a folder OSY:

$mkdir OSY

ii)create a file FIRST in OSY folder:

$cd OSY

$cat>FIRST or $ touch FIRST

iii) List/display all files and directories:

$ls

iv) to clear screen:

$clear

Unit - III CPU Scheduling

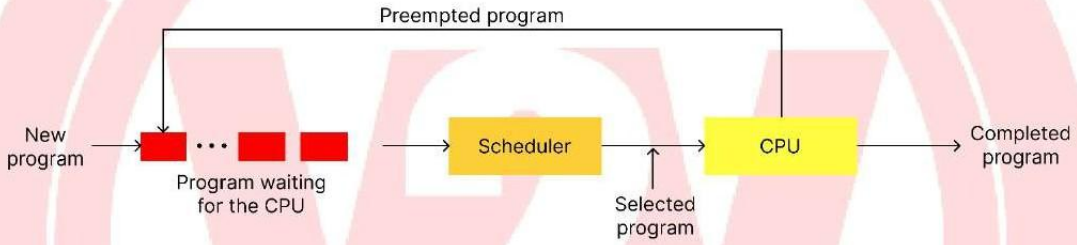
3.1 Scheduling

Basic concept

Scheduling is a key function of an operating system. It involves the process manager handling the removal of a running process from the CPU and selecting another process to be executed based on a specific strategy.

The scheduler is the kernel component (module/program) responsible for deciding which program should be executed on the CPU. It selects a program to be executed next for execution.

When the CPU becomes free, the scheduler chooses another program from the set of programs waiting to run on the CPU. The decision of which program gets the CPU and for how long is called scheduling.



A Schematic of Scheduling

Objectives of scheduling:

Depending on the system, the user and designer might expect the scheduler to have the following objectives for scheduling:

1) **Fairness:**
   Fairness is defined as, the degree to which each process gets an equal chance to execute
   A scheduler makes sure that each process gets its fair chance its fair share of the CPU time.

2) **Maximize Resource Utilization:**
   The scheduling techniques should keep the resources of the system busy.

3) **Avoid Indefinite Postponement:**
   A process should not experience an unbounded wait time before or while receiving service.

4) **Response Time:**
   A scheduler should minimize the response time for interactive users.

5. **Turnaround:**
   A scheduler should minimize the time so batch users must wait for an output time.

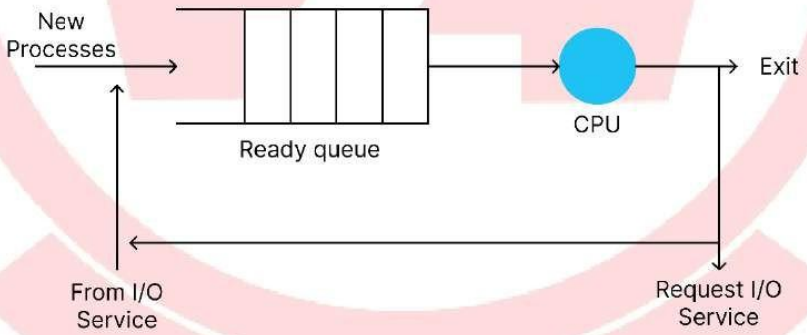
6. **Maximize Throughput:**
   A scheduling disc should maximize the number of jobs processed per unit time.

7. **Enforce Priorities:**
   If the system assign priorities to processes, the scheduling mechanism should favour the higher-priority processes.

- **Scheduling Model:**

CPU scheduling is the process of selecting a process from the ready queue and assigning the CPU to it. CPU scheduling algorithms decides which of the processes in the ready queue is to be allocated to the CPU. The CPU scheduler is the part of the operating system that selects the next process to which the CPU will be allocated de-allocates the CPU from the process currently executing, and allocates the CPU to the newly selected process. The algorithm used by the scheduler to carry out the selection of a process for execution is known as scheduling algorithm.

Every process in OS that requests a CPU service carries out following sequence of actions.



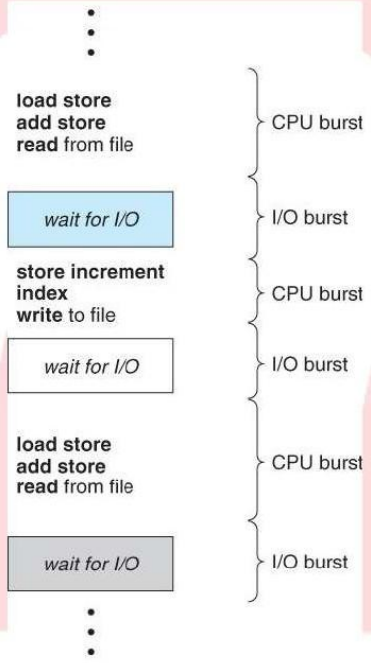
In first action, join the ready queue and wait for CPU service. In second action, execute (receive CPU service) for the duration of the current CPU burst or for the duration of the time slice (time out). In third action, join the I/O queue to wait for I/O service or return to the ready queue to wait for more CPU service. In fourth action, terminate and exit if service is completed i.e. if there are no more CPU or I/O burst. If more service is required, return to the ready queue to wait for more CPU service.

- **CPU and I/O burst cycle**

CPU burst cycle:
It is a time period when process is busy with CPU.

I/O burst cycle:
It is a time period when process is busy in working with I/O resources.

Diagram for I/O Burst and CPU Burst cycle



Execution is an Alternating sequence of CPU and I/O Burst

I/O Burst and CPU Burst Cycle:

A process execution consists of a cycle of CPU execution and I/O wait. A process starts its execution when CPU is assigned to it, so process execution begins with a CPU burst cycle. This is followed by an I/O burst cycle when a process is busy doing I/O operations. A process can frequently switch from CPU burst cycle to I/O burst cycle and vice versa. The complete execution of a process starts with CPU burst cycle, followed by I/O burst cycle, then followed by another CPU burst cycle, then followed by another I/O burst cycle and so on. The final CPU burst cycle ends with a system request to terminate execution.

➢ 3.2 Preemptive and Non - preemptive scheduling, scheduling criteria

- **Preemptive scheduling:**

In preemptive scheduling if a higher priority process is entering the system, the currently running process is stopped and the CPU transfers the control to the higher priority process. The currently running process may be interrupted and moved to the ready state by the operating system. The preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

Circumstances of pre-emptive:

- Process switch from running to ready state
- Process switch from waiting to ready state

Example:

- SJF (Shortest Job First)
- Priority RR (Round Robin)

- **Non-pre-emptive Scheduling:**

Once the CPU has been allocated to a process, the process keeps the CPU until it releases CPU either by terminating or by switching to waiting state. Non-pre-emptive algorithms are designed so that once a process enters the running state, it cannot be pre-empted until it completes its allotted time.

Circumstances of non pre-emptive:

- When process switches from running to waiting state
- When process terminates.

Example:

- FCFS (First come first serve)

State difference between preemptive scheduling and non-preemptive scheduling (S – 22, S - 23)

Preemptive scheduling	Non-preemptive scheduling
Even if CPU is allocated to one process, CPU can be preempted to other process if other process is having higher priority.	once, the CPU has been allocated to a process the process keeps the CPU until releases either by terminating or by switching to waiting state.
Preemptive scheduling is more complex.	Non-preemptive scheduling is simple.
Throughput is less	Throughput is high.

It is suitable for real time system	It is not suitable for real time system
Algorithm design is Complex	Algorithm design is simple
The system resources are used efficiently	The system resources are not used efficiently
Only the processes having higher priority are scheduled.	Processes having any priority can get scheduled.
It does not treat all processes as equal.	It treats all process as equal.
It has high overhead.	It has low overhead.
Circumstances for Pre-emptive:	**Circumstances for Non-preemptive:**
• process switch from running to ready state	• process switches from running to waiting state
• process switch from waiting to ready state	• process terminates
Example: Round Robin Example: Priority algorithms.	**Example:** FCFS algorithm.

Explain any four scheduling criteria. (W – 19, S – 22, W – 22, S - 23)

- **Scheduling criteria**

  - ✓ CPU Utilization
  - ✓ Throughput
  - ✓ Turnaround Time (TAT)
  - ✓ Waiting Time
  - ✓ Balanced Utilization
  - ✓ Response Time

1) CPU Utilization:

CPU utilization is defined as, the percentage of time the CPU is busy in executing processes. For higher utilization, CPU must be kept as busy as possible, that is, there must be some process running at all times. CPU utilization may range from 0 to 100 percent. In real system it should range from 40 percent to 90 percent.

2) Throughput:

Throughput is defined as, the total number of processes that a system can execute per unit of time. If the CPU is busy executing processes then work is being done. One measure of work is the number of processes that are completed per unit of time called throughput. For long processes, this rate may be one process per hour, for short transactions throughput might be 10 processes per second.

3) Turnaround Time (TAT):

The interval from the time of submission of a process to the time of completion is the turnaround time. From the point of view of a particular process, the important criterion is how long it takes to execute that process. Turnaround time is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU and doing I/O.

4) Waiting Time:

Waiting time is defined as, the time spent by a process while waiting in the ready queue. A process waits in ready queue till CPU is allocated to it. Once the CPU is allocated the process, it starts its execution and if required requests for resources. When I/O request completes, it goes back to ready queue. In ready queue again it waits for CPU allocation.

5) Balanced Utilization:

Balanced utilization is defined as, the percentage of time all the system resources are busy. It considers not only the CPU utilization but the utilization of I/O devices, memory, and all other resources. To get more work done by the system, the CPU and I/O devices must be kept running simultaneously.

6) Response Time:

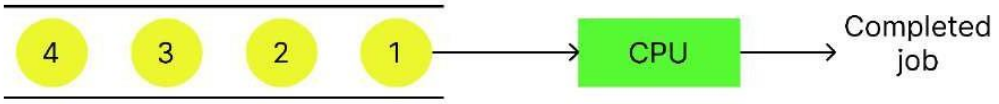
Response time is defined as, the time elapsed between the moment when a user initiates a request and the instant when the system starts responding to this request. It is the amount of time it takes to start responding, but not the time that it takes to output that response.

➢ 3.3 Types of Scheduling algorithms:

- First Come First Serve (FCFS)
- Shortest Job First (SJF)
- Shortest Remaining Time Next (SRTN)
- Round Robin (RR)
- Priority Scheduling
- Multilevel Queue Scheduling

1. First Come First Serve (FCFS)

It is the simplest type of algorithm in which the process that requests the CPU first is allocated the CPU first. In FCFS scheduling algorithm processes are scheduled in the order they are received. The FCFS algorithm is easily implemented by using a queue data structure for the ready queue.



It can be implemented with FIFO (First In First Out (FIFO)) queue. The FCFS scheduling algorithm is non-pre-emptive. Once the CPU has been allocated to a process, that process keeps the CPU until it wants to release the CPU, either by terminating or by requesting I/O.

Advantages of FCFS:

- FCFS is easier to understand and implement as processes are simply to be added at the end and removed from the front of queue.
- FCFS is well suited for batch systems.

Disadvantages of FCFS:

- Average waiting time is very large.
- FCFS is not an attractive alternative on its own for a single processor system.

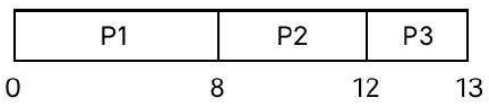
Example:

Find average waiting time and turn around time.

process	Arrival time	Burst time
P1	0	8
P2	1	4
P3	2	1

Process	Arrival time	Burst time	Completion time	Turnaround time	Waiting time
P1	0	8	8	8	0
P2	1	4	12	11	7
P3	2	1	13	11	10

Gantt chart:



Completion time:

Completion time is calculated by using Gantt chart

Turnaround time:

Turnaround time = Completion time - Arrival time

P1 -> 8 - 0 = 8 ms

P2 -> 12 - 1 = 11 ms

P3 -> 13 - 2 = 11 ms

Average turn around time:

Average turn around time = Turnaround time of all processes / No. of processes

= (8 + 11 + 11) / 3

= 10 ms

Waiting time:

Waiting time = Turnaround time - Burst time

P1 -> 8 - 8 = 0 ms

P2 -> 11 - 4 = 7 ms

P3 -> 11 - 1 = 10 ms

Average waiting time:

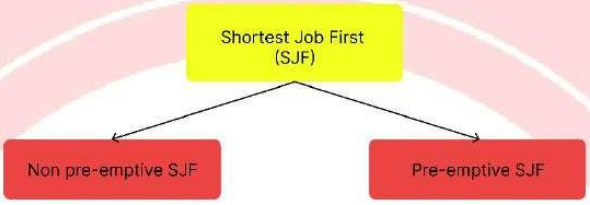
Average waiting time = waiting time of all processes / No. of processes

= (0 + 7 + 10) / 3

= 5.67 ms

2. Shortest Job First (SJF)

The SJF scheduling algorithm is also known as Shortest Process Next (SPN) scheduling algorithm or Shortest Request Next (SRN) scheduling algorithm that schedul the processes according to the length of the CPU burst they require. In SJF algorithm, the process with the shortest expected burst time is assigned to CPU. Hence, the name is shortest Job First. Jobs or processes are processed in the ascending order of their CPU burst times. Every time the job with smallest CPU burst-time is selected from the ready queue. If the two processes having same CPU burst time then they will be scheduled according to FCFS algorithm.



1. Non pre-emptive SJF:

In this method, if CPU is executing one job, it is not stopped in between before completion.

2. Pre-emptive SJF:

In this method, while CPU is executing a job, if a new job arrives with smaller burst time, then the current job is pre-empted and the new job is executed. It is also called Shortest Remaining Time First (SRTF).

Example:

Calculate average Turnaround time and waiting time from following:

Process	Arrival Time	Burst Time
P1	0	8
P2	1	4
P3	2	9
P4	3	5

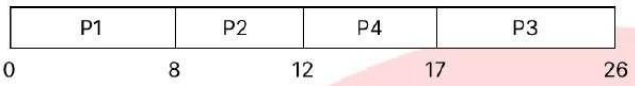
Answer:

Using non pre-emptive SJF

Process	Arrival Time	Burst Time	Completion Time	Turnaround Time	Waiting Time
P1	0	8	8	8	0

P2	1	4	12	11	7
P3	2	9	26	24	15
P4	3	5	17	14	9

Gantt chart:



Completion Time:

Compile completion time is calculated by using Gantt chart.

Turnaround time:

Turnaround time = completion time - arrival time

P1 -> 8 - 0 = 8 ms

P2 -> 12 - 1 = 11 ms

P3 -> 26 - 2 = 24 ms

P4 -> 17 - 3 = 14 ms

Average Turnaround time = Turnaround time of all processes / No. of processes

= (8 + 11 + 24 + 14) / 4

= 57 / 4

= 14.25 ms

Waiting Time:

Waiting Time = Turnaround Time - Burst Time

P1 -> 8 - 8 = 0 ms

P2 -> 11 - 4 = 7 ms

P3 -> 24 - 9 = 15 ms

P4 -> 14 - 5 = 9 ms

Average waiting time = Waiting time of all processes / No. of processes

= (0 + 7 + 15 + 9) / 4

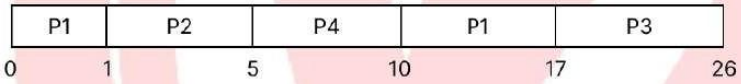
= 31 / 4

= 7.75 ms

Using pre-emptive scheduling

process	Arrival Time	Burst Time	completion Time	Turnaround Time	waiting Time
P1	0	8	17	17	
P2	1	4	5	4	
P3	2	9	26	24	
P4	3	5	10	7	

Gantt chart:



Completion time:

Completion time is calculated by using Gantt chart.

Turnaround Time:

Turnaround Time = completion Time - Arrival Time

P1 -> 17 - 0 = 17 ms

P2 -> 5 - 1 = 4 ms

P3 -> 26 - 2 = 24 ms

P4 -> 10 - 3 = 7 ms

Average Turnaround time = Turnaround time of all processes / No. of processes

= (17 + 4 + 24 + 7) / 4

= 13 ms

Waiting time:

waiting time = Turnaround Time - Burst Time

P1 -> 17 - 8 = 9 ms

P2 -> 4 - 4 = 0 ms

P3 -> 24 - 9 = 15 ms

P4 -> 7 - 5 = 2 ms

Average waiting Time = Waiting time of all processes / Number of processes

= (9 + 0 + 15 + 2) / 4

= 26 / 4

= 6.5 ms

Attempt:

How pre-emptive scheduling is better than non-pre-emptive scheduling by solving your problem using SJF (Solve it by using the preemptive SJF and non-pre-emptive SJF also). (W - 23)

Process	Arrival Time	Burst Time
P1	0	8
P2	1	4
P3	2	9
P4	3	5

Following are the reason, why pre-emptive scheduling is better than non-preemptive scheduling.

Pre-emptive scheduling is quite flexible because critical processes are allowed to access the CPU because they come in the ready queue and no matter which process is currently running.

Non-pre-emptive scheduling is tough because if an essential process is assigned to the ready queue, the CPU process is not be interrupted.

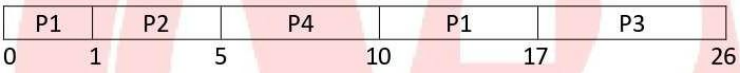
Pre-emptive scheduling allows a process to be interrupted in the midst of its execution, taking the CPU away and allocating it to another process.

Non-pre-emptive scheduling ensures that a process relinquishes control of the CPU only when its finishes with its current CPU burst.

Pre-emptive SJF:-

Process	Arrival Time	Burst Time	Waiting Time (ms)	Turnaround time (ms)
P1	0	8	10 - 1 = 9	17 - 0 = 17
P2	1	4	1 - 1 = 0	5 - 1 = 4
P3	2	9	17 - 2 = 15	26 - 2 = 24
P4	3	5	5 - 3 = 2	10 - 3 = 7

Gantt chart:



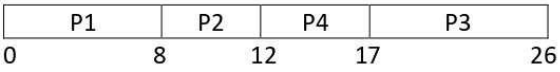
Average waiting time = Waiting time of all processes / Number of processes
= (9 + 0 + 15 + 2) / 4
= 26 / 4
= 6.5 ms

Average Turn-Around Time = Turn-Around time of all processes / Number of processes
= (17 + 4 + 24 + 7) / 4
= 52 / 4
= 13 ms

Non-pre-emptive SJF:

Process	Arrival Time	Burst Time	Waiting Time (ms)	Turnaround Time (ms)
P1	0	8	0	8 - 0 = 8
P2	1	4	8 - 1 = 7	12 - 1 = 11
P3	2	9	17 - 2 = 15	26 - 2 = 24
P4	3	5	12 - 3 = 9	17 - 3 = 14

Gantt chart:



Average waiting time = waiting time of all processes / Number of process
= (0 + 7 + 15 + 9) / 4
= 31 / 4
= 7.75 ms

Average Turn-Around time = Turn-Around time of all processes / Number of process
= (8 + 11 + 24 + 14) / 4
= 57 / 4
= 14.25 ms

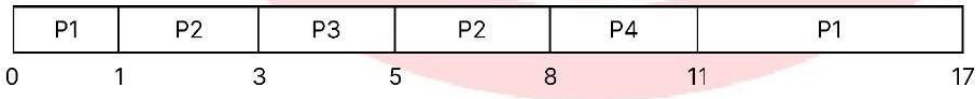
3. Shortest Remaining Time Next (SRTN)

SRTN scheduling algorithm is a pre-emptive form of Shortest Job First (SJF) scheduling algorithm. The shortest remaining time next also known as shortest Time to Go (STG) scheduling algorithm. In this algorithm a job is chosen whose burst time is the shortest. For a new job, its burst time is compared with burst time of current job. If new job needs less time to complete than the current job, then, the current job is blocked and the new job is run. It is used for batch systems.

Example:

Processes	Arrival Time	CPU Burst
P1	0	7
P2	1	5
P3	3	2
P4	4	3

Gantt chart:



Job	Arrival Time	Burst time	Finish Time	Turnaround Time	Waiting Time
P1	0	7	17	17	10
P2	1	5	8	7	2
P3	3	2	5	2	0
P4	4	3	11	7	4

			Average	33/4 = 8.25	16/4 = 4

Advantages :

1. It minimizes average waiting time.
2. More efficient for short processes.

Disadvantages :

1. It may cause starvation risk.
2. It may cause high overhead.
3. Difficult to implement.

Explain Round Robin algorithm with suitable example. (W - 19)

4. Round Robin (RR)

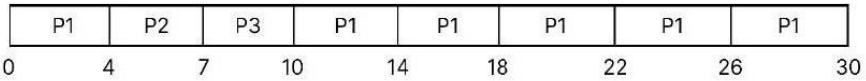
Round Robin is the pre-emptive process scheduling algorithm. The Round Robin scheduling algorithm is designed especially for time sharing systems. Processes are dispatched in a First In First Out (FIFO) sequence but each process is allowed to run for only a limited amount of time. A small unit of time called a Time Quantum or Time Slice is defined. The ready queue is treated as a circular queue. The CPU scheduler goes around the ready queue, allocating CPU to each process for a time interval of up to 1 time quantum. New processes are added to the tail of the ready queue.

Example:

Process	Burst Time
P1	24
P2	3
P3	3

Time quantum : 4ms

The resulting Round Robin schedule is as follows:



CPU is allocated to process P1 for 4ms. Since it requires another 20 milliseconds, it is preempted after the first time quantum and the CPU is given to the next process in the queue, process P2. Process P2 does not need 4 milliseconds, so it quits before its time quantum expires. The CPU is then given to the next process, process P3. Once each process has received 1 time quantum, the CPU returns to process P1 for an additional time quantum.

Advantages of Round Robin scheduling algorithm

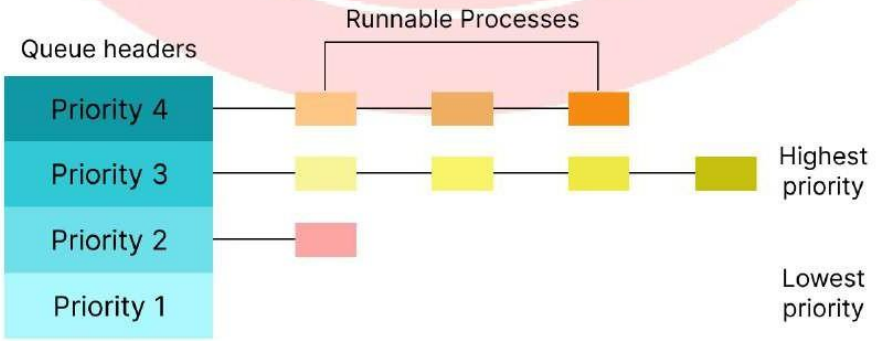
1. It is efficient for time sharing systems.
2. Round Robin increases the fairness among the processes.
3. In Round Robin scheduling algorithm overhead on processor is low.

Disadvantages of Round Robin scheduling algorithm

1. In Round Robin the processes may take long time to execute.
2. In Round Robin scheduling care must be taken in choosing quantum value.
3. Throughput in Round Robin scheduling algorithm is low if time quantum is too small.

5. Priority Scheduling

A priority is associated with each process and the CPU is allocated to the process with the highest priority, hence it is called Priority scheduling. Equal priority processes are scheduled in FCFS order. The priority scheduling can be either pre-emptive or non-pre-emptive. When process enters into the ready queue. Its priority is compared with the priority of the current running process. In a pre-emptive priority scheduling algorithm, at any time. The CPU is allocated to process if the priority of the newly arrived process is higher than the priority of the currently running process. In non-preemptive priority scheduling algorithm if priority of newly arrived process is higher than also currently running process with low priority is not get interrupted



Advantages of Priority scheduling Algorithm:

- It is simple to use
- Important processes are never made to wait because of the execution of less important processes.

Disadvantages of Priority scheduling Algorithm:

- It suffers from the problem of starvation of lower priority processes.
- Apriority scheduling can leave some low priority waiting processes indefinitely for CPU.

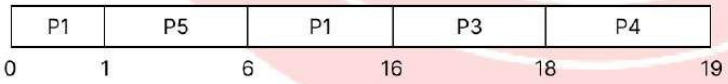
Example :

Consider the following set of processes assumed to have arrived at time 0 in the order P1, P2, P3, ---- Ps, with the length of the CPU burst time given in milliseconds. Calculate the average TAT and waiting Time

Process	Burst Time	Priority
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

Process	Burst Time	Priority	Completion Time	Turnaround Time	Waiting Time
P1	10	3	16	16	6
P2	1	1	1	1	0
P3	2	3	18	18	16
P4	1	4	19	19	18
P5	5	2	6	6	1

Gantt chart:



Completion Time:

Completion time is calculated by using Gantt chart

Turnaround Time:

Turnaround Time = Completion Time - Arrival Time

P1 = 16 - 0 = 16 ms

P2 = 1 - 0 = 1 ms

P3 = 18 - 0 = 18 ms

P4 = 19 - 0 = 19 ms

P5 = 6 - 0 = 6 ms

Average Turnaround Time = Turnaround time of all processes / Number of Processes

= (16 + 1 + 18 + 19 + 6) / 5

= 12ms

Waiting Time:

Waiting Time = Turnaround Time - Burst Time

P1 = 16 - 10 = 6 ms

P2 = 1 - 1 = 0 ms

P3 = 18 - 2 = 16 ms

P4 = 19 - 1 = 18 ms

P5 = 6 - 5 = 1 ms

Average Waiting Time = Waiting Time of all processes / Number of processes

= (6 + 0 + 16 + 18 + 1) / 5

= 8.2ms

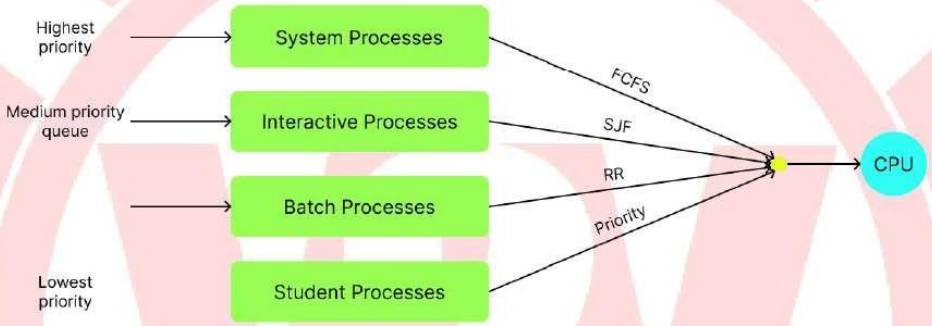
With neat diagram explain multilevel queue scheduling. (S - 23)

6. Multilevel Queue Scheduling

Multilevel queue scheduling classifies processes into different groups. It partitions the ready queue into several separate queues. The processes are permanently assigned to one queue based on some properties such as memory size, priority, process type, etc. Each queue has its own scheduling algorithm. In a system there are foreground processes and background processes. So system can divide processes into two queues: one for background and other for foreground. The interactive processes are known as foreground processes and the batch processes are known as background processes. Foreground queue can be scheduled with Round Robin algorithm where as background queue can be scheduled by First Come First Serve algorithm.

Example:-

Consider all the processes in the system are divided into four groups system, interactive, batch and student processes queue. Each queue contains processes. CPU based on scheduled queues may be priority, total burst time or process types.



Multilevel priority queue scheduling with scheduling algorithm

Advantages MLQ Scheduling Algorithm:

1. In MLQ the processes are permanently assigned to their respective queues and do not move between queues. This results is low scheduling overhead.
2. In MLQ one can apply different scheduling algorithms to different processes.
3. There are many processes which we are not able to put them in the one single queue which is solved by MLQ scheduling as we can now put them in different queues.

Disadvantages MLQ Scheduling Algorithm:

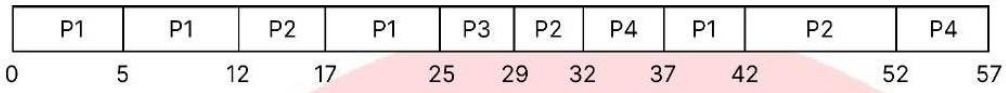
1. The processes in lower priority queues may have to starve for CPU in case processes are continuously arriving in higher priority queues.
2. In MLQ the process does not move from one queue to another queue.

Consider following table, which contains four processes P1, P2, P3, and P4 with their arrival times and required CPU burst (in milliseconds):

Process	Arrival Time	CPU Burst

P1	0	25
P2	12	18
P3	25	4
P4	32	10

Gantt Chart:



Waiting Time

- Waiting time of Process P1 = 5 + 12 = 17
- Waiting time of Process P2 = 12 + 10 = 22
- Waiting time of Process P3 = 0
- Waiting time of Process P4 = 52 - 37 = 15
- Average Waiting Time = (17 + 22 + 0 + 15)/4 = 54/4 = 13.5 msec.

Turnaround Time

- TAT of process P1 = 42 - 0 = 42
- TAT of process P2 = 52 - 12 = 40
- TAT of process P3 = 29 - 25 = 4
- TAT of process P4 = 57 - 32 = 25
- Average TAT = (42 + 40 + 4 + 25)/4 = 111/4 = 27.75 msec.

Compare Short Job First (SJF) and Shortest Remaining Time (SRTN) Scheduling (S - 24)

Short Job First (SJF)	Shortest Remaining Time (SRTN)
1. It is a non-preemptive algorithm	1. It is a preemptive algorithm
2. It involves less overhead than SRTN.	2. It involves more overheads than SJF.
3. It lead to comparatively lower throughput.	3. It leads to increased throughput as execution time is less.

4. It minimizes the average waiting time for each process.	4. It may or may not minimize the average waiting time for each process.
5. It involves lesser number of context switching.	5. It involves higher number of context switching.
6. Short processes are executed first and then following by longer processes.	6. Shorter processes run fast and longer processes show poor response time.

Following are different questions asked in exam regarding different types of scheduling :

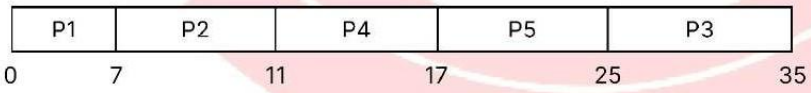
The jobs are scheduled for execution as follows:

Process	Arrival Time	Burst Time
P1	0	7
P2	1	4
P3	2	10
P4	3	6
P5	4	8

i) SJF ii) FCFS
Also find average waiting time using Ganatt chart.

Answer :

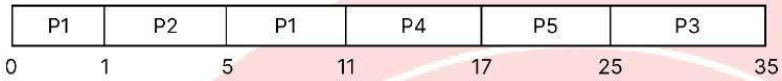
SJF Non-Preemptive SJF:
Gantt chart:



Process	Arrival Time	Burst Time	Waiting Time
P1	0	7	0
P2	1	4	7-1 = 6
P3	2	10	25-2 = 23
P4	3	6	11-3 = 8
P5	4	8	17-4 = 13

Average waiting time = waiting time of all process / total number of process
= (0+6+23+8+13)/5
= 50/5
= 10

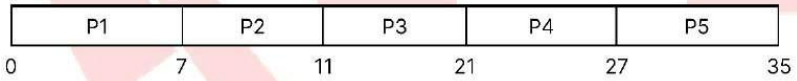
Preemptive SJF
Gantt Chart:



Process	Arrival Time	Burst Time	Waiting Time
P1	0	7	0 + (5 - 1) = 4
P2	1	4	1 - 1 = 0
P3	2	10	25 - 2 = 23
P4	3	6	11 - 3 = 8
P5	4	8	17 - 4 = 13

Average waiting Time = waiting time of all processes / Total number of processes
= (4+0+23+8+13) / 5
= 9.6

ii) FCFS:
Gantt chart:



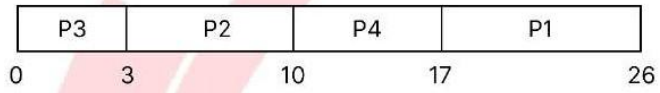
Process	Arrival Time	Burst Time	Waiting Time
P1	0	7	0 - 0 = 0
P2	1	4	7 - 1 = 6
P3	2	10	11 - 2 = 9
P4	3	6	21 - 3 = 18
P5	4	8	27 - 4 = 23

Average waiting time = waiting time of all processes / Total number of processes
= (0 + 6 + 9 + 18 + 23) / 5
= 56 / 5
= 11.2

Solve given problem by using SJF and FCFS scheduling algorithm using Gantt chart. Calculate the average waiting time for each algorithm.

Process	Burst time (in ms)
P1	9
P2	7
P3	3
P4	7

Gantt Chart SJF:



waiting Time:

P1 = 17

P2 = 3

P3 = 0

P4 = 10

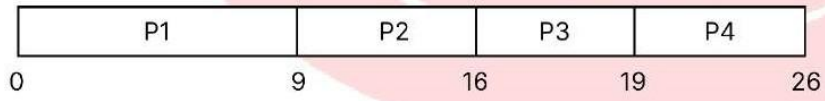
Average waiting time = waiting time of all processes / Number of processes

= (17+3+0+10) / 4

= 30/4

= 7.5 milliseconds (ms)

Gantt chart FCFS:



Waiting Time:

P1 = 0

P2 = 9

P3 = 16

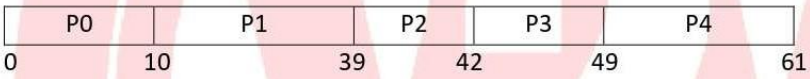
P4 = 19

Average waiting time = waiting time of all processes / Number of processes
= (0 + 9 + 16 + 19) / 4
= 44 / 4
= 11 milliseconds (ms)

Solve given problem by using FCFS Scheduling algorithm. Draw Correct Grantt chart & Calculate average waiting time and average turnaround time.

Process	Arrival Time	Burst Time
P0	0	10
P1	1	29
P2	2	3
P3	3	7
P4	4	12

Gantt Chart



FCFS
waiting time
P0 = 0
P1 = (10 - 1) = 9
P2 = (39 - 2) = 37
P3 = (42 - 3) = 39
P4 = (49 - 4) = 45

Average waiting time = Waiting time of all process / Number of process
= (0 + 9 + 37 + 39 + 45) / 5
= 26 ms

Turnaround Time
P0 = 10
P1 = 39 - 1 = 38
P2 = 42 - 2 = 40
P3 = 49 - 3 = 46
P4 = 61 - 4 = 57

Average Turnaround time = Turnaround time of all process / Number of process

= (10 + 38 + 40 + 46 + 57) / 5

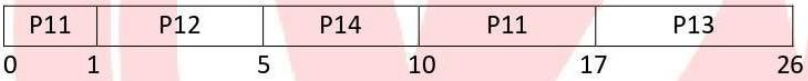
= 191 / 5

= 38.2 ms

Attempt Solve given problem by using i) Pre-emptive SJF ii) Round Robin (Time Slice = 3 ms)
Calculate average waiting time using Gannt chart.

Process	A.T.	B.T.(in ms)
P11	0	8
P12	1	4
P13	2	9
P14	3	5

i) Pre-emptive SJF:



Waiting Time = (Total Completion time - Burst time) - Arrival time

P11 = (17 - 8) - 0 = 9 ms.

P12 = (5 - 4) - 1 = 0 ms.

P13 = (26 - 9) - 2 = 15 ms.

P14 = (10 - 5) - 3 = 2 ms.

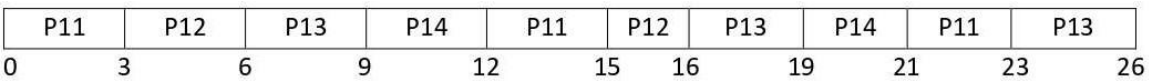
Average waiting time :

=(9 + 0 + 15 + 2) / 4

= 26 / 4

= 6.5 ms

ii) Round Robin (Time slice = 3 ms)



Waiting time :

P11 = (23 - 8) - 0 = 15 ms.

P12 = (16 - 4) - 1 = 11 ms.

P13 = (26 - 9) - 2 = 15 ms.

P14 = (21 - 5) - 3 = 13 ms.

Average waiting time:

=(15 + 11 + 15 + 13) / 4

= 54 / 4

= 13.5 ms

What is the average turnaround time for the following process using:

1. FCFS scheduling algorithm.
2. SJF non-preemptive scheduling algorithm.
3. Round Robin Scheduling algorithm.

Process	Arrival Time	Burst Time
P1	0	8
P2	1	4
P3	2	1

i) Using FCFS:

Gantt Chart:

P1	P2	P3

0 8 12 13

Process	Arrival time	Burst time	Completion time	Turnaround time
P1	0	8	8	8
P2	1	4	12	11
P3	2	1	13	11

Completion Time:

Completion Time is calculated by using Gantt chart.

Turnaround Time:

Turnaround Time = Completion Time - Arrival Time

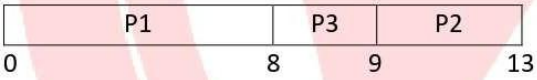
P1 = 8-0 = 8 ms
P2 = 12-1 = 11 ms
P3 = 13-2 = 11 ms

Average turnaround time = Turnaround time of all processes / Number of processes
= (8 + 11 + 11) / 3
= 10 ms / units

ii) using SJF (non-preemptive):

Process	Arrival time	Burst time	Completion time	Turnaround time
P1	0	8	8	8
P2	1	4	13	12
P3	2	1	9	7

Gantt chart:



Completion Time:

completion time is calculated by using Gantt chart

Turnaround Time:

Turnaround time = completion time - Arrival time
P1 = 8-0 = 8 ms
P2 = 13-1 = 12 ms
P3 = 9-2 = 7 ms

Average turnaround time = Turnaround time of all processes / Number of processes
= (8 + 12 + 7) / 3
= 9 ms / units

iii) using Round Robin: consider time Quantum = 2 units

Process	Arrival time	Burst time	Completion time	Turnaround time
P1	0	8	13	13
P2	1	4	9	8
P3	2	1	5	3

Gantt chart:

P1	P2	P3	P1	P2	P1	P1
0	2	4	5	7	9	11	13

Completion Time:
completion time is calculated by using Gantt chart.

Turnaround Time:
Turnaround Time = Completion Time - Arrival Time
P1 = 13 - 0 = 13 ms
P2 = 11 - 1 = 10 ms
P3 = 7 - 2 = 5 ms

Average Turnaround Time = Turnaround time of all processes / Number of processes
= (13 + 8 + 3) / 3
= 8 ms / units

- **Considering time quantum as 3 units:**

Process	Arrival time	Burst time	Completion time	Turnaround time
P1	0	8	13	13
P2	1	4	11	10
P3	2	1	7	5

Gantt chart:

P1	P2	P3	P1	P2	P1
0	3	6	7	10	11	13

Completion time:

Completion time is calculated by using gantt chart.

Turnaround time:

Turnaround time = completion time - Arrival time

P1 = 13 - 0 = 13 ms

P2 = 11 - 1 = 10 ms

P3 = 7 - 2 = 5 ms

Average Turnaround time = Turnaround time of all processes / Number of processes

= (13 + 10 + 5) / 3

= 9.33 ms / unit

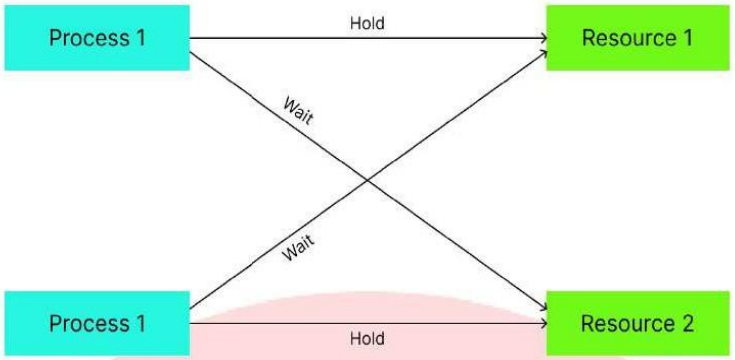
➢ 3.4 Deadlock:

Deadlock is a situation when two or more processes get locked and cannot processed further because of inter-dependability. Deadlock is defined as a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process. Processes involved in a deadlock remain blocked permanently and this affect OS performance. A dead deadlock arises when the four conditions hold true simultaneously in a system.

These four conditions are as follows:

i) Mutual Exclusion
ii) Hold and Wait
iii) No Pre-emption
iv) Circular Wait

Example of Deadlock :



①Process 1 is holding Resource 1 and is waiting for Resource 2.

② Process 2 is holding Resource 2 and is waiting for Resource 1.

③ Neither process can proceed because each one is waiting for a resource that the other holds.

Following are the conditions for deadlock:

1. **Mutual exclusion:**

At least one resource must be held in a non-sharable mode. That is, only one process at a time can use the resource.

2. **Hold and wait:**

A process must be holding at least one resource and waiting to acquire additional resources that are currently being held by other processes.

3. **No pre-emption:**

Resource cannot be pre-empted, i.e. resource can only be released voluntarily by the process holding it, after the process has completed its task. Resources previously granted cannot be forcibly taken away from a process.

4. **Circular wait:**

A set of waiting processes must exist such that P_o is waiting for a resource held by P_1, P_1 is waiting for a resource held by P_2, ..., Pn-1 is waiting for a resource held by Pn and Pn is waiting for a resource held by P_o. Each process is waiting for the resource held by other waiting processes in circular form.

Following are the methods of deadlock prevention :

1. Eliminate Mutual Exclusion:

The mutual exclusion condition must hold for non-sharable type of resources. Shareable resources do not require mutually exclusive access, and thus cannot be involved in deadlock. For example, a file shared on

network of computers, with read-only access can be open by multiple users at a time. It is not possible to prevent deadlock by denying the mutual exclusion condition.

2. Eliminate Hold and Wait:
One way to avoid this Hold and wait is when a process requests a resource it does not hold any other resources. One protocol that can be used requires each process to request and be allocated all its resources before it begin execution. Another protocol that can be used is, to allow a process to request resources only when the process has none. A process may request some resources and use them. Before it requests any additional resources, it must release all the resources that are currently allocated to it.

3. Eliminate No Preemption:
If a process that is holding some resources requests another resource that cannot be immediately allocated to it, then all resources currently being held are preempted. That is, this resources are implicitly released. The preempted resources are added to the list of resources for which the process is waiting. Process will be restarted only when all the resources, i.e. its old resources, as well as the new ones that it is requesting will be available.

4. Eliminating Circular Wait condition:
The circular wait condition of deadlock can be eliminated by assigning a priority number to each available resource and a process can request resources only in increasing order. If the priority number of a requested resource is greater than that of all the currently held resources, the request is granted. If the priority number of a requested resource is less than that of all the currently held resources, all the resources with greater priority number must be released first, before acquiring the new resource.

Number	Resource Name
0	Tape Drive
1	Printer
2	Plotter
3	Card Reader
4	Card Punch

System Models :

Resources and Processes: A system has a limited number of resources (like CPU, memory, printers, files) which are divided into types, each with multiple identical instances. Various processes compete for these resources.

Resource Requests and Release: Processes must request a resource before using it and release it after use. If a requested resource is not immediately available, the process must wait.

Resource Limitations: The total number of resources requested by processes cannot exceed the total number available in the system.

Resource Management Cycle: The typical sequence for a process using a resource is:

- **Request:** The process asks for a resource.
- **Use:** If granted, the process operates on the resource.
- **Release:** The process relinquishes the resource.

System Calls and Management: Resource requests and releases are handled through system calls provided by the operating system (e.g., request(), release(), open(), close(), allocate(), free()).

Synchronization Mechanisms: Resource management, particularly for resources not directly managed by the OS, can be achieved using synchronization mechanisms like semaphores (through wait() and signal() operations) or mutex locks.

Operating System's Role: The operating system is responsible for verifying that a process has indeed requested and been allocated a resource before allowing it to use it.

System Table for Resource Tracking: The OS maintains a system table that records the status of each resource:

- Whether the resource is free or allocated.
- If allocated, it also records which process the resource is allocated to.

Handling Unavailable Resources: If a process requests a resource that is currently allocated to another process, the requesting process is added to a queue of processes waiting for that particular resource.

Deadlock Handling :

Deadlock avoidance methods :

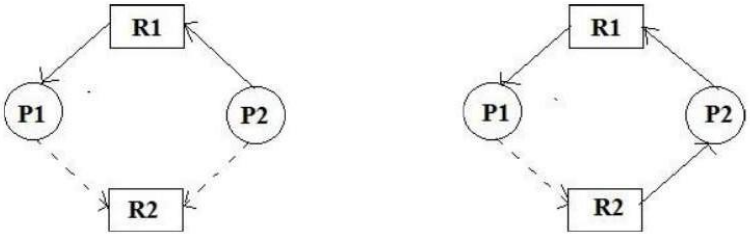
1. Safe State :

If a system is already in a safe state, we can try to stay away from an unsafe state and avoid deadlock. Deadlocks cannot be avoided in an unsafe state. A safe sequence of processes and allocation of resources ensures a safe state. Deadlock avoidance algorithms try not to allocate resources to a process if it will make the system in an unsafe state.

2. Resource Allocation Graph :-

A resource allocation graph is generally used to avoid deadlocks. If there are no cycles in the resource allocation graph, then there are no deadlocks. If there are cycles, there may be a deadlock. The resource allocation graph has request edges and assignment edges. Based on calm edges we can see if these is a chance for a cycle and then grant requests if the system will again be in a safe state.

Example :



If R2 is allocated to P2 and if P1 is request for R2, there will be a deadlock.

3. Banker's Algorithm :

In this algorithm, every process must tell upfront the maximum resource of each type it need. The Banker's algorithm can be divided into two parts: safety algorithm if a system is in a safe state or not. The resource request algorithm make an assumption of allocation and see if the system will be in a safe state. If the new state is unsafe, the resources are not allocated and the data structures are restored to their previous state. In this case the processes must wait for the resource.

Example :

5 Processes P_o through P_4 :

3 Resources types: A (10 Instances), B (5 Instances) and C (7 Instances)

Process	Allocation	Max	Available	Remaining
	A B C	A B C	A B C	A B C
P_o	0 1 0	7 5 3	3 3 2	7 4 3
P_1	2 0 0	3 2 2	5 3 2	1 2 2
P_2	3 0 2	9 0 2	7 4 3	6 0 0
P_3	2 1 1	2 2 2	7 4 5	0 1 1
P_4	0 0 2	4 3 3	10 4 7	4 3 1
			10 5 7	

The system in a safe state since the sequence $<P_1, P_3, P_4, P_2, P_o>$ satisfies safety criteria.

Deadlock avoidance- Banker's Algorithm

- The name was chosen because the algorithm could be used in a banking system to ensure that the bank never allocated its available cash in such a way that it could no longer satisfy the needs of all its customers. When a new process enters the system, it must declare the maximum number of instances of each resource type that it may need.
- This number may not exceed the total number of resources in the system.
- When a user requests a set of resources, the system must determine whether the allocation of these resources will leave the system in a safe state.

- If it will, the resources are allocated; otherwise, the process must wait until some other process releases enough resources.
- We need the following data structures, where n is the number of
- processes in the system and m is the number of resource types:

1. **Available:** A vector showing how many instance of each resource are currently free.
    a. Example: Available = [10 Printers, 5 CPU's, 7 RAM Units]
2. **Max:** An n x m matrix defines the maximum demand of each process. If Max[i] [j] equals k, then process P_i may request at most k instances of resource type R_i.
    a. Example: Max[Process 1] = [5 Printers, 2 CPU's, 4 RAM Units]
3. **Allocation:** An n x m matrix defines the number of resources of each type currently allocated to each process. This is declared in advance.
    a. Example: Allocation[Process 1] = [2 Printers, 1 CPU, 1 RAM]
4. **Need:** An n x m matrix indicates the remaining resource need of each process.
5. Note that Need = Max - Allocation.
    a. Example: Need[Process 1] = [3 Printers, 1 CPU, 3 RAM]

Safety Algorithm

We can now present the algorithm for finding out whether or not a system is in a safe state. This algorithm can be described as follows:

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

Initialize Work= Available and Finish[$_i$] =false for i = 0, 1, ... , n - 1.

Example :

WORK				Process	FINISH
A	B	C	D	0	F
				1	F
				2	F
				3	F
				4	F

2. Find an index i such that both
    a. Finish[$_i$] ==false
    b. Need$_i$ <= Work

If no such i exists, go to step 4.

3. Work = Work + Allocation;

Finish[i] = true

Go to step 2.

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

Following are the steps of Banker's algorithm to avoid deadlock.

Step 1: When a process requests for a resource, the OS allocates it on a trial basis.

Step 2: After trial allocation, the os updates all the matrices and vectors. This updating can be done by the OS in a separate work area in the memory.

Step 3: It compares free resources Vector with each row of matrix B on a vector to vector basis.

Step 4: If free resources is smaller than each of the row in Matrix B i.e. even if all free resources are allocated to any process in Matrix B and not a single process can complete its task then as concludes that the System is in unstable State.

Step 5: If free resources is greater than any row for a process in Matrix B the os allocates all required resources for that process on a trial basis. It assumes that after completion of process, it will release all the resources allocated to it. These resources can be added to the free vector.

Step 6: After execution of a process, it removes the row indicating executed process from both matrices.

Step 7: This algorithm will repeat the procedure Step 3 for each process from the matrices and finds that all processes can complete execution without entering unsafe state. For each request for any resource by a process os goes through all these trials of imaginary allocation and updates on After this if the system remains in the safe state, and then changes can be made in actual matrices.

Example:

Find out the safe sequence of processes using banker's algorithm for following problem

Process	Allocation	Max Need	Available	Remaining Need
	A B C	A B C	A B C	A B C
P0	0 0 1	2 2 2	4 2 2	2 2 1
P1	0 1 0	3 3 3	4 2 3	3 2 3
P2	1 0 0	1 1 1	4 3 3	0 1 1
P3	0 0 2	2 2 2	5 3 3	2 2 0
P4	0 0 0	3 3 3	0 3 5	3 1 3
			5 5 5	

A, B, C are representing resources.

Total Resources: A = 5

B = 5

C = 5

so, the safe sequence is

P0 -> P1 -> P2 -> P3 -> P4

Unit - IV Memory Management

➢ 4.1 Basic Memory Management

Memory management in operating system is important because of it directly affects the execution time of a process. The execution time of a process depends on the availability of data in the main memory. Therefore, an operating system must perform the memory management in such a manner that the essential data is always present in the main memory. An effective memory management system ensures accuracy, availability and consistency of the data imported from the secondary memory to the main memory.

Advantages :

- Memory management provide protection for memory.
- Increase the performance of the system

Disadvantages :

- Memory management is complex
- It may cause high overhead
- Memory management is dependent on hardware.

• Memory Partitioning

Explain partitioning and its types. (W – 19, S - 23)

In memory partitioning, memory is divided into a number of regions or partitions. These partitions can be of different size or same size. Each region may have one program to be executed. When a region is free, a program is selected from the job queue and loaded into the free region. Memory is divided into two sections, one for user and one for resident monitor of the operating system.

Types of partitioning:

i) fixed partitioning i.e. static partitioning

ii) variable partitioning i.e. dynamic partitioning.

1) Fixed Partitioning: (Static partitioning)

Explain fixed size memory partitioning. (W - 22)

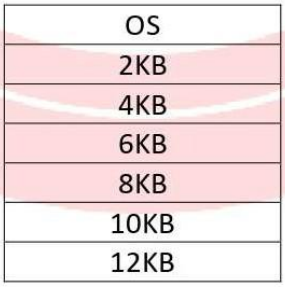
Main memory is divided into multiple partitions of fixed size at the time of System generation. A process may be loaded into a partition of equal size or greater size. Partition can be of equal size or unequal size. There are two alternatives for fixed partitioning.

a) Equal size partitioning
b) Unequal size partitioning

a) Equal size partitioning:

Main memory is divided into equal size partitions. Any process with less or equal size can be loaded in any available partition.

b) Unequal size partitioning:



Main memory is divided into multiple partitions of unequal size. Each process can be loaded into the smallest partition within which the process will fit.

Advantages of Static Memory Partitioning:

1. Simple to implement.
2. It requires minimal operating system software and processing overhead.
3. Partitions are fixed at the time of system generation.

Disadvantages of Static Memory partitioning

1. Memory wastage
2. Inefficient use of memory due to internal fragmentation.

2) Variable Partitioning: (Dynamic Partitioning)

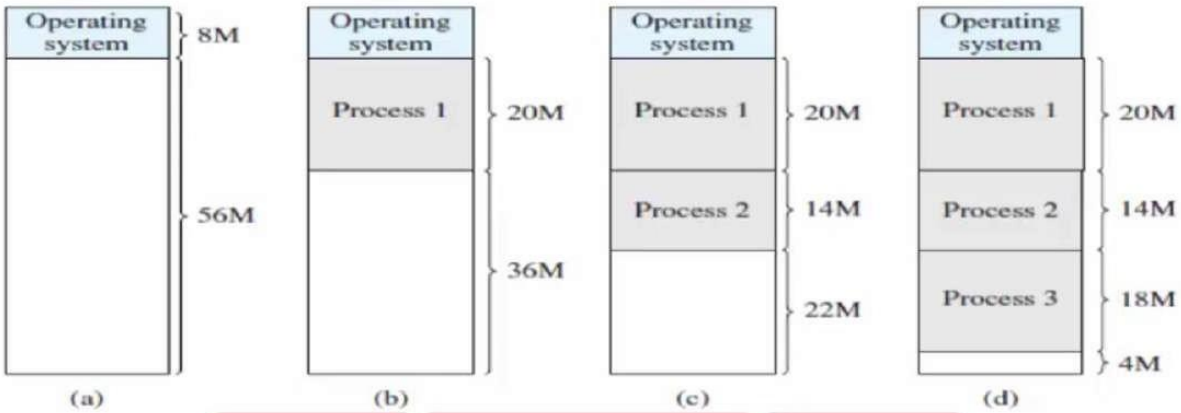
With suitable diagram, describe the concept of variable partitioning of memory. (S – 22, S - 24)

When a process enters in main memory, it is allocated exact size that is required by that process. So in this method, partitions can vary in size depending on memory space required by a process entering in main memory. Operating System maintains a table indicating which parts of memory are available and which are occupied. When new process arrives and it needs space, System searches for available memory space in main memory. If it is available, then memory is allocated to the process by creating a partition in memory.

For example:

Consider following table with process and memory space.

Process	Memory space
P1	20M
P2	14M
P3	18M



Dynamic Storage Allocation:

The set of holes is searched to determine which hole is best to allocate. The most common strategies used to select a free hole from the set of available holes are first fit, best fit and worst fit.

Types of the Dynamic Storage Allocation

1. First Fit
2. Best Fit
3. Worst Fit

- **First Fit:** Allocate the first hole (or free space) that is big enough for the new process.
- **Best Fit:** Allocate the smallest hole that is big enough. We search the entire list unless the list is kept ordered by size. This strategy produces the smallest left over hole.
- **Worst Fit:** Allocate the largest hole. We must search the entire list, unless it is sorted by size.

Advantages of Dynamic Memory Partitioning:

1. No internal fragmentation.
2. More efficient use of main memory.

Disadvantages of Dynamic Memory partitioning:

1. It suffers from external fragmentation.
2. It needs compaction.

While hole is taken for next segment request for 8KB in a swapping system for First Fit, Best Fit, and Worst Fit. (W - 23)

OS
4 KB
9 KB
20 KB
16 KB
8 KB
2 KB
6 KB

First Fit :

Allocate the first free block to the new process.

OS
4 KB
< FREE > 1 KB 8 KB
20 KB
16 KB
8 KB
2 KB
6 KB

Best Fit :

Allocate the smallest free block that is big enough to accommodate new process.

OS
4 KB
9 KB
20 KB
16 KB
< FREE> 0 KB 8 KB

2 KB
6 KB

Worst Fit :

Allocate the largest free block to the new process.

OS
4 KB
9 KB
< FREE > 12 KB
8 KB
16 KB
8 KB
2 KB
6 KB

Consider the following memory map and assume a new process P4 comes with memory requirements of 6 KB

Locate (Draw) this process in memory using

i. First Fit

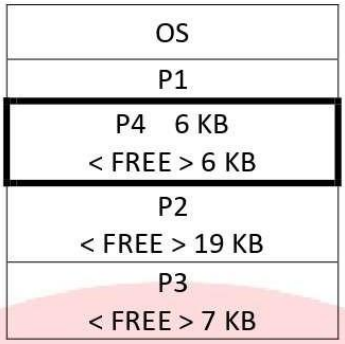
ii. Best Fit

iii. Worst Fit (W - 22)

OS
P1
< FREE > 12 KB
P2
< FREE > 19 KB
P3
< FREE > 7 KB

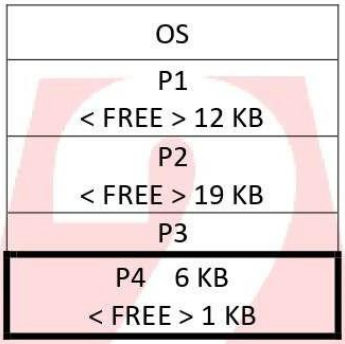
First Fit :

Allocate the first free block to the new process.



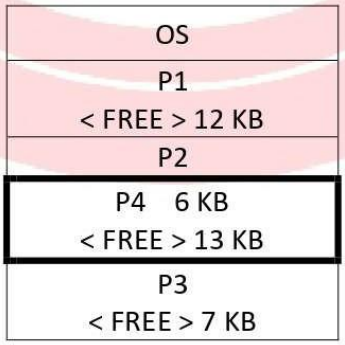
Best Fit :

Allocate the smallest free block that is big enough to accommodate new process.



Worst Fit :

Allocate the largest free block to the new process.



1. Free space management :

List free space management techniques? Describe any one in detail. (W – 19, S – 22, W – 22, W – 23, S - 24)

The process of looking at free and managing the free blocks of the disk is called free space management. Files are created and deleted frequently during the operation of a computer system. Since there is only a limited amount of disk space, it is necessary to reuse the space from deleted files for new files. To keep track of free disk space, the file system maintains a free space list. The free space list records all disk blocks, which are free. To create a file, we search the free space list for the required amount of space on and allocate it to the new file. This space is then removed from the free space list. When a file is deleted, its disk space is added to the free space list.

Following are the techniques of free space management:

1. Bit vector
2. Linked list
3. Grouping
4. Counting

1. Bit Vector :

Describe concept of free space management technique using bit map method (W - 23)

The free space list is implemented as a bit map or bit vector. Each block is represented by 1 bit. If the block is free, the bit is 0, if the block is allocated, the bit is 1.

Example :

Consider a disk where blocks 2, 3, 4, 5, 8, 9, 10, 11, 12, 13 are free and the rest of the blocks are allocated.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	0	0	0	1	1	0	0	0	0	0	0	1	1

The free space bit map would be: 1100001100000011

0 = Free Block.
1 = Allocated Block.

Advantages of Bit map / Bit vector :

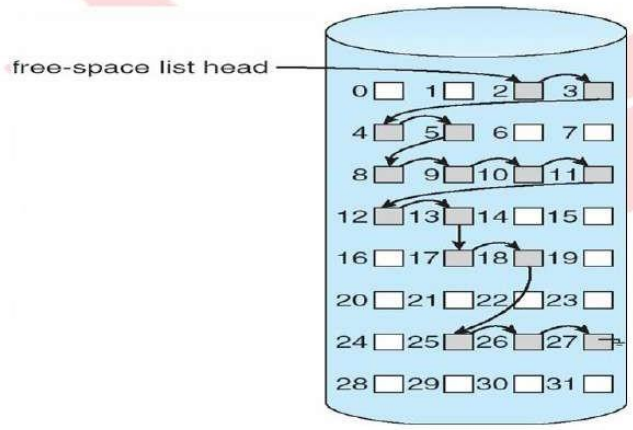
1. It is simple and efficient method to find the first free block or n-consecutive free block on the disk.
2. A bit map requires lesser space as it uses 1 bit per block.

Disadvantages of Bit map / Bit vector :

1. Extra Space required to store bit map.
2. It may require a special hardware support to do bit operation i.e. finding out which bit is 1 and which bit is 0.

2. Linked List :

Linked list is another technique for free space management in which, the free disk blocks are linked together i.e. a free block contains a pointer to the next free block. In linked list technique, link all the disk blocks together, keeping a pointer to the first free block. This block contains a pointer to the next free disk block, and so on.



In this example we would keep a pointer to block 2, as the first free block. Block 2 would contain a pointer to block 3, which point to block 4, which would point to block 5 and so on.

Advantages of Linked Free Space List :

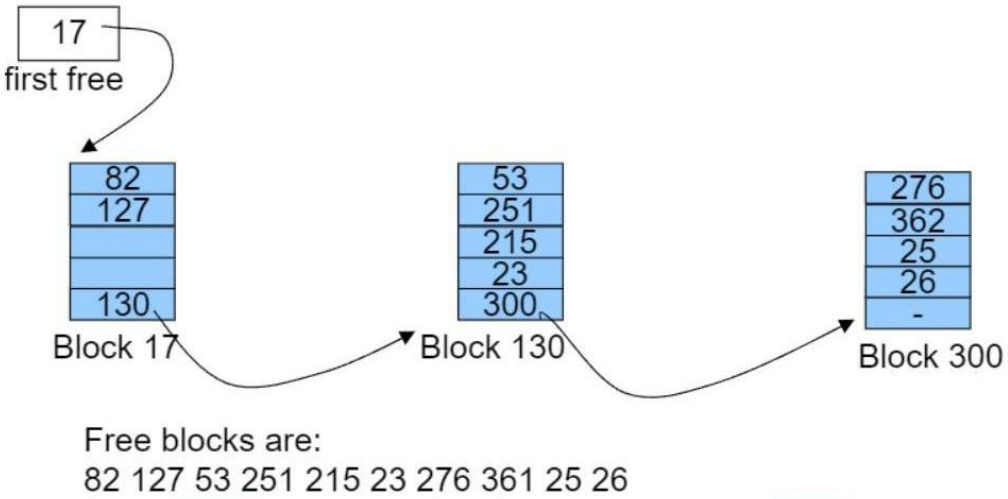
1. If a file is to be allocated a free block, the operating system can simply allocate the first block in the free space list and move the head pointer to the next free block in the list.
2. No disk fragmentation as every block is utilized.

Disadvantages of Linked Free Space Lists :

1. It is not very efficient scheme as to traverse the list.
2. Pointers use here will also consume disk space. Additional memory is therefore required

3. Grouping

Grouping is a free space management technique. This approach stores the address of the free blocks in the first free block. The first free block stores the address of some, say n free blocks. Out of these n blocks, the first n-1 blocks are actually free and the last block contains the address of next free n blocks. Shows an example of grouping free space management technique, in which a disk block contains addresses of many free blocks and a block containing free block pointers will get freed when those blocks are used.



Advantages :

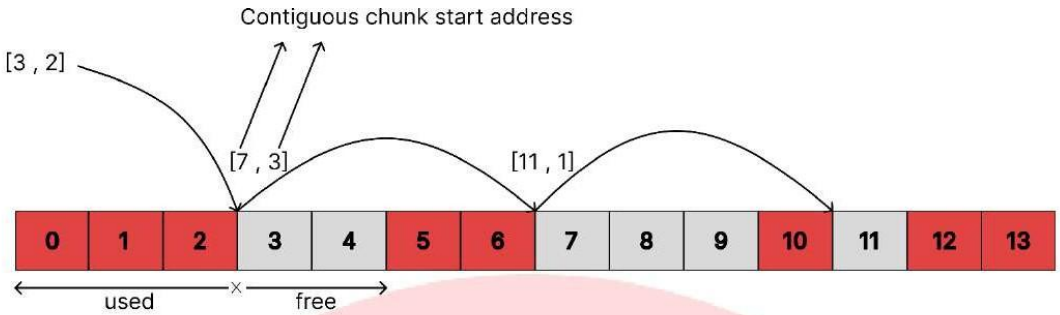
1. Finding free blocks in massive amount can be done easily using this method.
2. Reduce redundancy.

Disadvantages :

1. We need to alter the entire list, if any of the block of the list is occupied.
2. It increases the complexity.

4. Counting

Counting is also a technique for free space management. Generally several contiguous blocks may be allocated or freed simultaneously, particularly when contiguous allocation is used. Thus rather than keeping a list of 'n' free disk addresses, we keep the address of first free block and number 'n' of free contiguous blocks. Each entry in the free space list then consists of a disk address and a count. Although each entry requires more space than would a simple disk address, the overall list will be shorter as long as the count is generally greater than 1. Below fig shows an example of Conti Counting Free Space management technique. It besides the free block Pointer, keep a counter saying how many block are free contiguously after that free block.



Advantages :

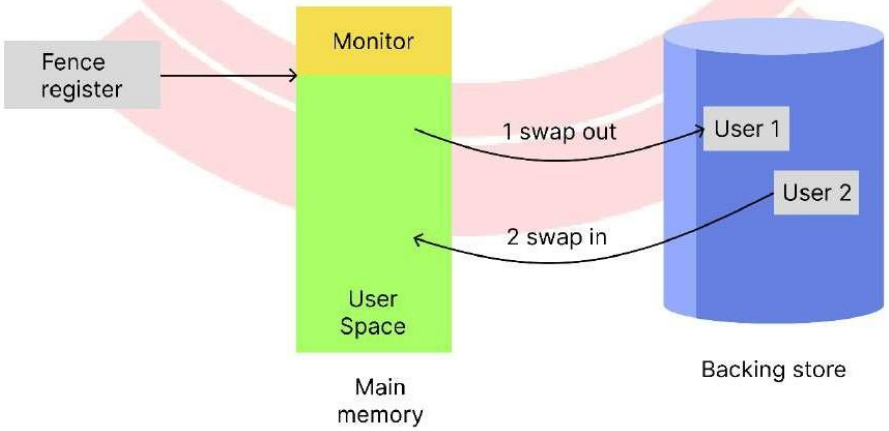
1. It provides efficient utilization of space
2. It allows fast allocation and deallocation
3. It reduces fragmentation

Disadvantages :

1. Only contiguous blocks are required
2. It may increase the overhead

➢ 4.2 Swapping

Explain the following terms with respect to memory management:
i) Dynamic relocation
ii) Swapping (S - 24)



Swapping is mechanism in which a process can be swapped temporarily out of main memory (or move) to secondary storage (disk) and make that memory available to other processes. At some late time, the system swaps back the process from the secondary storage to main memory. For eg. In a multiprogramming environment when number of processes is queued for execution and if the system is implemented with a round robin algorithm then when time quantum / time slice expires then the process will be swapped out and the new process will be swapped into the memory space that just has been freed.

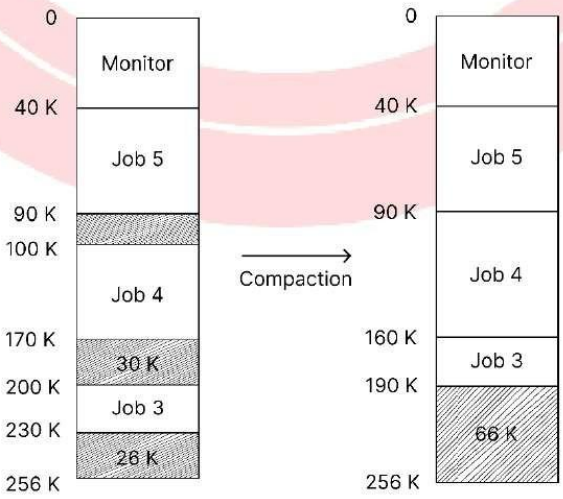
2. Compaction :

Compaction is a method used to overcome the external fragmentation problem. All free blocks are brought together as one large block of free space. The collection of free space from multiple non-contiguous blocks into one large free block in a system's memory is called compaction.

The goal compaction is to shuffle the memory contents to place all free memory together in one large block. For example, the memory map of (e) can be compacted as shown in below Fig. The three holes of sizes 10K, 30K and 26K can be compacted into one hole of 66K. Compaction is not always possible.

In below Fig. we moved job4 and job3 for these programs to be able to work in their new locations, all internal addresses must be relocated. Compaction is possible only if relocation is dynamic, at execution time, using base and limit registers. The simplest compaction algorithm is to simply move all jobs towards one end of memory, all holes move in the other direction, producing an large hole of available memory.

Compaction can be quite expensive. Compaction changes the allocation of memory to make free space contiguous and hence useful. Compaction also consumes system resources.



Advantages:

1. **Eliminates External Fragmentation:** Consolidates free space, making it easier to allocate larger processes.
2. **Efficient Memory Usage:** Maximizes memory utilization by creating a single large free block.
3. **Improved System Performance:** Reduces allocation failures from fragmented space.
4. **Allows Large Memory Allocation:** Enables large programs to run even with scattered free memory.

Disadvantages:

1. **High CPU Overhead:** Requires significant CPU time for relocating processes and updating references.
2. **System Downtime:** Can cause other tasks to wait, leading to temporary system inefficiency.
3. **Complexity:** Demands careful management of relocation information to maintain data integrity and avoid errors.

3. Fragmentation :

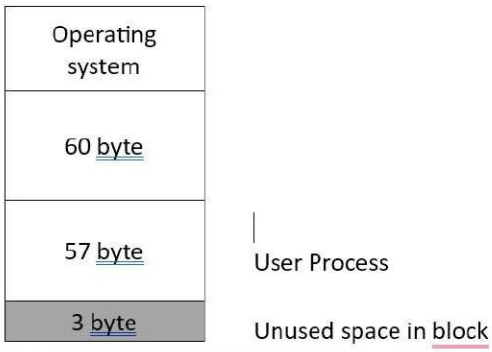
Define the term fragmentation in terms of memory. (S – 22, S - 23)

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

Types of Fragmentation :-

1. Internal Fragmentation
2. External Fragmentation.

1. Internal Fragmentation



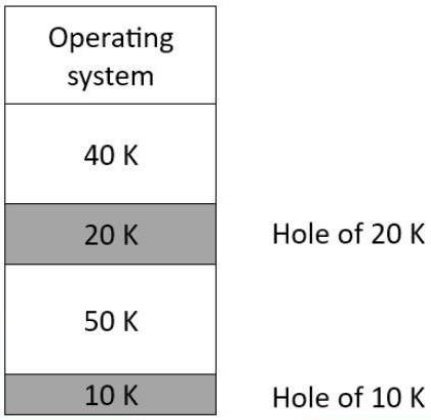
1. Wasting of memory within a partition, due to a difference in size of a partition and of the object resident within it, is called internal fragmentation.
2. Internal fragmentation occurs when the memory allocator leaves extra space empty inside block of memory that has been allocated for a client.

Example :

1. A client that needs 57 bytes of memory is given 60 bytes
2. The extra bytes that the client doesn't need go to waste,

2. External Fragmentation :

Wasting of memory between partitions, due to scattering of the free space into a number of discontinuous areas, is called external fragmentation. External fragmentation exists when enough total memory space exists to satisfy a request, but it is not contiguous, storage is fragmented into large number of small holes.



For Example :-

There is a hole of 20k and 10k is available in multiple partition allocation schemes. The next process request for 30k of memory. Actually 30k of memory is free which satisfy the request but hole is not contiguous. So there is an external fragmentation of memory.

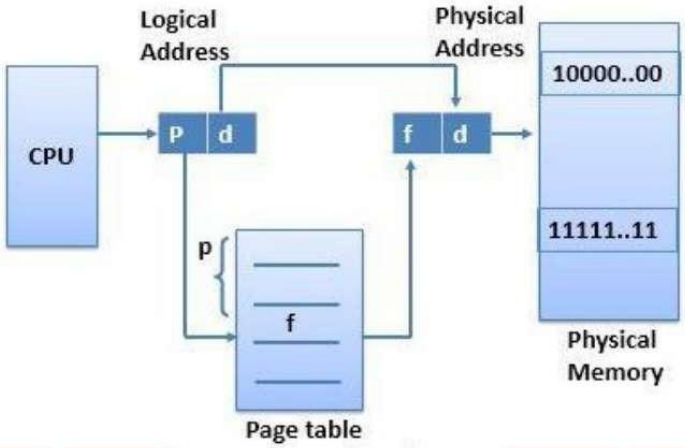
➢ 4.3 Non-contiguous Memory Management Techniques:

Define paging and segmentation (W - 23)

- **Paging:**

  Define paging. (S - 23)

Paging is a memory management technique by which a computer stores and retrieves data from secondary storage to main memory. In paging, the operating system retrieves data from secondary storage in same-size blocks called pages. The main idea behind the paging is to divide each process in the form of pages. The main memory will also be divided in the form of frames. One page of the process is to be stored in one of the frames of the memory. Pages of the process are brought into the main memory only when they are required otherwise they reside in the secondary storage. The sizes of each frame must be equal. and page must be equal. considering the fact that Pages are mapped to the frames in Paging.



Advantages of Page Memory Allocation:

1. Allow jobs to be allocated in non-contiguous memory locations.
2. Paging eliminates fragmentation
3. Memory used more efficiently.
4. Support higher degree of multiprogramming.

Disadvantages of Paged Memory Allocation:

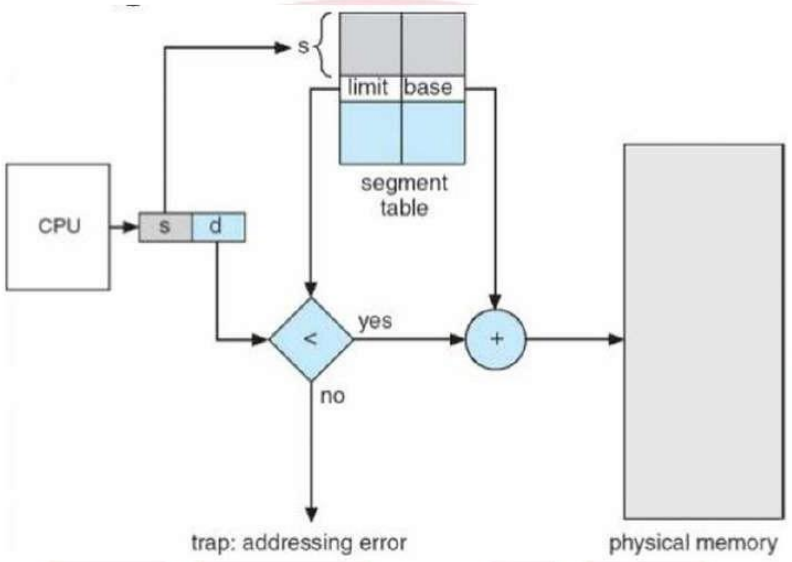
1. Page Address mapping hardware usually increases the cost of the computer.
2. Internal fragmentation still exists, though in last Page.
3. Requires the entire job to be stored in memory location
4. Size of page is crucial (not too small, not too large).
5. Memory must be used to store the various tables like page table, memory map table etc.

- **Segmentation:**

In Operating System, Segmentation is a memory management technique in which the memory is divided into the variable size parts. Each part is known as a segment which can be allocated to a process. The detail about each segment are stored in a table called a

segment table. Segment table is stored in one (or many) of the segments. Segment table contains mainly two information about segment:

1. **Base:** It is the base address of the segment.
2. **Limit:** It is the length of the segment.



Working of Segmentation:

Each entry of segment table has a segment base and a segment limit. The offset of the logical address must be between 0 and the segment limit. If it is not, we trap to the operating system. If this offset is legal, it is added to the segment base to produce the address in physical memory of the desired word.

Advantages of Segmentation:

1. Segmentation can eliminate fragmentation.
2. It provides a virtual memory.
3. It supports Dynamic linking and loading.
4. It provides a convenient way of organizing programs and data to the programmer.

Disadvantages of Segmentation:

1. It suffers from external fragmentation.

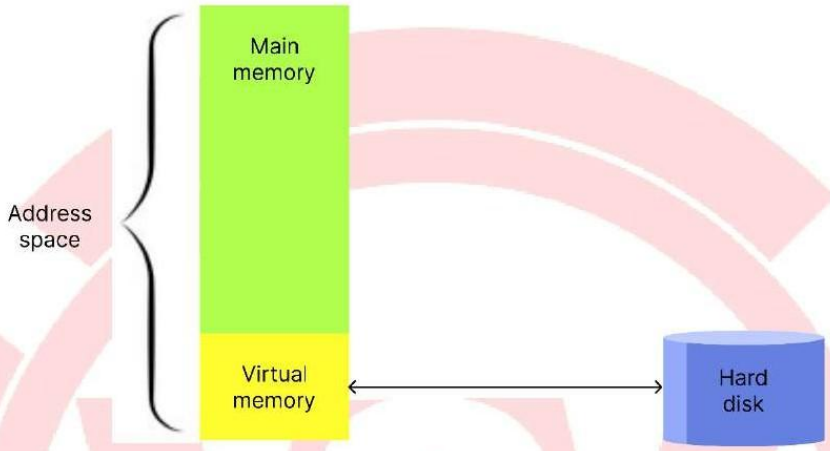
2. Conversion from logical address to physical address is not a simple function.
3. It increase complexity of operating system.
4. It increase hardware cost.

Difference between paging and segmentation (W – 22, S - 23)

Parameter	Paging	Segmentation
1) Individual Memory	In paging, we break a process address space into blocks known as pages.	In the case of segmentation, we break a process address space into blocks known as sections/segments.
2) Memory Size	The pages are blocks of fixed size.	The sections/segments are blocks of varying sizes.
3) Speed	This technique is comparatively much faster in accessing memory.	This technique is comparatively much slower in accessing memory than paging.
4) Size	The available memory determines the individual page sizes.	The user determines the individual segment sizes.
5) Fragmentation	The paging technique may underutilize some of the pages, thus leading to internal fragmentation blocks at all.	The segmentation technique may not use some internal memory blocks at all. Thus, it may lead to external fragmentation.
6) Logical Address	A logical address divides into page offset and page number in the case of paging.	A logical address divides into section offset and section number in the case of segmentation.
7) Data Storage	In the case of paging, the page table leads to the storage of the page data.	In the case of segmentation, the segment table leads to the storage of the segmentation data.

➢ 4.4 Virtual Memory :

Define virtual memory (W – 19, S - 23)



Virtual memory is a memory management technique that provides an idealized abstraction of the storage resources that are actually available on a given machine which creates the illusion to users of a very large (main) memory. Virtual memory is the separation of user logical memory from physical memory. This separation allows an extremely large virtual memory to be provided for programmers when only a small physical memory is available. Virtual memory makes the task of programming much easier, because the programmer no longer needs to worry about the amount of physical memory available for execution of program. It is the process of increasing the apparent size of a computer's RAM by using a section of the hard disk storage as an extension of RAM.

Example:

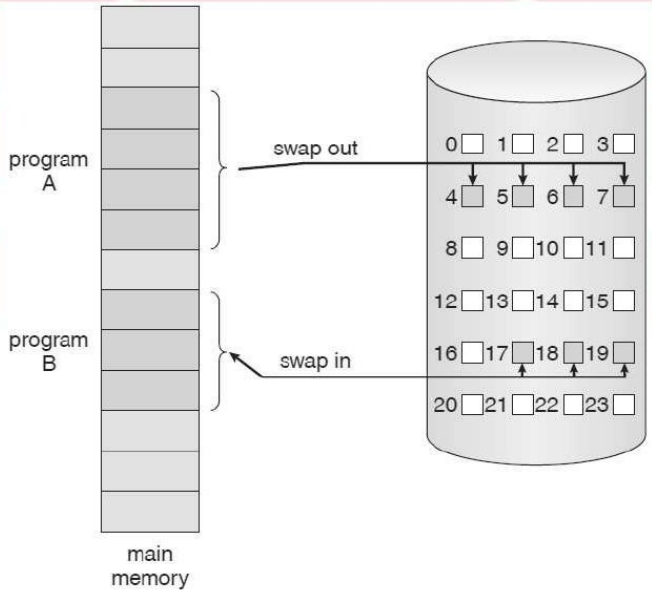
Consider, an e-mail program, a web browser and a word processor is loaded into RAM simultaneously; the 64 MB space is not enough to store all these programs. Without a virtual memory a message "You cannot load any more applications. Please close an application to load a new one." would be displayed. By using virtual memory a computer can identify data in RAM that is not actively being used by running programs and temporarily move this data to the hard disk. This process frees up RAM space for other active applications and processes.

Advantages :

1. Allow the programmer to use more memory than actual physical memory in a system.

2. Allow the swapping of processes to and from physical memory.
3. It enables multitasking.
4. Helps in keeping the system stable i.e. avoid system crash.

- **Demand paging**



A demand - Paging system is similar to a Paging system with swapping. Where processes reside in secondary memory (usually a disk). When we want to execute a process, we swap it into memory. Rather than swapping the entire process into memory, however, we use a lazy swapper called pager. A lazy swapper never swaps a page into memory unless that page will be needed. When a process is to be swapped in the pager guesses which pages will be used before the process is swapped out again. Instead of swapping in a whole process, the pager brings only those necessary pages into memory.

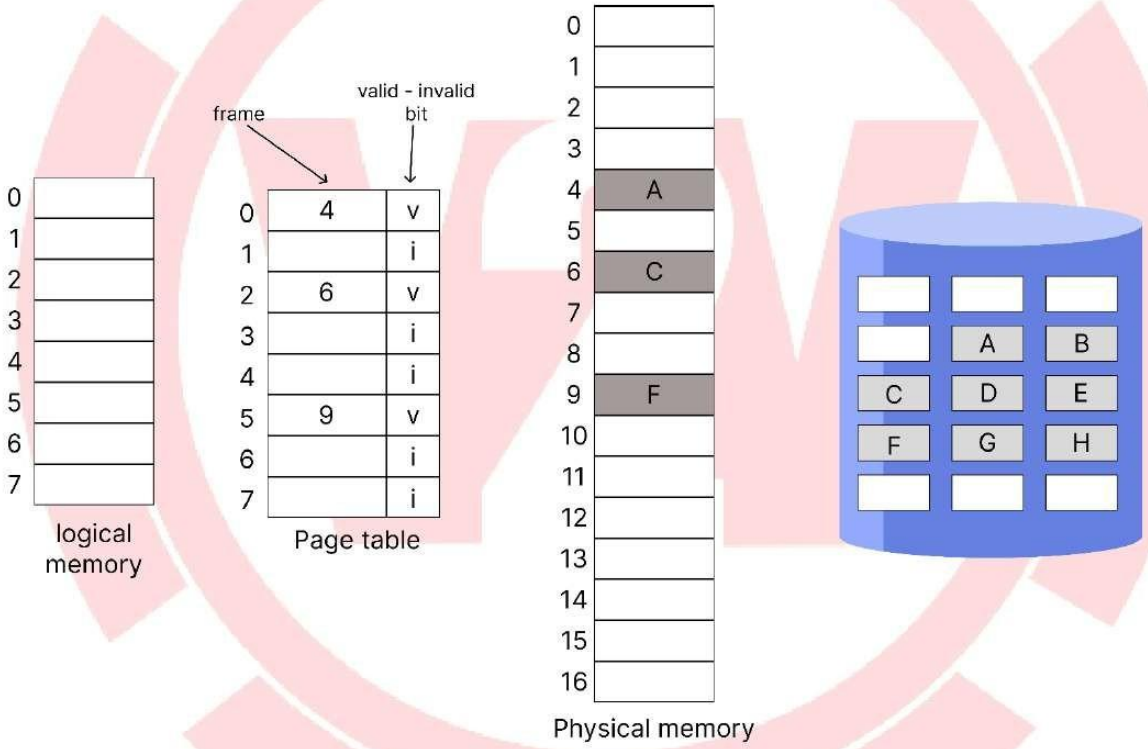
Advantages of Demand Paging :

1. Large virtual memory.
2. more efficient use of memory.
3. There is no limit on degree of multiprogramming.
4. It saves memory.

Disadvantages Demand Paging :

1. Accessing data not in memory causes delays.
2. It is complex to implement.

- **Page Fault**



A page fault occurs when a program accesses a page that has been mapped in address space, but has not been loaded in the physical memory. When the page (data) requested by a program is not available in the memory, it is called as a page fault. It is possible that not all pages of the program were brought into memory. Some pages are loaded into memory and some pages are kept on the disk. To distinguish between the pages that are in memory and the pages that are on the disk, a valid-invalid bit is provided. Pages that are not loaded into memory are marked as invalid in the page table, using the invalid bit. The bit is set to valid if the associated page is in memory. Access to a page marked invalid causes a page fault.

➢ 4.5 Page Replacement Algorithm

- **First In First Out (FIFO)**

The simplest page replacement algorithm is a FIFO. A FIFO replacement algorithm associates with each page the time when that page was brought into memory. When a page must be replaced, the oldest page is chosen. FIFO queue is created to hold all pages in memory. We replace the page at the head of the queue. When a page is brought into memory, we insert it at the tail of the queue.

For example, consider the following reference string: problem of FIFO Algorithm: consider the following reference string :

1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

Reference string Frames	1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1	4	4	4	5	5	5	5	5	5
2		2	2	2	1	1	1	1	1	3	3	3
3			3	3	3	2	2	2	2	2	4	4
Page fault	F	F	F	F	F	F	F			F	F	

Fig Page fault for using three frames.

Reference string Frames	1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1	1	1	1	5	5	5	5	4	4
2		2	2	2	2	2	2	1	1	1	1	5
3			3	3	3	3	3	3	2	2	2	2
4				4	4	4	4	4	4	3	3	3
Page fault	F	F	F	F			F	F	F	F	F	F

i) the number of faults for four frames (10) is greater than the number of faults for three frames (9).
ii) This most unexpected result is known as Belady's anomaly.
iii) For some page replacement algorithms, the page fault rate may increase as the number of allocated frames increases.

Advantages of FIFO Page Replacement Algorithm:

1. It is simple to implement.
2. It is easiest algorithm
3. Easy to understand and execute.

Disadvantages of FIFO Page Replacement Algorithm:

1. It is not very effective
2. System needs to track of each frame.
3. It suffers from Belady's anomaly.
4. Bad replacement choice increases the page fault rate and slow process execution.

- **Least Recently Used (LRU)**

  Explain LRU page replacement algorithm for following reference string. 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
  Calculate the page fault. (W - 19)

The Least Recently Used (LRU) page replacement algorithm replaces the page that has not been used for the longest period of time. LRU replacement associates with each page the time of that page's last use. When a page must be replaced, LRU chooses the page that has not been used for the longest period of time. An LRU page-replacement algorithm may require substantial hardware assistance.

Following are the different approaches to implement LRU replacement algorithm:

1. Using counters for LRU:

We can implement LRU using a counter for each page frame. Whenever a page is referenced, its counter is incremented with the current timestamp. When a page fault occurs, the page with the smallest counter value is replaced.

2. Using Stack for LRU :

Another approach to implementing LRU replacement is to keep a stack of page number. Whenever a page is referenced, it is removed from the stack and put on the top. In this way, that most recently used page is always at the top of the stack and the least recently used page is always at the bottom.

Example :

Reference String:

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

The below solved sum can also be solved same as like LRU table (Only the table structure will be same logic of solving is different) in some places sums are solved in this way also

LRU:

Assume Frame Size = 3

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2		2		4	4	4	0			1		1	*	1		*
	0	0	0	*	0	*	0	0	3	3	*		3		0		0	*	
		1	1		3		3	2	2	2		*	2	*	2		7		

Page Fault = 12

Assume frame size = 4

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	7		3		3		*		*		3				7		
	0	0	0	*	0	*	0			*			0		*		0	*	
		1	1		1		4						1			*	1		*
			2		2		2	*				*	2	*			2		

Page fault = 08

Advantages of LRU Page Replacement Algorithm :

- LRU is actually quite a good algorithm.
- It never suffers from Belady's anomaly.
- LRU algorithm is very feasible to implement.

Disadvantages of LRU Page Replacement Algorithm :

- LRU algorithm is that it requires additional data structure and hardware support.
- Its implementation is not very easy.

- **Optimal**

An optimal page replacement algorithm has the lowest page fault rate of all algorithms and would never suffer from Belady's anomaly. Optimal replacement algorithm states replace that page which will not be used for the longest period of time.

For example, Consider the following reference string.
7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

- The first three reference cause faults which fill the three empty frames.
- The reference to page 2 replaces page 7, because 7 will be used until reference 18, whereas page 0 will be used at 5 and page 1 at 14.
- The reference to page 3 replaces page 1, as page 1 will be the last of the three pages in memory to be reference referenced again
- with only nice Page fault, optimal replacement is much better than a FIFO algorithm, which had 15 faults.
- The optimal page replacement algorithm is difficult to implement because it requires future knowledge of the reference string.

Frame	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
F1	7	7	7	2	2	2	2	2	2	2	2	2	2	2	2	2	2	7	7	7
F2		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F3			1	1	1	3	3	4	4	3	3	3	3	1	1	1	1	1	1	1
Page fault	F	F	F	F		F		F		F				F				F		

Total Page Fault = 9
Total page hits = 11

Advantages of optimal page Replacement Algorithm

1. It give the smallest number of Page faults.
2. It never suffers from Belady's anomaly.
3. Twice as good as FIFO Page Replacement Algorithm.

Disadvantage of optimal page Replacement Algorithm:

1. This algorithm is difficult to implement.
2. It is only use as a theoretical part of page replacement.
3. It requires future knowledge of reference string.

Find out the total number of page faults using:

- **Least Recently Used page Replacement**
- **Optimal page Replacement**

Page Replacement algorithms of memory management, if the pages are coming in the order:

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1 (W - 23)

i) LRU

considering frame size is 3:

Ref	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
F1	7	7	7	2	2	2	2	4	4	4	0	0	0	1	1	1	1	1	1	1
F2		0	0	0	0	0	0	0	0	3	3	3	3	3	3	0	0	0	0	0
F3			1	1	1	3	3	3	2	2	2	2	2	2	2	2	2	7	7	7
Fault	F	F	F	F		F		F	F	F	F			F		F		F		

Total page faults using LRU page Replacement / Fault = 12

ii) optimal:

Considering Frame size is 3:

Ref	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
F1	7	7	7	2	2	2	2	2	2	2	2	2	2	2	2	2	2	7	7	7
F2		0	0	0	0	0	0	4	4	4	0	0	0	0	0	0	0	0	0	0
F3			1	1	1	3	3	3	3	3	3	3	3	1	1	1	1	1	1	1
Fault	F	F	F	F		F		F			F			F				F		

Total page faults using optimal page Replacement / page Fault = 09

Given a page reference string with three (3) page frames. Calculate the page fault with 'optimal' and 'LRU' page replacement algorithm respectively. (W - 22)

Page Reference String :
7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

i. Optimal Page Replacement Algorithm

REF	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
F1	7	7	7	2	2	2	2	2	2	2	2	2	2	2	2	2	2	7	7	7
F2		0	0	0	0	0	0	4	4	4	0	0	0	0	0	0	0	0	0	0
F3			1	1	1	3	3	3	3	3	3	3	3	1	1	1	1	1	1	1
Fault	F	F	F	F		F		F			F			F				F		

Total page Fault = 9

ii. LRU

REF	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
F1	7	7	7	2	2	2	2	4	4	4	0	0	0	1	1	1	1	1	1	1
F2		0	0	0	0	0	0	0	0	3	3	3	3	3	3	0	0	0	0	0
F3			1	1	1	3	3	3	2	2	2	2	2	2	2	2	2	7	7	7
Fault	F	F	F	F		F		F	F	F	F			F		F		F		

Total page Fault = 12

For the page reference string 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1 Calculate the page faults applying: i) optimal ii) LRU iii) FIFO page Replacement algorithms for a memory with three frames. (S - 23)

i) optimal

Ref	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
F1	7	7	7	2	2	2	2	2	2	2	2	2	2	2	2	2	2	7	7	7
F2		0	0	0	0	0	0	4	4	4	0	0	0	0	0	0	0	0	0	0
F3			1	1	1	3	3	3	3	3	3	3	3	1	1	1	1	1	1	1
Fault	F	F	F	F		F		F			F			F				F		

Total Page faults - 9

ii) LRU

Ref	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
F1	7	7	7	2	2	2	2	4	4	4	0	0	0	1	1	1	1	1	1	1
F2		0	0	0	0	0	0	0	0	3	3	3	3	3	3	0	0	0	0	0
F3			1	1	1	3	3	3	2	2	2	2	2	2	2	2	2	7	7	7
Fault	F	F	F	F		F		F	F	F	F			F		F		F		

Total page faults - 12

iii) FIFO

REF	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
F1	7	7	7	2	2	2	2	4	4	4	0	0	0	0	0	0	0	7	7	7
F2		0	0	0	0	3	3	3	2	2	2	2	2	1	1	1	1	1	0	0
F3			1	1	1	1	0	0	0	3	3	3	3	3	2	2	2	2	2	1
Fault	F	F	F	F		F	F	F	F	F	F			F	F			F	F	F

Total page faults – 15

Consider the following page reference string arrival with three page frames:- 5, 6, 7, 8, 9, 8, 7, 5, 8, 9, 8, 7, 9, 6, 5, 6.
Calculate the number of page faults with optimal and FIFO (First In First Out) page replacement algorithms. (S - 22)

Optimal :

REF	5	6	7	8	9	7	8	5	9	7	8	7	9	6	5	6
F1	5	5	5	5	9	9	9	9	9	9	9	9	9	9	9	9
F2		6	6	8	8	8	8	5	5	5	8	8	8	8	5	5
F3			7	7	7	7	7	7	7	7	7	7	7	6	6	6
Fault	F	F	F	F	F			F			F			F	F	

Number of Hits: 7
Page Faults: 9

FIFO :

REF	5	6	7	8	9	7	8	5	9	7	8	7	9	6	5	6
F1	5	5	5	8	8	8	8	8	8	7	7	7	7	6	6	6
F2		6	6	6	9	9	9	9	9	9	8	8	8	8	5	5
F3			7	7	7	7	7	5	5	5	5	5	9	9	9	9
Fault	F	F	F	F	F			F		F	F		F	F	F	

Number of Hits: 5
Page Faults: 11

Consider the string: 0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3, 4, 5, 6, 7 with Frame Size 3 and calculate page Fault in both the cases using FIFO algorithm. (S - 24)

FIFO

i. Frame Size = 3

Ref	0	1	2	3	0	1	2	3	0	1	2	3	4	5	6	7
F1	0	0	0	3	3	3	2	2	2	1	1	1	4	4	4	7
F2		1	1	1	0	0	0	3	3	3	2	2	2	5	5	5
F3			2	2	2	1	1	1	0	0	0	3	3	3	6	6
Fault	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F

Total Page Fault = 16

ii. Frame size : 4

Ref	0	1	2	3	0	1	2	3	0	1	2	3	4	5	6	7
F1	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4	4
F2		1	1	1	1	1	1	1	1	1	1	1	1	5	5	5
F3			2	2	2	2	2	2	2	2	2	2	2	2	6	6
F4				3	3	3	3	3	3	3	3	3	3	3	3	7
Fault	F	F	F	F									F	F	F	F

Total page fault = 8

Unit - V File Management

➢ 5.1 File Concepts

In an operating system, a file is a collection of data or information that is stored on a storage device (such as a hard disk, SSD, or USB drive). Files are used to store data in a structural format. We can perform different types of operations on file like creating file, Reading file, writing file, Deleting file, Renaming file, Moving file, Copying file. A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks. Commonly files represent programs (source and object forms) and data. Data files may be numeric, alphabetic, alphanumeric or binary. In general a file is a sequence of bits, bytes, lines or records whose meaning is defined by the file's creator and user. The information in a file is defined by its creator.

- **File Attributes**

Describe any four file attributes (W – 19, S - 23)

1. **Name:**
   The symbolic file name is the only information kept in human readable form.

2. **Identifiers:**
   File System gives a unique tag or number that identifies file within file system and which is used to refer files internally.

3. **Type:**
   This information is needed for those system that support different types.

4. **Location:**
   This information is a pointer to a device and to the location of the file on the device.

5. **Size:**
   The current size of the file (in bytes, words or blocks) and possibly the maximum allowed size are included in this attribute.

6. **Protection:**
   Access control information determines that who can do reading, writing, executing and so on.

7. **Time, date and user Identification:**
   This information may be kept for creation, last modification and last use. These data can be useful for protection, security and usage monitoring.

- **Operations**

List any four operations performed on a file (W – 19, W-22, W -23, S - 24)
Write any four systems call related to file management (W - 22)

1. Creating a file
2. Writing file
3. Reading a file
4. Repositioning within a file
5. Deleting a file
6. Truncating a file
7. Appending new information to the end of the file
8. Renaming an existing file
9. Creating copy of a file, copy file to another I/O device such as printer or display.

1. **Creating a File:** This involves two main steps. First, space is allocated within the file system. Second, a new entry for the file is created in the directory, which includes the file's name and its location.

2. **Writing a File:** To write to a file, a system call is made, specifying both the file name and the information to be written. The system then searches the directory for the file's location. The write pointer is updated after each write operation.

3. **Reading a File:** To read from a file, a system call specifies the file's name and the memory location where the data should be placed. The system maintains a read pointer to indicate the next block of the file to be read. Once data is read, the read pointer is updated.

4. **Repositioning within a File:** This operation, also known as a file seek, involves searching the directory for the appropriate file entry and then setting the current file position to a specified value. It does not require any actual data movement.

5. **Deleting a File:** To delete a file, the directory is searched for the named file. Once found, all associated file space is released, and the directory entry is erased.

6. **Truncating a File:** Instead of deleting and recreating a file, this function allows a user to erase the contents of a file while retaining its attributes.

- **File types and File system structure**

In an operating system, a file is a collection of data or information that is stored on a storage device (such as a hard disk, SSD, or USB drive). Files are used to store data in a structural format. We can perform different types of operations on file like creating file, Reading file, writing file, Deleting file, Renaming file, Moving file, Copying file. A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks. Commonly files represent programs (source and object forms) and data. Data files may be numeric, alphabetic, alphanumeric or binary. In general a file is a sequence of bits, bytes, lines or records whose meaning is defined by the file's creator and user. The information in a file is defined by its creator.

File Types :

The file name is split into two parts: a name of file and extension. For example "student.xls" the file name is student and it is created in MS - Excel application. The system uses the extension to indicate the type of the file and the type of operations that can be done on that file.

The following table indicates the common file types.

File Type	Usual Extension	Function
Executable	exe, com, bin, none	Ready to run machine language program.
Object	obj, o	Compiled, machine language, not linked.
Source code	c, cc, pas, asm, f77	Source code in various languages.
Text	text, doc	Textual data documents.
Batch	bat, sh	Commands to the command interpreter.
Word Processor	wp, rtf, tex, doc, etc.	Various Word Processor formats.
Library	lib, a, dll	Libraries of routines for programmers.

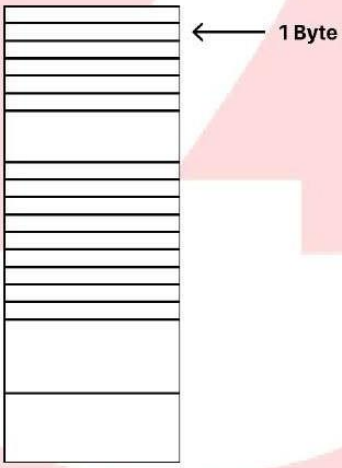
Print or view	ps, gif, dvi, pdf	ASCII or binary file in a format for printing or viewing.
Archive	arc, zip, tar	Related files grouped into one file; sometimes compressed, for archiving or storage.
Multimedia	avi, mp3, mov, mkv, mpeg, rm	Binary file containing audio or audio/video information.

File Structure :

Files can be structured in any of several ways.

1) Stream of Bytes :

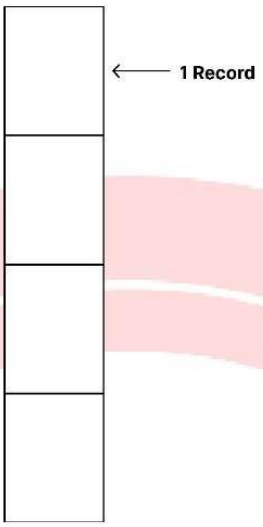
The file is treated as a continuous sequence of bytes. Data is stored in a linear fashion with no inherent structure. Commonly used for binary files where data is read and written in chunks.



2) Records :

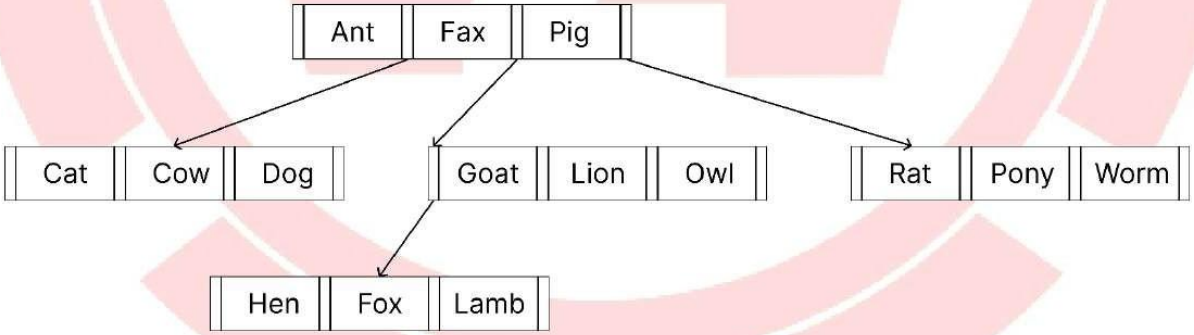
The file is divided into fixed or variable -sized records. Each record is a logical unit, often representing a single entry like a row in a database. Makes it easier to retrieve, update, or delete specific records.

Example :- employee record



3) Tree of Records :

Data is organized hierarchically, often in a tree structure. Each record can have "child" records, forming Parent-child relationships. Efficient for data retrieval, especially for hierarchical data structures like directories or XML data.

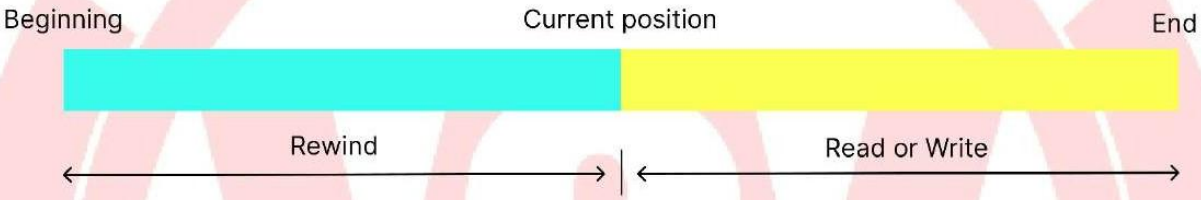


➢ 5.2 Accessing Methods

Describe sequential and direct access method (W - 19)

Sequential File Access :

Information from the file is processed in order i.e. one record after another. It is commonly used access mode. For example, editors and compilers access file in sequence. A read operation read information from the file in a sequence i.e reads the next portion of the file and automatically advances a file pointer. A write operation writes information into the file in a sequence i.e. appends to the end of the file and advances the end of the newly written material. In some operating system, a program may be able to skip forward or backward in records for some integer n.



Sequential file access

Advantages of sequential files:

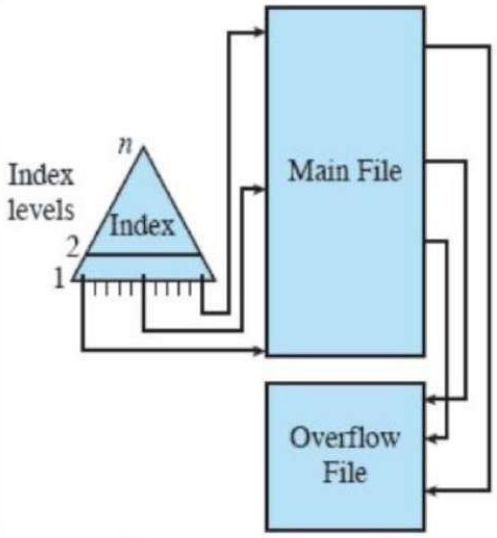
1. Easy to access the next record.
2. Data organization is very simple.
3. Automatic backup copy is created.

Disadvantages of sequential file:

1. Wastage of memory space because of master file and transaction file.
2. It is more time consuming since, reading, writing and searching always start from beginning of file.

Indexed Sequential File Access

An indexed sequential file is a sequential file in which the records are indexed. An indexed sequential file is an improvement over a sequential file. Two features are added in this file namely, an index to the file and an overflow file. Indexing of records provides the facility of searching the records randomly. An indexed file is a simple sequential file that contains an index as its records. Entries in indexed files are made up of two fields, the key field, which is the same as the key field in the main file and a pointer pointing to some record in the main file. To find a specific field in the main file, the index is searched for the highest key value, which is equivalent to the desired value. The pointer related to key field start searching the record at location it indicates. The search continues in the main file at the location indicated by the pointer.



Advantages of Index Sequential File :

1. Variable length records are allowed.
2. Indexed sequential file may be updated in sequential or random mode.
3. very fast operation.

Disadvantages of Index sequential File :

1. as the file grows, a performance deteriorates rapidly.
2. When a new record is added to the main file, all of the index files must be updated.
3. consumes large memory space for maintaining index files.

Direct access :

It is also called as relative access. A file is made up of fixed length logical records that allow programs to read and write records rapidly in no particular order. Direct access method is based on disk model of file which allows random access to any file block. For direct access, a file is viewed as a numbered sequence of block or record so we can directly read block 14, then block 53, and so on. Read n operation is used to read the nth block from the file whereas write n is used to write in that block. The block number provided by the user to the operating system is a relative block number. A relative block number is an index relative to the beginning of the file.

Direct file access

Advantages of Direct File Access:

1. Using this method we can access any records randomly.
2. It gives faster retrieval of records.

Disadvantages of Direct File Access:

1. Wastage of storage space, if hashing algorithm is not chosen properly.
2. This method is complex and expensive.

➢ 5.3 File Allocation Methods

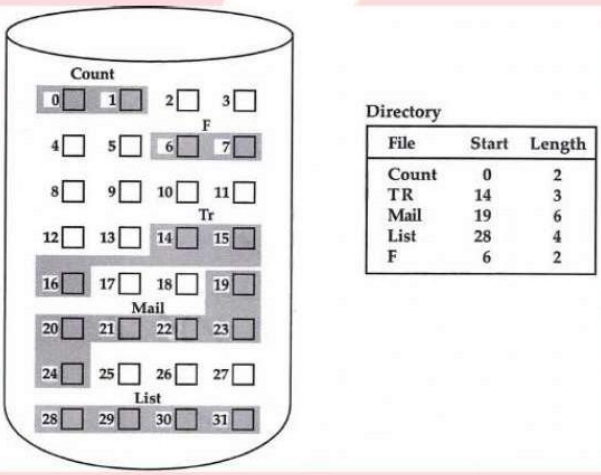
From the user's point of view, a file is an abstract data type. It can be created, opened, written, read, closed and deleted without any real concern for its implementation. The implementation of a file is a problem for the operating system. The main problem is how to allocate space to these files so that disk space is effectively utilized and file can be quickly accessed.

Three major methods of allocating disk space are in wide use:

i) Contiguous file allocation.
ii) Linked file allocation.
iii) Indexed file allocation.

• Contiguous Allocation

Enlist different file allocation methods? Explain contiguous allocation method in detail (W – 19, S – 23, S - 24)



Directory

File	Start	Length
Count	0	2
TR	14	3
Mail	19	6
List	28	4
F	6	2

The Contiguous allocation method requires each file to occupy a set of contiguous address on the disk. Contiguous allocation of a file is defined by the disk address of the first block and its length. If the file is 'n' blocks long and starts at location 'b', then it occupies blocks b, b+1, b+2, ... b+n-1. Contiguous allocation supports both sequential and direct access. The

difficulty with Contiguous allocation is finding space for a new file. If file to be created are 'n' blocks long, We must search free space list for 'n' free Contiguous blocks.

Advantages of Contiguous File Allocation Method

① Supports both sequential and direct access methods.

② Contiguous allocation is the best form of allocation for sequential files.

③ It is also easy to retrieve a single block from a file...

④ Reading all blocks belonging to each file is very fast.

⑤ provides good performance.

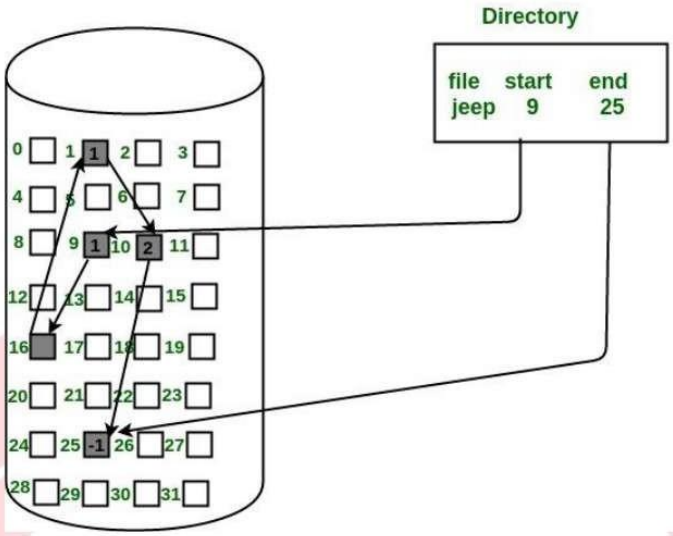
Disadvantages of Contiguous File Allocation Method

① Suffers from external Fragmentation.

② Very difficult to find contiguous blocks of space for new files.

③ Compaction may be required and it can be very expensive.

- **Linked file allocation :**
  Describe linked file allocation method with suitable example. Also list its one advantage (S -22, W – 22, W - 23)



Linked allocation solves all problems of contiguous allocation. This allocation is on the basis of an individual block. Each block contains a pointer to the next block in the chain. The disk block can be scattered anywhere on the disk. The directory contains a pointer to the first and last block of the file. To create a new file, simply create a new entry in the directory.

Advantages of Linked File Allocation Method :

① Any free blocks can be added to a chain.

② These is no external fragmentation.

③ Best suited for sequential files that are to be processed sequentially.

④ No need to know the size of the file in advance.

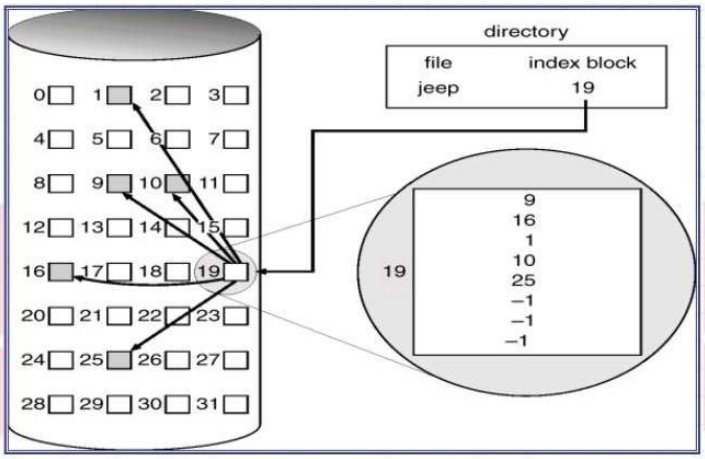
Disadvantages of Linked File Allocation Method :

① This method requires more space to store pointers.

② This method cannot support direct access.

③ It is not an efficient scheme because the list traversal needs to read each block which is quite time consuming.

- **Indexed File Allocation**



In this method, each file has its own index block. This index block is an array of disk block address. When a file is created, an index block and other disk blocks according to the file size are allocated to that file. Pointer to each allocated block is stored in the index block of that file. It contains file name and address of index block. When any block is allocated to the file, its address is updated in the index block. Each i^{th} entry in the index block points to the i^{th} block of the file. To find and read the i^{th} block, we use Pointer to that block from index block.

Advantages of Indexed File Allocation Method :

1. Does not suffer from external fragmentation.
2. Support both sequential and direct access to the file.
3. No Need for user to know size of the file in advance.
4. Entire block is available for data as no space is occupied by pointers.

Disadvantages of Indexed File Allocation Method :

1. It required lot of space for keeping pointers so wasted space of memory.
2. Indexed allocation is more complex and time consuming.
3. keeping index in memory requires space.

5.4 Directory Structure

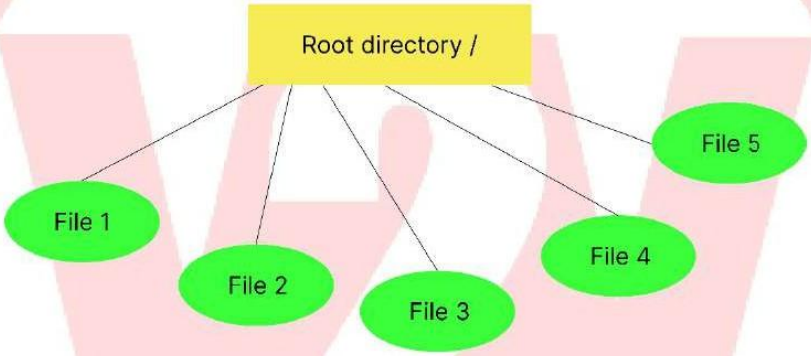
Describe following directory structures in short with neat sketches:

i) Single level

ii) Two level

iii) Tree structured (S – 22, W – 22, S – 23, W – 23, S - 24)

• Single Level Directory Structure



It is the simplest form of directory structure is having one directory containing all the files. Sometimes it is called the root directory. In this directory structure unique name must be assigned to each file of the directory. The single level directory structure appears as the list of files or a sequential file having the file names serving as the key. Single level directory structure was implemented in the older versions of single user system.

Advantages of single level directory structure

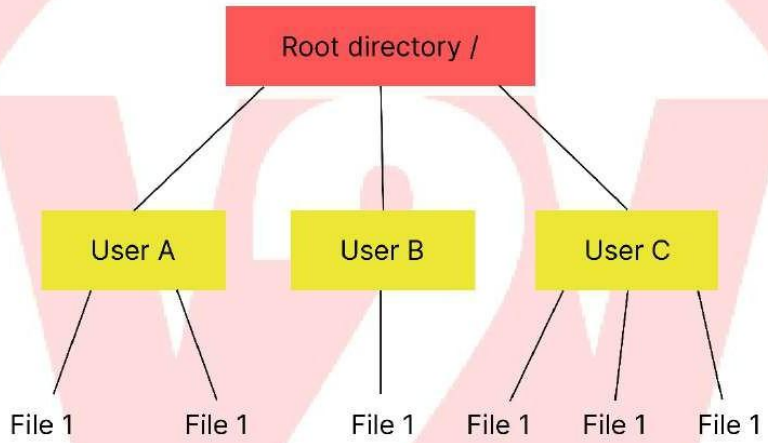
1. Single level directory structure is easy to implement and maintain.
2. It is simple directory structure.
3. Single level directory structure, the operations like creation, searching, deletion, updating are very easy and faster.

Disadvantages of Single Level directory structure

1. It having only one directory in a system so there may chance of name collision.

2. In single level directory structure, it is difficult to keep track of the files.
3. This directory is not used on multi- user systems.
4. The file such as graphics, text etc. are inconvenient for this data structure.

- **Two-level directory**



In two level directory structure, a separate directory is provided to each user and all these directories are contained and indexed in the master directory. The user directory represents a list of files of a specific user. In this directory structure, each user has its private directory known as user file directory (UFD). When a user refers to a particular file, only his own UFD is searched. Thus different users may have files with the same name. Two level directory structure used on a multi-user computer or on a simple network of personal computers that shared a common file server over a local area network.

A two-level directory as a tree of height 2:

i) The root of the tree is the MFD.

ii) Its direct descendants are the UFDs.

Advantages of Two Level Directory Structure

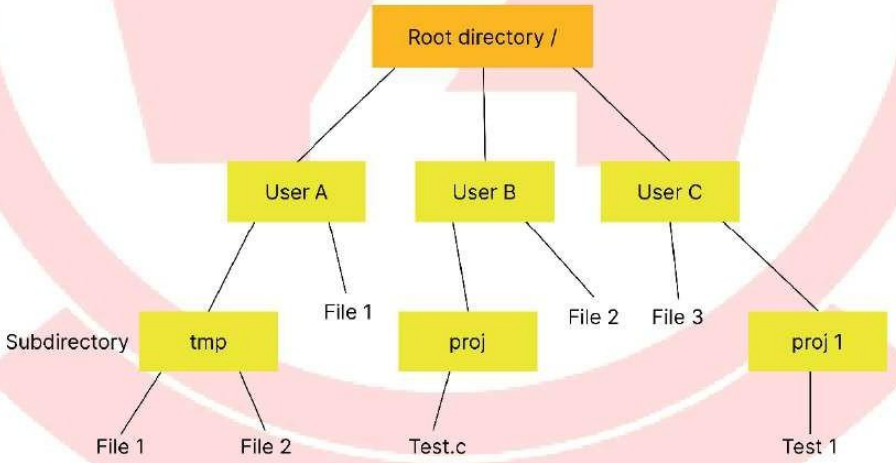
1. It solves the file name collision problem by creating own user directory.

2. This method isolates one user from another and protects user's files.
3. Different users may have files with same name.

Disadvantages of Two Level Directory:

1. Still it not very scalable, two files of the same type cannot be grouped together in the same user.
2. Sharing of files by different users is difficult.

- **Tree Structure**



In this directory structure users can create their own sub-directories and organize their files. The tree has a root directory and every file has a unique path name. A directory contains a set of files or subdirectories. All directories have the same internal format. One bit in each directory defines the entry as a file (0) or as a subdirectory (1). Each process has a current

directory. Current directory contains files that are currently required by the process. When reference is made to a file, the current directory is searched. If a file needed that is not in the current directory, then the user usually must either specify path name or change the current directory.

Advantages of Tree Structured Directory

1. User can create directory as well as subdirectory.
2. It provides a better structure to the system.
3. Managing millions of files is easy with tree structured directory.

Disadvantages of Tree Structured Directory

1. The tree structure can create duplicate copies of the files.
2. The users could not share files or directories.
3. It is inefficient, because accessing a file may go under multiple directories.4.
4. Search time may become unnecessarily long.