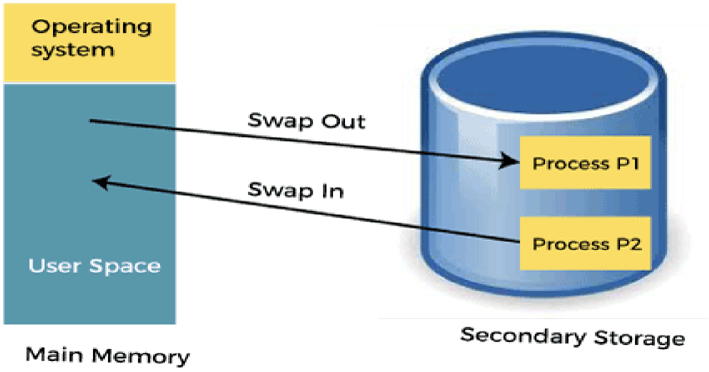
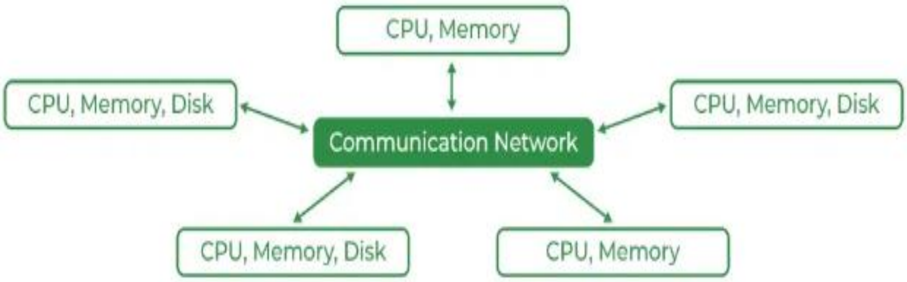
St. Peter’s Engineering College (Autonomous) Dullapally (P), Medchal, Hyderabad – 500100. QUESTION BANK					Dept.	:	AIML
					Academic Year 2024-25		
Subject Code	:	AS22-05PC03	Subject	:	OPERATING SYSTEMS		
Class/Section	:	B. Tech.	Year	:	II	Semester	: II

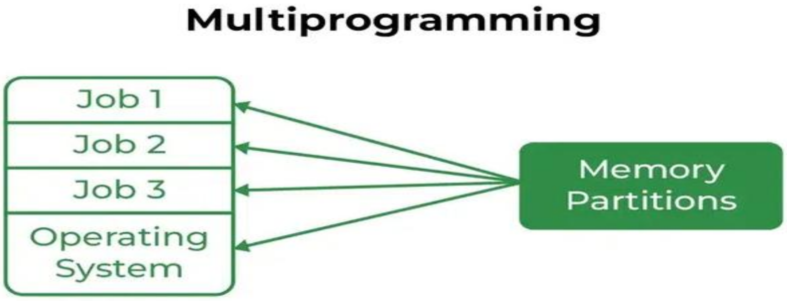
BLOOMS LEVEL					
Remember	L1	Understand	L2	Apply	L3
Analyze	L4	Evaluate	L5	Create	L6

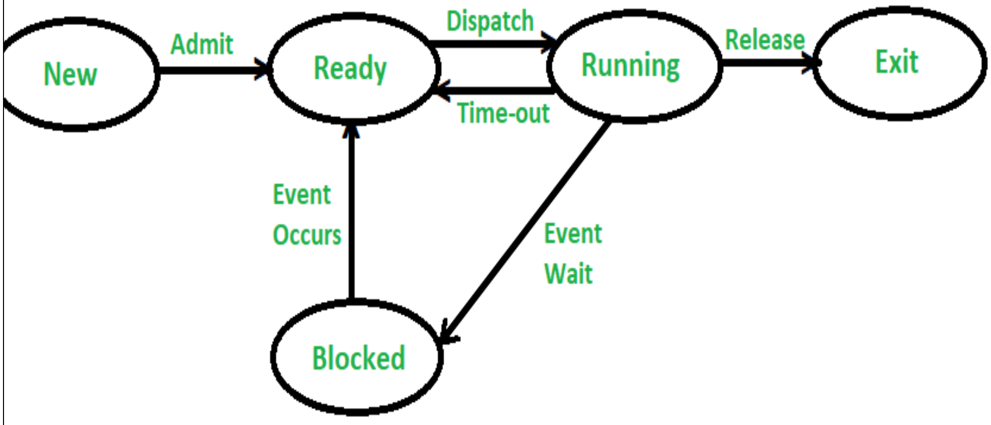
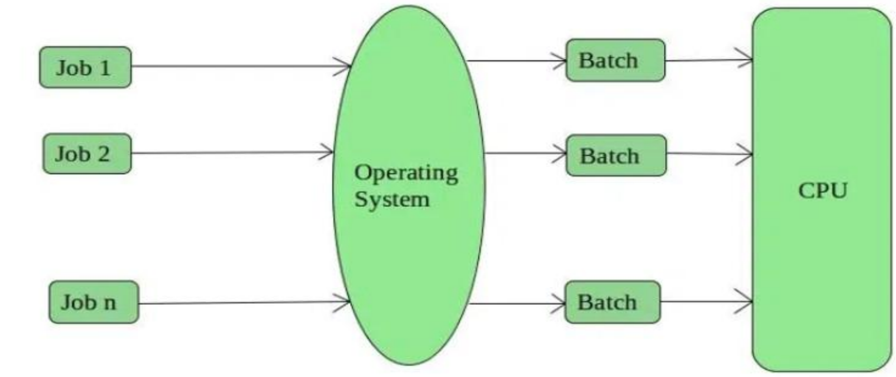
Q. No	Question (s)	Marks	BL	CO
	UNIT – I			
1	**a) Define OS**. OS stands for Operating System. It is software that manages computer hardware and provides common services for computer programs	1M	L1	C225.1
	b) List out the versions of windows OS • Windows 7 • Windows 8 • Windows 8.1 • Windows 10	1M	L1	C225.1
	c) List out the types of OS. 1. Batch Operating System 2. Multi-Programming System 3. Multi-Processing System 4. Multi-Tasking Operating System 5. Time-Sharing Operating System 6. Distributed Operating System 7. Real-Time Operating System	1M	L1	C225.1
	d)Define uni-programming system A uni-programming system is an operating system that allows only one program to run at a time. It does not support multitasking.	1M	L1	C225.1
	e)Define system call A system call is a request made by a program to the operating system for a specific service, such as reading or writing data to a file, creating a new process, or allocating memory	1M	L1	C225.1
2	**a) Explain about Real Time Operating Systems (RTOS) and its types.** They prioritize deterministic behaviour and timely response over other factors like throughput or efficiency.	3M	L4	C225.1

	RTOS types include: **a. Hard Real-Time Operating Systems:** These guarantee that tasks are completed within a specified deadline. Failure to meet deadlines can lead to system failure**.** **b. Soft Real-Time Operating Systems:** These prioritize timely execution of tasks but can tolerate occasional missed deadlines without catastrophic consequences. **c. Firm Real-Time Operating Systems:** These fall between hard and soft real-time systems, where some deadlines may be missed without system failure, but overall performance			
	b) Define a process? Explain process control block. A process is a program in execution. It consists of the program code, data, and resources like CPU time, memory, and I/O devices allocated to it during execution. The Process Control Block (PCB) is a data structure used by the operating system to manage information about each process, including its current state, program counter, CPU registers, scheduling information, and memory management data	3M	L4	C225.1
	c) Explain in detail about system calls. System calls are interfaces provided by the operating system that allow user-level processes to request services from the kernel. They provide a way for user programs to interact with hardware devices, perform I/O operations, manage files, allocate memory, and perform other privileged operations that are restricted to the operating system kernel	3M	L4	C225.1
	d) Explain in detail about process concept. The process concept involves the execution of a program in a sequential manner, where each process has its own address space, resources, and execution state. Processes can interact with each other through inter-process communication mechanisms provided by the operating system, such as pipes, sockets, and shared memory.	3M	L4	C225.1
	e) Explain in detail about thread concept in detail. A thread is a lightweight process that shares the same address space and resources as other threads within the same process. Threads allow for concurrent execution of multiple tasks within a single process, enabling parallelism and efficient resource utilization. Threads share data and resources more easily than processes but require proper synchronization to avoid race conditions and other concurrency issues.	3M	L4	C225.1
3	**a)Explain in detail about components of a computer system.** a. Central Processing Unit (CPU): This is the brain of the computer where all	5M	L4	C225.1

	calculations and data processing occur. b. Memory (RAM): Random Access Memory stores data and instructions that the CPU needs to access quickly. c. Storage (Hard Drive, SSD): This is where data is permanently stored even when the computer is turned off. d. Input devices: These allow users to interact with the computer, such as keyboards, mice, and touch screens. e. Output devices: These display information processed by the computer, such as monitors, printers, and speakers. f. Motherboard: This is the main circuit board that connects all the components of the computer together. g. Power supply: This provides electrical power to the computer system.			
	b) Explain in detail about memory management. Memory management is the process of controlling and coordinating computer memory, assigning portions called blocks to various running programs to optimize overall system performance. It involves tasks like allocating memory to processes, freeing up memory when it's no longer needed, and ensuring efficient memory usage through techniques like virtual memory, caching, and swapping i) **Main Memory Management** ❖ Main memory is a flexible and volatile type of storage device. It is a large sequence of bytes and addresses used to store volatile data. ❖ Main memory is also called Random Access Memory (RAM), which is the fastest computer storage available on PCs. ❖ It is costly and low in terms of storage as compared to secondary storage devices. **ii) Secondary Memory Management** ❖ The most important task of a computer system is to execute programs. ❖ These programs help you to access the data from the main memory during execution. ❖ This memory of the computer is very small to store all data and programs permanently. ❖ The computer system offers secondary storage to back up the main memory.	5M	L4	C225.1

	 Operating system Swap Out Swap In Process P1 Process P2 User Space Main Memory Secondary Storage			
	c)**Explain in detail about distributed systems and its advantages.** ❖ Various autonomous interconnected computers communicate with each other using a shared communication network. ❖ Independent systems possess their own memory unit and CPU. ❖ These are referred to as loosely coupled systems or distributed systems . ❖ These systems processors differ in size and function. ❖ The major benefit of working with these types of the operating system is that it is always possible that one user can access the files or software which are not actually present on his system but some other system connected within this network i.e., remote access is enabled within the devices connected in that network. Architecture of Distributed OS  CPU, Memory CPU, Memory, Disk Communication Network CPU, Memory, Disk CPU, Memory, Disk CPU, Memory **Advantages of Distributed Operating System** ❖ Failure of one will not affect the other network communication, as all systems are independent of each other. ❖ Electronic mail increases the data exchange speed. ❖ Since resources are being shared, computation is highly fast and durable. **Disadvantages of Distributed Operating System** ❖ Failure of the main network will stop the entire communication. ❖ To establish distributed systems the language is used not well-defined yet.	5M	L4	C225.1

	❖ These types of systems are not readily available as they are very expensive. Not only that the underlying software is highly complex and not understood well yet.			
	d)Explain in detail about multi-programming operating systems ❖ Multiprogramming Operating Systems can be simply illustrated as more than one program is present in the main memory and any one of them can be kept in execution. ❖ This is basically used for better utilization of resources.  Multiprogramming Job 1 Job 2 Job 3 Operating System Memory Partitions **Advantages of Multi-Programming Operating System** ❖ Multi Programming increases the Throughput of the System. ❖ It helps in reducing the response time. **Disadvantages of Multi-Programming Operating System** ❖ any facility for user interaction of system resources with the system.	5M	L4	C225.1
	e) Explain in detail about Process Management. ❖ The process management component is a procedure for managing many processes running simultaneously on the operating system. ❖ Every running software application program has one or more processes associated with them. ❖ For example, when you use a search engine like Chrome, there is a process running for that browser program. ❖ Process management keeps processes running efficiently. ❖ It also uses memory allocated to them and shutting them down when needed. ❖ The execution of a process must be sequential so, at least one instruction should be executed on behalf of the process.	5M	L4	C225.1

	 New Admit Ready Dispatch Time-out Running Release Exit Event Occurs Event Wait Blocked			
4	**a) Define OS and explain types of OS.** OS stands for Operating System. It is software that manages computer hardware and provides common services for computer programs **Types of Operating Systems** 1. Batch Operating System 2. Multi-Programming System 3. Multi-Processing System 4. Multi-Tasking Operating System 5. Time-Sharing Operating System 6. Distributed Operating System 7. Real-Time Operating System **Batch Operating System:** ❖ This type of operating system does not interact with the computer directly. ❖ There is an operator which takes similar jobs having the same requirements and groups them into batches. ❖ It is the responsibility of the operator to sort jobs with similar needs. ❖ Batch Operating System is designed to manage and execute a large number of jobs efficiently by processing them in groups.  Job 1 Job 2 Job n Operating System Batch Batch Batch CPU **Advantages of Batch Operating System:**	10M	L4	C225.1

- Multiple users can share the batch systems.
- The idle time for the batch system is very less.

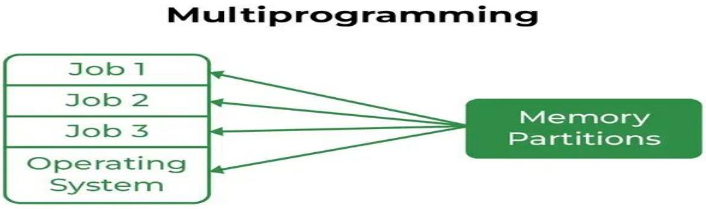
Disadvantages of Batch Operating System

- Batch systems are hard to debug.
- It is sometimes costly.

Example of Batch Operating Systems: Payroll Systems, Bank Statements, etc.

2. Multi-Programming System

- Multiprogramming Operating Systems can be simply illustrated as more than one program is present in the main memory and any one of them can be kept in execution.
- This is basically used for better utilization of resources.



Advantages of Multi-Programming Operating System

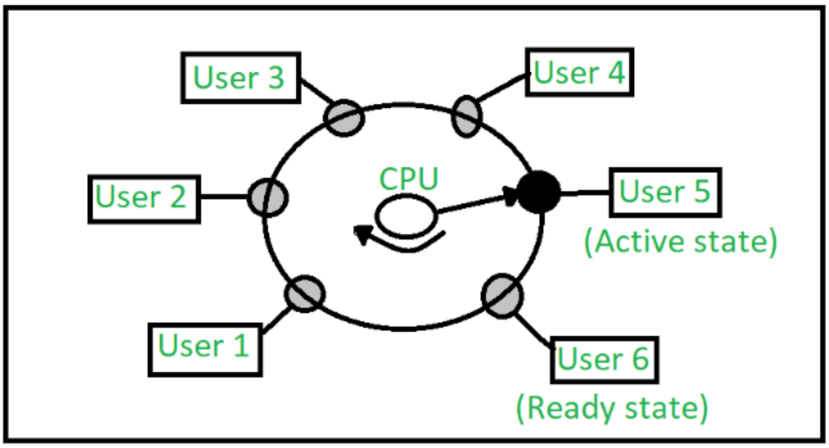
- Multi Programming increases the Throughput of the System.
- It helps in reducing the response time.

Disadvantages of Multi-Programming Operating System

- any facility for user interaction of system resources with the system.

3. Time-Sharing Operating Systems

- Each task is given some time to execute so that all the tasks work smoothly.
- Each user gets the time of the CPU as they use a single system.
- These systems are also known as Multitasking Systems.
- The task can be from a single user or different users also.
- The time that each task gets to execute is called quantum.



Advantages of Time-Sharing OS

- Each task gets an equal opportunity.

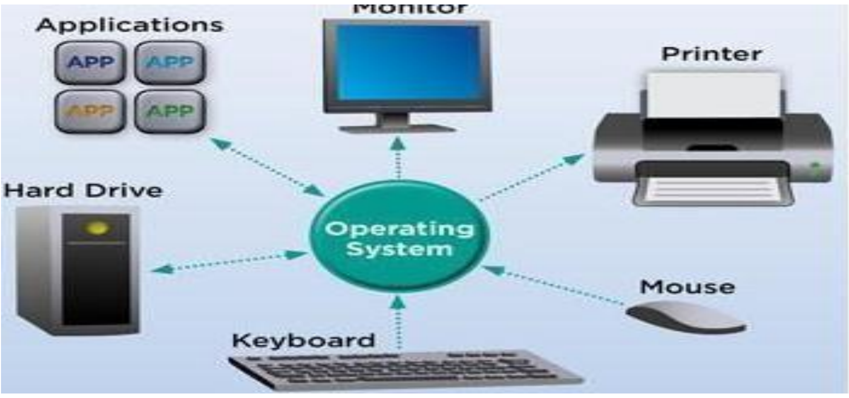
- Fewer chances of duplication of software.

Disadvantages of Time-Sharing OS

- Reliability problem.
- One must have to take care of the security and integrity of user programs and data.

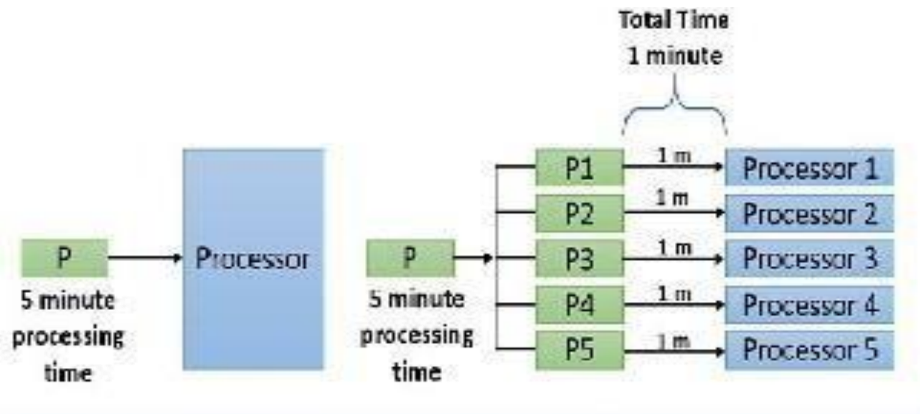
4. Personal Computers

- Personal computer operating system provides a good interface to a single user.
- Personal computer operating systems are widely used for word processing, spreadsheets and Internet access.
- Personal computer operating system is made only for personal. You can say that your laptops, computer systems, tablets etc. are your personal computers and the operating system such as windows 7, windows 10, android, etc. are your personal computer operating system.



5. Parallel Processing

- Parallel processing requires multiple processors and all the processor works simultaneously in the system.
- Here, the task is divided into subparts and these subparts are then distributed among the available processors in the system. Parallel processing completes the job on the shortest possible time.
- All the processors in the parallel processing environment should run on the same operating system.
- All processors here are tightly coupled and are packed in one casing.



Single Processor Vs Multiprocessor in Parallel processing

Advantages

- It saves time and money as many resources working together will reduce the time and potential costs.

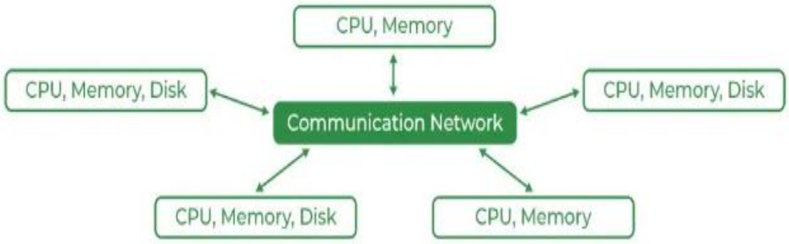
Disadvantages

- It addresses such as communication and synchronization between multiple sub-tasks and processes which is difficult to achieve.

6. Distributed Operating System

- Independent systems possess their own memory unit and CPU.
- These are referred to as loosely coupled systems or distributed systems .
- The major benefit of working with these types of the operating system is that it is always possible that one user can access the files or software which are not actually present on his system but some other system connected within this network i.e., remote access is enabled within the devices connected in that network.

Architecture of Distributed OS



Advantages of Distributed Operating System

- Failure of one will not affect the other network communication, as all systems are independent of each other.
- Electronic mail increases the data exchange speed.

Disadvantages of Distributed Operating System

- Failure of the main network will stop the entire communication.

- To establish distributed systems the language is used not well-defined yet.

7. Real-Time Operating System

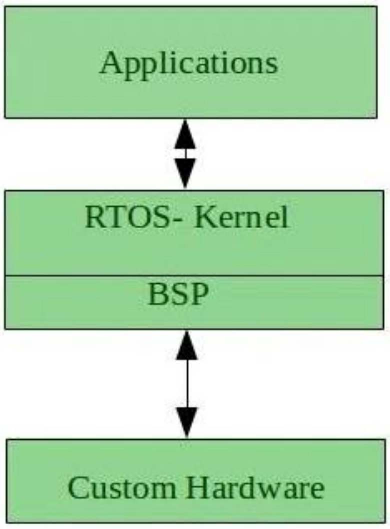
- These types of OSs serve real-time systems.
- The time interval required to process and respond to inputs is very small.
- Real-time systems are used when there are time requirements that are very strict like missile systems, air traffic control systems, robots, etc.

Types of Real-Time Operating Systems

Hard Real-Time Systems:

- Hard Real-Time OSs are meant for applications where time constraints are very strict and even the shortest possible delay is not acceptable.

Soft Real-Time Systems: These OSs are for applications where time-constraint is less strict.



Advantages of RTOS

- **Maximum Consumption**
- **Task Shifting**
- **Error Free**
- **Memory Allocation**

Disadvantages of RTOS

- **Complex Algorithms**
- **Device driver and interrupt signals**
- **Thread Priority**

	b) What are the services provided by the operating systems (OS)? ❖ An Operating System provides services to both the users and to the programs.	10M	L5	C225.1

- It provides programs an environment to execute.
- It provides users the services to execute the programs in a convenient manner.

Common services:

1. Program execution
2. I/O operations
3. File System manipulation
4. Communication
5. Error Detection
6. Resource Allocation
7. Protection

Program execution

- A process includes the complete execution context (code to execute, data to manipulate, registers, OS resources in use). Following are the major activities of an operating system with respect to program management −
  1.Loads a program into memory.
  2.Executes the program.
  3.Handles program's execution.
  4.Provides a mechanism for process synchronization.
  5.Provides a mechanism for process communication.
  6.Provides a mechanism for deadlock handling.

I/O Operation

- An I/O subsystem comprises of I/O devices and their corresponding driver software. Drivers hide the peculiarities of specific hardware devices from the users.
- I/O operation means read or write operation with any file or any specific I/O device.

File system manipulation

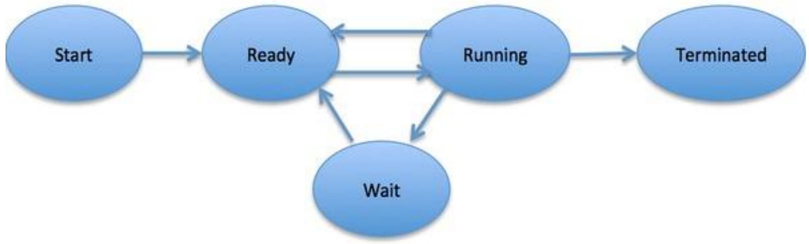
- A file represents a collection of related information. Computers can store files on the disk (secondary storage), for long-term storage purpose.
- Examples of storage media include magnetic tape, magnetic disk and optical disk drives like CD, DVD.
- A file system is normally organized into directories for easy navigation and usage.

Communication

- In case of distributed systems which are a collection of processors that do not share memory, peripheral devices, or a clock, the operating system manages communications between all the processes.
- Multiple processes communicate with one another through communication lines in the network.
- Communication may be implemented by two methods, either by Shared Memory or by Message Passing.

Error handling

- Errors can occur anytime and anywhere. An error may occur in CPU, in I/O devices or in the memory hardware. Following are the major activities of an operating system with respect to error handling −
- The OS constantly checks for possible errors.

	❖ The OS takes an appropriate action to ensure correct and consistent computing. **Resource Management** ❖ In case of multi-user or multi-tasking environment, resources such as main memory, CPU cycles and files storage are to be allocated to each user or job. Following are the major activities of an operating system with respect to resource management − ❖ The OS manages all kinds of resources using schedulers. ❖ CPU scheduling algorithms are used for better utilization of CPU. **Protection** ❖ Considering a computer system having multiple users and concurrent execution of multiple processes, the various processes must be protected from each other's activities. ❖ Protection refers to a mechanism or a way to control the access of programs, processes, or users to the resources defined by a computer system. Following are the major activities of an operating system with respect to protection 1.The OS ensures that all access to system resources is controlled. 2.The OS ensures that external I/O devices are protected from invalid access attempts. 3.The OS provides authentication features for each user by means of passwords.			
	c)**Explain process concept and scheduling** A process is defined as a sequence of instructions executed in a predefined order. **Process Life cycle:**  **1. Start** ❖ This is the initial state when a process is first started/created. **2. Ready** ❖ The process is waiting to be assigned to a processor. ❖ Ready processes are waiting to have the processor allocated to them by the operating system so that they can run.	10M	L4	C225.1

- Process may come into this state after Start state or while running it by but interrupted by the scheduler to assign CPU to some other process.

3. Running

- Once the process has been assigned to a processor by the OS scheduler, the process state is set to running and the processor executes its instructions.

4.Waiting

- Process moves into the waiting state if it needs to wait for a resource, such as waiting for user input, or waiting for a file to become available.

5.Terminated or Exit

- Once the process finishes its execution, or it is terminated by the operating system, it is moved to the terminated state where it waits to be removed from main memory.

Process Scheduling:

- The operating system can use different scheduling algorithms to schedule processes.
- Here are some commonly used timing algorithms:

1. First-Come, First-Served (FCFS):

- This is the simplest scheduling algorithm, where the process is executed on a first-come, first-served basis.
- FCFS is non-pre-emptive, which means that once a process starts executing, it continues until it is finished or waiting for I/O.

2. Shortest Job First (SJF):

- SJF is a proactive scheduling algorithm that selects the process with the shortest burst time.
- The burst time is the time a process takes to complete its execution.
- SJF minimizes the average waiting time of processes.

3. Round Robin (RR):

- Round Robin is a proactive scheduling algorithm that reserves a fixed amount of time in a round for each process.
- If a process does not complete its execution within the specified time, it is blocked and added to the end of the queue.
- RR ensures fair distribution of CPU time to all processes and avoids starvation.

4. Priority Scheduling:

- This scheduling algorithm assigns priority to each process and the process with the highest priority is executed first.
- Priority can be set based on process type, importance, or resource requirements.

5. Multilevel Queue:

- This scheduling algorithm divides the ready queue into several separate queues, each queue having a different priority.
- Processes are queued based on their priority, and each queue uses its own scheduling algorithm.

Q. No	Question (s)	Marks	BL	CO
UNIT – II				
1	a) Define non-preemptive scheduling. Non-preemptive scheduling: Non-preemptive scheduling is a scheduling technique where once a process starts executing, it cannot be interrupted until it completes or voluntarily relinquishes control of the CPU	1M	L1	C225.2
	b) Define turnaround time (TAT). Turnaround time (TAT): Turnaround time is the total time taken from the submission of a process to its completion, including both the waiting time and the execution time	1M	L1	C225.2
	c) Define mutual-exclusion. Mutual-exclusion: Mutual-exclusion refers to a synchronization technique that ensures that only one process at a time can access a shared resource or a critical section, preventing simultaneous access by multiple processes.	1M	L1	C225.2
	d) Define deadlock. Deadlock is a situation in which two or more processes are unable to proceed because each is waiting for another to release a resource.	1M	L1	C225.2
	e) Define completion time. Completion time is the time at which a process finishes its execution and exits the system.	1M	L1	C225.2
2	**a) Discuss briefly about CPU scheduling.** **CPU Scheduling**: CPU scheduling involves the process of selecting which process/thread should be executed next by the CPU. It aims to maximize CPU utilization, throughput, and minimize response time and waiting time for processes in a system.	3M	L2	C225.2
	b) Discuss about scheduling criteria in CPU scheduling. Scheduling criteria in CPU scheduling include factors such as CPU utilization, throughput, turnaround time, waiting time, response time, fairness, and prioritization. These criteria guide the selection of scheduling algorithms to achieve desired performance goals.	3M	L2	C225.2
	c) Discuss about multi-processing systems. Multi-processing systems involve the simultaneous execution of multiple processes or threads by utilizing multiple CPUs or CPU cores. This allows for increased throughput, better resource utilization, and improved system responsiveness by distributing workload across multiple processors.	3M	L2	C225.2
	d) Discuss about deadlock. Deadlock occurs in a system when two or more processes are unable to proceed because each is waiting for the other to release a resource, resulting in a cyclic dependency. Deadlocks can lead	3M	L2	C225.2

<table>
<tr><td></td><td>to system halts and resource starvation.</td><td></td><td></td><td></td></tr>
<tr><td></td><td><b>e) Discuss about deadlock with an example.</b>
Consider a scenario where Process A holds Resource X and requests Resource Y, while Process B holds Resource Y and requests Resource X. Both processes cannot proceed as they are waiting for the resources held by the other. This results in a deadlock situation where neither process can release its held resource, leading to a system deadlock.</td><td>3M</td><td>L2</td><td>C225.2</td></tr>
<tr><td>3</td><td><b>a) Explain detail about any CPU scheduling technique.</b>
CPU scheduling is the process of determining which processes in the ready queue should be allocated the CPU for execution. One common technique is "Round Robin" scheduling. In Round Robin, each process is assigned a fixed time slice or quantum to execute. When a process's time slice expires, it is moved to the end of the ready queue, allowing other processes to have a turn. This technique ensures fairness as each process gets a chance to execute, preventing starvation. However, it may not be efficient for tasks with varying execution times or I/O-bound processes
<b>Example:</b>
Following is the example of round robin scheduling.
<table>
<tr><th>Process Id</th><th>Arrival Time</th><th>Burst Time</th></tr>
<tr><td>P1</td><td>0</td><td>10</td></tr>
<tr><td>P2</td><td>1</td><td>8</td></tr>
<tr><td>P3</td><td>2</td><td>7</td></tr>
</table>
<b>Time Slot</b> is 5 Sec.
First <b>P1</b> is executed for 5 seconds, left burst time is 5 sec
Then <b>P2</b> is executed for 5 seconds, left burst time is 3 sec
Then <b>P3</b> is executed for 5 seconds, left burst time is 2 sec
Then <b>P1</b> is executed for 5 seconds, execution of <b>P1</b> is completed.
Then <b>P2</b> is executed for 3 seconds, execution of <b>P2</b> is completed.
Then <b>P1</b> is executed for 2 sec, execution <b>P3</b> is completed.
Execution of all processes completed
<table>
<tr><th>Process Id</th><th>Burst Time</th><th>Wait Time</th><th>Turn Around Time</th></tr>
<tr><td>P1</td><td>10</td><td>20</td><td>10</td></tr>
<tr><td>P2</td><td>8</td><td>22</td><td>14</td></tr>
<tr><td>P3</td><td>7</td><td>23</td><td>16</td></tr>
</table>
<b>Average Waiting Time</b> = (20 + 22 + 23)/3 = 65/3 = 21.666666
<b>Average Turnaround Time</b> = (10 + 14 + 16)/3 = 40/3 = 13.333333
<b>Advantages:</b></td><td>5M</td><td>L4</td><td>C225.2</td></tr>
</table>

	• There is no starvation of resources as each process gets equal share. • Every Process gets equal allocation of CPU. • Increases Performance time in terms of response time.			
	b) Explain in detail about deadlock detection. Deadlock detection is a mechanism used to identify if a system has entered a deadlock state, where each process is waiting for a resource held by another process, creating a cycle of dependencies. To detect deadlock, various algorithms like resource allocation graph and wait-for graph are employed. These algorithms analyze the resource allocation and wait-for relationships among processes and resources to identify cycles indicative of deadlock. Once a deadlock is detected, systems can take actions such as process termination or resource preemption to resolve it.	5M	L4	C225.2
	c) Explain in detail about necessary condition for deadlock. In a deadlock, processes never finish executing, and system resources are tied up, preventing other jobs from starting **Necessary Conditions:** There are four conditions that must be met in order to achieve deadlock as follows. **Mutual Exclusion –** At least one resource must be kept in a non-shareable state; if another process requests it, it must wait for it to be released. **Hold and Wait –** A process must hold at least one resource while also waiting for at least one resource that another process is currently holding. **No preemption –** Once a process holds a resource (i.e. after its request is granted), that resource cannot be taken away from that process until the process voluntarily releases it. **Circular Wait –** There must be a set of processes P0, P1, P2,…, PN such that every P[I] is waiting for P[(I + 1) percent (N + 1)]. (It is important to note that this condition implies the hold-and-wait condition, but dealing with the four conditions is easier if they are considered separately).	5M	L4	C225.2
	d) Explain in detail about a deadlock prevention. Deadlocks can be avoided by avoiding at least one of the four necessary conditions: as follows. **Condition-1:** **Mutual Exclusion:** ❖ Read-only files, for example, do not cause deadlocks. ❖ Unfortunately, some resources, such as printers and tape drives, require a single process to have exclusive access to them.	5M	L4	C225.2

	Condition-2 : **Hold and Wait:** ❖ To avoid this condition, processes must be prevented from holding one or more resources while also waiting for one or more others. There are a few possibilities here: ❖ Make it a requirement that all processes request all resources at the same time. This can be a waste of system resources if a process requires one resource early in its execution but does not require another until much later. **Condition-3 :** **No Preemption :** When possible, preemption of process resource allocations can help to avoid deadlocks. ❖ One approach is that if a process is forced to wait when requesting a new resource, all other resources previously held by this process are implicitly released (preempted), forcing this process to re-acquire the old resources alongside the new resources in a single request, as discussed previously. ❖ Another approach is that when a resource is requested, and it is not available, the system looks to see what other processes are currently using those resources and are themselves blocked while waiting for another resource. **Condition-4 :** **Circular Wait :** ❖ To avoid circular waits, number all resources and insist that processes request resources is strictly increasing (or decreasing) order. ❖ To put it another way, before requesting resource Rj, a process must first release all Ri such that I >= j. ❖ The relative ordering of the various resources is a significant challenge in this scheme.			
	e) Discuss about round-robin scheduling algorithm. **Round Robin Scheduling Algorithm:** Step 1: Start the Program. Step 2: Input the number of processes. Step 3: Input the burst time and arrival time of each process and the limit of the time slot. Step 4: Push all processes into the ready queue according to their arrival time. Then execute each process upto time slot and push left over process in queue again for execution. Step 5: After a process is completely executed, print its turn around time and waiting time.	5M	L2	C225.2

Example:

Following is the example of round robin scheduling.

Process Id	Arrival Time	Burst Time
P1	0	10
P2	1	8
P3	2	7

Time Slot is 5 Sec.

First **P1** is executed for 5 seconds, left burst time is 5 sec
Then **P2** is executed for 5 seconds, left burst time is 3 sec
Then **P3** is executed for 5 seconds, left burst time is 2 sec
Then **P1** is executed for 5 seconds, execution of **P1** is completed.
Then **P2** is executed for 3 seconds, execution of **P2** is completed.
Then **P1** is executed for 2 sec, execution **P3** is completed.
Execution of all processes completed

Process Id	Burst Time	Wait Time	Turn Around Time
P1	10	20	10
P2	8	22	14
P3	7	23	16

Average Waiting Time = (20 + 22 + 23)/3 = 65/3 = 21.666666

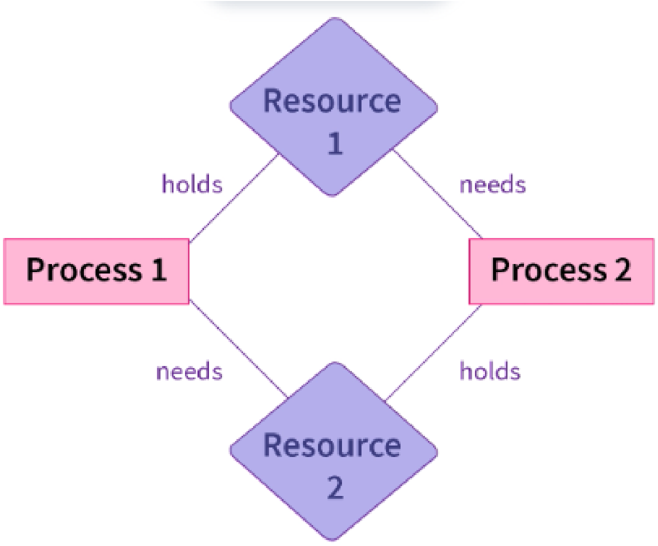
Average Turnaround Time = (10 + 14 + 16)/3 = 40/3 = 13.333333

Advantages:

- There is no starvation of resources as each process gets equal share.
- Every Process gets equal allocation of CPU.
- Increases Performance time in terms of response time.

4	a) Explain in detail about banker's algorithm. ❖ This is a specific algorithm that is used to avoid deadlocks in systems that have a fixed number of resources. ❖ It involves assigning a "safety level" to each process, based on the resources it has requested and the resources it currently holds. ❖ If a process requests a resource and its safety level becomes negative, the resource is not granted, and the process must wait. ❖ This helps to prevent deadlocks by ensuring that processes only request resources that they can safely acquire.	10M	L4	C225.2

	Banker’s algorithm comprises of two algorithms: ❖ Safety algorithm ❖ Resource request algorithm **Safety Algorithm** A safety algorithm is an algorithm used to find whether or not a system is in its safe state. The algorithm is as follows: 1. Let Work and Finish be vectors of length m and n, respectively. Initially, Work = Available Finish[i] =false for i = 0, 1, ... , n - 1. This means, initially, no process has finished and the number of available resources is represented by the Available array. 2. Find an index i such that both Finish[i] ==false Need(i) <= Work If there is no such i present, then proceed to step 4. 3. Perform the following: Work = Work + Allocation(i) Finish[i] = true Go to step 2. When an unfinished process is found, then the resources are allocated and the process is marked finished. And then, the loop is repeated to check the same for all other processes. 4. If Finish[i] == true for all i, then the system is in a safe state. That means if all processes are finished, then the system is in safe state. **Resource Request Algorithm** 1. If Request(i) <= Need(i), then go to step 2;else raise an error condition, since the process has exceeded its maximum claim. 2. If Request(i) <= Available(i) then go to step 3; else Pi must have to wait as resources are not available. 3. Now we will assume that resources are assigned to process Pi and thus perform the following steps: Available = Available - Request(i); Allocation(i) = Allocation(i) + Request(i); Need(i) = Need(i) - Request(i); If the resulting resource allocation state comes out to be safe, then the transaction is completed and, process Pi is allocated its resources. But in this case, if the new state is unsafe, then Pi waits for Requesti, and the old resource-allocation state is restored.			
	b) Explain in detail about deadlock. ❖ A deadlock occurs when a set of processes is stalled because each process is holding a resource and waiting for another process to acquire another resource.	10M	L4	C225.2

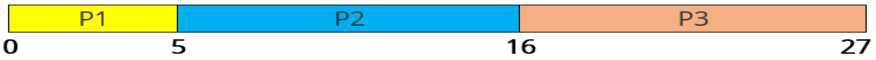
	❖ In the below figure, there are two processes and two resources. Process 1 holds "Resource 1" and needs "Resource 2" while Process 2 holds "Resource 2" and requires "Resource 1". ❖ This creates a situation of deadlock because none of the two processes can be executed. Since the resources are non-shareable they can only be used by one process at a time(Mutual Exclusion). ❖ Each process is holding a resource and waiting for the other process the release the resource it requires. None of the two processes releases their resources before their execution and this creates a circular wait.  Resource 1 holds needs Process 1 Process 2 needs holds Resource 2			
	c) Explain in detail about any two CPU scheduling techniques. **1)FCFS Scheduling Algorithm:** ❖ The CPU scheduling algorithm First Come, First Served (**FCFS**), also known as First In, First Out (**FIFO**), allocates the CPU to the processes in the order they are queued in the ready queue. ❖ **FCFS** uses non-preemptive scheduling, which means that once a CPU has been assigned to a process, it stays assigned to that process until it is either terminated or may be interrupted by an I/O interrupt. **Problem Solution** 1. Enter all the processes and their burst time. 2. Find waiting time, **WT** of all the processes. 3. For the 1st process, **WT = 0**. 4. For all the next processes i, **WT[i] = BT[i-1] + WT[i-1]**. 5. Calculate Turnaround **time = WT + BT** for all the processes. 6. Calculate **average waiting time** = total waiting time/no. of	10M	L4	C225.2

processes.
7. Calculate **average turnaround time** = total turnaround time/no. of processes.

Example:

Process	**Arrival Time**	**Burst Time**
P1	0	5
P2	0	11
P3	0	11

Gantt Chart:



Waiting Time: Time Difference between turnaround time and burst time.

Waiting Time = Turnaround Time – Burst Time

P1 waiting time: 0
P2 waiting time: 5
P3 waiting time: 16

Average Waiting Time = (0 + 5 + 16)/3 = 21/3 = 7

Turnaround Time: Difference between completion time and arrival time.

Turnaround Time = Completion Time – Arrival Time

P1 turnaround time: 5-0 = 5
P2 turnaround time: 16-0 = 16
P3 turnaround time: 27-0 = 27
Average Turnaround Time = (5+16+27)/3 = 16

Round Robin algorithm:

❖ **Round Robin Scheduling** is a CPU scheduling algorithm in which each process is executed for a fixed time slot. Since the resources are snatched after the time slot, round robin is preemptive.

Step 1: Start the Program.
Step 2: Input the number of processes.
Step 3: Input the burst time and arrival time of each process and the limit of the time slot.
Step 4: Push all processes into the ready queue according to their arrival time. Then execute each process upto time slot and push left over process in queue again for execution.
Step 5: After a process is completely executed, print its turn around time and waiting time.

Example:

Following is the example of round robin scheduling.

Process Id	Arrival Time	Burst Time
P1	0	10
P2	1	8
P3	2	7

Time Slot is 5 Sec.

First **P1** is executed for 5 seconds, left burst time is 5 sec
Then **P2** is executed for 5 seconds, left burst time is 3 sec
Then **P3** is executed for 5 seconds, left burst time is 2 sec
Then **P1** is executed for 5 seconds, execution of **P1** is completed.
Then **P2** is executed for 3 seconds, execution of **P2** is completed.
Then **P1** is executed for 2 sec, execution **P3** is completed.
Execution of all processes completed

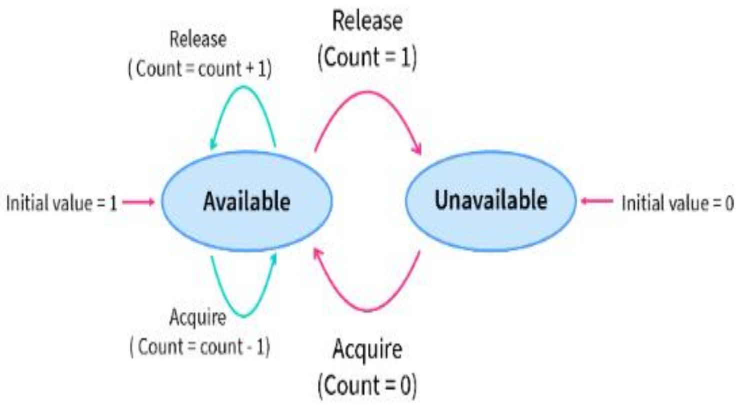
Process Id	Burst Time	Wait Time	Turn Around Time
P1	10	20	10
P2	8	22	14
P3	7	23	16

Average Waiting Time = (20 + 22 + 23)/3 = 65/3 = 21.666666
Average Turnaround Time = (10 + 14 + 16)/3 = 40/3 = 13.333333

Advantages:

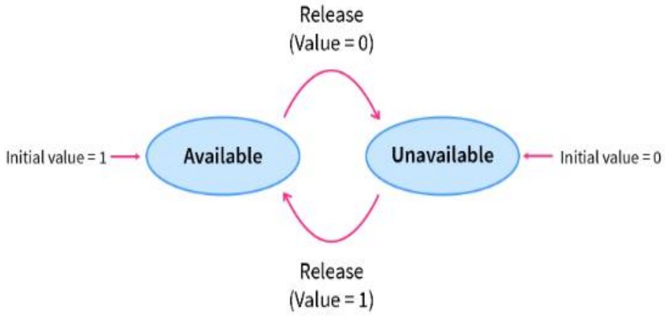
- There is no starvation of resources as each process gets equal share.
- Every Process gets equal allocation of CPU.
- Increases Performance time in terms of response time.

Q. No	Question (s)	Marks	BL	CO
UNIT – III				
1	**a) What is the critical problem in process synchronization?** The critical problem in process synchronization is ensuring that multiple processes or threads coordinate their access to shared resources in a way that prevents conflicts and maintains consistency	1M	L5	C225.3
	b) Define semaphore in the context of process synchronization. Semaphore: in the context of process synchronization, is a synchronization primitive used to control access to shared resources by multiple processes or threads. It allows processes to signal each other to indicate the availability of resources.	1M	L1	C225.3
	c) What is the primary purpose of using monitors in process Synchronization? The primary purpose of using monitors in process synchronization is to provide a high-level abstraction for managing shared resources and coordinating access to them.	1M	L5	C225.3
	d) Define IPC in context of computer system. Inter-Process Communication, refers to the mechanisms and techniques used by operating systems to allow processes to communicate and synchronize with each other. This includes methods such as message passing, shared memory, and synchronization primitives	1M	L1	C225.3
	e) What are FIFO's in the context of IPC. FIFOs, in the context of IPC, refer to First-In-First-Out communication channels used for inter-process communication. FIFOs provide a way for processes to communicate by writing data to and reading data from a shared buffer in a first-in-first-out manner.	1M	L5	C225.3
2	a) Distinguish between binary and counting semaphores. **Types of Semaphores** There are two main types of semaphores i.e. counting semaphores and binary semaphores. Details about these are given as follows − **Counting Semaphores** Counting semaphores can be used to control access to a given resource consisting of a finite number of instances. The semaphore is initialized to the number of resources available. Its value can range over an unrestricted domain.	3M	L2	C225.3



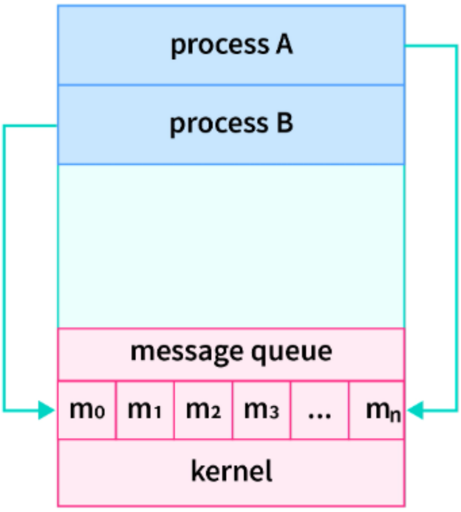
Binary Semaphores

This is also known as a mutex lock, as they are locks that provide mutual exclusion. It can have only two values – 0 and 1. Its value is initialized to 1. It is used to implement the solution of critical section problems with multiple processes and a single resource.



	b) Discuss the role of monitors in providing a higher level of Synchronization abstraction. Monitors play a crucial role in providing a higher level of **synchronization abstraction** in concurrent programming. They are a programming construct that helps manage access to shared resources in a safe and structured way. Here's a breakdown of their role and importance: **Definition of a Monitor** A **monitor** is a synchronization construct that combines: • **Mutual exclusion** (only one thread can execute inside the monitor at a time)	3M	L2	C225.3

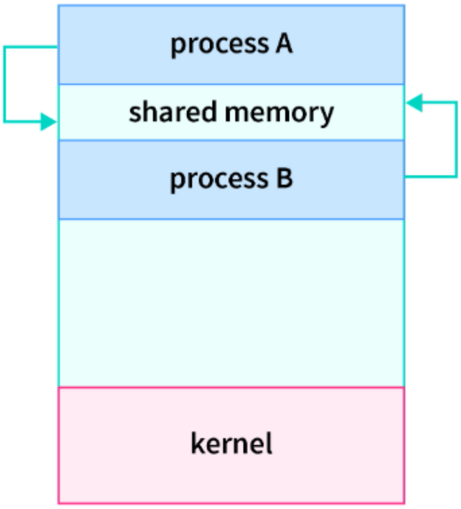
	• **Condition variables** (to allow threads to wait and be notified when a condition becomes true) Monitors are typically provided as part of the programming language (e.g., Java, Python), which enforces access rules and makes the design of concurrent systems safer and less error-prone. **Advantages:** • **Encapsulation**: The shared resource and the code that manipulates it are encapsulated within the monitor. • **Automatic mutual exclusion**: Only one thread can be active in the monitor at any time, preventing race conditions. • **Condition synchronization**: Threads can wait for specific conditions (via `wait()`, `signal()`, or `notify()`), allowing fine-grained control over execution flow. • **Simplified logic**: Developers deal with high-level constructs instead of manually managing locks.			
	c) Explain message queues and shared memory as IPC mechanisms. **a) Message Queues:** • Message queues are a more advanced form of pipes. • They allow processes to send messages to each other, and they can be used to communicate between processes that are not running on the same machine. • Message queues are a good choice for communication between processes that need to be decoupled from each other. • In Message Queue IPC, each message has a priority associated with it, and messages are retrieved from the queue in order of their priority. • This allows processes to prioritize the delivery of important messages and ensures that critical messages are not blocked by less important messages in the queue. • Message Queue IPC provides a flexible and scalable method of communication between processes, as messages can be sent and received asynchronously, allowing processes to continue executing while they wait for messages to arrive.	3M	L4	C225.3



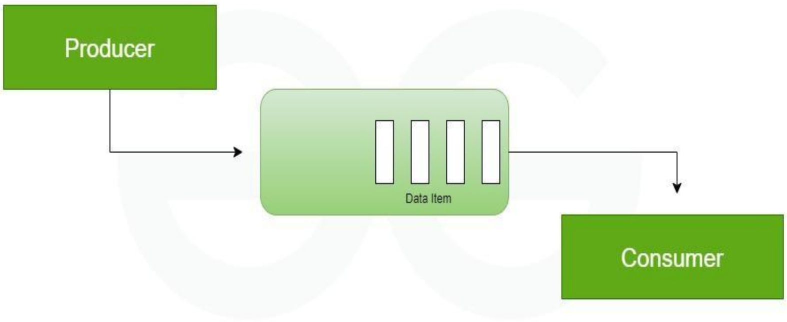
b) Shared Memory

- Shared memory is a region of memory that is accessible to multiple processes. This allows processes to communicate with each other by reading and writing data from the shared memory region.
- Shared memory is a fast and efficient way for processes to communicate, but it can be difficult to use if the processes are not carefully synchronized.
- There are two main types of shared memory:
    - **Anonymous shared memory:** Anonymous shared memory is not associated with any file or other system object. It is created by the operating system and is only accessible to the processes that created it.
    - **Mapped shared memory:** Mapped shared memory is associated with a file or other system object. It is created by mapping a file into the address space of one or more processes.

Multiple processes can access a common shared memory. Multiple processes communicate by shared memory, where one process makes changes at a time and then others view the change. Shared memory does not use kernel.

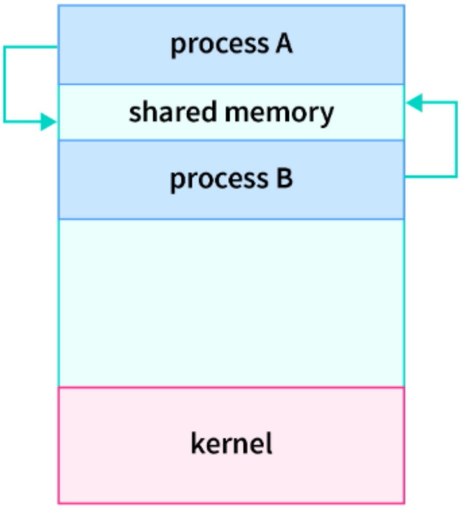
				
	d) Discuss the advantages of using pipes in IPC. 1. Simplified Data Flow: Pipes provide a straightforward way to connect the output of one process to the input of another, facilitating data transfer between them. 2. Streamlined Complex Tasks: They are excellent for tasks where the output of one process needs to be processed by another sequentially. For example, you can use pipes to chain multiple commands in a Unix-like shell to process data efficiently. 3. Efficient Data Transfer: Pipes are a relatively fast and efficient method of IPC, especially for smaller amounts of data. 4. Security: Unlike shared memory, pipes do not pose the same security risks as they are not directly exposed to other processes. 5. Easy to Use: Pipes are a straightforward and easy-to-implement IPC mechanism, making them a popular choice for many applications. 6. No Synchronization Required (for unidirectional flow): Since pipes are unidirectional, they do not require complex synchronization mechanisms for data transfer. 7. No Shared Resource Conflicts: Because pipes are a dedicated channel for a specific pair of processes, there's no chance of conflicts with other processes trying to access the same resource. 8. Ideal for Chaining Commands: Pipes are particularly well-suited for Unix-like shells and remote terminals, allowing users to chain multiple commands together and store the output results along the way.	3M	L2	C225.3
	e) Explain the concept of IPC using sockets between process on different systems. One of the ways to manage interprocess communication is by using sockets. They provide point-to-point, two-way communication between two processes. Sockets are an endpoint of communication and	3M	L4	C225.3

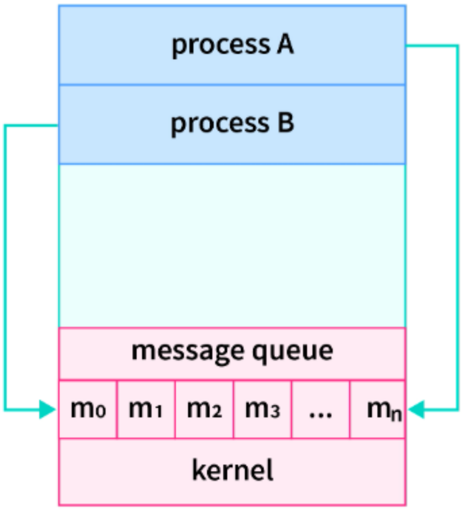
	a name can be bound to them. A socket can be associated with one or more processes. Types of Sockets The different types of sockets are given as follows − Sequential Packet Socket: This type of socket provides a reliable connection for datagrams whose maximum length is fixed This connection is two-way as well as sequenced. Datagram Socket: A two-way flow of messages is supported by the datagram socket. The receiver in a datagram socket may receive messages in a different order than that in which they were sent. The operation of datagram sockets is similar to that of passing letters from the source to the destination through a mail. Stream Socket: Stream sockets operate like a telephone conversation and provide a two-way and reliable flow of data with no record boundaries. This data flow is also sequenced and unduplicated. Raw Socket: The underlying communication protocols can be accessed using the raw sockets.			
3	**a) Explain the producer-consumer problem and how it can be Solved using synchronization primitives.** **Producer-consumer problem** is also called as **Bounded Buffer Problem** ♦ Because the buffer pool has a maximum size, this problem is often called the Bounded buffer problem. ♦ This problem is generalized in terms of the Producer Consumer problem, where a finite buffer pool is used to exchange messages between producer and consumer processes. ♦ Solution to this problem is, creating two counting semaphores "full" and "empty" to keep track of the current number of full and empty buffers respectively. ♦ In this Producers mainly produces a product and consumers consume the product, but both can use of one of the containers each time. ♦ The main complexity of this problem is that we must have to maintain the count for both empty and full containers that are available.	5M	L4	C225.3

	b) Explain in detail about semaphores in process synchronization.
	Semaphores are a synchronization primitive used to control access to shared resources in concurrent programming. They consist of a counter and two operations: wait (P) and signal (V).
	Wait(s)
	{
	While(S<=0); // no operations
	S--;
	}
	Signal(S)
	{
	S++;
	}
	Mutual Exclusion implementation
	do
	{
	Wait(mutex);
	Critical section
	Signal(mutex);
	Remainder section
	}while(1);
	The wait operation decrements the semaphore counter and blocks if the counter becomes negative, while the signal operation increments the counter and unblocks waiting processes if necessary.
	Types of semaphores:
	1) Binary semaphore: Value may be either ‘0’ or ‘1’.
	2) Counting semaphore: for counting the availability of resources in the computer (example Keyboard, mouse, printer, scanner etc)
	Semaphores can be used to implement critical sections, mutual exclusion, and synchronization between processes or threads.
		5M	L4	C225.3

	c) Discuss the role of shared memory in inter-process communication and its challenges in ensuring data integrity	5M	L2	C225.3

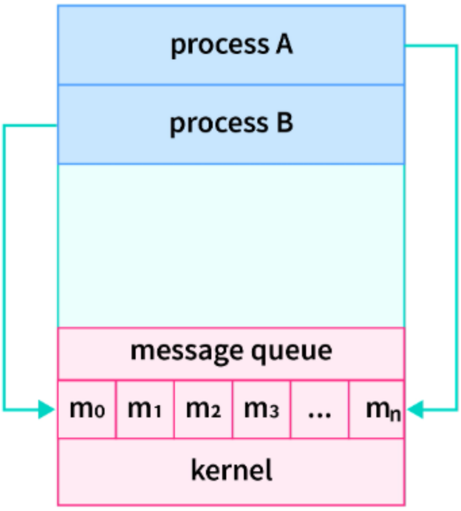
	• Shared memory is a region of memory that is accessible to multiple processes. This allows processes to communicate with each other by reading and writing data from the shared memory region. • Shared memory is a fast and efficient way for processes to communicate, but it can be difficult to use if the processes are not carefully synchronized. • There are two main types of shared memory: ○ **Anonymous shared memory:** Anonymous shared memory is not associated with any file or other system object. It is created by the operating system and is only accessible to the processes that created it. ○ **Mapped shared memory:** Mapped shared memory is associated with a file or other system object. It is created by mapping a file into the address space of one or more processes. Multiple processes can access a common shared memory. Multiple processes communicate by shared memory, where one process makes changes at a time and then others view the change. Shared memory does not use kernel.  			
	d) **Explain the implementation and advantages of message queues in inter-process communication**. • Message queues are a more advanced form of pipes. • They allow processes to send messages to each other, and they can be used to communicate between processes that are not running on the same machine. • Message queues are a good choice for communication between processes that need to be decoupled from each other. • In Message Queue IPC, each message has a priority associated with it, and messages are retrieved from the queue in order of their priority.	5M	L4	C225.3

	- This allows processes to prioritize the delivery of important messages and ensures that critical messages are not blocked by less important messages in the queue. - Message Queue IPC provides a flexible and scalable method of communication between processes, as messages can be sent and received asynchronously, allowing processes to continue executing while they wait for messages to arrive.  process A process B message queue m0 m1 m2 m3 ... mn kernel **Advantages of Message Queues** a) Asynchronous Communication b) Improved Scalability c) Increased Reliability and Fault Tolerance			
	e) Explain in detail about FIFO's and message queues as IPC Mechanisms. **FIFO:** - FIFO (First In First Out) is a type of message queue that guarantees that messages are delivered in the order they were sent. - It involves the use of a FIFO buffer, which acts as a queue for exchanging data between processes. - Used to communicate between two processes that are not related. - In the FIFO method, one process writes data to the FIFO buffer, and another process reads the data from the buffer in the order in which it was written. - Full-duplex method - Process P1 is able to communicate with Process P2, and vice versa.	5M	L4	C225.3

- The main advantage of the FIFO method is that it provides a simple way for processes to communicate, as data is exchanged sequentially, and there is no need for processes to coordinate their access to the FIFO buffer.
- However, the FIFO method can also introduce limitations, as it may result in slow performance if the buffer becomes full and data must be written to the disk, or if the buffer becomes empty and data must be read from the disk.

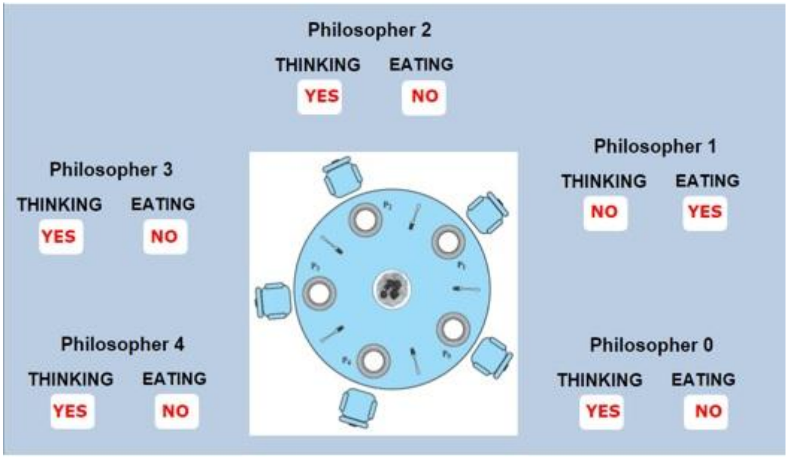
Message Queues:

- Message queues are a more advanced form of pipes.
- They allow processes to send messages to each other, and they can be used to communicate between processes that are not running on the same machine.
- Message queues are a good choice for communication between processes that need to be decoupled from each other.
- In Message Queue IPC, each message has a priority associated with it, and messages are retrieved from the queue in order of their priority.
- This allows processes to prioritize the delivery of important messages and ensures that critical messages are not blocked by less important messages in the queue.
- Message Queue IPC provides a flexible and scalable method of communication between processes, as messages can be sent and received asynchronously, allowing processes to continue executing while they wait for messages to arrive.



4	**a) Explain the classical problems of synchronization, such as the dining philosophers or producer-consumer.** **Dining Philosophers Problem**	10M	L4	C225.3

- The dining philosopher's problem involves the allocation of limited resources to a group of processes in a deadlock-free and starvation-free manner.
- There are five philosophers sitting around a table, in which there are five chopsticks/forks kept beside them and a bowl of rice in the centre, When a philosopher wants to eat, he uses two chopsticks - one from their left and one from their right. When a philosopher wants to think, he keeps down both chopsticks at their original place.



For example, let's consider P0, P1, P2, P3, and P4 as the philosophers or processes and C0, C1, C2, C3, and C4 as the 5 chopsticks or resources between each philosopher. Now if P0 wants to eat, both resources/chopsticks C0 and C1 must be free, which would leave P1 and P4 void of the resource and the process wouldn't be executed, which indicates there are limited resources(C0,C1..) for multiple processes(P0, P1..), and this problem is known as the Dining Philosopher Problem.

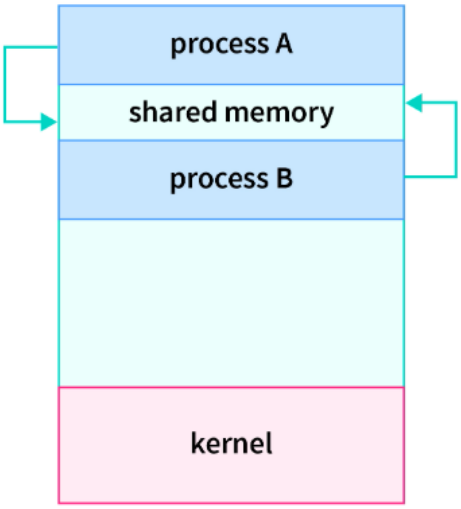
	b) Explain the benefits and drawbacks of using monitors as a synchronization mechanism compared to semaphores.	10M	L4	C225.3

S. No.	**Semaphore**	**Monitor**
1.	It is an integer variable.	It is an abstract data type.
2.	The value of this integer variable tells about the number of shared resources that are available in the system.	It contains shared variables.
3.	When any process has access to the shared resources, it performs the ?wait' operation	It also contains a set of procedures that operate upon the shared variable.

	(using wait method) on the semaphore.	
4.	When a process releases the shared resources, it performs the ?signal' operation (using signal method) on the semaphore.	When a process wishes to access the shared variables in the monitor, it has to do so using procedures.
5.	It doesn't have condition variables.	It has condition variables.

semaphores and monitors are types process synchronization tools in operating systems, however they are quite different from each other, as descried in the above table. The most significant difference that you should note here is that a semaphore is an integer variable that indicates the number of resources available in the system, whereas a monitor is an abstract data type that allows only one process to be executed at a time.

	f) Discuss the role of shared memory in inter-process communication and its challenges in ensuring data integrity • Shared memory is a region of memory that is accessible to multiple processes. This allows processes to communicate with each other by reading and writing data from the shared memory region. • Shared memory is a fast and efficient way for processes to communicate, but it can be difficult to use if the processes are not carefully synchronized. • There are two main types of shared memory: o **Anonymous shared memory:** Anonymous shared memory is not associated with any file or other system object. It is created by the operating system and is only accessible to the processes that created it. o **Mapped shared memory:** Mapped shared memory is associated with a file or other system object. It is created by mapping a file into the address space of one or more processes. Multiple processes can access a common shared memory. Multiple processes communicate by shared memory, where one process makes changes at a time and then others view the change. Shared memory does not use kernel.	10M	L2	C225.3

Q. No	Question (s)	Marks	BL	CO
UNIT – IV				
1	a) **Define segmentation?** Dividing a process's memory into logical segments to manage and protect different parts of the program	1M	L1	C225.4
	b) **Define swapping** Moving entire processes in and out of main memory to optimize resource utilization.	1M	L1	C225.4
	c) **Define paging?** Dividing a process's physical memory into fixed-size blocks called pages, allowing for efficient memory management and address translation	1M	L1	C225.4

	d) Define logical address space. The virtual memory space seen by a process, which may not correspond directly to physical memory addresses	1M	L 1	C 2 2 5. 4
	e) Define page replacement. The process of selecting a page in memory to be replaced when additional pages need to be brought in, often based on specific algorithms like LRU (Least Recently Used).	1M	L 1	C 2 2 5. 4
2	**a) Explain about physical address space** It refers to the actual memory addresses where data is stored in physical memory. Every byte of data in a computer system has a unique physical address. The physical address space represents the range of addresses accessible by the hardware without any translation. It's the tangible space where data resides in the RAM modules, typically represented in hexadecimal format, and its size is limited by the architecture of the computer.	3M	L 4	C 2 2 5. 4
	b) Explain about swapping Swapping is a memory management technique used by operating systems to efficiently utilize memory resources. When the system runs out of physical memory (RAM) to accommodate all the processes, it moves some pages of data from the RAM to the secondary storage, usually a hard disk or SSD. This process is called swapping. It allows the operating system to free up space in RAM for other processes, ensuring smooth operation of the system. Swapping can slow down performance due to the relatively slow speed of secondary storage compared to RAM.	3M	L 4	C 2 2 5. 4
	c) Explain about segmentation with paging Segmentation with paging is a memory management technique that combines the advantages of both segmentation and paging. Segmentation divides the logical address space into variable-sized segments, while paging divides the physical memory into fixed-sized blocks called pages. In this technique, each segment of a process is divided into pages, and the mapping from logical addresses to physical addresses is performed in two steps: first, the logical address is translated to a segment number and an offset within that segment, and then the segment number is translated to a physical address using a page table for that segment.	3M	L 4	C 2 2 5. 4
	d) Explain about demand paging Demand paging is a memory management scheme where pages are only loaded into memory when they are demanded by a process. Instead of loading the entire program into memory at once, only the necessary pages are loaded when they are needed during the execution of the program. This technique helps in conserving memory resources as only the required portions of a program are brought into memory, reducing the initial load time and overall memory usage. If a requested page is not in memory, a page fault occurs, and the operating system loads the required page from secondary storage into memory.	3M	L 4	C 2 2 5. 4

e) Explain about physical address space

It refers to the actual memory addresses where data is stored in physical memory. Every byte of data in a computer system has a unique physical address.

The physical address space represents the range of addresses accessible by the hardware without any translation. It's the tangible space where data resides in the RAM modules, typically represented in hexadecimal format, and its size is limited by the architecture of the computer.

3M | L 4 | C 2 2 5. 4

3

a) Distinguish between logical and physical address space

4.1 **Logical versus Physical Address Space:**

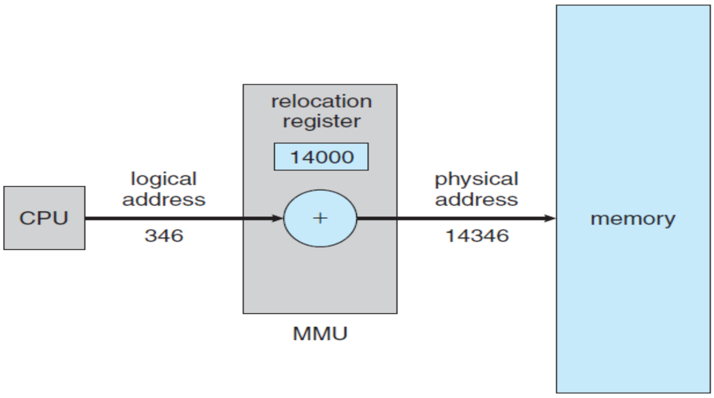
a) Logical Address:

- A logical address in an operating system is also known as a virtual address. It represents a location in the computer's memory from the perspective of the executing process.

S. No	Logical Address	Physical Address
1	Logical address is rendered by CPU.	Physical address is like a location that is present in the main memory.
2	It is a collection of all logical addresses rendered by the CPU.	It is a collection of all physical addresses mapped to the connected logical addresses.
3	Logical address of the program is visible to the users.	We cannot view the physical address of the program.
4	Logical address is generated by the CPU.	Physical address is computed by MMU.
5	We can easily utilise the logical address to access the physical address.	We can use the physical address indirectly.

- These addresses are generated by the CPU during program execution. Logical addresses are used by programs to access data and instructions stored in memory.

5M | L 3 | C 2 2 5. 4



b) Physical Address:

A physical address is the actual location in the computer's memory hardware, such as RAM or a storage device. Unlike logical addresses, which are generated by the CPU for processes, physical addresses are fixed and represent the true location of data in memory.

To continue with our previous example, when you open a document using your word processing application, the OS's memory manager translates the logical addresses generated by the CPU into physical addresses to fetch the actual content of the document from RAM. This translation is crucial because it ensures that each process can access the correct data without causing conflicts.

c) **Difference between Logical Address and Physical Address in Operating System**

b) Explain about segmentation 5M L4 C225.4

4.5 Segmentation:

Segmentation divides processes into smaller subparts known as modules. The divided segments need not be placed in contiguous memory. Since there is no contiguous memory allocation, internal fragmentation does not take place. The length of the segments of the program and memory is decided by the purpose of the segment in the user program.

We can say that logical address space or the main memory is a collection of segments. Hence, segmentation was introduced in which the code is divided into modules so that related code can be combined in one single block.

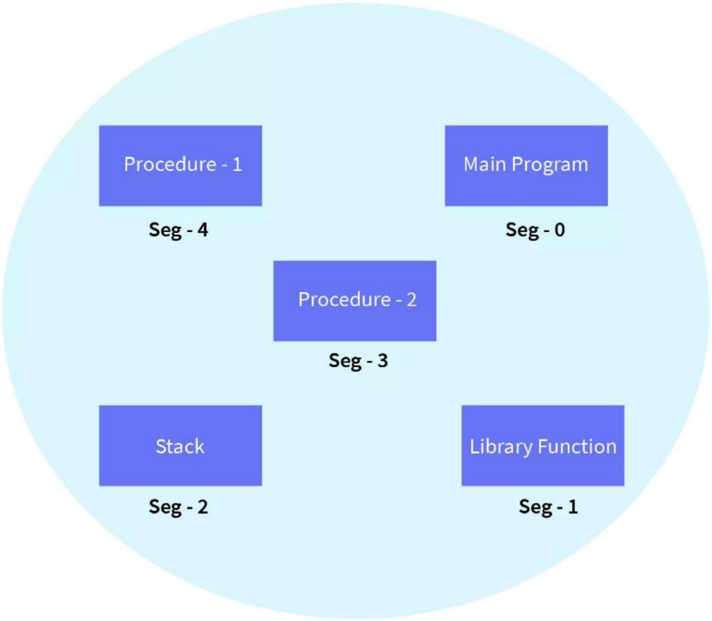
a) Types of Segmentation

Segmentation can be divided into two types:

i) **Virtual Memory Segmentation:** Virtual Memory Segmentation divides the processes into n number of segments. All the segments are not divided at a time. Virtual Memory Segmentation may or may not take place at the run time of a program.

ii) **Simple Segmentation:** Simple Segmentation also divides the processes into n number of segments but the segmentation is done all together at once. Simple segmentation takes place at the run time of a program. Simple segmentation may scatter the segments into the memory such that one segment of the process can be at a different location than the other (in a noncontinuous manner).

- In the case of the paging technique, a function or piece of code is divided into pages without considering that the relative parts of code can also get divided.
- Hence, for the process in execution, the CPU must load more than one page into the frames so that the complete related code is there for execution.
- Paging took more pages for a process to be loaded into the main memory.
- Other memory management techniques have also an important drawback - the actual view of physical memory is separated from the user's view of physical memory.
- Segmentation helps in overcoming the problem by dividing the user's program into segments according to the specific need.



Advantages of Segmentation in OS:

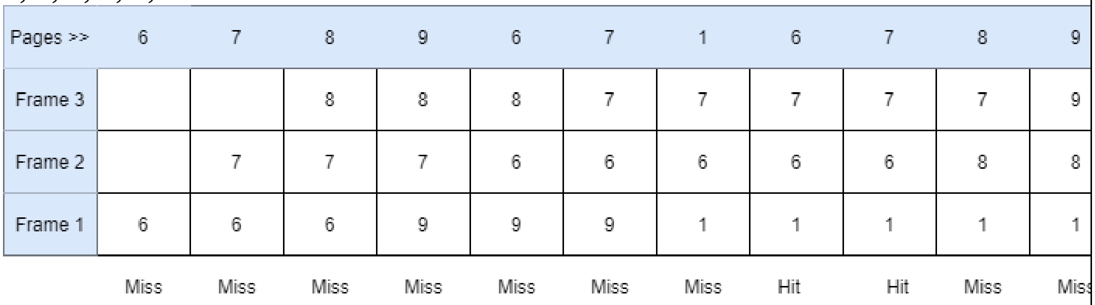
- No internal fragmentation is there in segmentation.
- Segment Table is used to store the records of the segments. The segment table itself consumes small memory as compared to a page table in paging.
- Segmentation provides better CPU utilization as an entire module is loaded at once.
- Segmentation is near to the user's view of physical memory. Segmentation allows users to partition the user programs into modules. These modules are nothing but the independent codes of the current process.

	c) Explain FIFO and OPR algorithms with examples **a) First In First Out:** • This algorithm is similar to the operations of the queue.	5M	L 4	C 2 2

5. 4

- All the pages are stored in the queue in the order they are allocated frames in the main memory.
- The one which is allocated first stays in the front of the queue.
- The one which is allocated the memory first is replaced first.
- The one which is at the front of the queue is removed at the time of replacement.

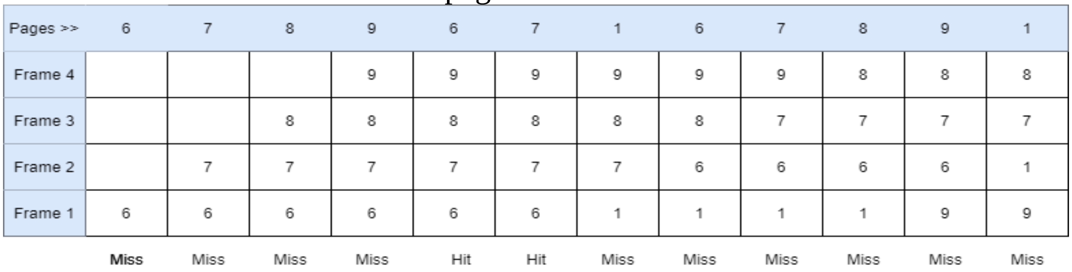
Example: Consider the Pages referenced by the CPU in the order are 6, 7, 8, 9, 6, 7, 1, 6, 7, 8, 9, 1



Pages >>	6	7	8	9	6	7	1	6	7	8	9
Frame 3			8	8	8	7	7	7	7	7	9
Frame 2		7	7	7	6	6	6	6	6	8	8
Frame 1	6	6	6	9	9	9	1	1	1	1	1
	Miss	Miss	Miss	Miss	Miss	Miss	Miss	Hit	Hit	Miss	Mis

- As in the above figure shown, Let there are 3 frames in the memory.
- 6, 7, 8 are allocated to the vacant slots as they are not in memory.
- When 9 comes page fault occurs, it replaces 6 which is the oldest in memory or front element of the queue.
- Then 6 comes (Page Fault), it replaces 7 which is the oldest page in memory now.
- Similarly, 7 replaces 8, 1 replaces 9.
- Then 6 comes which is already in memory (Page Hit).
- Then 7 comes (Page Hit).
- Then 8 replaces 6, 9 replaces 7. Then 1 comes (Page Hit).
- Number of Page Faults = 9

"While using the First In First Out algorithm, the number of page faults increases by increasing the number of frames. This phenomenon is called Belady's Anomaly". Let's take the same above order of pages with 4 frames.

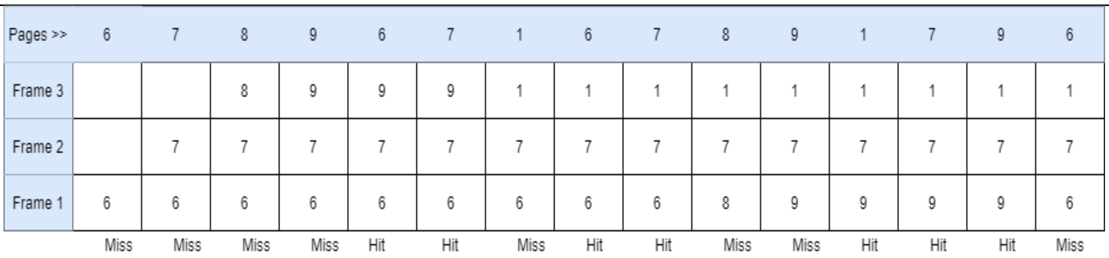


Pages >>	6	7	8	9	6	7	1	6	7	8	9	1
Frame 4				9	9	9	9	9	9	8	8	8
Frame 3			8	8	8	8	8	8	7	7	7	7
Frame 2		7	7	7	7	7	7	6	6	6	6	1
Frame 1	6	6	6	6	6	6	1	1	1	1	9	9
	Miss	Miss	Miss	Miss	Hit	Hit	Miss	Miss	Miss	Miss	Miss	Miss

- In the above picture shown, it can be seen that the number of page faults is 10.
- There were 9 page faults with 3 frames and 10 page faults with 4 frames.
- The number of page faults increased by increasing the number of frames.

b) Optimal Page Replacement –

- In this algorithm, the page which would be used after the longest interval is replaced.
- In other words, the page which is farthest to come in the upcoming sequence is replaced.
- Example:
  Consider the Pages referenced by the CPU in the order are 6, 7, 8, 9, 6, 7, 1, 6, 7, 8, 9, 1, 7, 9, 6.



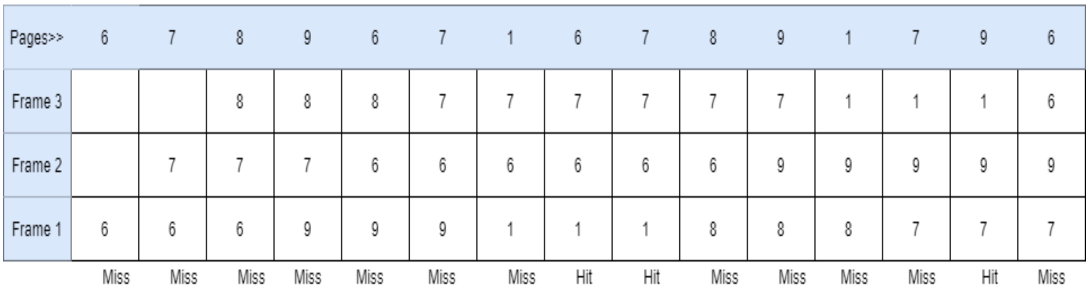
Pages >>	6	7	8	9	6	7	1	6	7	8	9	1	7	9	6
Frame 3			8	9	9	9	1	1	1	1	1	1	1	1	1
Frame 2		7	7	7	7	7	7	7	7	7	7	7	7	7	7
Frame 1	6	6	6	6	6	6	6	6	6	8	9	9	9	9	6
	Miss	Miss	Miss	Miss	Hit	Hit	Miss	Hit	Hit	Miss	Miss	Hit	Hit	Hit	Miss

- First, all the frames are empty. 6, 7, 8 are allocated to the frames (Page Fault).
- Now, 9 comes and replaces 8 as it is the farthest in the upcoming sequence. 6 and 7 would come earlier than that so not replaced.
- Then, 6 comes which is already present (Page Hit).
- Then 7 comes (Page Hit).
- Then 1 replaces 9 similarly (Page Fault).
- Then 6 comes (Page Hit), 7 comes (Page Hit).
- Then 8 replaces 6 (Page Fault) and 9 replaces 8 (Page Fault).
- Then 1, 7, 9 come respectively which are already present in the memory.
- Then 6 replaces 9 (Page Fault), it can also replace 7 and 1 as no other page is present in the upcoming sequence.
- The number of Page Faults = 8
- This is the most optimal algorithm but is impractical because it is impossible to predict the upcoming page references.

d) Explain in detail about LRU Page Replacement Algorithm. 5M L4 C225.4

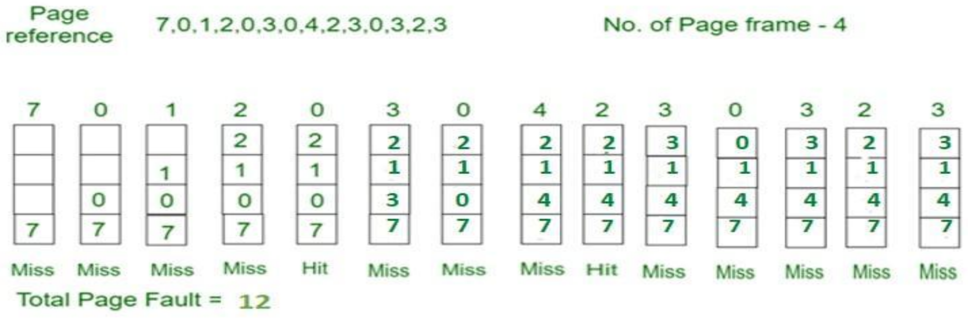
c) Least Recently Used –

- This algorithm works on previous data.
- The page which is used the earliest is replaced or which appears the earliest in the sequence is replaced.
- Example:
- Consider the Pages referenced by the CPU in the order are 6, 7, 8, 9, 6, 7, 1, 6, 7, 8, 9, 1, 7, 9, 6



Pages>>	6	7	8	9	6	7	1	6	7	8	9	1	7	9	6
Frame 3			8	8	8	7	7	7	7	7	7	1	1	1	6
Frame 2		7	7	7	6	6	6	6	6	6	9	9	9	9	9
Frame 1	6	6	6	9	9	9	1	1	1	8	8	8	7	7	7
	Miss	Miss	Miss	Miss	Miss	Miss	Miss	Hit	Hit	Miss	Miss	Miss	Miss	Hit	Miss

- First, all the frames are empty. 6, 7, 8 are allocated to the frames (Page Fault).
- Now, 9 comes and replaces 6 which is used the earliest (Page Fault).
- Then, 6 replaces 7, 7 replaces 8, 1 replaces 9 (Page Fault).
- Then 6 comes which is already present (Page Hit).
- Then 7 comes (Page Hit).
- Then 8 replaces 1, 9 replaces 6, 1 replaces 7, and 7 replaces 8 (Page Fault).
- Then 9 comes (Page Hit).

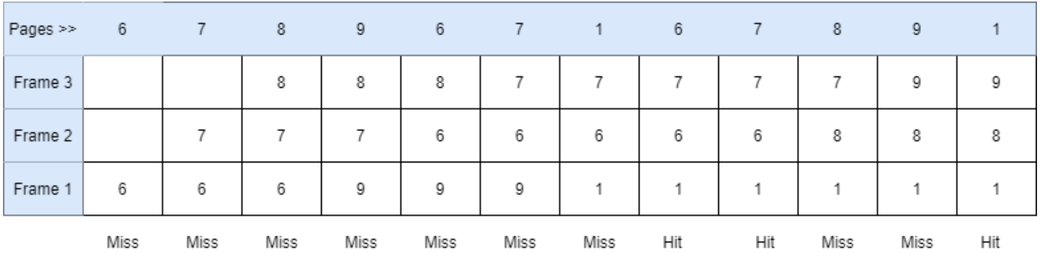
	- Then 6 replaces 1 (Page Fault). - The number of Page Faults = 12			
	e) Explain in detail about any one Replacement Algorithm. **Most Recently Used (MRU)** - In this algorithm, page will be replaced which has been used recently. Belady's anomaly can occur in this algorithm. - Example 4: Consider the page reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 3 with 4-page frames. Find number of page faults using MRU Page Replacement Algorithm.  Page reference 7,0,1,2,0,3,0,4,2,3,0,3,2,3 No. of Page frame - 4 Miss Miss Miss Miss Hit Miss Miss Miss Hit Miss Miss Miss Miss Miss Total Page Fault = 12 - Initially, all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> 4 Page faults - 0 is already their so–> 0 page fault - when 3 comes it will take place of 0 because it is most recently used —> 1 Page fault - when 0 comes it will take place of 3 —> 1 Page fault - when 4 comes it will take place of 0 —> 1 Page fault - 2 is already in memory so —> 0 Page fault - when 3 comes it will take place of 2 —> 1 Page fault - when 0 comes it will take place of 3 —> 1 Page fault - when 3 comes it will take place of 0 —> 1 Page fault - when 2 comes it will take place of 3 —> 1 Page fault - when 3 comes it will take place of 2 —> 1 Page fault	5M	L 4	C 2 2 5. 4
4	**a) Explain about page replacement algorithms.** **Page Replacement algorithm:** - Page Replacement Algorithm is used when a page fault occurs. - Page Fault means the page referenced by the CPU is not present in the main memory. - When the CPU generates the reference of a page, if there is any vacant frame available in the main memory then the page is loaded in that vacant frame. - In another case, if there is no vacant frame available in the main memory, it is required to replace one of the pages in the main memory with the page referenced by the CPU. - Page Replacement Algorithm is used to decide which page will be replaced to allocate memory to the current referenced page. - Different Page Replacement Algorithms suggest different ways to decide which page is to be replaced.	10 M	L 4	C 2 2 5. 4

- The main objective of these algorithms is to reduce the number of page faults.

d) **First In First Out:**

- This algorithm is similar to the operations of the queue.
- All the pages are stored in the queue in the order they are allocated frames in the main memory.
- The one which is allocated first stays in the front of the queue.
- The one which is allocated the memory first is replaced first.
- The one which is at the front of the queue is removed at the time of replacement.

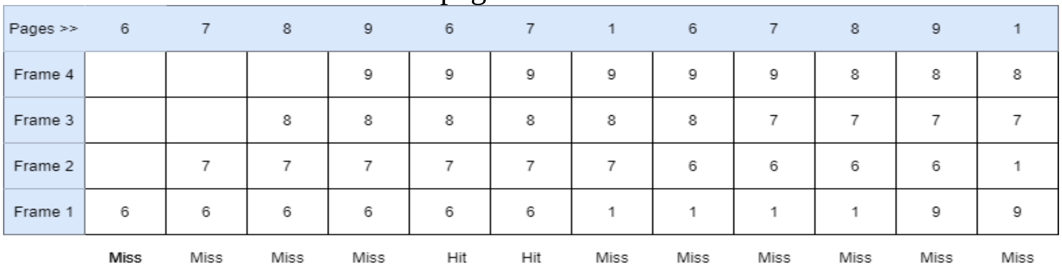
Example: Consider the Pages referenced by the CPU in the order are 6, 7, 8, 9, 6, 7, 1, 6, 7, 8, 9, 1



Pages >>	6	7	8	9	6	7	1	6	7	8	9	1
Frame 3			8	8	8	7	7	7	7	7	9	9
Frame 2		7	7	7	6	6	6	6	6	8	8	8
Frame 1	6	6	6	9	9	9	1	1	1	1	1	1
	Miss	Miss	Miss	Miss	Miss	Miss	Miss	Hit	Hit	Miss	Miss	Hit

- As in the above figure shown, Let there are 3 frames in the memory.
- 6, 7, 8 are allocated to the vacant slots as they are not in memory.
- When 9 comes page fault occurs, it replaces 6 which is the oldest in memory or front element of the queue.
- Then 6 comes (Page Fault), it replaces 7 which is the oldest page in memory now.
- Similarly, 7 replaces 8, 1 replaces 9.
- Then 6 comes which is already in memory (Page Hit).
- Then 7 comes (Page Hit).
- Then 8 replaces 6, 9 replaces 7. Then 1 comes (Page Hit).
- Number of Page Faults = 9

"While using the First In First Out algorithm, the number of page faults increases by increasing the number of frames. This phenomenon is called Belady's Anomaly".
Let's take the same above order of pages with 4 frames.



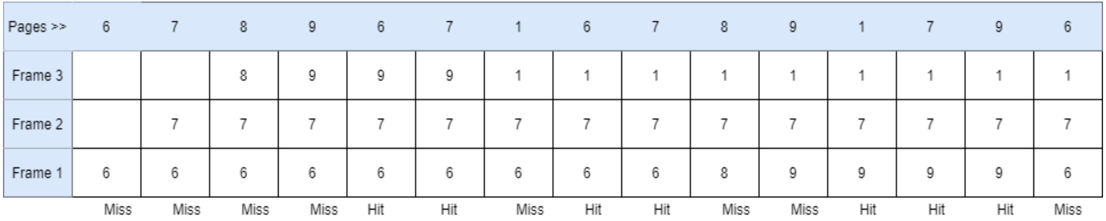
Pages >>	6	7	8	9	6	7	1	6	7	8	9	1
Frame 4				9	9	9	9	9	9	8	8	8
Frame 3			8	8	8	8	8	8	7	7	7	7
Frame 2		7	7	7	7	7	7	6	6	6	6	1
Frame 1	6	6	6	6	6	6	1	1	1	1	9	9
	Miss	Miss	Miss	Miss	Hit	Hit	Miss	Miss	Miss	Miss	Miss	Miss

- In the above picture shown, it can be seen that the number of page faults is 10.
- There were 9 page faults with 3 frames and 10 page faults with 4 frames.
- The number of page faults increased by increasing the number of frames.

e) **Optimal Page Replacement –**

- In this algorithm, the page which would be used after the longest interval is replaced.
- In other words, the page which is farthest to come in the upcoming sequence is replaced.
- Example:

Consider the Pages referenced by the CPU in the order are 6, 7, 8, 9, 6, 7, 1, 6, 7, 8, 9, 1, 7, 9, 6.

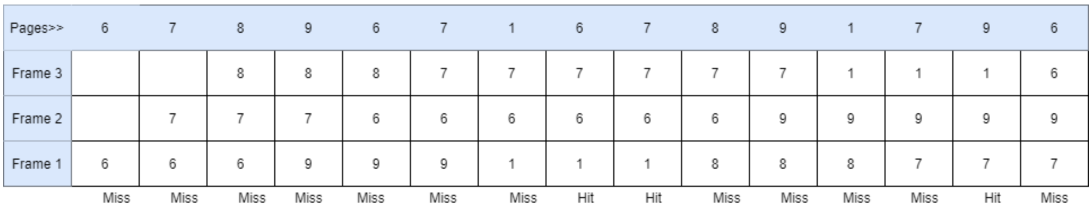


Pages >>	6	7	8	9	6	7	1	6	7	8	9	1	7	9	6
Frame 3			8	9	9	9	1	1	1	1	1	1	1	1	1
Frame 2		7	7	7	7	7	7	7	7	7	7	7	7	7	7
Frame 1	6	6	6	6	6	6	6	6	6	8	9	9	9	9	6
	Miss	Miss	Miss	Miss	Hit	Hit	Miss	Hit	Hit	Miss	Miss	Hit	Hit	Hit	Miss

- First, all the frames are empty. 6, 7, 8 are allocated to the frames (Page Fault).
- Now, 9 comes and replaces 8 as it is the farthest in the upcoming sequence. 6 and 7 would come earlier than that so not replaced.
- Then, 6 comes which is already present (Page Hit).
- Then 7 comes (Page Hit).
- Then 1 replaces 9 similarly (Page Fault).
- Then 6 comes (Page Hit), 7 comes (Page Hit).
- Then 8 replaces 6 (Page Fault) and 9 replaces 8 (Page Fault).
- Then 1, 7, 9 come respectively which are already present in the memory.
- Then 6 replaces 9 (Page Fault), it can also replace 7 and 1 as no other page is present in the upcoming sequence.
- The number of Page Faults = 8
- This is the most optimal algorithm but is impractical because it is impossible to predict the upcoming page references.

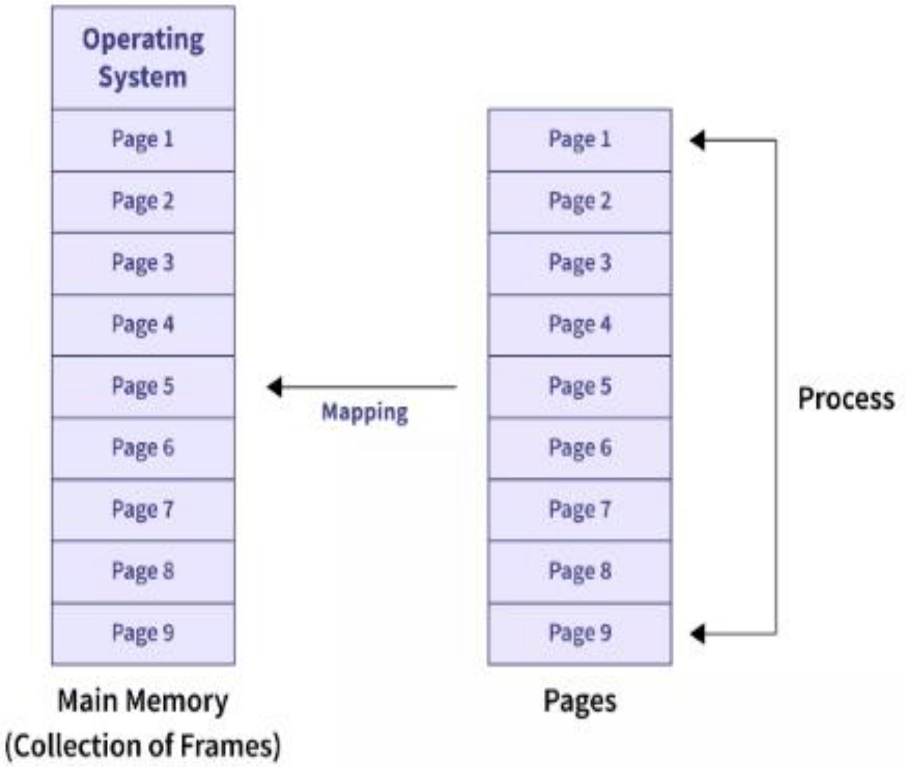
f) Least Recently Used –

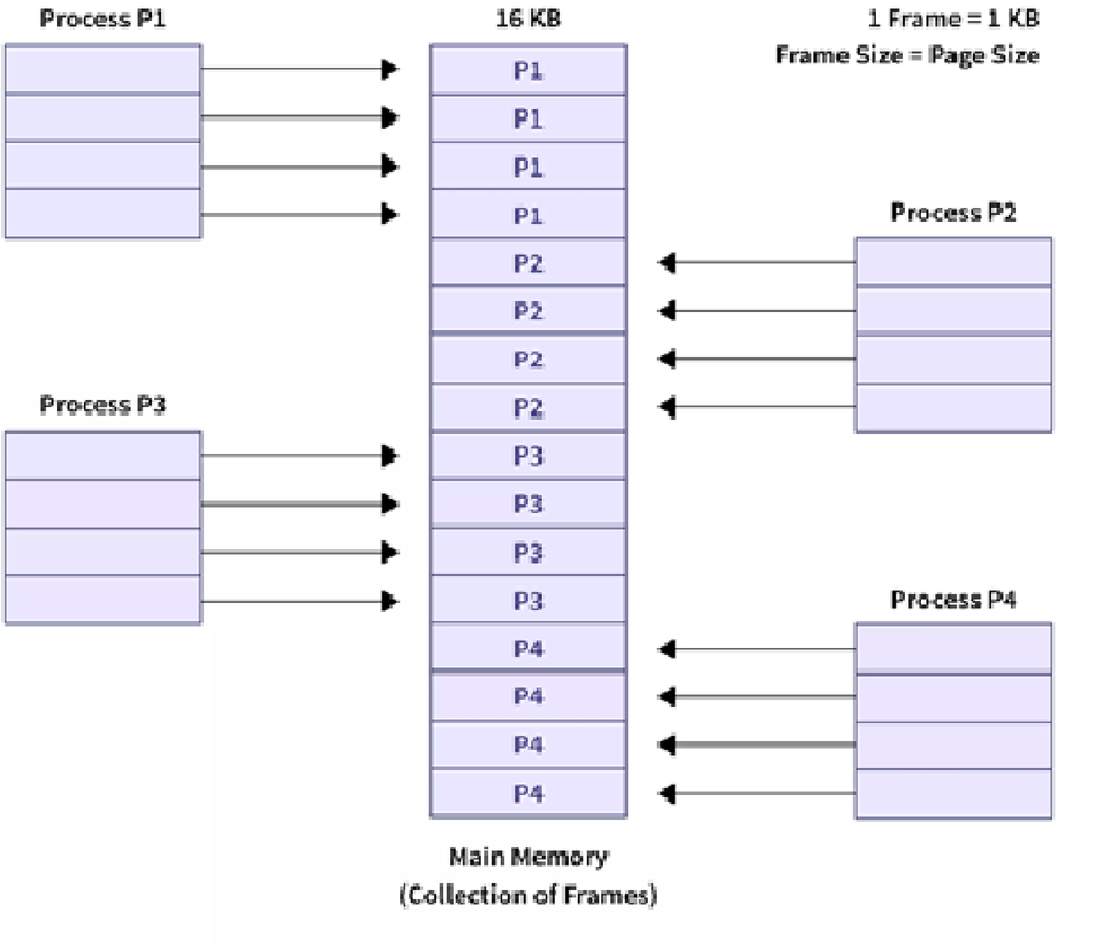
- This algorithm works on previous data.
- The page which is used the earliest is replaced or which appears the earliest in the sequence is replaced.
- Example:
- Consider the Pages referenced by the CPU in the order are 6, 7, 8, 9, 6, 7, 1, 6, 7, 8, 9, 1, 7, 9, 6



Pages>>	6	7	8	9	6	7	1	6	7	8	9	1	7	9	6
Frame 3			8	8	8	7	7	7	7	7	7	1	1	1	6
Frame 2		7	7	7	6	6	6	6	6	6	9	9	9	9	9
Frame 1	6	6	6	9	9	9	1	1	1	8	8	8	7	7	7
	Miss	Miss	Miss	Miss	Miss	Miss	Miss	Hit	Hit	Miss	Miss	Miss	Miss	Hit	Miss

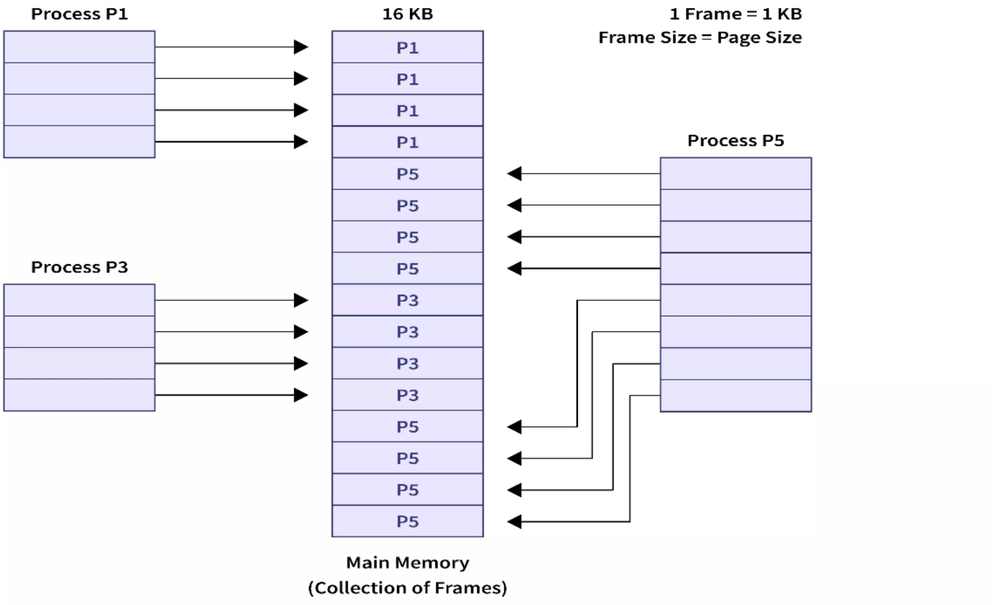
- First, all the frames are empty. 6, 7, 8 are allocated to the frames (Page Fault).
- Now, 9 comes and replaces 6 which is used the earliest (Page Fault).
- Then, 6 replaces 7, 7 replaces 8, 1 replaces 9 (Page Fault).
- Then 6 comes which is already present (Page Hit).
- Then 7 comes (Page Hit).
- Then 8 replaces 1, 9 replaces 6, 1 replaces 7, and 7 replaces 8 (Page Fault).
- Then 9 comes (Page Hit).
- Then 6 replaces 1 (Page Fault).
- The number of Page Faults = 12

	b) Explain about memory management techniques. **Paging:** Paging is a technique that divides memory into fixed-sized blocks. The main memory is divided into blocks known as Frames and the logical memory is divided into blocks known as Pages. Paging requires extra time for the address conversion, so we use a special hardware cache memory known as TLB. This concept of Paging in OS includes dividing each process in the form of pages of equal size and also, the main memory is divided in the form of frames of fixed size. Now, each page of the process when retrieved into the main memory, is stored in one frame of the memory, and hence, it is also important to have the pages and frames of equal size for mapping and maximum utilization of the memory. Its main advantage is that the pages can be stored at different locations of the memory and not necessarily in a contiguous manner, though priority is always set to firstly find the contiguous frames for allocating the pages.  Operating System Page 1 Page 2 Page 3 Page 4 Page 5 Page 6 Page 7 Page 8 Page 9 Mapping Process Main Memory (Collection of Frames) Pages **If a process has n pages in the secondary memory, then there must be n frames available in the main memory for mapping.** **Example to understand Paging in OS** **CASE-1 (Contiguous Allocation of Pages)**	10 M	L 4	C225.4



As we can see in the above image, we have main memory divided into 16 frames of the size of 1KB each. Also, there are 4 processes available in the secondary (local) memory: P1, P2, P3, and P4 of a size of 4KB each. Clearly, each process needs to be further subdivided into pages of size of 1KB each, so that one page can be easily mapped to one frame of the main memory. This divides each process into 4 pages and the total for 4 processes gives 16 pages of 1KB each. Initially, all the frames were empty and therefore, pages will be allocated here in a contiguous manner.

CASE-2 (Non-Contiguous Allocation of Pages)

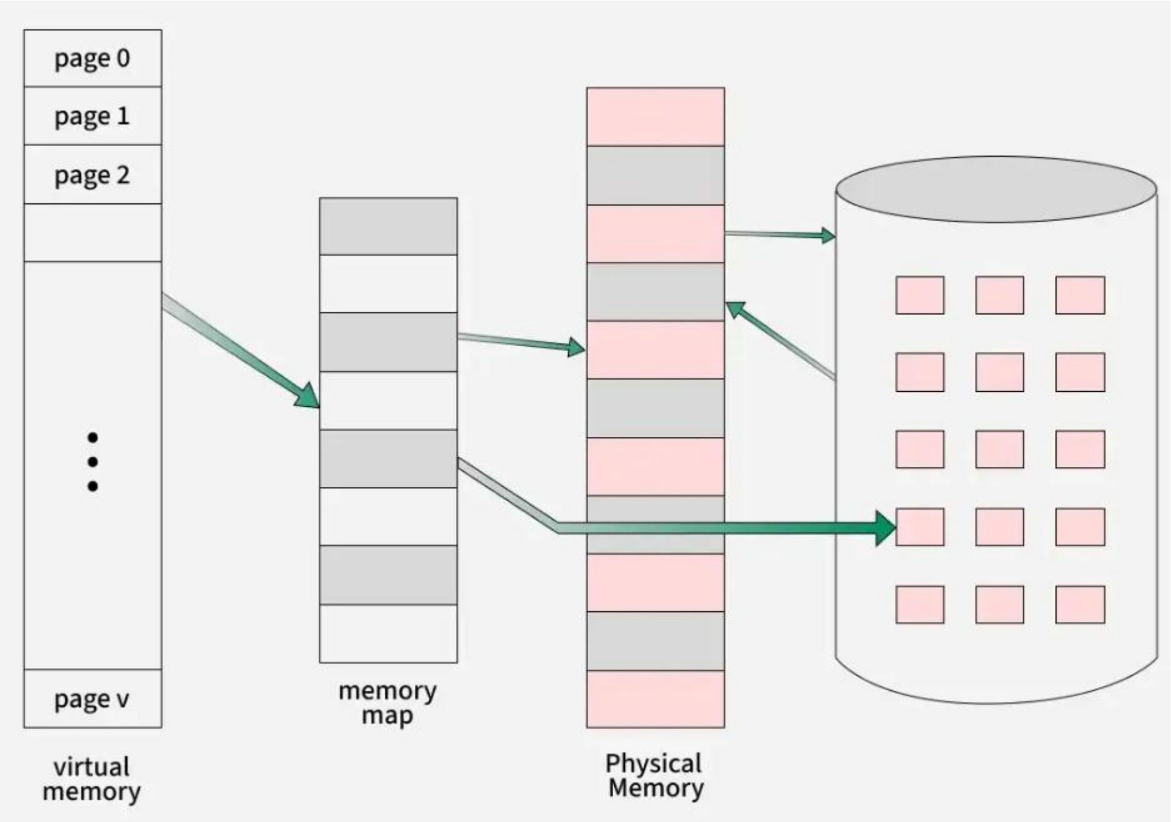


Let us assume that in Case-1, processes P2 and P4 are moved to the waiting state after some time and leave behind the empty space of 8 frames. **In Case-2, we have another process P5 of size 8KB (8 pages) waiting inside the ready queue to be allocated.** We know that with Paging, we can store the pages at different locations of the memory, and here, we have8non-contiguous frames available. Therefore, we can easily load the 8 pages of the process P5 in the place of P2 and P4 which we can observe in the above image.

Memory Management Unit

MMU is a computer hardware component that responsible for all memory and caching operations which are associated with the CPU. MMU usually exist in CP but in some cases it operates on separate integrated circuit (IC) chip. Here's an overview of how the MMU, specifically in the context of paging, functions in an operating system:

- **Virtual Memory Translation:** The MMU translates virtual addresses generated by a process into physical addresses in RAM (Random Access Memory). This translation allows processes to run as if they have access to a large, continuous block of memory, even if the physical memory is fragmented or limited.
- **Paging:** Paging is a memory management scheme where physical memory is divided into fixed-size blocks called "frames," and logical memory is divided into equally sized blocks called "pages." The MMU maps pages to frames, enabling efficient allocation and management of memory.
- **Page Tables:** The MMU uses page tables to maintain the mapping between virtual pages and physical frames. Each process has its own page table, which keeps track of the virtual-to-physical address translations.

	• **Page Faults:** When a process accesses a page that is not in physical memory, a page fault occurs. The MMU triggers an interrupt, and the operating system must load the required page from secondary storage (usually a hard disk) into a free frame in RAM. • **Page Replacement:** If all frames in physical memory are in use, the operating system must choose a page to evict. This process, known as page replacement, is often managed using algorithms like the Least Recently Used (LRU) or the Second Chance algorithm.			
	c) Explain in detail about Virtual Memory. Virtual memory is a memory management technique used by operating systems to give the appearance of a large, continuous block of memory to applications, even if the physical memory (RAM) is limited. It allows larger applications to run on systems with less RAM. • The main objective of virtual memory is to support multiprogramming, The main advantage that virtual memory provides is, a running process does not need to be entirely in memory. • Programs can be larger than the available physical memory. Virtual Memory provides an abstraction of main memory, eliminating concerns about storage limitations. A memory hierarchy, consisting of a computer system's memory and a disk, enables a process to operate with only some portion  page 0 page 1 page 2 page v virtual memory memory map Physical Memory **Types of Virtual Memory**	10 M	L 4	C 225.4

In a computer, virtual memory is managed by the Memory Management Unit (MMU), which is often built into the CPU. The CPU generates virtual addresses that the MMU translates into physical addresses.

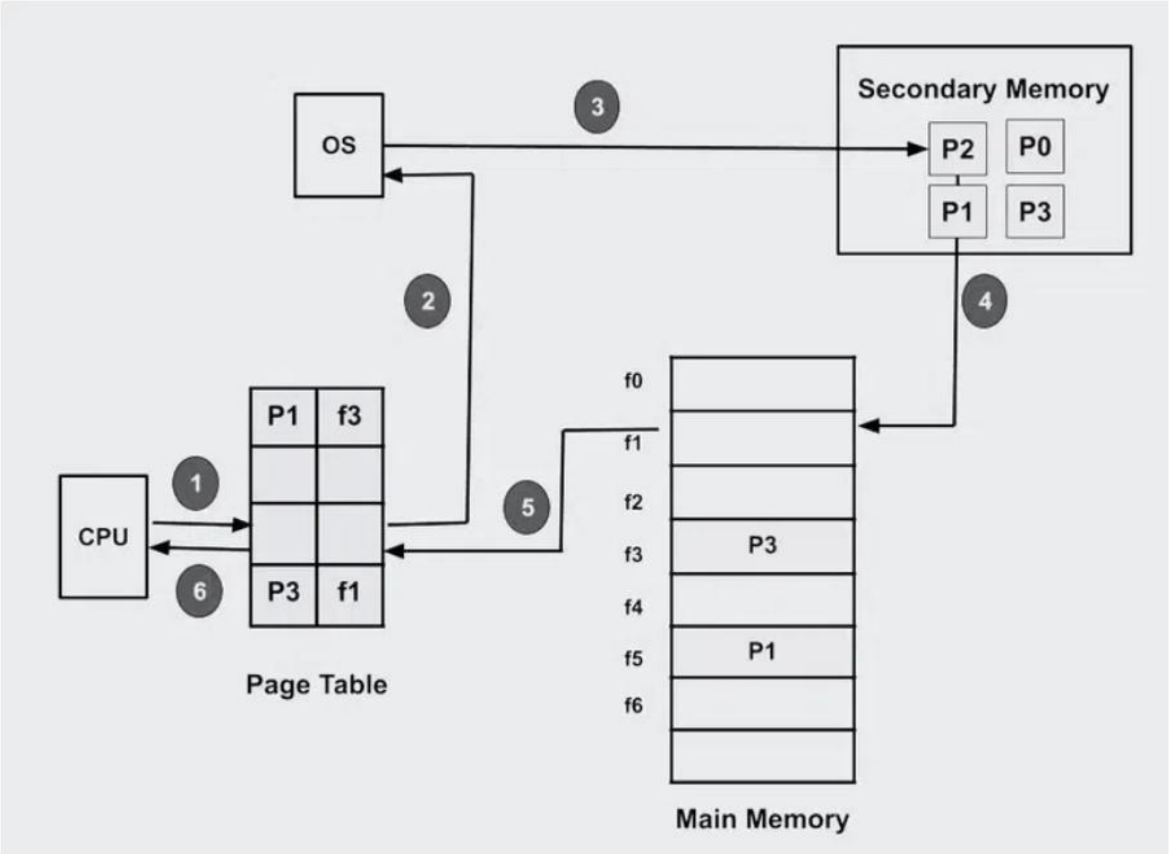
There are two main types of virtual memory:

- Paging
- Segmentation

Paging

Paging divides memory into small fixed-size blocks called pages. When the computer runs out of RAM, pages that aren't currently in use are moved to the hard drive, into an area called a swap file. The swap file acts as an extension of RAM. When a page is needed again, it is swapped back into RAM, a process known as page swapping. This ensures that the operating system (OS) and applications have enough memory to run.

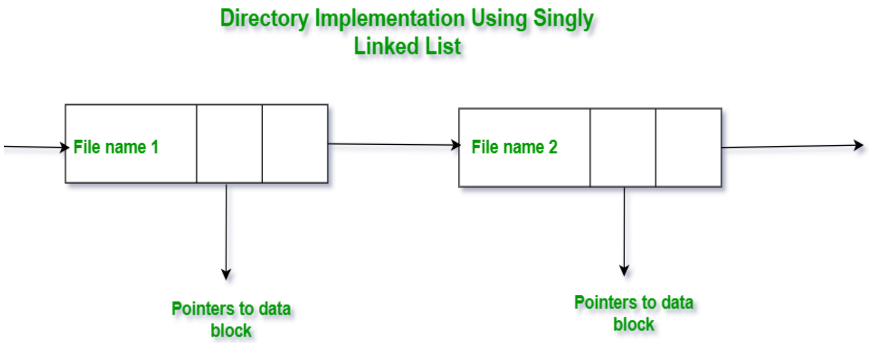
Demand Paging: The process of loading the page into memory on demand (whenever a page fault occurs) is known as demand paging. The process includes the following steps are as follows:



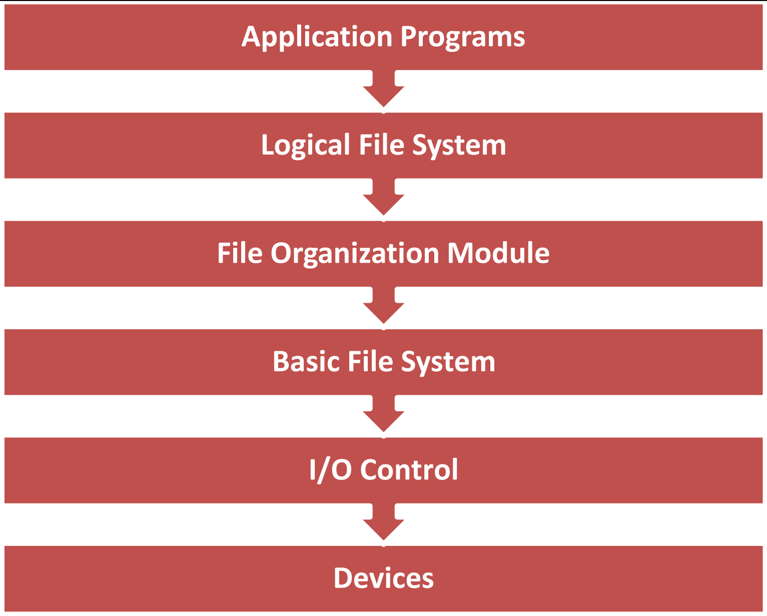
- If the CPU tries to refer to a page that is currently not available in the main memory, it generates an interrupt indicating a memory access fault.

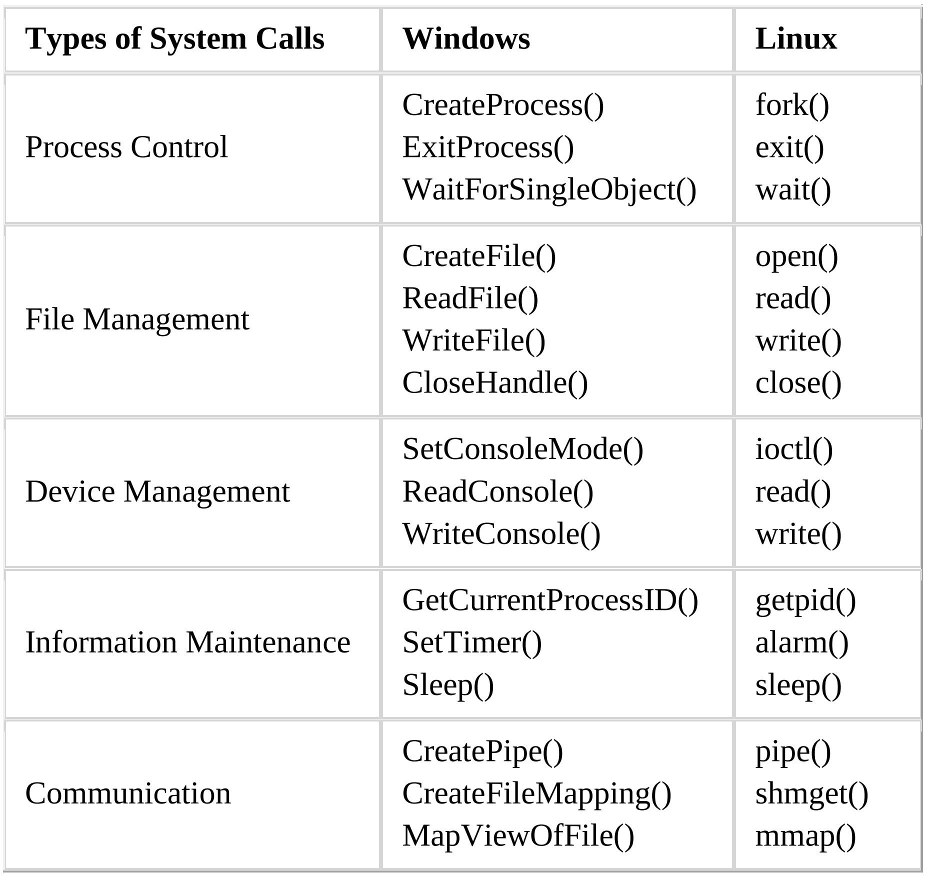
	• The OS puts the interrupted process in a blocking state. For the execution to proceed the OS must bring the required page into the memory. • The OS will search for the required page in the logical address space. • The required page will be brought from logical address space to physical address space. The page replacement algorithms are used for the decision-making of replacing the page in physical address space. • The page table will be updated accordingly. • The signal will be sent to the CPU to continue the program execution and it will place the process back into the ready state.			

Q. No	Question (s)	Marks	BL	CO
UNIT – V				
1	**a) Define a file.** A file is a named collection of data stored on a storage device like a hard drive or SSD. It's a basic unit of storage, providing a logical view of information. Files can contain various data types, such as text, images, audio, video, or executable code. The operating system manages files through operations like creating, reading, writing, deleting, and more	1M	L1	C225.5
	b) Define file system mounting? file system mounting is the process of making a file system, storage device, or network share accessible to the system's directory structure. This is done by attaching the file system to a specific directory, called the mount point, and making it accessible as if it were part of the main file system.	1M	L1	C225.6
	c) List out the merits of Linux. Linux offers numerous advantages, including being open-source, free, secure, stable, and highly customizable. It's also known for its broad compatibility, community support, and efficiency.	1M	L1	C225.5
	d) Define the usage of system calls. System calls are the interface that allows user programs to request services from the operating system's kernel. They enable programs to perform actions that require special privileges or access to hardware, memory, and other low-level resources that user-level processes cannot directly manage.	1M	L1	C225.6
	e) Define stat system calls. The stat system call in an operating system is a mechanism for retrieving metadata about a file or directory. It provides information like file size, type, ownership, permissions, and timestamps. This information is stored in a stat structure, which is then passed back to the calling program	1M	L1	C225.5
2	**a) Discuss about Two Level Directory Structure.** A two-level directory structure in an operating system has a root directory and user-specific directories. Each user gets their own directory, where they can store their files. This structure separates users' files, allowing them to have the same file names within their individual directories.	3M	L2	C225.5
	b) Explain Tree structure directory. In an operating system, a tree-structured directory organization is a hierarchical method for organizing files and folders. It resembles an inverted tree, with a root directory at the top and branches (subdirectories) extending downwards. This structure provides a logical way to manage files and folders, allowing users to create subdirectories within their own directories.	3M	L4	C225.5
	c) What is directory? Explain about directory Implementation. Directory implementation in the operating system can be done using	3M	L4	C225.6

	Singly Linked List and Hash table. The efficiency, reliability, and performance of a file system are greatly affected by the selection of directory-allocation and directory-management algorithms. There are numerous ways in which the directories can be implemented. But we need to choose an appropriate directory implementation algorithm that enhances the performance of the system. **Directory Implementation using Singly Linked List** The implementation of directories using a singly linked list is easy to program but is time-consuming to execute. Here we implement a directory by using a linear list of filenames with pointers to the data blocks.  Directory Implementation Using Singly Linked List File name 1 File name 2 Pointers to data block Pointers to data block ***Directory Implementation Using Singly Linked List*** • To create a new file the entire list has to be checked such that the new directory does not exist previously. • The new directory then can be added to the end of the list or at the beginning of the list. • In order to delete a file, we first search the directory with the name of the file to be deleted. After searching we can delete that file by releasing the space allocated to it. • To reuse the directory entry we can mark that entry as unused or we can append it to the list of free directories. • To delete a file linked list is the best choice as it takes less time.			
	d) Discuss the objectives for file management system. The primary objective of a filing system is to organize and store documents for easy access, retrieval, and management, ensuring that information is readily available when needed. This includes safeguarding documents, providing a systematic arrangement, and facilitating the retrieval of specific information quickly and efficiently. Here's a more detailed look at the objectives: 1. Organization and Retrieval: • A well-designed filing system makes it easy to find and retrieve specific documents or information within a collection. • This organization can be achieved through various methods like alphabetical, numerical, or subject-based classification. • Proper labeling and indexing are crucial for effective retrieval. 2. Security and Preservation: • Filing systems protect documents from damage, loss, or	3M	L2	C225.6

	unauthorized access. • They ensure that important records are kept safe and secure for future use. • This includes protecting against physical threats like fire, water, or pests, as well as digital security measures. 3. Accountability and Compliance: • Filing systems help maintain a record of transactions, correspondence, and other relevant information, which is important for accountability and audit purposes. • They can also be used to track compliance with legal and regulatory requirements. 4. Efficiency and Productivity: • A well-organized filing system saves time and effort by making it easier to locate information. • This can improve productivity by allowing staff to access information quickly and efficiently. • It also helps streamline workflows and reduces the risk of errors. 5. Decision Making and Planning: • Filing systems provide a historical record of past activities and decisions, which can be valuable for future planning and decision-making. • They can also help identify trends, patterns, and insights that can inform strategic planning. 6. Customer Service and Communication: • A well-organized filing system can improve customer service by providing quick access to relevant information. • It can also facilitate better communication with customers by ensuring that all relevant information is readily available. 7. Legal and Evidential Purposes: • Filing systems can be used to document and preserve evidence in legal proceedings. • They can also be used to provide proof of transactions and agreements.			
	e) Explain briefly about system calls with examples.	3M	L4	C225.5
3	**a) Discuss about file system structure.** **File System Structure:** ♦ File System provide efficient access to the disk by allowing data to be stored, located and retrieved in a convenient way. ♦ A file System must be able to store the file, locate the file and retrieve the file. ♦ Most of the Operating Systems use layering approach for every task including file systems. ♦ Every layer of the file system is responsible for some activities.	5M	L2	C225.6

	 Application Programs Logical File System File Organization Module Basic File System I/O Control Devices ♦ When an application program asks for a file, the first request is directed to the logical file system. ♦ The logical file system contains the Meta data of the file and directory structure. If the application program doesn't have the required permissions of the file then this layer will throw an error. ♦ Logical file systems also verify the path to the file. ♦ Generally, files are divided into various logical blocks. ♦ Files are to be stored in the hard disk and to be retrieved from the hard disk. Hard disk is divided into various tracks and sectors. ♦ Therefore, in order to store and retrieve the files, the logical blocks need to be mapped to physical blocks. ♦ This mapping is done by File organization module. It is also responsible for free space management. ♦ Once File organization module decided which physical block the application program needs, it passes this information to basic file system. ♦ The basic file system is responsible for issuing the commands to I/O control in order to fetch those blocks. ♦ I/O controls contain the codes by using which it can access hard disk. These codes are known as device drivers. ♦ I/O controls are also responsible for handling interrupts.			
	b) Explain about Lseek, stat and IOCTL system calls.	5M	L4	C225.5

<table>
<tr>
<td></td>
<td><b>c) Explain the usage of open, create, read, and write system calls.</b>

<table><tr><th>Types of System Calls</th><th>Windows</th><th>Linux</th></tr><tr><td>Process Control</td><td>CreateProcess()
ExitProcess()
WaitForSingleObject()</td><td>fork()
exit()
wait()</td></tr><tr><td>File Management</td><td>CreateFile()
ReadFile()
WriteFile()
CloseHandle()</td><td>open()
read()
write()
close()</td></tr><tr><td>Device Management</td><td>SetConsoleMode()
ReadConsole()
WriteConsole()</td><td>ioctl()
read()
write()</td></tr><tr><td>Information Maintenance</td><td>GetCurrentProcessID()
SetTimer()
Sleep()</td><td>getpid()
alarm()
sleep()</td></tr><tr><td>Communication</td><td>CreatePipe()
CreateFileMapping()
MapViewOfFile()</td><td>pipe()
shmget()
mmap()</td></tr></table></td>
<td>5M</td>
<td>L4</td>
<td>C225.6</td>
</tr>
<tr>
<td></td>
<td><b>d) Explain the concept of file sharing ,what are the criteria to be followed in system to implement file sharing.</b>
File Sharing in an Operating System(OS) denotes how information and files are shared between different users, computers, or devices on a network; and files are units of data that are stored in a computer in the form of documents/images/videos or any others types of information needed.

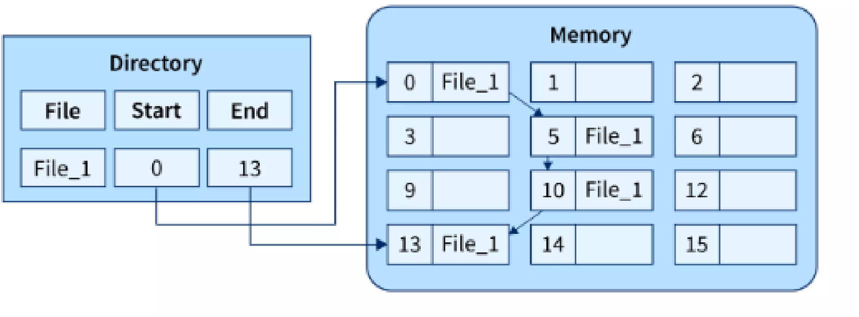
For Example: Suppose letting your computer talk to another computer and exchange pictures, documents, or any useful data. This is generally useful when one wants to work on a project with others, send files to friends, or simply shift stuff to another device. Our OS provides ways to do this like email attachments, cloud services, etc. to make the sharing process easier and more secure.

Now, file sharing is nothing like a magical bridge between Computer A to Computer B allowing them to swap some files with each other.

Primary Terminology Related to File Sharing

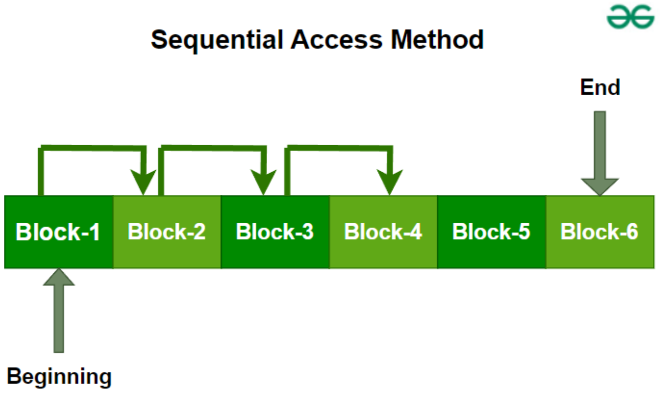
Let's see what are the various ways to achieve this, but there are some important terminologies one should know beforehand. Let's discuss</td>
<td>5M</td>
<td>L4</td>
<td>C225.5</td>
</tr>
</table>

	those primary terminologies first: • Folder/Directory: It is basically like a container for all of our files on a computer. The folder can contain files and even other folders maintaining like hierarchical structure for organizing data. • Networking: It is involved in connecting computers or devices where we need to share the resources. Networks can be local (LAN) or global (Internet). • IP Address: It is numerical data given to every connected device on the network • Protocol: It is given as the set of rules which drives the communication between devices on a network. In the context of file sharing, protocols define how files are transferred between computers. • File Transfer Protocol (FTP): FTP is a standard network protocol used to transfer files between a client and a server on a computer network.			
	e) Explain about linked allocation method of a file. **a) Linked File Allocation:** ♦ The Linked file allocation overcomes the drawback of contiguous file allocation. ♦ Here the file which we store on the hard disk is stored in a scattered manner according to the space available on the hard disk. ♦ Now, you must be thinking about how the OS remembers that all the scattered blocks belong to the same file. ♦ So as the name linked File Allocation suggests, the pointers are used to point to the next block of the same file, therefore along with the entry of each file each block also stores the pointer to the next block.	5M	L4	C225.6

	 Directory File Start End File_1 0 13 Memory 0 File_1 1 2 3 5 File_1 6 9 10 File_1 12 13 File_1 14 15 ♦ In the above image on the right, we have a memory diagram where we can see memory blocks. ♦ On the left side, we have a directory where we have the information like the address of the first memory block and the last memory block. ♦ In this allocation, the starting block given is 0 and the ending block is 15, therefore the OS searches the empty blocks between 0 and 15 and stores the files in available blocks, but along with that it also stores the pointer to the next block in the present block. ♦ Hence it requires some extra space to store that link.			
4	**a) Explain in detail about File Access Methods.** **2 File Access Methods:** • File access methods in an operating system are the techniques and processes used to read from and write to files stored on a computer's storage devices. • There are several ways to access this information in the file. Some systems provide only one access method for files. • Other systems, such as those of IBM, support many access methods, and choosing the right one for a particular application is a major design problem. • These methods determine how data is organized, retrieved, and modified within a file system. Understanding file access methods is crucial for efficient data management and system performance. • In this article, we are going to discuss different types of methods to access the file. There are three ways to access a file in a computer system: 1. Sequential-Access 2. Direct Access 3. Index sequential Method	10M	L4	C225.5

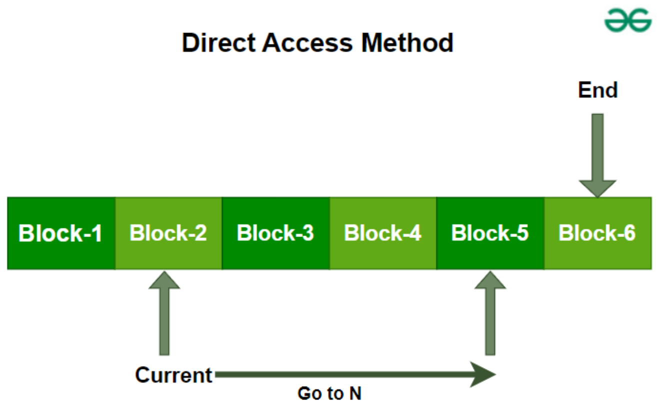
Sequential Access:

- It is the simplest access method. Information in the file is processed in order, one record after the other.
- This mode of access is by far the most common; for example, the editor and compiler usually access the file in this fashion.
- Read and write make up the bulk of the operation on a file.
- A read operation -read next- reads the next position of the file and automatically advances a file pointer, which keeps track of the I/O location.
- Similarly, for the -write next- append to the end of the file and advance to the newly written material.



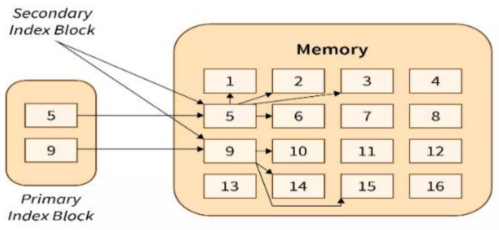
Direct Access Method:

- Another method is direct access method also known as relative access method.
- A fixed-length logical record that allows the program to read and write record rapidly in no particular order.
- The direct access is based on the disk model of a file since disk allows random access to any file block.
- For direct access, the file is viewed as a numbered sequence of block or record.
- Thus, we may read block 14 then block 59, and then we can write block 17.
- There is no restriction on the order of reading and writing for a direct access file.
- A block number provided by the user to the operating system is normally a relative block number, the first relative block of the file is 0 and then 1 and so on.



Index Sequential method:

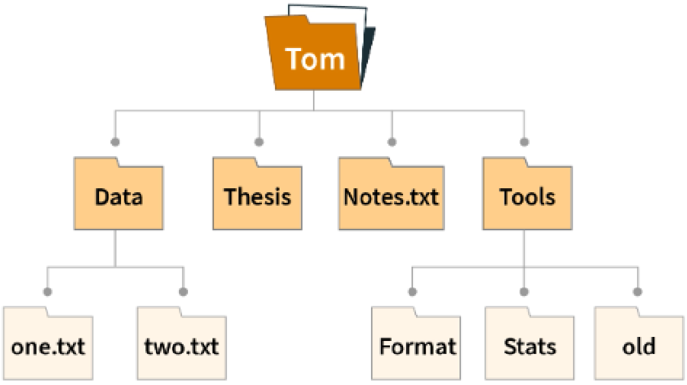
- It is the other method of accessing a file that is built on the top of the sequential access method.
- These methods construct an index for the file.
- The index, like an index in the back of a book, contains the pointer to the various blocks.
- To find a record in the file, we first search the index, and then by the help of pointer we access the file directly.



Indexed Sequential Access Of File

	b) Explain in detail about different File Operations. **File System concepts:** ♦ A file system in OS dictates how the contents of a storage medium are stored and organized. ♦ These storage media (such as secondary memory, external drives, etc) could be computer secondary memory, flash memory, etc. ♦ The contents are either files or directories. ♦ Most of the time, a storage device has a number of partitions. ♦ Each of these partitions is formatted with an empty filesystem for that device. ♦ A filesystem helps in separating the data on the storage into comparatively smaller and simpler segments.	10M	L4	C225.6

- These chunks are files and directories.
- The filesystem also provides for storing data related to files, such as their name, extension, permissions, etc.



a) Attributes:

A file is named, for the convenience of its human users, and is referred to by its name. A name is usually a string of characters, such as example.c. Some systems differentiate between uppercase and lowercase characters in names,

A file's attributes vary from one operating system to another but typically consist of these:

- Name - The symbolic file name is the only information kept in human readable form.
- Identifier - This unique tag, usually a number, identifies the file within the file system; it is the non-human-readable name for the file.
- Type - This information is needed for systems that support different types of files.
- Location - This information is a pointer to a device and to the location of the file on that device.
- Size - The current size of the file (in bytes, words, or blocks) and possibly the maximum allowed size are included in this attribute.
- Protection - Access-control information determines who can do reading, writing, executing, and so on.
- 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.

b) File operations:

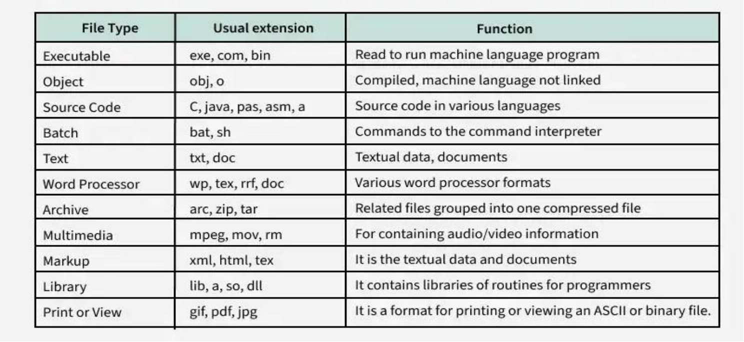
A file is an abstract data type. To define a file properly, we need to consider the operations that can be performed on files. The operating system can provide system calls to create, write, read, reposition, delete, and truncate files. Let's examine what the operating system must do to perform each of these six basic file operations. It should then be easy to see how other similar operations, such as renaming a file, can be implemented.

- **Creating a file:**
  - Two steps are necessary to create a file. First, space in the file system must be found for the file.

- **Writing a file.**
  - To write a file, we make a system call specifying both the name of the file and the information to be written to the file.
  - Given the name of the file, the system searches the directory to find the file’s location.
  - The system must keep a write pointer to the location in the file where the next write is to take place.
  - The write pointer must be updated whenever a write occurs.
- **Reading a file.**
  - To read from a file, we use a system call that specifies the name of the file and where (in memory) the next block of the file should be put.
  - Again, the directory is searched for the associated entry, and the system needs to keep a read pointer to the location in the file where the next read is to take place.
  - Once the read has taken place, the read pointer is updated.
  - Because a process is usually either reading from or writing to a file, the current operation location can be kept as a per-process current file - position pointer.
  - Both the read and write operations use this same pointer, saving space and reducing system complexity.
- **Repositioning within a file.**
  - The directory is searched for the appropriate entry, and the current-file-position pointer is repositioned to a given value.
  - Repositioning within a file need not involve any actual I/O. This file operation is also known as a file seek.
- **Deleting a file.**
  - To delete a file, we search the directory for the named file. Having found the associated directory entry,
  - we release all file space, so that it can be reused by other files, and erase the directory entry.
- **Truncating a file.**
  - The user may want to erase the contents of a file but keep its attributes.
  - Rather than forcing the user to delete the file and then recreate it, this function allows all attributes to remain unchanged.

c) File Types:

- If an operating system recognizes the type of a file, it can then operate on the file in reasonable ways.
- A common technique for implementing file types is to include the type as part of the file name.
- The name is split into two parts-a name and an extension, usually separated by a period character.



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

d) File Structure:

- A File Structure needs to be predefined format in such a way that an operating system understands
- It has an exclusively defined structure, which is based on its type.
- Three types of files structure in OS:
- A text file: It is a series of characters that is organized in lines.
- An object file: It is a series of bytes that is organized into blocks.
- A source file: It is a series of functions and processes.

c) Explain in detail about Linked Allocation method of a File.

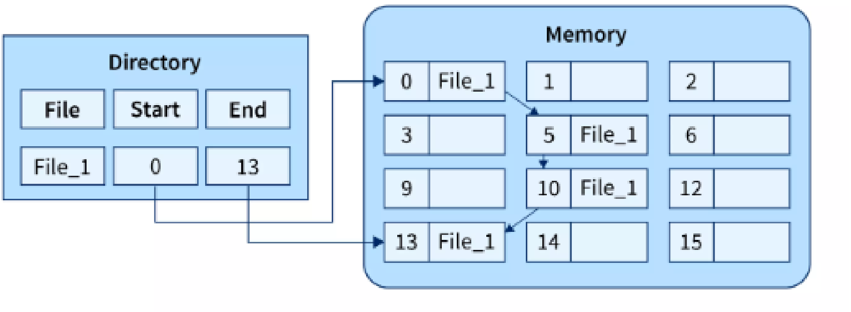
10M

b) Linked File Allocation:

- ♦ The Linked file allocation overcomes the drawback of contiguous file allocation.
- ♦ Here the file which we store on the hard disk is stored in a scattered manner according to the space available on the hard disk.
- ♦ Now, you must be thinking about how the OS remembers that all the scattered blocks belong to the same file.
- ♦ So as the name linked File Allocation suggests, the pointers are used to point to the next block of the same file, therefore along with the entry of each file each block also stores the pointer to the next block.

L4

C225.5



- In the above image on the right, we have a memory diagram where we can see memory blocks.
- On the left side, we have a directory where we have the information like the address of the first memory block and the last memory block.
- In this allocation, the starting block given is 0 and the ending block is 15, therefore the OS searches the empty blocks between 0 and 15 and stores the files in available blocks, but along with that it also stores the pointer to the next block in the present block.
- Hence it requires some extra space to store that link.