V2V EdTech LLP | OSY (315319) | Super 25 Questions with Solution

OSY – Super 25 – VIMP Questions with Solution by Akshay Sir (V2V Class)

Unit Wise - 2 Marks Questions

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
v2vedtech.com
V2V EdTech LLP
v2vedtech

Questions

1. Enlist types of operating system.
Types of operating system
a. Batch Systems
b. Multiprogramming
c. Multitasking
d. Time-Sharing Systems
e. Desktop Systems
f. Distributed system
g. Clustered system
h. Real Time system

2. Distinguish Between:
Compare between CLI and GUI based OS
Compare between Time sharing operating system and multiprogramming operative system.
Multitasking vs Multiprogramming
Time sharing vs Real Time System

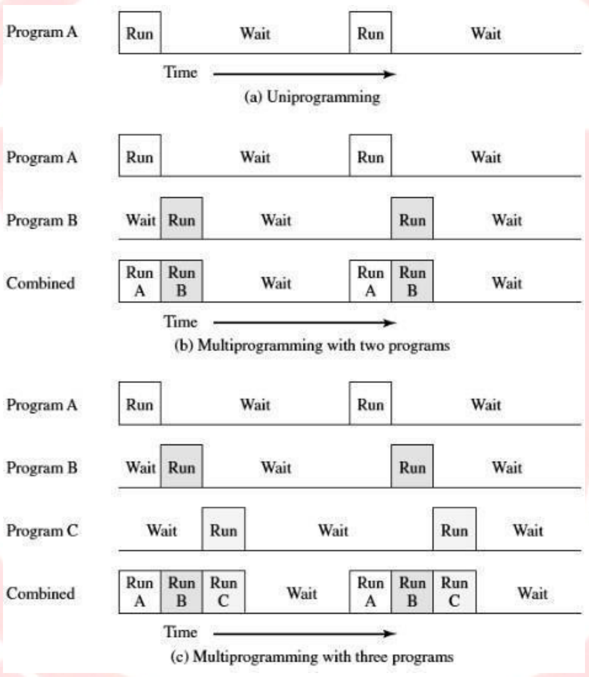
3. Explain System Call? Types of System Call?
4. Explain services provided by Operating System?
5. Draw the structure of unix operating system. Concept of Spooling
6. Explain Process State Diagram with neat labeled diagram?
7. Define Process?
8. Define Scheduler? Explain types of Scheduler?
9. Define Context Switching with neat labeled diagram?
10. Explain Inter Process Communication ? Explain shared Memory System and Message Passing System
11. Difference between
a. Process and Program
b. Process and Thread
c. User level Thread and Kernel Level Thread
12. Threads? Explain Multithreading Models with neat labeled diagrams
13. Execute process Commands (top, ps, kill, wait, sleep, exit, nice)
14. Explain Scheduling Criteria
15. Difference between Preemptive and Non-Preemptive Scheduling
16. Algorithms and Numericals on below topic:
FCFS
SJF
SRTN
RR
Priority Scheduling
Multilevel Queue Scheduling

17. Questions on Deadlock
Define Deadlock? Necessary conditions leading to deadlock?
Define Deadlock? Explain Deadlock Prevention?

1. Enlist types of operating system. Ans. **Types of operating system**

1. Batch Systems
2. Multiprogramming
3. Multitasking
4. Time-Sharing Systems
5. Desktop Systems
6. Distributed system
7. Clustered system
8. Real Time system

Explain multiprogramming OS in detail.
Ans.



1. In multiprogramming, more than one program lies in the memory.
2. The scheduler selects the jobs to be placed in ready queue from a number of programs.
3. The ready queue is placed in memory and the existence of more than one program in main memory is known as multiprogramming.
4. Since there is only one processor, there multiple programs cannot be executed at a time.
5. Instead, the operating system executes part of one program, then the part of another and so on.
6. Example of multiprogramming: user can open word, excel, access and other applications in a system.

Explain Real Time OS. Explain its types
Ans. A real-time operating system (RTOS) is a special-purpose operating system used in

computers that has strict time constraints for any job to be performed. It is employed mostly in those systems in which the results of the computations are used to influence a process while it is executing. Whenever an event external to the computer occurs, it is communicated to the computer with the help of some sensor used to monitor the event. The sensor produces the signal that is interpreted by the operating system as an interrupt. On receiving an interrupt, the operating system invokes a specific process or a set of processes to serve the interrupt.

Types of Real time OS

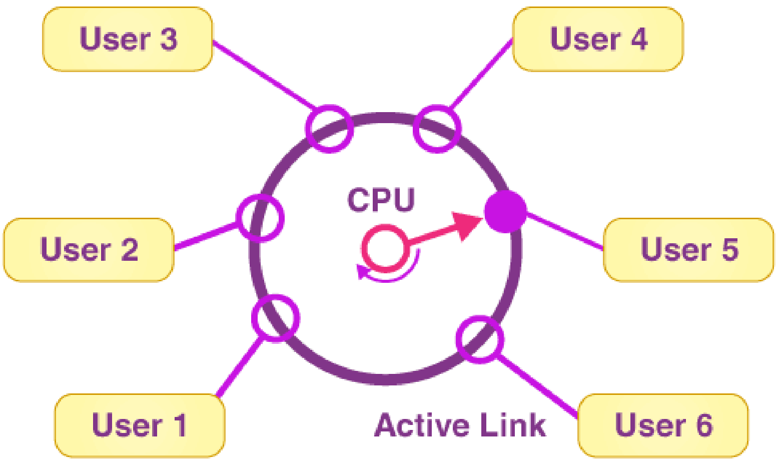
Hard Real-Time operating system: In Hard RTOS, all critical tasks must be completed within the specified time duration, i.e., within the given deadline. Not meeting the deadline would result in critical failures such as damage to equipment or even loss of human life. Consider an on-stock trading software. If someone wants to sell a particular share, the system must ensure that command is performed within a given critical time. Otherwise, if the market falls abruptly, it may cause a huge loss to the trader.

Soft Real-Time operating system: Soft RTOS accepts a few delays via the means of the Operating system. In this kind of RTOS, there may be a closing date assigned for a particular job, but a delay for a small amount of time is acceptable. So, cut off dates are treated softly via means of this kind of RTOS. For Example, This type of system is used in Online Transaction systems and Livestock price quotation Systems.

Firm Real-Time operating system: In Firm RTOS additionally want to observe the deadlines. However, lacking a closing date might not have a massive effect, however may want to purposely undesired effects, like a massive discount within the fine of a product. For Example, this system is used in various forms of Multimedia applications.

Describe working of time sharing system with neat diagram.

Ans.



1) In time sharing system, the CPU executes multiple jobs by switching among them.

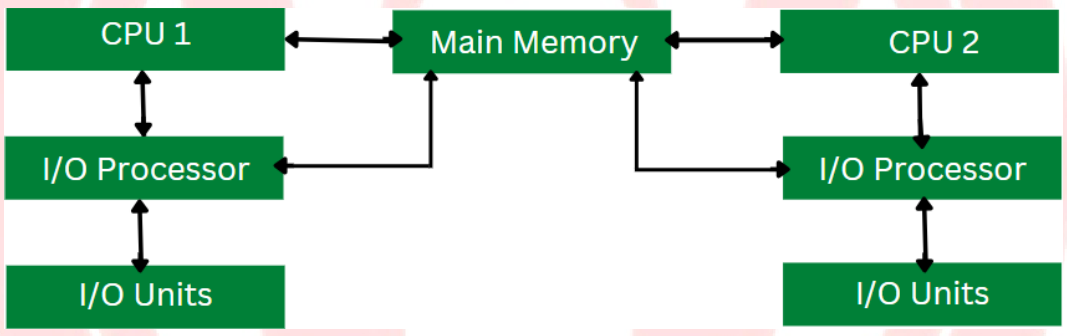
2) The switches occur so frequently that the users can interact with each program while it is running.

3) It includes an interactive computer system which provides direct communication between the user and the system.

4) A time-sharing system allows many users to share the computer resources simultaneously.
5) The time-sharing system provides the direct access to many users where CPU time is divided among all the users on scheduled basis.
6) The operating system allocates a time slice to each user.
7) When this time is expired, it passes control to the next user on the system.
8) 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.
9) In this time slice each user gets attention of the CPU.

Describe multiprocessor OS with its advantages (any two)

Ans. A multiprocessing operating system is defined as a type of operating system that makes use of more than one CPU to improve performance. Multiple processors work parallelly in multi-processing operating systems to perform the given task. All the available processors are connected to peripheral devices, computer buses, physical memory, and clocks. The main aim of the multi-processing operating system is to increase the speed of execution of the system. The use of a multiprocessing operating system improves the overall performance of the system.



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 compared to multiple single systems. Because the processors can share peripherals cabinets and power supplies.
 It increases reliability: if functions can be distributed properly among several processors, then the failure of one processor will not halt the system, but rather will only slow it down.
Minimum hardware required.
 Higher performance due to parallel processing.

2. Difference Between

a. Compare between CLI and GUI based OS

Ans.

Parameter	Command Line Interface (CLI)	Graphic User Interface (GUI)
Definition	Interaction is by typing commands.	Interaction with devices is by graphics and visual components and icons.
Understanding	Commands need to be memorized.	Visual indicators and icons are easy to understand.
Memory	Less memory is required for storage.	More memory is required as visual components are involved.
Working Speed	Use of keyboard for commands makes CLI quicker.	Use of mouse for interaction makes it slow.
Resources Used	Only keyboard.	Mouse and keyboard both can be used.
Accuracy	High.	Comparatively low.
Flexibility	Command Line Interface does not change; remains the same over time.	Structure and design can change with updates.

b. Compare between Time sharing operating system and multiprogramming operative system.

Ans.

Time-Sharing OS	Multi-programming OS
In this time-sharing Operating System, many users/processes are allocated with computer resources in respective time slots.	Multiprogramming operating system allows execution of multiple processes by monitoring their process states and switching in between processes.
Processor's time is shared with multiple users; that's why it is called a time-sharing operating system.	Processor and memory underutilization problem is resolved, and multiple programs run on CPU; that's why it is called multiprogramming.
In this process, two or more users can use a processor in their terminal.	In this, the process can be executed by a single processor.
Time sharing OS has fixed time slice.	Multi-programming OS has no fixed time slice.
Time sharing system minimizes response time.	Multiprogramming system maximizes processor use.
Example: Windows NT.	Example: Mac OS.

c. Compare Time Sharing vs Real time System

Ans.

Real Time System	Time Sharing System
In real time system, a job has to be completed within fixed deadline (time allowed).	In time sharing system, fixed time is given to each process and all the processes are arranged in a queue.
A real-time system has well-defined, fixed time	It requires more complicated CPU scheduling.

constraints.	Algorithms.
If job is not completed within the given time then system may extend time for doing the operations.	If job is not completed within the given time, then it jumps to the next job leaving the previous job unfinished. After processing to each job, it again give the same time for unfinished job.
Response time is important	Response time is not important.
Process deals with single application at a time.	Process deals with more than one application simultaneously.
It focuses on accomplishing a computational task before its specified deadline.	Emphasis on providing a quick response to a request.

d. Multitasking vs Multiprogramming

Ans.

Multiprogramming	Multitasking
Several programs stay in memory together, and CPU switches between them.	CPU handles multiple tasks or jobs at the same time.
CPU cannot execute more than one program at the same instant (just switches between them).	CPU gives the impression of executing multiple tasks at the same time.
Based on context switching mechanism.	Based on time-sharing + context sharing mechanism.
User cannot interact with the system while programs are running.	User can interact with the system.
Takes more time to execute processes.	Takes less time compared to multiprogramming.
More CPU idle time compared to multitasking.	Less CPU idle time, better utilization.

3. Explain System Calls? Types of System Calls?

Ans. 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.

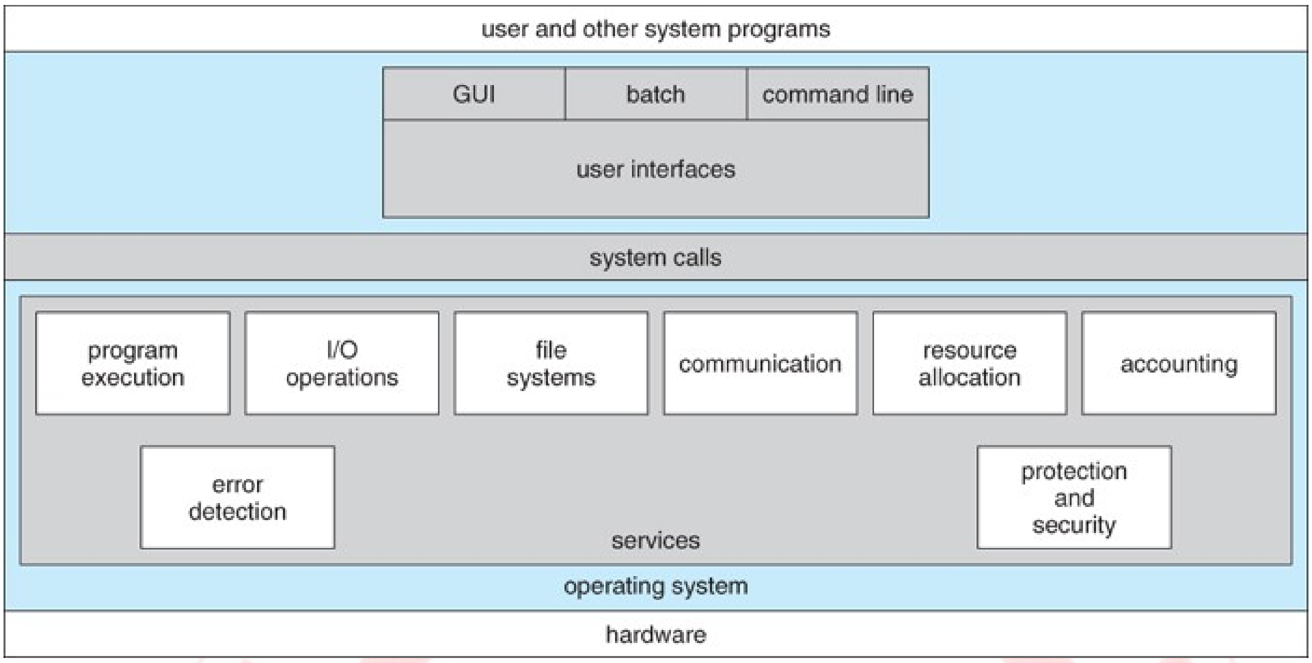
1. **File Management:**
   - These system calls manage file operations like creating, deleting, reading, and writing files.
   - **Examples:**

   - create() – To create a new file.
   - delete() – To delete an existing file.
   - open() – To open a file.
   - close() – To close an open file.
   - read() – To read data from a file.
   - write() – To write data to a file.
   - get() and set file attribute – To manage file attributes.

2. **Process Management:**
   - These system calls handle processes, including their creation, execution, and termination.
   - **Examples:**
     - create process() – To create a new process.
     - terminate() – To terminate a process.
     - load() – To load a process into memory.
     - execute() – To execute a process.
     - abort() – To abort a running process.
     - wait() – To wait for a process to complete.
     - allocate and free memory() – To manage memory for processes.
3. **Interprocess Communication (IPC):**
   - These system calls facilitate communication between processes.
   - **Examples:**
     - send message() and receive message() – For message passing.
     - create socket() – To establish a communication endpoint.
     - attach shared memory() – To use shared memory for communication.
     - connect() and disconnect() – To manage connections.
4. **I/O Device Management:**
   - These system calls manage input/output devices and their operations.
   - **Examples:**
     - request() – To request a device.
     - release() – To release a device.
     - get device attributes() and set device attributes() – To get or set device properties.
     - read() and write() – To handle data transfer.
5. **Information Processing and Maintenance:**
   - These system calls handle system-related information and maintenance tasks.
   - **Examples:**
     - set time and date() – To set system time and date.
     - get time and date() – To retrieve system time and date.
     - get system data() and set system data() – To manage system information.
     - get process information() – To access process-related data.

4. Explain services provided by Operating System?

Ans.



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:

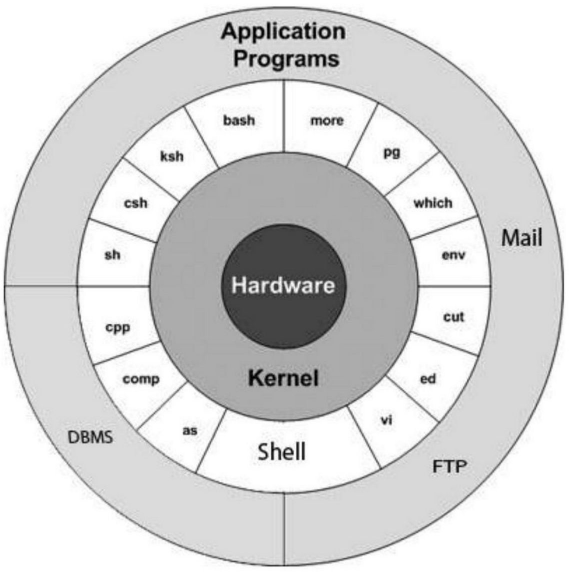
Operating System keeps track of usages of various computer resources allocated to users.

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.

5. Draw the structure of UNIX operating system. Concept of Spooling

Ans.



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.

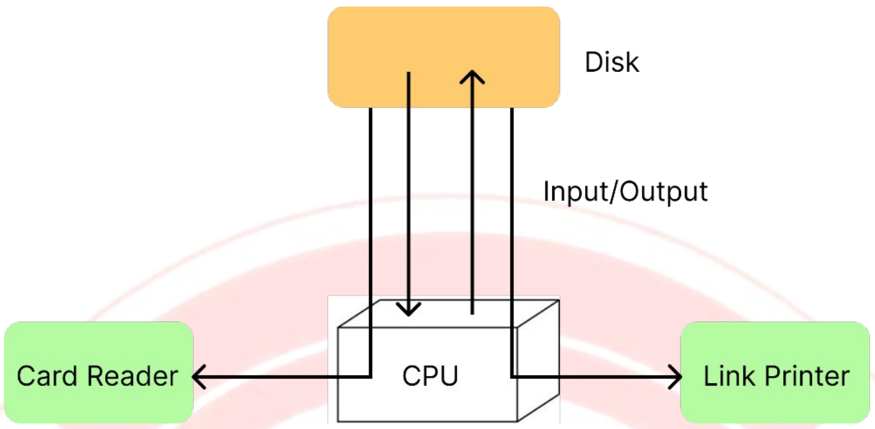
3) Commands and Utilities:

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.

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.

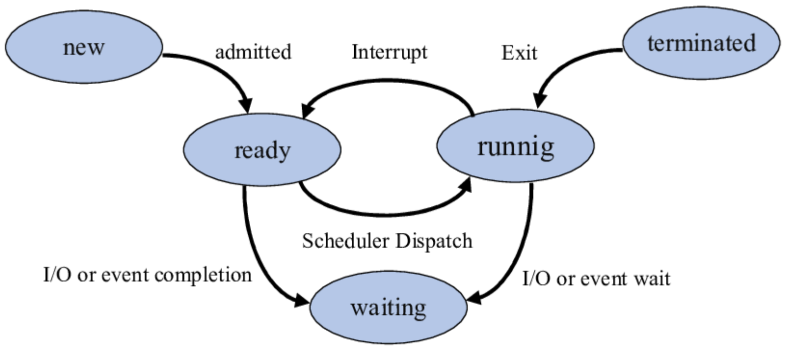
Advantages of Spooling :

- 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.

6. Explain Process State Diagram with neat labeled diagram?
Ans.



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

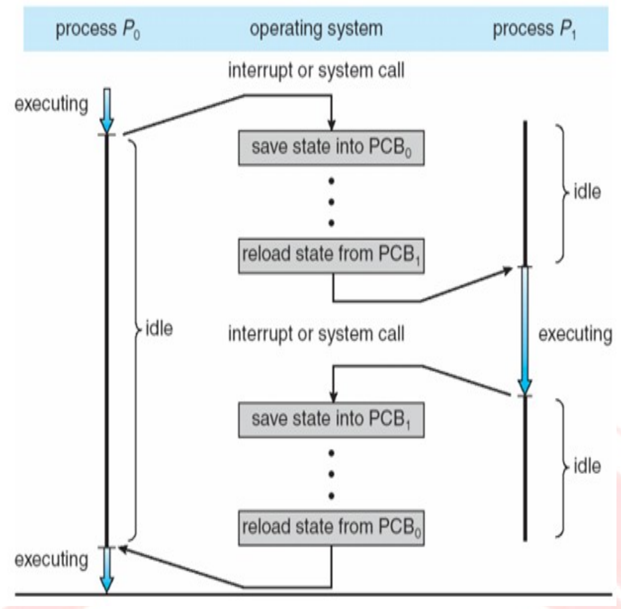
Ready: When the process is loaded into the main memory, it is ready for execution. In this state the process is waiting for processor allocation.
Running: When CPU is available, system selects one process from main memory and executes all the instructions from that process. So, when a process is in execution, it is in running state. In single user system, only one process can be in the running state. In multiuser system, there can be multiple processes which are in the running state.
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.
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.

7. Define Process
Ans. **Process:** A process is defined as, a program under execution, which competes for the CPU time and other resources. A process is a program in execution. Process is also called as job, task and unit of work.

8. Define Scheduler. Explain the types of Scheduler
Ans. 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.

Long term scheduler: Long term scheduler is also known as job scheduler. It chooses the processes from the pool (secondary memory) and keeps them in the ready queue maintained in the primary memory. Long Term scheduler mainly controls the degree of Multiprogramming. The purpose of long term scheduler is to choose a perfect mix of IO bound and CPU bound processes among the jobs present in the pool.
Short term scheduler: Short term scheduler is also known as CPU scheduler. It selects one of the Jobs from the ready queue and dispatch to the CPU for the execution. A scheduling algorithm is used to select which job is going to be dispatched for the execution. The Job of the short term scheduler can be very critical in the sense that if it selects job whose CPU burst time is very high then all the jobs after that, will have to wait in the ready queue for a very long time.
Medium term scheduler: Medium term scheduler takes care of the swapped out processes. If the running state processes needs some IO time for the completion then there is a need to change its state from running to waiting. Medium term scheduler is used for this purpose. It removes the process from the running state to make room for the other processes. Such processes are the swapped out processes and this procedure is called swapping. The medium term scheduler is responsible for suspending and resuming the processes.

9. Define Context Switching with neat labeled diagram
Ans.



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.

10. Explain Inter process communication? Explain shared memory system and message passing system

Ans.

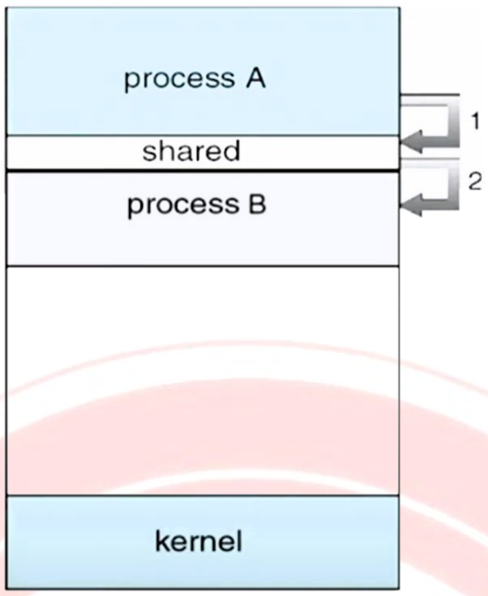
Cooperating processes require an inter-process communication (IPC) mechanism that will allow them to exchange data and information.

There are two models of IPC.

(a) Shared memory

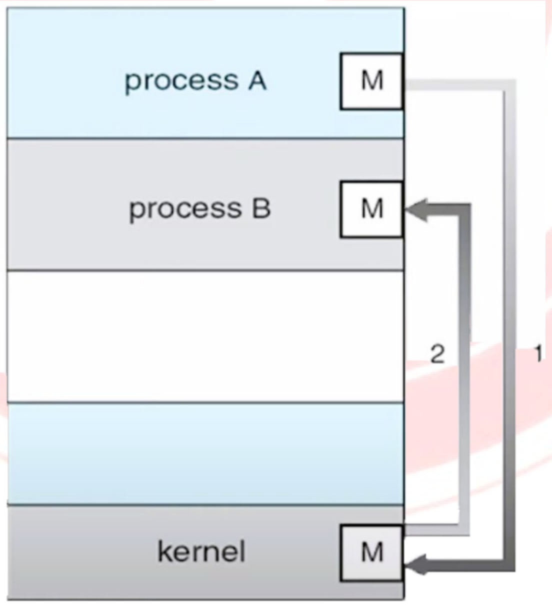
(b) Message passing

(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.

11. Difference between

a. Process and Program

Ans.

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.

b. Process and Thread

Ans.

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

c. User Level Thread and Kernel Level Thread

Ans.

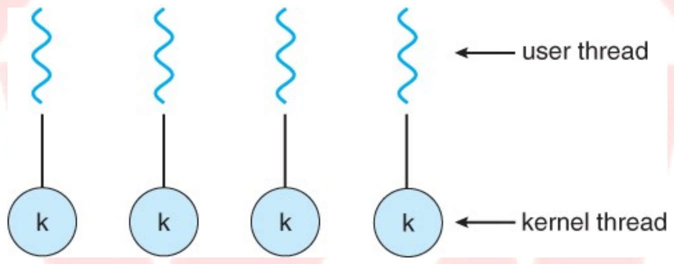
User-Level Thread	Kernel-Level Thread
Implemented and managed by the user, not by OS.	Implemented and managed by the operating system.
OS does not recognize user threads.	OS is fully aware of kernel threads.
Faster to create and manage (no system calls).	Slower because kernel involvement is required.
If one thread blocks → whole process gets blocked.	If one thread blocks → another thread can continue.
No hardware support needed for context switching.	Hardware support required for context switching.
Not suitable for multiprocessor systems.	Best for multiprocessor systems.
Example: Java threads, POSIX threads.	Example: Windows, Solaris threads.

12. Explain Multitasking Models with Neat Labeled Diagrams

Ans. 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.

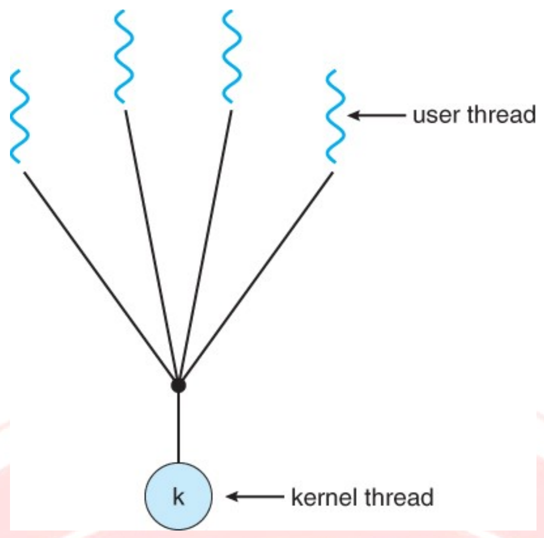
Advantages of one-to-one Model:

- 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.

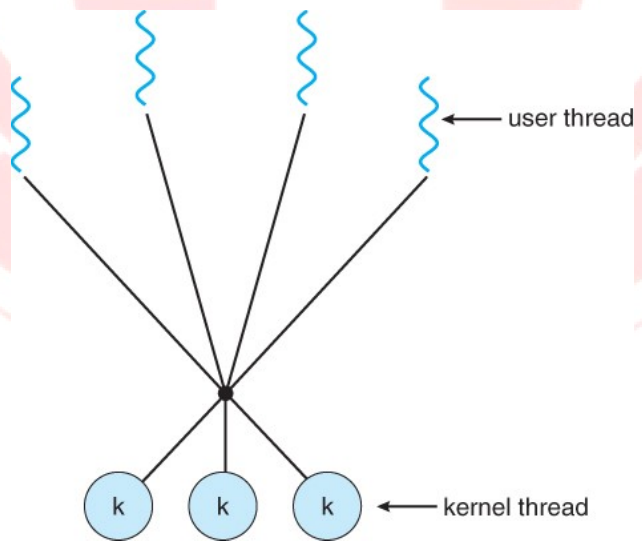
Advantages of many-to-one Model:

- 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.

13. Execute process Commands (top, ps, kill, wait, sleep, exit, nice)

Ans.

a. 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

b. ps

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.

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.

Syntax: $ ps [options]
Example: $ ps

c. kill

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

d. 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.

e. 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.

f. 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.

g. 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.

14. Explain Scheduling Criteria
Ans. 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.

15. Difference between preemptive and non-preemptive scheduling

Ans.

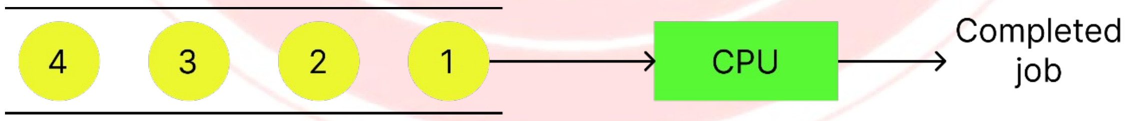
Preemptive scheduling	Non-preemptive scheduling
Even if CPU is allocated to one process, CPU can be preempted to other process if other process is having higher priority.	once, the CPU has been allocated to a process the process keeps the CPU until releases either by terminating or by switching to waiting 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: • process switch from running to ready state • process switch from waiting to ready state	Circumstances for Non-preemptive: • process switches from running to waiting state • process terminates
Example: Round Robin Example: Priority algorithms.	Example: FCFS algorithm.

16. Algorithms and Numericals on below topic:

a. FCFS : - https://youtu.be/EBRxOZxHS6k

Ans.

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.

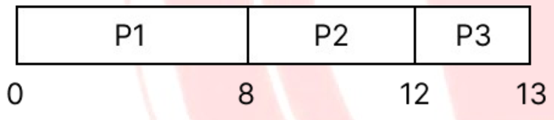
Example:

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

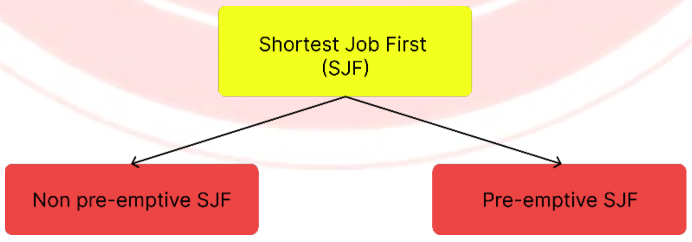
= (0 + 7 + 10) / 3

= 5.67 ms

b. SJF: - https://youtu.be/ZqllcDY8eaQ

Ans.

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:

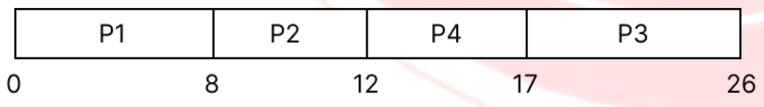
Process	Arrival Time	Burst Time
P1	0	8
P2	1	4
P3	2	9
P4	3	5

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

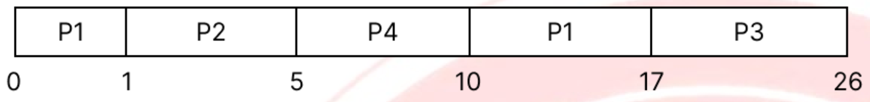
= 7.75 ms

Using pre-emptive scheduling

proces	Arrival Time	Burst Time	completion Time	Turnaround Time	waiting Time

s					
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

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).

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:

P1	P2	P4	P1	P3

0 1 5 10 17 26

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:

P1	P2	P4	P3

0 8 12 17 26

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

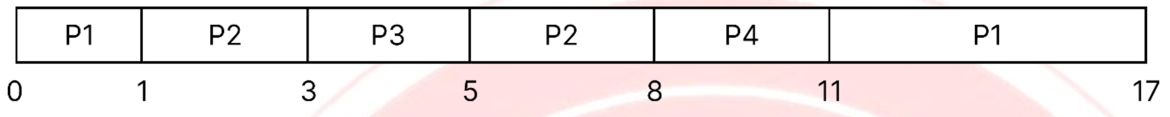
c. SRTN: - https://youtu.be/VQgOrwwManI
Ans.

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.

d. RR: - https://youtu.be/KR9Ywv02O24

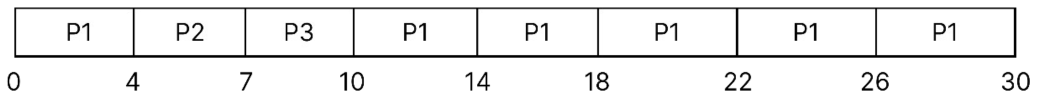
Ans. Round Robin is the pre-emptive process scheduling algorithm. The Round Robin scheduling algorithm is designed especially for time sharing systems. Processes are dispatched in a First In First Out (FIFO) sequence but each process is allowed to run for only a limited amount of time. A small unit of 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.

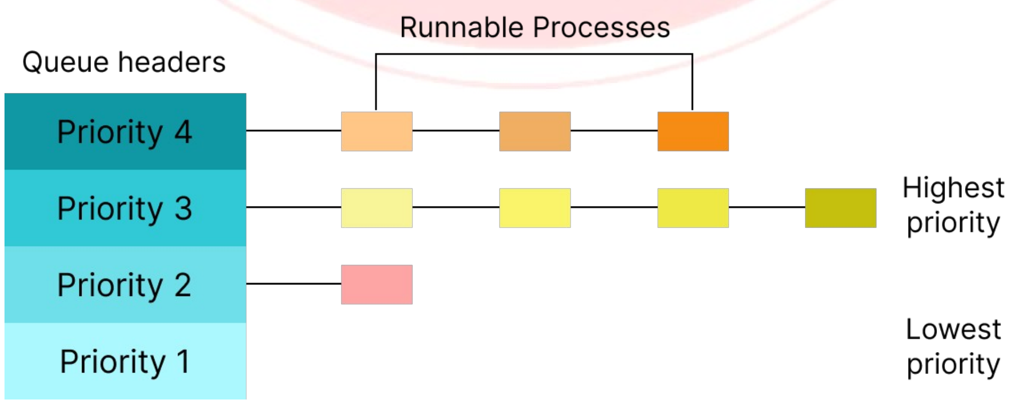
Disadvantages of Round Robin scheduling algorithm

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.

e. Priority Scheduling: -

Pre-emptive: - https://youtu.be/1M2DKDE2sE8?si=h0iJDaXM_vN_1i82
Non – Preemptive: - https://youtu.be/bB0U9BLfO7o?si=aPIX8FYc9c4qaz8u

Ans. 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 :

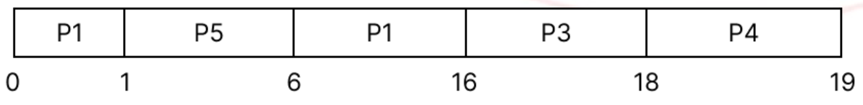
Consider the following set of processes assumed to have arrived at time 0 in the order P1, P2, P3, ---- Ps, with the length of the CPU burst time given in milliseconds. Calculate the average TAT and waiting Time

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

~~Here's the text from the image:~~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

~~Export to Sheets~~

Gantt chart:



~~| P2 | P5 | P1 | P3 | P4 | |---|---|---|---|---| | 0 | 1 | 6 | 16 | 18 | 19 |~~

Completion Time:

Completion time is calculated by using Gantt chart

Turnaround Time:

Turnaround Time = Completion Time - ~~Arrivel~~ 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

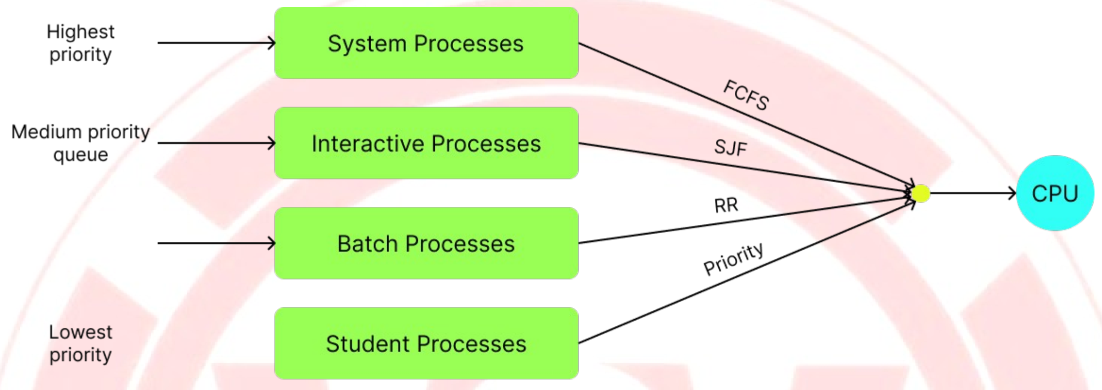
= 8.2ms

f. Multilevel Queue Scheduling: - https://youtu.be/06O4l04Ey68

Ans. 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:

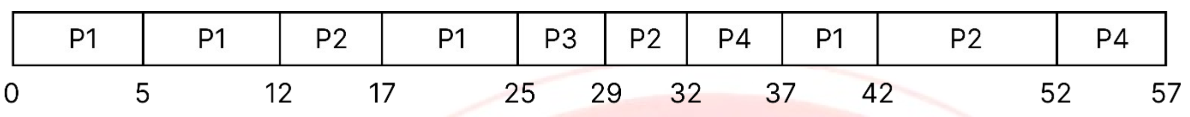
1. The processes in lower priority queues may have to starve for CPU in case processes are continuously arriving in higher priority queues.
2. In MLQ the process does not move from one queue to another queue.

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.

17. Questions on Deadlock

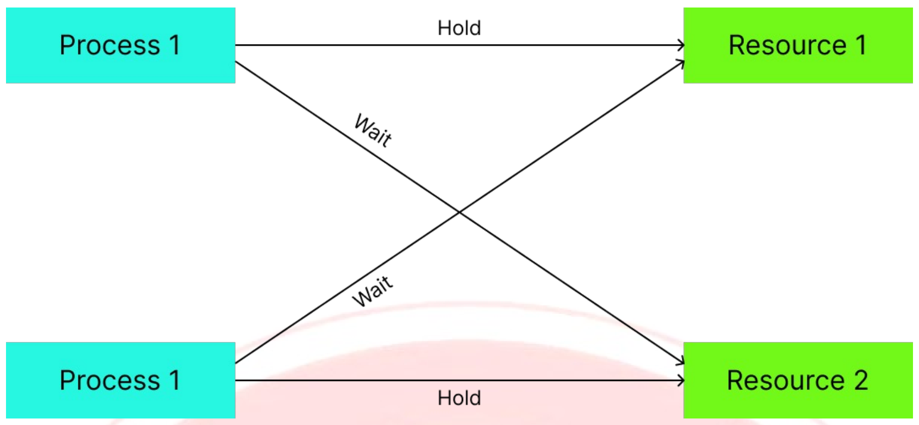
a. Define Deadlock. Explain necessary conditions leading to deadlock.

Ans. 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_0 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_0. Each process is waiting for the resource held by other waiting processes in circular form.

b. Explain Deadlock Prevention

Ans. **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

c. Explain Deadlock Avoidance

Ans. **Deadlock avoidance methods :**

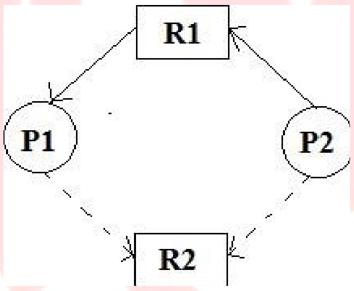
1. Safe State :

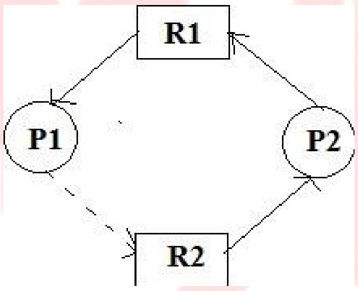
If a system is already in a safe state, we can try to stay away from an unsafe state and avoid deadlock. Deadlocks cannot be avoided in an unsafe state. A safe sequence of processes and allocation of resources ensures a safe state. Deadlock avoidance algorithms try not to allocate resources to a process 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_0 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_0	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_0> satisfies safety criteria.

Deadlock avoidance- Banker's Algorithm: - https://youtu.be/r3KP96UEjb0

- 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

18. Explain Memory Partitioning

Ans. 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)

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:

OS
8KB
8KB
8KB
8KB
8KB
8KB
8KB
8KB

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:

OS
2KB
4KB
6KB
8KB
10KB
12KB

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

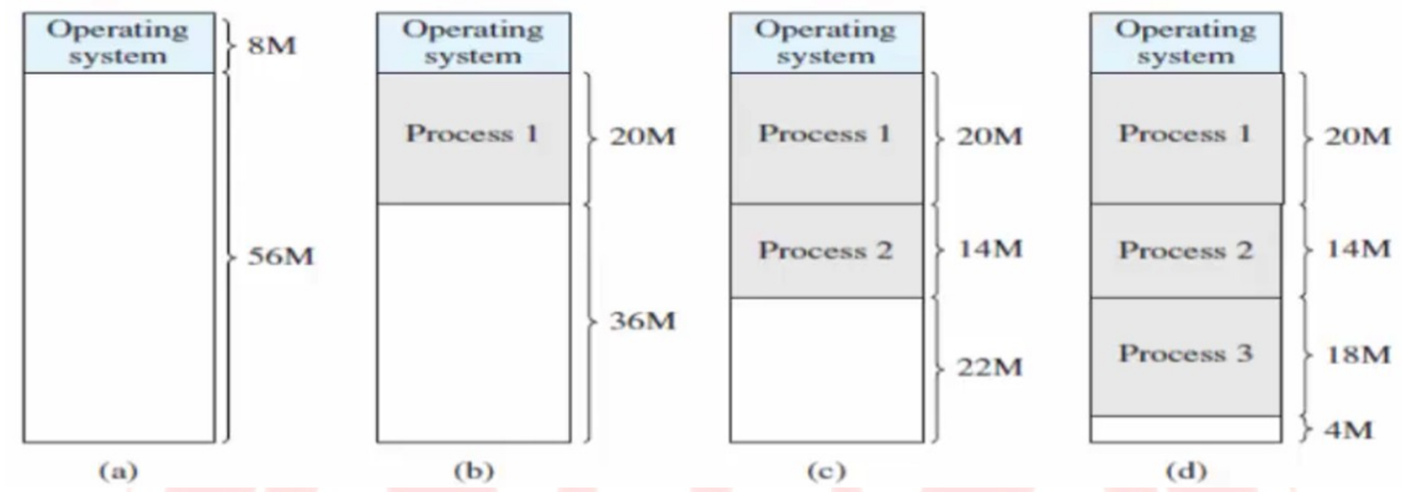
1. Memory wastage
2. Inefficient use of memory due to internal fragmentation.

2) Variable Partitioning: (Dynamic Partitioning)

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.

19. Free Space Management Technique

Ans. 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 :

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 :

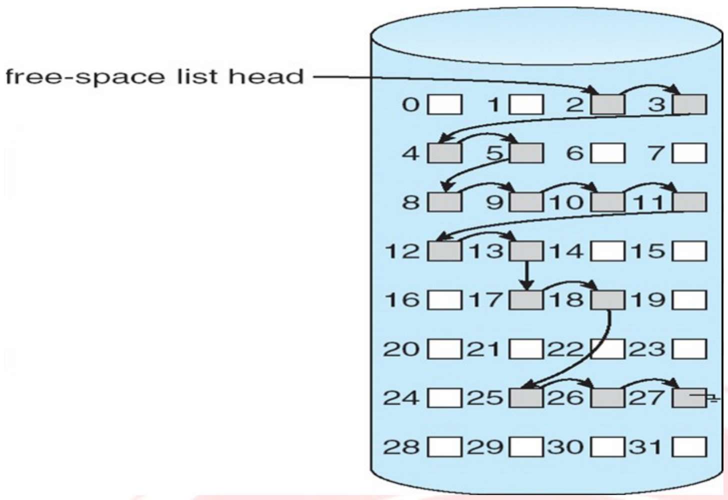
1. It is simple and efficient method to find the first free block or n-consecutive free block on the disk.
2. A bit map requires lesser space as it uses 1 bit per block.

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 :

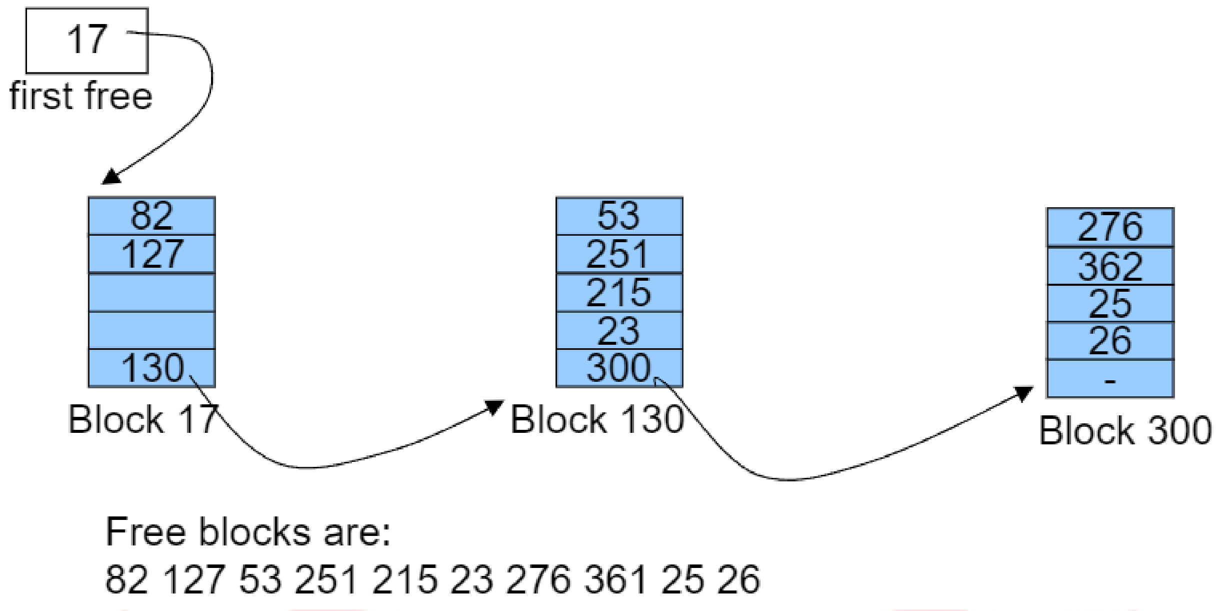
1. If a file is to be allocated a free block, the operating system can simply allocate the first block in the free space list and move the head pointer to the next free block in the list.
2. No disk fragmentation as every block is utilized.

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 :

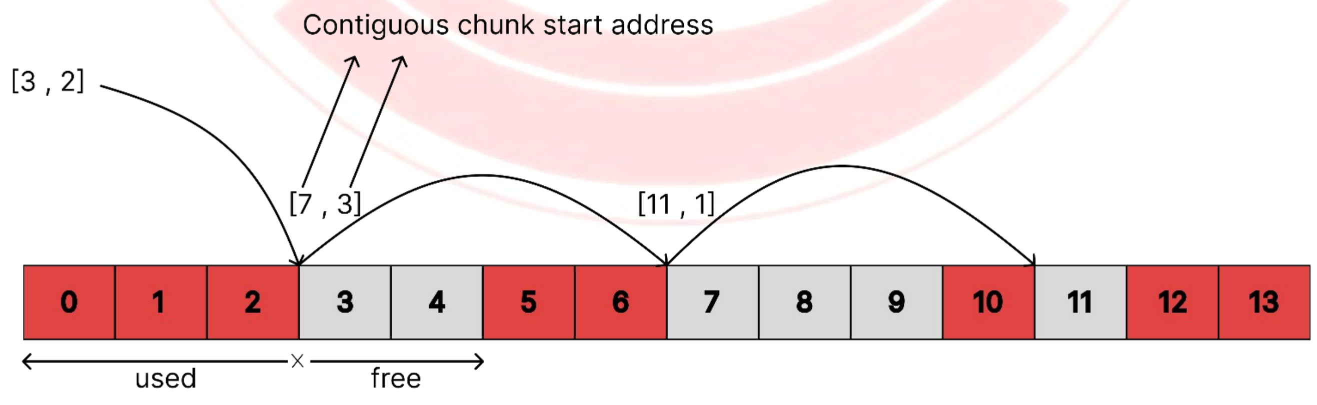
1. Finding free blocks in massive amount can be done easily using this method.
2. Reduce redundancy.

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

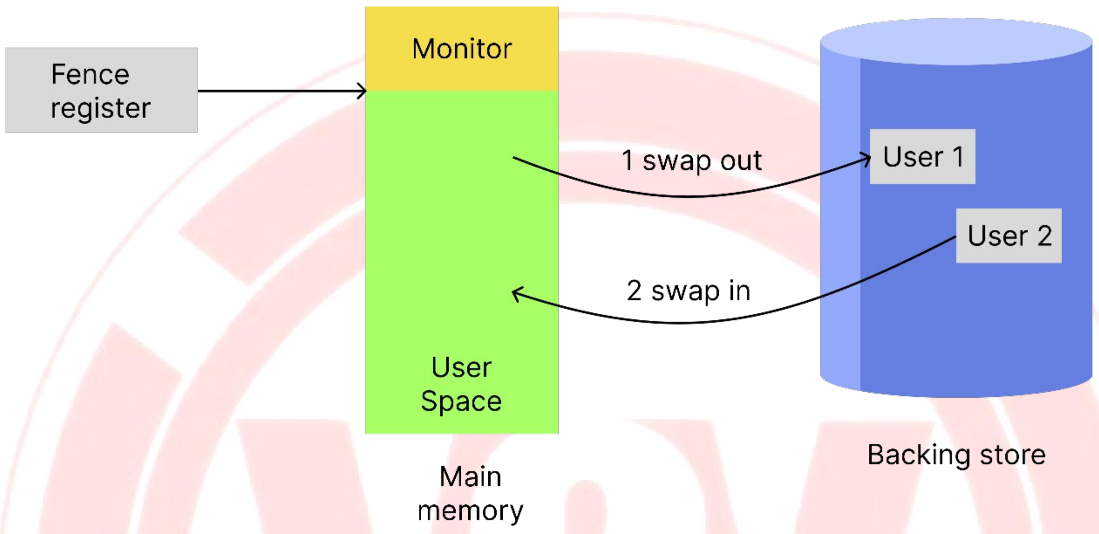
Disadvantages :

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

20. Explain the following terms:

a. Swapping

Ans.



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.

b. Fragmentation

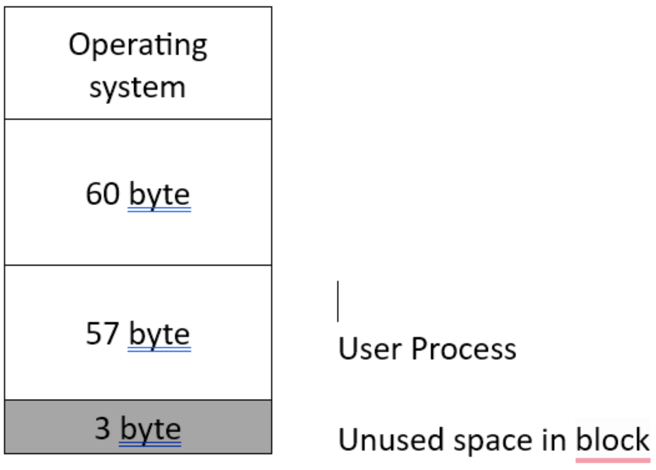
Ans. A. **Fragmentation :**

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

Types of Fragmentation :-

1. Internal Fragmentation
2. External Fragmentation.

1. Internal Fragmentation



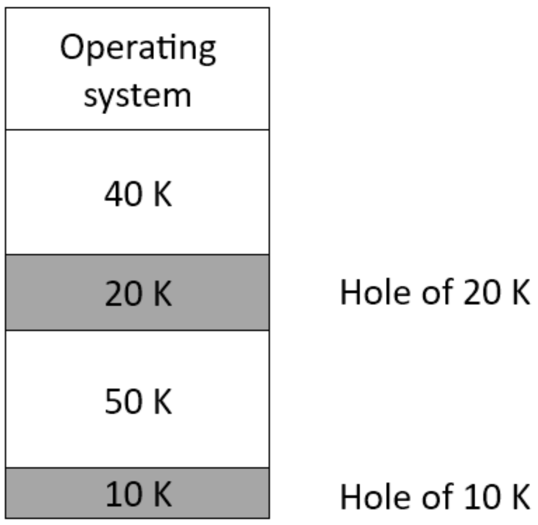
1. Wasting of memory within a partition, due to a difference in size of a partition and of the object resident within it, is called internal fragmentation.
2. Internal fragmentation occurs when the memory allocator leaves extra space empty inside block of memory that has been allocated for a client.

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.



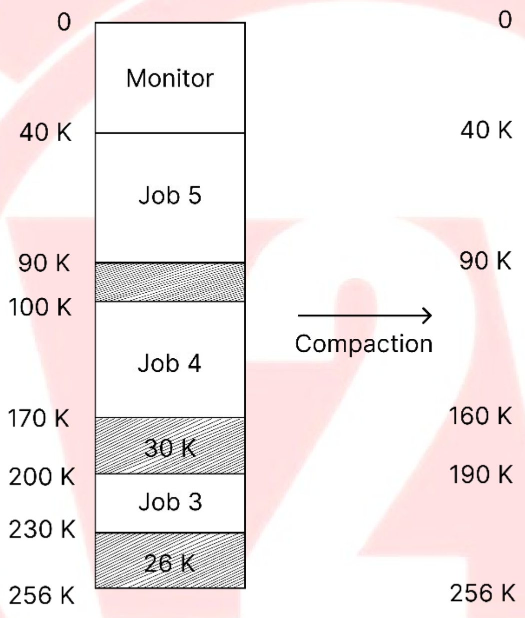
c. Compaction

Ans. 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.

21. Difference between Paging and Segmentation

Ans.

Parameter	Paging	Segmentation
1) Individual Memory	In paging, we break a process address space into blocks known as	In the case of segmentation, we break a process address space into blocks known as

	pages.	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.

22. Algorithms:

a. Partitioning Algorithm: First Fit, Best Fit and Worst Fit: - https://youtu.be/Xz0m5bm70xk

Ans.

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:

3. No internal fragmentation.
4. More efficient use of main memory.

Disadvantages of Dynamic Memory partitioning:

3. It suffers from external fragmentation.
4. It needs compaction.

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

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

First Fit :

Allocate the first free block to the new process.

OS
P1
P4 6 KB < FREE > 6 KB
P2 < FREE > 19 KB
P3 < FREE > 7 KB

Best Fit :

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

OS
P1 < FREE > 12 KB
P2

< FREE > 19 KB
P3
P4 6 KB < FREE > 1 KB

Worst Fit :
Allocate the largest free block to the new process.

OS
P1 < FREE > 12 KB
P2
P4 6 KB < FREE > 13 KB
P3 < FREE > 7 KB

b. Page Replacement Algorithm

- Ans. **First In First Out (FIFO): - https://youtu.be/vtTAjuDDU9M**

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): - https://youtu.be/RDCtEIRMwn0**

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.
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: - https://youtu.be/C8D1dd6P9GA

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

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.

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.

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.

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.

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

23. Explain file operations and file attributes

Ans.

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.

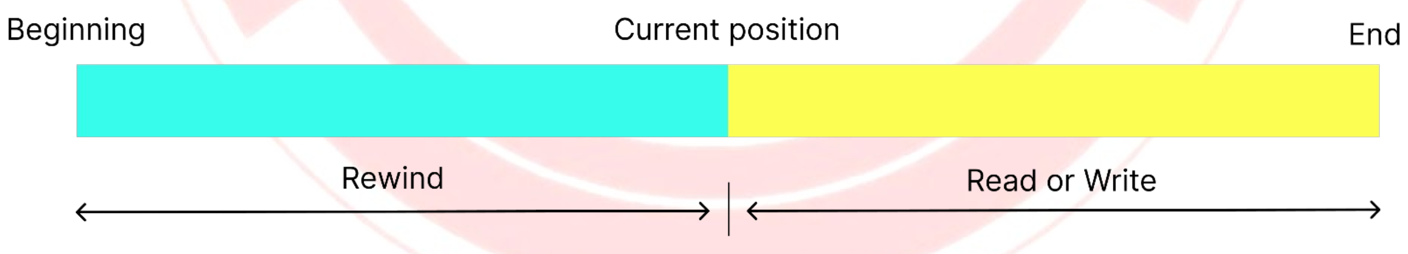
3. Reading a File: To read from a file, a system call specifies the file's name and the memory location where the data should be placed. The system maintains a read pointer to indicate the next block of the file to be read. Once data is read, the read pointer is updated.

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.

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

24. Explain File Access methods
Ans.
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

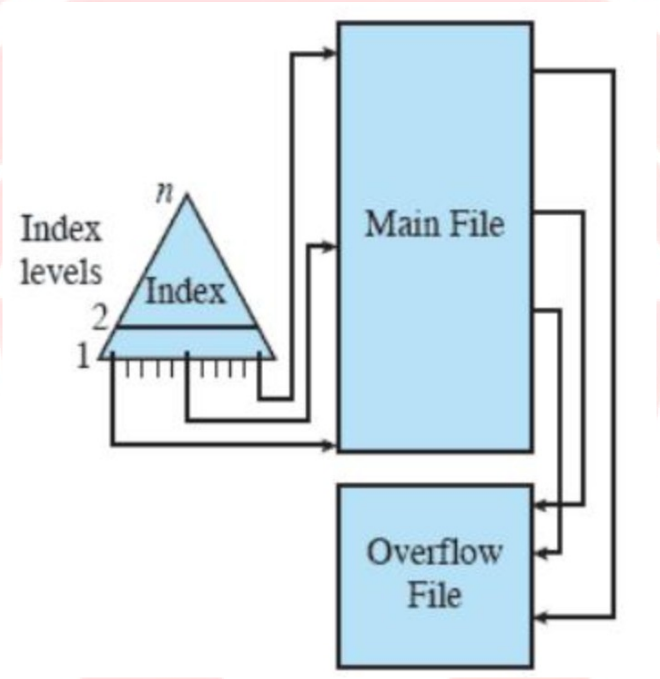
Advantages of sequential files:
1. Easy to access the next record.
2. Data organization is very simple.
3. Automatic backup copy is created.

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.

Block 0
Block 0
Block 0

Block 0
Block 0

Direct file access

Advantages of Direct File Access:

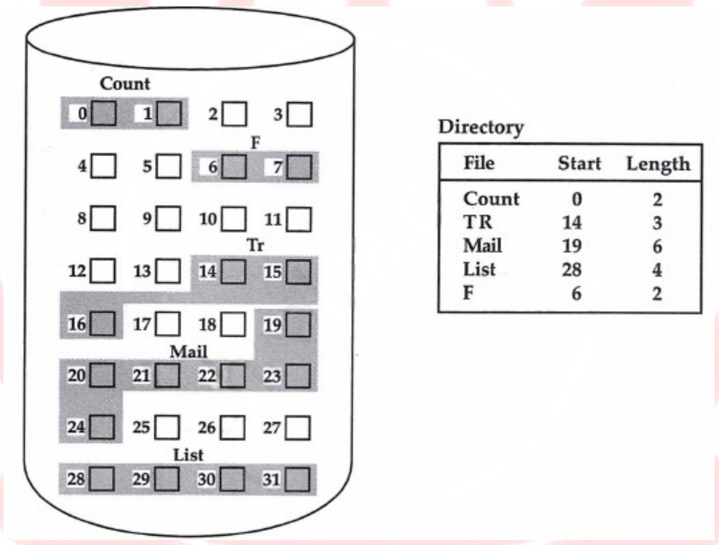
1. Using this method we can access any records randomly.
2. It gives faster retrieval of records.

Disadvantages of Direct File Access:

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

25. Explain File Allocation Methods

Ans. **Contiguous Allocation**



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

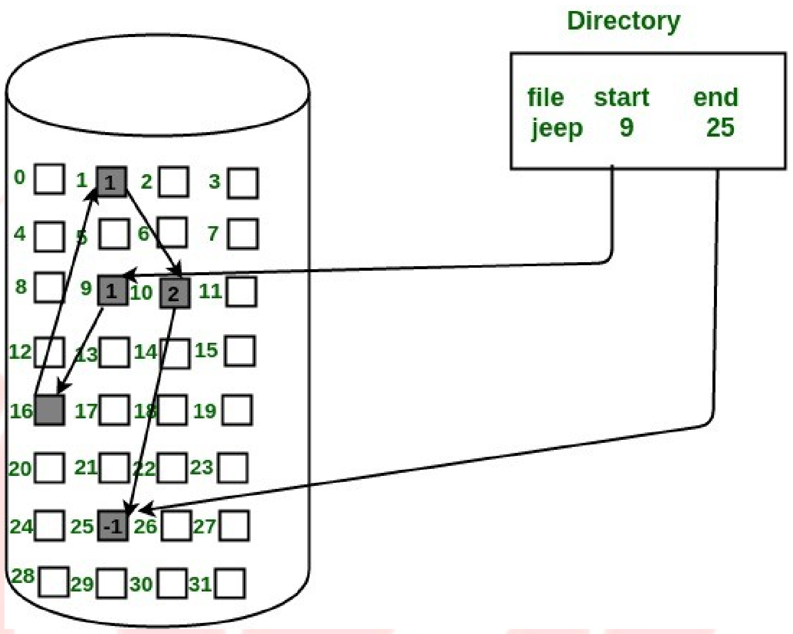
① Supports both sequential and direct access methods.
② Contiguous allocation is the best form of allocation for sequential files.
③ It is also easy to retrieve a single block from a file...
④ Reading all blocks belonging to each file is very fast.
⑤ provides good performance.

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.

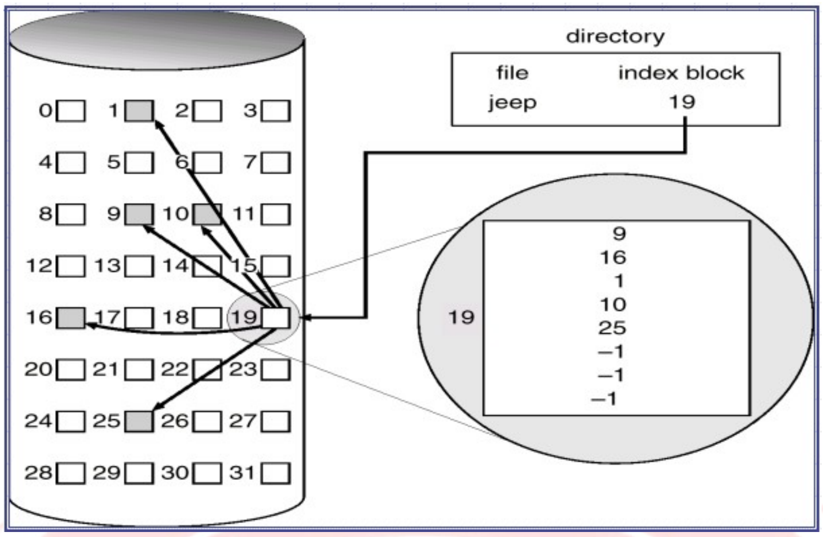
Advantages of Linked File Allocation Method :

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 ith entry in the index block points to the ith block of the file. To find and read the ith block, we use Pointer to that block from index block.

Advantages of Indexed File Allocation Method :

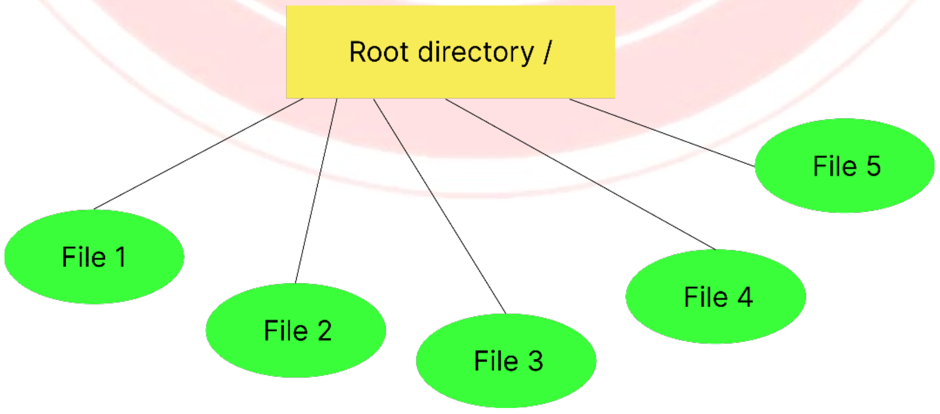
1. Does not suffer from external fragmentation.
2. Support both sequential and direct access to the file.
3. No Need for user to know size of the file in advance.
4. Entire block is available for data as no space is occupied by pointers.

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.

26. Explain Directory Structure

Ans. 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.

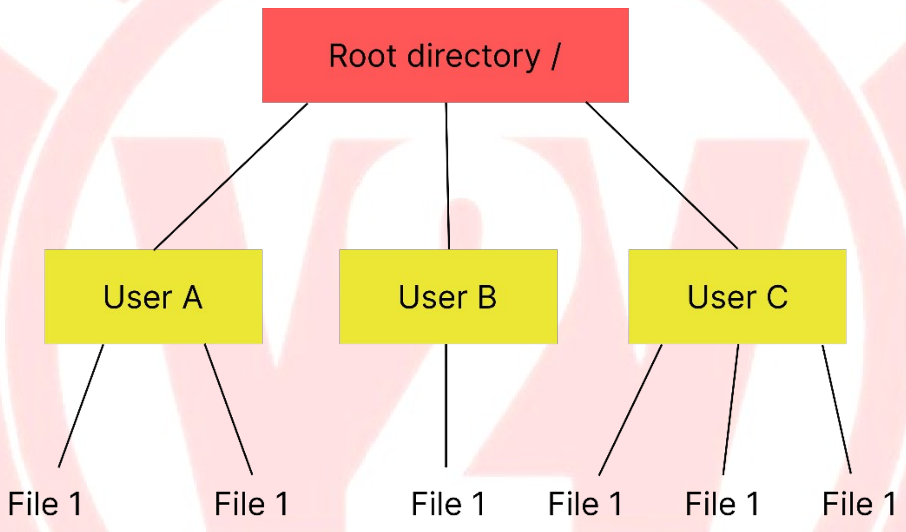
Advantages of single level directory structure

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

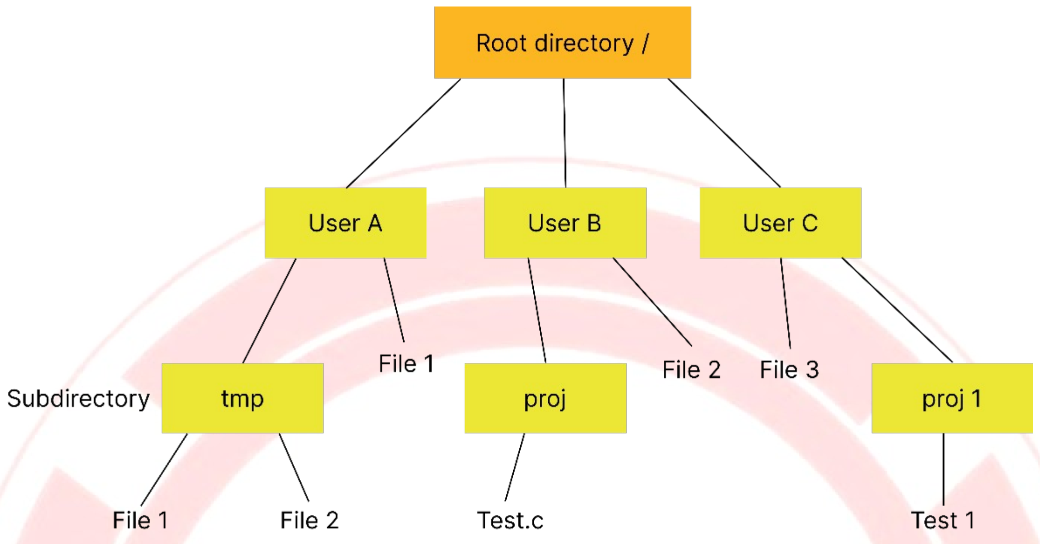
1. It solves the file name collision problem by creating own user directory.
2. This method isolates one user from another and protects user's files.
3. Different users may have files with same name.

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.

Search time may become unnecessarily long.