Operating System

(A)

1. **What is an Operating system?**
   Ans.
   An Operating System (OS) is an interface between a computer user and computer hardware. An operating system is a software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.
2. **Write the components of operating system.**
   Ans.
   An operating system consists of the following components:

   - Management of processes
   - File Administration
   - Network Management
   - Main Memory Management
   - Management of Secondary Storage
   - Management of I/O Devices
   - Security Management
   - Command Interpreter System

3. **What is a distributed system**? **Explain why distributed system are desirable?**
   Ans.
   Distributed System:
   Distributed System is a collection of autonomous computer systems that are physically separated but are connected by a centralized computer network that is equipped with distributed system software. The autonomous computers will communicate among each system by sharing resources and files and performing the tasks assigned to them.
   Example:
   Telephone and cellular networks are also examples of distributed networks.

   Distributed system's desirability:
   Distributed systems offer a number of advantages over monolithic, or single, systems, including:

   Greater flexibility: It is easier to add computing power as the need for services grows. In most cases today, you can add servers to a distributed system on the fly.
   Reliability: A well-designed distributed system can withstand failures in one or more of its nodes without severely impacting performance. In a monolithic system, the entire application goes down if the server goes down.
   Enhanced speed: Heavy traffic can bog down single servers when traffic gets heavy, impacting performance for everyone. The scalability of distributed databases and other

distributed systems makes them easier to maintain and also sustain high-performance levels.

Geo-distribution: Distributed content delivery is both intuitive for any internet user, and vital for global organizations.

That's why, **distributed system are desirable.**

4. **Classify operating system? Describe multiprogramming and time sharing operating system.**

   Ans.

   Operating systems can be classified as follows:

   i. **Multi-user:** is the one that concede two or more users to use their programs at the same time. Some of O.S permits hundreds or even thousands of users simultaneously.
   ii. **Single-User:** just allows one user to use the programs at one time.
   iii. **Multiprocessor:** Supports opening the same program more than just in one CPU.
   iv. **Multitasking:** Allows multiple programs running at the same time.
   v. **Single-tasking**: Allows different parts of a single program running at any one time.
   vi. **Real time**: Responds to input instantly. Operating systems such as DOS and UNIX, do not work in real time.

   Multiprogramming Operating System:

   In the multi-programming system, one or multiple programs can be loaded into its main memory for getting to execute. Main objective of multiprogramming is to manage entire resources of the system. Multiprogramming operating system has ability to execute multiple programs with using of only one processor machine . One example is User can use MS-Excel , download apps, transfer data from one point to another point, Firefox or Google Chrome browser, and more at a same time.

   Time-Sharing Operating System:

   Time-Sharing Operating System requires high specification hardware and uses lots of resources. CPU idle time is reduced in Time-sharing operating system. Reliability, Data mixing, and communication are some problems faced in Time-sharing operating systems. UNIX and LINUX are examples of Time-sharing operating systems.

5. **Discuss the functions of operating system.**

   Ans.

   The main operation performed by operating system is to carries out is the allocation of resources and services, such as allocation of the following −

- Memory
- Devices
- Processors
- Information

The operating system includes programs that are helpful to manage these resources, such as a traffic controller, a scheduler, memory management module, I/O programs, and a file system.

Functions of Operating Systems

Let us discuss the function of the operating system (OS) in detail.

Security:

The operating system uses a password protection to protect user data it also prevents unauthorized access to programs and user data, but for external functionality we need to install malware software to protect the system.

Control over system performance:

The operating system monitors overall system setup to help in improving the performance and it also records the response time between service requests and system response so that it has a complete view of the system. This can help improve performance by providing important information that is needed at the time of troubleshooting problems.

Job Accounting:

Operating systems always keep track of time and resources that are used by various tasks and users, this information can be used to track resource usage for a particular user or a group of users.

Error detecting aids:

Operating systems constantly monitor the system which helps us to detect errors and also avoid the malfunctioning of computer systems.

Coordination between other software and users:

Operating systems help in coordinate and assign interpreters, compilers, assemblers, and other software to the various users of the computer systems.

Memory Management:

The operating system controls the primary memory or main memory. Primary memory is a large array of bytes or words where each byte or word is assigned a certain address. It is a fast storage, and it can be accessed directly by the CPU which is present inside the system. If a program wants to be executed, it should be first loaded in the main memory.

The following activities are performed by operating system for memory management –

- It keeps track of primary memory.
- Memory addresses that have already been allocated and the memory addresses of the memory that has not yet been used.
- In multiprogramming, the OS decides for how long the process must stay and the order in which processes are granted access to memory.
- It allocates the memory to a process when the process requests it and deallocates the memory when the process has terminated.

Processor Management:

The OS manages the order in which processes have access to the processor, and how much processing time that each process must stay in the multiprogramming environment. This is called process scheduling.

The following activities are performed by operating system for processor management –

- Keeps track of the status of processes.
- The program to track the status is known as traffic controller.

- It allocates the CPU and deallocates the processor when it is not required.

Device Management:

An OS manages device communication through respective drivers.
The following activities are performed by the operating system for device management.

- Keeping track of all devices connected to the system.
- The OS designates a program that is responsible for every device which is called the Input/output controller.
- It decides which process gets access to which device and for how long. It then allocates the devices in an effective and efficient way and de-allocates devices when they are not required.

File Management:

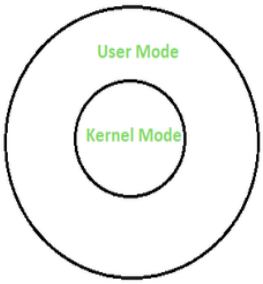
A file system is arranged into directories for efficient navigation and usage. These directories contain other directories and other files.

The following activities are performed by operating system for file management activities −

- It keeps track of where information is stored, user access settings and status of every file and more.
- These facilities are called the file system

6. **What do you understand by user mode and kernel mode of operations?**
   Ans.
   User Mode: When a Program is booted up on an Operating system let's say windows, then it launches the program in user mode. And when a user-mode program requests to run, a process and virtual address space (address space for that process) is created for it by windows. User-mode programs are less privileged than user-mode applications and are not allowed to access the system resources directly. For instance, if an application under user-mode wants to access system resources, it will have to first go through the Operating system kernel by using syscalls.

   Kernel Mode: The kernel is the core program on which all the other operating system components rely, it is used to access the hardware components and schedule which processes should run on a computer system and when, and it also manages the application software and hardware interaction. Hence it is the most privileged program, unlike other programs it can directly interact with the hardware. When programs running under user mode need hardware access for example webcam, then first it has to go through the kernel by using a syscall, and to carry out these requests the CPU switches from user mode to kernel mode at the time of execution. After finally completing the execution of the process the CPU again switches back to the user mode.



7. **Why we build distributed system?4-10.**
   Ans.
   Distributed systems provide scalability and improved performance in ways that monolithic systems can't, and because they can draw on the capabilities of other computing devices and processes, distributed systems can offer features that would be difficult or impossible to develop on a single system.
   That's why ,we **build distributed system.**
8. **What is the main advantage of multiprogramming.-3-10.**
   Ans.
   Multiprogramming is a system where multiple programs are executed on a computer at the same time. The main advantage of multiprogramming is that it allows for better utilization of the computer's resources and results in more efficient use of the computer. With multiprogramming, the CPU is never idle as it is constantly switching between multiple programs, thereby maximizing its use. This results in improved performance and a more responsive system as multiple tasks can be executed simultaneously. Additionally, multiprogramming makes it possible to execute multiple programs concurrently, which is especially useful in environments where multiple users are sharing a single computer. By allowing multiple programs to run at the same time, multiprogramming helps to improve the overall performance and efficiency of the computer system.

9. **What is the difference between multiprogramming and multiprocessing?**
   Ans.

Sr. No.	Multiprocessing	Multiprogramming
1	Multiprocessing refers to processing of multiple processes at same time by multiple CPUs.	Multiprogramming keeps several programs in main memory at the same time and execute them concurrently utilizing single CPU.
2	It utilizes multiple CPUs.	It utilizes single CPU.
3	It permits parallel processing.	Context switching takes place.
4	Less time taken to process the jobs.	More Time taken to process the jobs.
5	It facilitates much efficient utilization of devices of the computer system.	Less efficient than multiprocessing.

10. Write the operating system activities in connection with memory management and storage management.

Ans.

The three major activities of the operating system with regard to memory management are:

- Keeping track of which parts of memory are currently being used and by whom
- Deciding which processes are to be loaded into memory when memory space becomes available
- Allocating and deallocating memory space as needed

The three major activities of the operating system with regard to secondary storage management are:

- Free space management
- Storage allocation
- Disk scheduling

11. What are the purpose of an operating system?

Ans.

The purpose of an operating system (OS) is to provide a platform for managing and controlling computer hardware and software resources and to act as an interface between the user and the hardware. Some of the key functions of an operating system include:

Resource management: The OS manages and allocates resources such as memory, processing power, and storage, among multiple applications and processes.

Memory management: The OS manages the physical and virtual memory of the system, providing a virtual address space for each process and ensuring that processes have sufficient memory to run.

Process management: The OS creates, schedules, and manages processes, ensuring that processes receive the necessary resources and have access to the system in a controlled manner.

File management: The OS provides a file system for organizing and storing data and manages access to the file system, ensuring data integrity and security.

Security: The OS provides security features to prevent unauthorized access and protect against malicious software, ensuring that the system and user data are protected.

User interface: The OS provides a graphical user interface (GUI) and/or command-line interface (CLI) for users to interact with the system and launch applications.

Hardware management: The OS communicates with and manages the underlying hardware, ensuring that hardware components work together seamlessly and that the system runs smoothly.

The purpose of an operating system is to provide a stable, efficient, and user-friendly environment for running applications and performing computing tasks.

12. Explain the reasons for building distributed systems.

Ans.

Distributed systems are computer systems that consist of multiple, interconnected nodes that work together to achieve a common goal. There are several reasons for building distributed systems, including:

Scalability: Distributed systems can scale horizontally by adding more nodes, allowing the system to handle increasing loads and grow as needed.

Reliability: By distributing resources and tasks among multiple nodes, a distributed system can be more resilient to failures and ensure that the system continues to function even if individual nodes fail.

Performance: Distributed systems can distribute processing and storage resources among multiple nodes, improving system performance and response time.

Flexibility: Distributed systems can be designed in a flexible and modular manner, making it easier to add new features, make changes to the system, and adapt to new requirements.

Resource sharing: Distributed systems allow resources such as processing power, memory, and storage to be shared among multiple nodes, improving resource utilization and reducing waste.

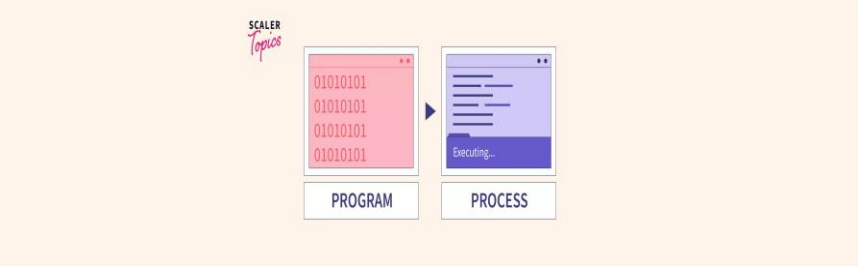
Geographical distribution: Distributed systems can support nodes located in different physical locations, allowing users to access resources and services from anywhere in the world.

Cost-effectiveness: By using commodity hardware and avoiding the need for expensive, centralized systems, distributed systems can be more cost-effective and provide a better return on investment.

The reasons for building distributed systems are driven by the need to improve system scalability, reliability, performance, flexibility, and cost-effectiveness, while also providing access to shared resources and services from anywhere in the world.

(B)

1. **What do you mean by job and process?**
   Ans.
   Job:
   Job means an application program and it is not a system program.
   For example, a job could be the running of an application program such as a weekly payroll program.
   Process:
   A process refers to a program under execution. This program may be an application or system program.
   For example, when you want to search something on web then you start a browser. So, this can be process.
2. **Define process and program.**
   Ans.



   Process:
   A process is an active instance of the program which is started when the program is executed
   Program:
   A program is a passive entity that contains the set of codes required to perform a certain task.

   Once a program is executed, a process is started by the program. The process executes the instructions written in the program.
3. **What are the activities of process management?**
   Ans.

   The five major activities of the operating system with regard to process management are:

- The creation and deletion of both user and system processes
- The suspension and resumption of processes
- The provision of mechanisms for process synchronization
- The provision of mechanisms for process communication
- The provision of mechanisms for deadlock handling

4. **Describe different status of a process.**

   Ans.

   The process executes when it changes the state. The state of a process is defined by the current activity of the process.

   Each process may be in any one of the following states −

   New − The process is being created.
   Running − In this state the instructions are being executed.
   Waiting − The process is in waiting state until an event occurs like I/O operation completion or receiving a signal.
   Ready − The process is waiting to be assigned to a processor.
   Terminated − the process has finished execution.

5. **Briefly write the activities of operating system in connection with process management.**

   Ans.

   The five major activities of the operating system with regard to process management are:

   - The creation and deletion of both user and system processes
   - The suspension and resumption of processes
   - The provision of mechanisms for process synchronization
   - The provision of mechanisms for process communication
   - The provision of mechanisms for deadlock handling

6. **Describe process control blocks.**

   Ans.

   Process Control Block is a data structure that contains information of the process related to it. The process control block is also known as a task control block, entry of the process table, etc.

   It is very important for process management as the data structuring for processes is done in terms of the PCB. It also defines the current state of the operating system.

Process Control Block (PCB)

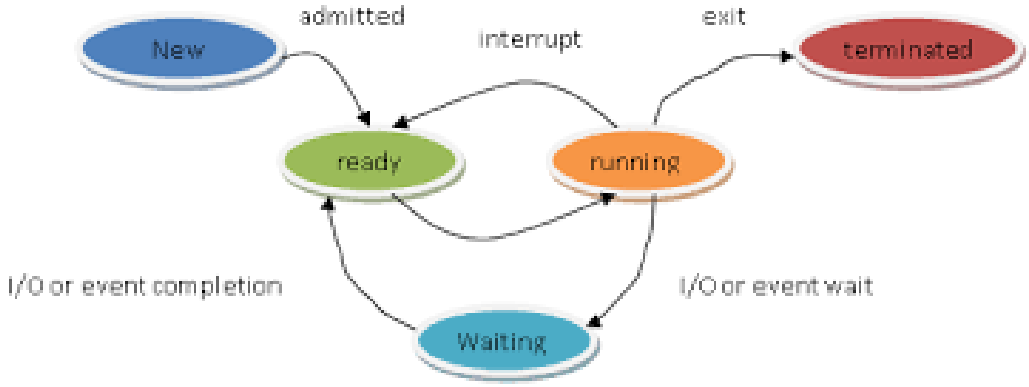
7. **Define process. What are possible states of process.**
   Ans.
   A process is an active instance of the program which is started when the program is executed.
   The process executes when it changes the state. The state of a process is defined by the current activity of the process.
   Each process may be in any one of the following states −

- **New** − The process is being created.
- **Running** − In this state the instructions are being executed.
- **Waiting** − The process is in waiting state until an event occurs like I/O operation completion or receiving a signal.
- **Ready** − The process is waiting to be assigned to a processor.
- **Terminated** − the process has finished execution.

8. Define context switch.
   Ans.
   Context Switch involves storing the context or state of a process so that it can be reloaded when required and execution can be resumed from the same point as earlier. This is a feature of a multitasking operating system and allows a single CPU to be shared by multiple processes.
9. **Define process. Draw the process state diagram and explain it.**
   Ans.
   Process:
   A process is an active instance of the program which is started when the program is executed.

   The process executes when it changes the state. The state of a process is defined by the current activity of the process.
   A process changes its state during its execution. Each process may be in one of the following states:

i. **New:** when a new process is being created.
ii. **Running:** A process is said to be in running state when instructions are being executed.
iii. **Waiting:** The process is waiting for some event to occur (such as an I/O operation).
iv. **Ready:** The process is waiting for processor.
v. **Terminated:** The process has finished execution.



10. .What is the difference between short term, medium term and long term scheduler.
Ans.

S. No.	Long-Term Scheduler	Short-Term Scheduler	Medium-Term Scheduler
1	It is a Job Scheduler	It is a CPU Scheduler	It is a process swapping scheduler
2	It takes process from the job pool	It takes process from the ready state	It takes process from running or wait/dead state
3	Its speed is lesser than short-term scheduler	It is fastest among the two other schedulers	Its speed is in between long-term and short-term
4	It controls the degree of multiprogramming	It has less control over the degree of multiprogramming	It reduces the degree of multiprogramming

AfterAcademy

11. Write the reasons of process cooperation.
Ans.

There may be many reasons for the requirement of cooperating processes. Some of these are given as follows –

Modularity:

Modularity involves dividing complicated tasks into smaller subtasks. These subtasks can completed by different cooperating processes. This leads to faster and more efficient completion of the required tasks.

Information Sharing:

Sharing of information between multiple processes can be accomplished using cooperating processes. This may include access to the same files. A mechanism is required so that the processes can access the files in parallel to each other.

Convenience:

There are many tasks that a user needs to do such as compiling, printing, editing etc. It is convenient if these tasks can be managed by cooperating processes.

Computation Speedup:

Subtasks of a single task can be performed parallely using cooperating processes. This increases the computation speedup as the task can be executed faster. However, this is only possible if the system has multiple processing elements.

12. What are the difference between process and thread.

Ans.

Process	Thread
A process is the instance of a program executed by one or many threads.	A thread is a basic unit of CPU utilisation, which consists of its own thread ID, a program counter, a register and a stack.
A process may contain multiple threads depending on the operating system.	A thread is the smallest unit of execution within a process.
In a multiprocessing environment, multiple processes do not share resources such as memory with each other.	Multiple threads of a given process running concurrently can share resources such as memory with each other.
Processes take more time for context switching.	Threads take less time for context switching.
Processes take more time for creation and termination.	Threads take lesser time for creation and termination.
Processes consume more resources.	Threads consume fewer resources.
No other process can execute until the first process gets unblocked.	Another thread in the same task can run while one thread is blocked and waiting.
Process communication is complex.	Thread communication is much easier and efficient.

13. What do you mean by thread? Is there any advantage of thread over process?

Ans.

Thread:

A thread is a single sequential flow of control within a program. The real excitement surrounding threads is not about a single sequential thread. Rather, it's about the use of multiple threads running at the same time and performing different tasks in a single program.

Advantage of thread over process :

Unlike processes, threads share data and information. They do, however, have their own stack.

We can create more than one thread by using just one system call. To further simplify things, thread management requires few or even no system calls because we don't need extra mechanisms such as IPC to maintain communication between threads.

14. Write the operations that perform operating system on process.
Ans.

Operations on Process

The two main operations perform on Process are as follows –

Process Creation

There should be four principle events which cause processes to be created.

System initialization

Numerous processes are created when an operating system is booted. Some of them are −

- **Foreground processes** − Processes that interact with users and perform work for them.
- **Background processes** − It is also called as daemons and not associated with particular users, but instead has some specific function.

  Execution of a process-creation system call by a running process

The running process will issue system calls to create one or more new processes to help it do its job.

A user request to create a new process

A new process is created with the help of an existing process executing a process creation system call.

In UNIX, the system call which is used to create a new process is fork()

In Windows, CreateProcess(), which has 10 parameters to handle both process creation and loading the correct program into the new process.

Initiation of a batch job

Users are going to submit batch jobs to the system.

When the operating system creates a new process and runs the next job from the input queue in it.

Process Termination

Process is going to be terminated by a call to kill in UNIX or Terminate Process in windows.

Process is terminated due to following reason −

- **Normal exit** − Most processes terminate when they have completed their work and execute a system call to exit.
- **Error exit** − The third type of error occurs due to program bugs like executing an illegal instruction, referencing, or dividing by zero.
- **Fatal exit** − A termination of a process occurs when it discovers a fatal error.
- **Killed by another process** − A process executes a system call to kill some other process

15. Explain the situation when CPU switches from a process to another process.

Ans.

A context switching is a process that involves switching of the CPU from one process or task to another. In this phenomenon, the execution of the process that is present in the running state is suspended by the kernel and another process that is present in the ready state is executed by the CPU.

Context Switching Triggers

There are three major triggers for context switching. These are given as follows −

- **Multitasking**: In a multitasking environment, a process is switched out of the CPU so another process can be run. The state of the old process is saved and the state of the new process is loaded. On a pre-emptive system, processes may be switched out by the scheduler.

- **Interrupt Handling:** The hardware switches a part of the context when an interrupt occurs. This happens automatically. Only some of the context is changed to minimize the time required to handle the interrupt.

- **User and Kernel Mode Switching:** A context switch may take place when a transition between the user mode and kernel mode is required in the operating system.

Context Switching Steps

The steps involved in context switching are as follows −

- Save the context of the process that is currently running on the CPU. Update the process control block and other important fields.
- Move the process control block of the above process into the relevant queue such as the ready queue, I/O queue etc.
- Select a new process for execution.
- Update the process control block of the selected process. This includes updating the process state to running.
- Update the memory management data structures as required.
- Restore the context of the process that was previously running when it is loaded again on the processor. This is done by loading the previous values of the process control block and registers.

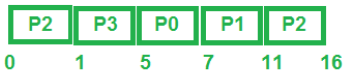
Context Switching Cost

Context Switching leads to an overhead cost because of TLB flushes, sharing the cache between multiple tasks, running the task scheduler etc. Context switching between two threads of the same process is faster than between two different processes as threads have the same virtual memory maps. Because of this TLB flushing is not required.

(C)

1. **What do you mean by preemptive and non-preemptive process?**
   Ans.
   Preemptive Process:
   Preemption as used with respect to operating systems means the ability of the operating system to preempt (that is, stop or pause) a currently scheduled task in favour of a higher priority task. The resource being scheduled may be the processor or I/O, among others.

Process	Arrival Time	CPU Burst Time (in millisec.)
P0	3	2
P1	2	4
P2	0	6
P3	1	4

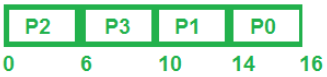


Preemptive Scheduling

Non-preemptive process:

Non-preemptive Scheduling is used when a process terminates, or a process switches from running to the waiting state. In this scheduling, once the resources (CPU cycles) are allocated to a process, the process holds the CPU till it gets terminated or reaches a waiting state.

Process	Arrival Time	CPU Burst Time (in millisec.)
P0	3	2
P1	2	4
P2	0	6
P3	1	4



Non-Preemtive Scheduling

2. **Define turnaround time, waiting time, and response time.**

Ans.

Turnaround time:

- TAT refers to the total time interval present between the time of process submission and the time of its completion.
- The difference between the time of completion and the time of arrival is known as the Turn Around Time of the process.

CT (Completion Time) – It is the exact time when a process completes the execution part.
AT (Arrival Time) – It is the time of arrival of a process in its ready state (before its execution).
Thus, **CT – AT = TAT**

Waiting time:

WT refers to the total time that a process spends while waiting in a ready queue until it gets the CPU (for the I/O completion).

- WT refers to the total time that a process spends while waiting in a ready queue before reaching the CPU.
- The difference between (time) of the turn around and burst time is known as the waiting time of a process.

BT (Burst Time) – It is the total time that a process requires for its overall execution.
Thus, **TAT – BT = WT**
Now, we can also easily calculate the Turn Around Time using the Burst Time and the Waiting Time.
Here, **BT + WT = TAT**
Response time:

Response time, in the context of computer technology, is the elapsed time between an inquiry on a system and the response to that inquiry. Used as a measurement of system performance, response time may refer to service requests in a variety of technologies. Low response times may be critical to successful computing.

3. **Describe SJF scheduling for preemptive and non-preemptive process.**
   Ans.

   Shortest Job First (SJF) is a scheduling algorithm that assigns the shortest tasks to the processor first.

   Non-preemptive SJF: In this type of SJF, once a task has started executing, it cannot be preempted by a shorter task. It is used when the execution time of a task is known beforehand and is often used in batch systems where the tasks have a well-defined execution time.

   Preemptive SJF: In this type of SJF, the executing task can be preempted by a shorter task that arrives later. It is used when the execution time of a task is not known beforehand and it is used in real-time systems where tasks have varying execution times. This type of SJF is more complex as it requires the ability to interrupt the execution of a task to start a shorter one
4. **Write the function of dispatcher module.**
   Ans.

   The dispatcher is a module in an operating system's kernel that is responsible for allocating the CPU to processes and threads. It performs the following functions:

   Context Switch: The dispatcher saves the context of the current process and loads the context of the next process to be executed.

   Scheduling Decisions: The dispatcher makes scheduling decisions based on the scheduling algorithm being used (e.g. SJF, Round Robin, etc.). It selects the next process to run and allocates the CPU to that process.

   Load Balancing: The dispatcher ensures that the load on the CPU is balanced across all processes. It monitors the utilization of the CPU and adjusts the scheduling algorithm to balance the load, if necessary.

Interrupt Handling: The dispatcher handles interrupts and exceptions that occur during the execution of processes. It saves the context of the interrupted process and starts executing a suitable handler.

Process Management: The dispatcher manages the process table, which contains information about all processes in the system. It adds, deletes, and updates entries in the process table as processes are created, terminated, or changed.

Overall, the dispatcher is responsible for allocating the CPU to processes and managing the context switch between processes. It is a critical component of the operating system's kernel and has a significant impact on system performance and responsiveness.

5. **Describe the criteria that uses to select a schedule algorithm.**
   Ans.

   When selecting a scheduling algorithm, several criteria are considered, including:

   CPU Utilization: The scheduling algorithm should aim to keep the CPU as busy as possible, to maximize its utilization.

   Throughput: The scheduling algorithm should maximize the number of processes that are completed per unit time, to improve the throughput of the system.

   Turnaround Time: The scheduling algorithm should minimize the time it takes for a process to complete, to reduce the turnaround time.

   Waiting Time: The scheduling algorithm should minimize the time that processes spend waiting in the ready queue, to reduce the waiting time.

   Response Time: The scheduling algorithm should minimize the time it takes for a process to receive its first response, to improve the response time of the system.

   Fairness: The scheduling algorithm should ensure that each process gets a fair share of the CPU, to avoid starvation and ensure fairness.

   Priority: The scheduling algorithm should consider the priority of processes and allocate the CPU to high-priority processes first, to meet real-time requirements.

   Scalability: The scheduling algorithm should scale well as the number of processes increases, to maintain good performance even under heavy load.

   Overhead: The scheduling algorithm should minimize overhead, such as context switching overhead and the overhead of maintaining data structures.

Different scheduling algorithms may perform better or worse in different situations, so the criteria used to select a scheduling algorithm depend on the specific requirements of the system. For example, real-time systems may prioritize response time and priority over other criteria, while batch systems may prioritize throughput and utilization.

6. **Write the advantages of process cooperation.**
Ans.

There are several advantages of process cooperation:

Improved resource utilization: By sharing resources, multiple processes can utilize a single resource more efficiently and effectively, improving overall system performance.

Improved reliability: When processes work together, the system becomes more resilient to individual process failures. In case one process fails, the other processes can compensate and continue to function normally.

Better system response: By dividing a complex task into smaller, cooperative processes, the system can respond to requests and complete tasks faster.

More flexible system design: Process cooperation allows the system to be designed in a modular and flexible way, making it easier to add new features and make changes to existing ones.

Improved security: By limiting the access and privilege of individual processes, cooperative processes can enhance the security of the system and prevent malicious processes from compromising the system.

Better load balancing: By distributing the workload among multiple cooperative processes, the system can balance the load and reduce the risk of overloading a single process.

Simplified development: Process cooperation can simplify the development process by breaking down complex tasks into smaller, more manageable components, making it easier to develop, test, and maintain the system.

7. **Briefly write the direct and indirect method for inter process communication.**
Ans.

Inter-process communication (IPC) refers to the exchange of data between processes. There are two main methods for inter-process communication: direct and indirect methods.

Direct method: In this method, processes communicate directly with each other using shared memory or message passing.

Shared memory: In this method, two or more processes share a common area of memory for communication. The processes can read and write to this shared memory, allowing them to exchange information directly.

Message passing: In this method, processes send messages to each other through a message queue. The receiver process retrieves messages from the queue and processes them.

Indirect method: In this method, processes communicate indirectly through an intermediate agent such as a file, pipe, or socket.

File communication: In this method, processes write to and read from a common file.

Pipe communication: In this method, processes communicate through a pipe, which is a unidirectional communication channel.

Socket communication: In this method, processes communicate through a socket, which is a bidirectional communication endpoint for network communication.

Each method has its own advantages and disadvantages, and the choice of method depends on the specific requirements of the application and the operating system.

(D)

1. When does CPU scheduling decision take place?

Ans.

The CPU scheduling decision takes place when the operating system has to choose which process to execute next. This decision is made by the CPU scheduler, which is a component of the operating system. The CPU scheduler makes this decision based on the scheduling algorithm in use and the state of the processes in the system.

There are several key moments when the CPU scheduling decision takes place:

- When a process switches from the running state to the waiting state: When a process completes its execution or when it has to wait for an event, such as the completion of an I/O operation, the CPU scheduler makes a decision on which process to execute next.
- When a process switches from the waiting state to the ready state: When an event that a process was waiting for occurs, the process becomes ready to run again. The CPU scheduler makes a decision on which process to execute next.

- When a new process is created: When a new process is created, the CPU scheduler decides whether to allocate CPU resources to the new process or to continue executing the current process.

  The CPU scheduling decision takes place at these key moments in order to ensure that the CPU is being used efficiently and that the system is responsive to the needs of the user and the processes running on the system.

2 .**Explain the criteria for comparing CPU scheduling algorithm.**

Ans.

There are several criteria to compare CPU scheduling algorithms, including:

CPU utilization: The algorithm should keep the CPU as busy as possible, maximizing utilization.

Throughput: The number of processes that are completed per unit time, high throughput is desired.

Turnaround time: The time between the submission of a process and its completion, the lower the turnaround time, the better.

Waiting time: The time a process has to wait in the ready queue before it is scheduled to run, the lower the waiting time, the better.

Response time: The time from when a request was submitted until the first response is produced, the lower the response time, the better.

Fairness: The algorithm should ensure that each process gets a fair share of the CPU time, without one process monopolizing the CPU.

Overhead: The amount of CPU time spent on scheduling and managing processes, the lower the overhead, the better.

3 .List the factors that affect a scheduling mechanism of processes.
Ans.

The following are the factors that affect a scheduling mechanism of processes:

Arrival time: The arrival time of a process affects its priority and position in the queue. Processes that arrive earlier may have a higher priority compared to processes that arrive later.

CPU Burst Time: The CPU burst time of a process is the amount of time it requires to complete its execution. A process with a short CPU burst time will complete faster compared to a process with a long CPU burst time.

Priority: A process's priority affects its order in the queue. Higher priority processes are executed before lower priority processes.

Memory requirements: The memory requirements of a process can affect its priority and scheduling. Processes that require a large amount of memory may have a lower priority compared to processes that require less memory.

I/O requests: A process with frequent I/O requests may have to wait for I/O operations to complete, which can affect its scheduling.

Multitasking requirements: The multitasking requirements of a process can affect its scheduling. Processes that require multiple tasks to be performed simultaneously may have a higher priority compared to processes that can be executed sequentially.

Deadlines: A process with a deadline may have a higher priority compared to processes without deadlines.

Preemption: The ability of a scheduling mechanism to preempt a running process and execute a higher priority process can affect the scheduling of processes.

Load on the system: The load on the system, including the number of processes and the amount of available resources, can affect the scheduling of processes.

4. Define the following: i) Dispatcher-1-10; ii) Throughput iii) CPU utilization and iv) Response time.
Ans.

i) **Dispatcher:** The dispatcher is a component of the operating system that is responsible for switching the CPU from one process to another. It performs context switching, which involves saving the state of the current process and loading the state of the next process to be executed.

ii) Throughput: Throughput refers to the number of processes that are completed in a given time frame. It is a measure of the efficiency of the CPU scheduling mechanism and is expressed as the number of processes per unit time.

iii) CPU utilization: CPU utilization refers to the amount of CPU time used effectively. It is a measure of how busy the CPU is and is expressed as a percentage of the total CPU time available.

iv) Response time: Response time refers to the time taken by the system to respond to a request made by a process. It is the time elapsed between the submission of a request and the completion of the response. Response time is an important factor in the evaluation of the performance of a CPU scheduling mechanism as it affects the perceived performance of the system by the end-user.

5. Describe priority scheduling algorithm.

Ans.

Priority Scheduling is a CPU scheduling algorithm in which each process is assigned a priority, and the process with the highest priority is executed first. The priority of a process can be assigned based on various factors such as the process's CPU burst time, memory requirements, I/O requests, and user priority.

The algorithm can be implemented in two ways:

Preemptive Priority Scheduling: In this method, if a process with a higher priority enters the ready queue, the scheduler preempts the currently executing process and switches to the higher priority process.

Non-Preemptive Priority Scheduling: In this method, once a process has been assigned to the CPU, it cannot be preempted until it has completed its execution or is blocked for I/O operations.

If two processes have the same priority, they are executed in a round-robin fashion. The scheduler selects the process with the highest priority from the ready queue for execution.

However, Priority Scheduling can suffer from the problem of priority inversion, where a low-priority process can hold up a high-priority process by holding resources required by the high-priority process. To prevent this, various methods such as aging, priority inheritance, and priority boosting can be used.

Overall, Priority Scheduling is an efficient algorithm that can provide quick response times for high-priority processes while ensuring that low-priority processes receive a fair share of CPU time.

6. Explain Round-Robin algorithm.

Ans.

Round-Robin (RR) is a CPU scheduling algorithm that is designed to distribute CPU time fairly among processes in the ready queue. It operates by dividing the CPU time slice, also known as a time quantum, into equal units, and allocating each process a time quantum in a circular manner.

In the Round-Robin algorithm, the scheduler selects the process at the front of the ready queue and assigns the CPU to it for a fixed time quantum. If the process has not completed its execution within the time quantum, it is moved to the end of the ready queue, and the scheduler selects the next process in the queue for execution. This process continues until all processes have completed their execution.

The time quantum determines the granularity of the algorithm. A large time quantum results in lower overhead and better CPU utilization, but can lead to longer waiting times for processes. A small time quantum results in higher overhead and lower CPU utilization, but ensures that processes receive a fair share of CPU time.

Round-Robin is an efficient algorithm for time-sharing systems, where multiple processes need to share the CPU simultaneously. It provides a good balance between fairness and efficiency, and is widely used in operating systems, especially for scheduling interactive processes.

7.What do you mean by trap?
Ans.

A trap, also known as an exception or a software interrupt, is a mechanism in operating systems that allows a process to transfer control to the operating system kernel. A trap is generated by the process to request a service from the operating system or to indicate an error condition.

When a trap occurs, the operating system saves the current state of the process and transfers control to the appropriate interrupt handler in the kernel. The handler performs the necessary operations and returns control to the process when it has completed its work.

Traps are used to handle a variety of situations, including system calls, I/O operations, page faults, and other errors. By using traps, processes can request services from the operating system without having to switch to kernel mode, which saves time and reduces overhead.

Traps are a fundamental mechanism for implementing the interface between processes and the operating system, and play a key role in maintaining the stability and security of an operating system.

(E)

1 .**Define critical section problem.**

Ans.

The critical section problem in operating systems is a problem of synchronizing access to shared resources by multiple processes or threads. It is a central issue in concurrent programming, as it requires that a process or thread be able to execute a critical section of code that accesses a shared resource, such as a shared data structure or a peripheral device, in an atomic manner, i.e., without interference from other processes or threads. The critical section problem requires a solution that ensures that only one process or thread can enter the critical section at a time, preventing race conditions and preserving data integrity. This synchronization is typically achieved through the use of semaphores, locks, or other synchronization mechanisms provided by the operating system.

2 .Explain all three requirements to solve the critical section problem.

Ans.

There are three requirements to solve the critical section problem in operating systems:

Mutual Exclusion: At any given time, only one process should be allowed to execute in its critical section. No other process should be executing in its critical section at the same time.

Progress: If no process is executing in its critical section, and some processes wish to enter their critical section, then only one of them can enter, and the others have to wait.

Bounded Waiting: A process should not wait indefinitely to enter its critical section. There should exist a bound on the number of times that other processes are allowed to enter their critical section before a process is allowed to enter its critical section.

These three requirements are collectively known as the Mutual Exclusion, Progress, and Bounded Waiting requirements, and they form the basis for solving the critical section problem in operating systems. The solution should ensure that these requirements are met, even in the presence of concurrent access from multiple processes or threads, and in the face of hardware failures, such as power failures, and software failures, such as crashes or bugs.

3 .Write the solution of critical section problem for two processes for software technique.

Ans.

There are several software-based techniques for solving the critical section problem for two processes. One of the simplest and most widely used techniques is the Peterson's Algorithm, which is based on the use of two variables, "turn" and "flag", and a hardware instruction "test and set".

Here is the pseudocode for the Peterson's Algorithm:

Code:

```
process 0:

flag[0] = true

turn = 1

while (flag[1] and turn == 1)

    ; // do nothing

critical section

flag[0] = false
```

process 1:

flag[1] = true

turn = 0

while (flag[0] and turn == 0)

    ; // do nothing

critical section

flag[1] = false

The algorithm ensures mutual exclusion by using the "flag" array to indicate whether a process is interested in entering the critical section. The "turn" variable is used to resolve the tie between two processes that both want to enter the critical section at the same time. The "test and set" instruction is used to atomically set the value of the "flag" array, ensuring that only one process can set its flag to "true" at a time. The "while" loops are used to implement the bounded waiting requirement, ensuring that each process waits for the other to complete its critical section before entering its own critical section.

This algorithm provides a solution to the critical section problem for two processes, but it is not suitable for more than two processes, as the mechanism for resolving the tie between processes becomes more complex.

4 .What is semaphore? Explain how semaphore can be implemented.

Ans.

A semaphore is a synchronization mechanism used in operating systems to control access to shared resources by multiple processes or threads. It is a variable that is used to represent the availability of a shared resource, and it provides operations to change its value in a way that ensures that only one process or thread can access the shared resource at a time.

Semaphores can be implemented in several ways, including using busy waiting, using blocking and unblocking operations, or using a combination of both.

Busy waiting: In this approach, a process repeatedly checks the value of the semaphore and waits until the semaphore becomes available. This approach is also known as "spin lock".

Blocking and unblocking operations: In this approach, a process blocks (waits) when the semaphore is not available, and unblocks (wakes up) when the semaphore becomes available. This approach is implemented using system calls such as wait() and signal() or using similar operations provided by the operating system or programming language.

Semaphores can be binary semaphores or counting semaphores, depending on the value they can represent. Binary semaphores can only have two values, indicating either that a shared resource is available or that it is not, while counting semaphores can have a value greater than two, indicating the number of available units of a shared resource.

Semaphores are widely used in operating systems and concurrent programming, as they provide a convenient mechanism for synchronizing access to shared resources and ensuring that critical sections of code are executed atomically.

5 .**What is meaning of the term of busy waiting.**

Ans.

Busy waiting is a synchronization technique in computer science, where a process or thread repeatedly checks the status of a shared resource, such as a semaphore or a lock, without releasing the CPU. The process or thread continues to spin, or loop, until the resource becomes available, at which point it can proceed to access the resource.

Busy waiting is used as a synchronization mechanism in certain situations, where a process or thread needs to wait for a shared resource to become available, but cannot block, as this would cause the process or thread to release the CPU and consume significant amounts of system resources.

However, busy waiting is often considered to be a suboptimal solution, as it consumes a large amount of CPU cycles and can cause performance issues, such as high CPU utilization, long wait times, and decreased system responsiveness. It is generally preferred to use blocking and unblocking operations, such as wait() and signal(), or other similar mechanisms, which allow a process or thread to block and release the CPU, saving system resources and improving performance.

6 .**What is Starvation? How the system can deal with starvation.**

Ans.

Starvation is a problem in computer science, where a process or thread is prevented from making progress due to a lack of resources or the unavailability of shared resources. In other words, it occurs when a process or thread is unable to obtain the resources it needs to complete its execution, even though these resources are available.

Starvation can occur in various systems, including operating systems, databases, and concurrent programming, and it can have serious consequences, such as degraded system performance, decreased responsiveness, and even system failure.

The system can deal with starvation in several ways, including:

Priority scheduling: In this approach, processes or threads are assigned different priority levels, and the resources are allocated to the highest priority process or thread first. This ensures that lower priority processes or threads are not starved, as they will eventually get access to the resources they need.

Fair sharing: In this approach, resources are shared equitably between processes or threads, ensuring that no process or thread is favored over the others. This can be achieved by using algorithms such as round-robin scheduling or proportional sharing.

Deadlock detection and prevention: In this approach, the system uses algorithms to detect and prevent deadlocks, which can cause starvation by blocking processes or threads. The system can detect deadlocks by monitoring the state of shared resources and processes or threads, and can prevent deadlocks by releasing resources or killing processes or threads that are causing the deadlock.

Time slicing: In this approach, the CPU is allocated to processes or threads for a limited time period, known as a time slice. This ensures that processes or threads are not starved, as they will eventually get access to the CPU.

By using these or similar mechanisms, the system can deal with starvation, ensuring that all processes or threads have access to the resources they need to make progress and complete their execution.

7 .Write how semaphores mitigate criteria section problem.

Ans.

Semaphores are used to solve the critical section problem in operating systems by providing a mechanism to synchronize access to shared resources among multiple processes or threads.

Semaphores provide two operations, wait() and signal(), which allow processes or threads to control access to the shared resource. The wait() operation decrements the value of the semaphore and blocks the process or thread if the value of the semaphore becomes negative, indicating that the shared resource is not available. The signal() operation increments the value of the semaphore and unblocks a blocked process or thread if the value of the semaphore becomes positive, indicating that the shared resource is available.

By using semaphores, processes or threads can ensure that they access the shared resource in a mutually exclusive manner, avoiding race conditions and ensuring that the critical section of

code is executed atomically. This ensures that the three requirements of the critical section problem are met: mutual exclusion, progress, and bounded waiting.

Mutual exclusion: By using the wait() and signal() operations, processes or threads can control access to the shared resource in a mutually exclusive manner, ensuring that only one process or thread can access the shared resource at a time.

Progress: The signal() operation unblocks a blocked process or thread, ensuring that a process or thread can make progress even if the shared resource is not immediately available.

Bounded waiting: The wait() operation blocks a process or thread if the shared resource is not available, ensuring that the waiting process or thread consumes a limited amount of system resources and does not cause an unbounded wait.

By mitigating the critical section problem, semaphores provide a powerful and widely used mechanism for synchronizing access to shared resources in operating systems and concurrent programming.

(F)

1. What do you mean by deadlock? Explain the necessary condition for deadlock.

Ans.

A deadlock is a state in computer programming where two or more processes are blocked indefinitely because each process is waiting for the other to release a resource.

The necessary conditions for a deadlock are:

Mutual Exclusion: The resources involved in the deadlock must be exclusive to each process, meaning that only one process can use the resource at a time.

Hold and Wait: A process holding a resource must be able to request additional resources while waiting.

No Preemption: The resources cannot be forcibly taken away from a process.

Circular Wait: There must be a circular chain of processes, such that each process is waiting for a resource held by the next process in the chain.

2. **Explain deadlock prevention method.**

Ans.

Deadlocks can be prevented by avoiding or breaking one of the four necessary conditions for a deadlock. Here are some methods to prevent deadlocks:

Mutual Exclusion Elimination: Avoid using exclusive resources and instead use sharable resources that multiple processes can access simultaneously.

Hold and Wait Prevention: Impose a rule that a process cannot request a new resource if it already holds any resources.

Preemption: Use a mechanism to forcibly remove resources from a process and allocate it to another process in need.

Circular Wait Prevention: Impose a total ordering of all resource types, and require processes to request resources in that order.

Timeouts: Implement a timeout for resource requests, so that if a process is waiting for a resource for too long, it can be assumed that the resource is not available, and the process can be rolled back to a safe state.

Resource Allocation Graph: Use a graph-based algorithm to detect and prevent deadlocks by keeping track of the allocation of resources to processes and detecting any circular dependencies.

3. **What are the methods for handling deadlock?**

Ans.

There are several methods for handling deadlocks, including:

Prevention: Implementing algorithms or protocols to prevent deadlocks from occurring in the first place, as described in the previous answer.

Detection: Monitoring the system and detecting when a deadlock has occurred, then taking action to resolve it.

Avoidance: Allocating resources in such a way that deadlocks cannot occur, for example by avoiding resource allocation cycles.

Recovery: Releasing resources that are involved in the deadlock, either by rolling back the process that caused the deadlock or by killing one or more processes to break the deadlock.

Resolution: Manually resolving the deadlock by releasing resources or terminating processes. This approach may require human intervention and may not always be feasible.

Time-Out: Setting a time limit for resource allocation, and releasing the resources if the time limit is exceeded. This can be useful in situations where deadlocks are likely to occur and resolving them automatically is not possible.

Priority-Based Resource Allocation: Assigning a priority to each process and allocating resources to the process with the highest priority first. This method can reduce the likelihood of deadlocks.

4. Describe and explain with an example the Banker's algorithm for deadlock avoidance.

Ans.

The Banker's algorithm is a method used in operating systems to prevent deadlocks (a state in which two or more processes are blocked because each process is holding a resource the other process needs) by ensuring that the system will never enter a deadlock state.

Here's how it works:

Define the maximum number of resources of each type that a process may request and the current availability of each resource type.

For each process, define the maximum number of resources it may have and the number of resources currently allocated to it.

Before allocating a resource to a process, check if the process's current resource allocation plus the requested resource will exceed the process's maximum limit. If so, the allocation is not granted.

Check if the resource requested can be granted without causing the system to enter a deadlocked state by checking if the total resources requested by all processes plus the resource requested by the current process will exceed the total available resources. If so, the allocation is not granted.

Example:

Let's say there are three processes (P1, P2, P3) and three resources (R1, R2, R3).

Maximum resources:

P1: (7, 5, 3)

P2: (3, 2, 2)

P3: (9, 0, 2)

Total resources: (10, 5, 5)

Current allocation:

P1: (0, 1, 2)

P2: (2, 0, 0)

P3: (3, 0, 2)

Available resources: (5, 4, 1)

Let's say P1 requests (1, 0, 2) resources.

Current allocation: (0, 1, 2) + (1, 0, 2) = (1, 1, 4)

Maximum limit: (7, 5, 3)

The current allocation does not exceed the maximum limit.

Total resources requested by all processes: (1, 1, 4) + (2, 0, 0) + (3, 0, 2) = (6, 1, 6)

The total resources requested does not exceed the total available resources.

The resource request by P1 is granted.

By following the Banker's algorithm, the system ensures that it will never enter a deadlock state and that each process has access to the resources it needs to function.

5. **Define resource allocation graph. Explain resource request algorithm.**

Ans.

A resource allocation graph is a graphical representation used in operating systems to illustrate the allocation of resources among processes. In this graph, each process is represented as a node, and each resource is represented as an edge between the process and the resource it holds.

The resource request algorithm is a method used in the Banker's algorithm to determine whether a process can be granted a request for a resource. The algorithm performs the following steps:

Check if the requested resources can be granted to the process without causing the system to exceed its maximum resource limits. If the process's maximum limit is exceeded, the request is denied.

Check if the requested resources can be granted without causing the system to enter a deadlock state. This is done by checking if the available resources will still be greater than or equal to the

sum of the resources requested by all processes after granting the request. If the available resources will become negative, the request is denied.

If both of the above checks pass, the requested resources are granted to the process. The process's current resource allocation is updated and the available resources are reduced by the amount of resources granted.

Example:

Let's say there are three processes (P1, P2, P3) and three resources (R1, R2, R3).

Maximum resources:

P1: (7, 5, 3)

P2: (3, 2, 2)

P3: (9, 0, 2)

Total resources: (10, 5, 5)

Current allocation:

P1: (0, 1, 2)

P2: (2, 0, 0)

P3: (3, 0, 2)

Available resources: (5, 4, 1)

Let's say P1 requests (1, 0, 2) resources.

Maximum limit: (7, 5, 3)

Current allocation: (0, 1, 2) + (1, 0, 2) = (1, 1, 4)

The current allocation does not exceed the maximum limit.

Total resources requested by all processes: (1, 1, 4) + (2, 0, 0) + (3, 0, 2) = (6, 1, 6)

The total resources requested does not exceed the total available resources.

The resource request by P1 is granted.

The resource request algorithm ensures that resources are allocated to processes in a safe and efficient manner, preventing deadlocks and ensuring that the system runs smoothly.

6. **Briefly explain how a system can be recovered from a deadlock.**

Ans.

A deadlock in a system occurs when two or more processes are blocked and waiting for each other to release resources. To recover from a deadlock, the system can use one of the following methods:

- **Preemption:** This method involves forcibly releasing a resource from a blocked process and allocating it to another process that needs it. This can break the deadlock, but it requires a lot of overhead and may result in loss of data.
- **Rollback:** This method involves rolling back one or more processes to a previous state and releasing their resources. This breaks the deadlock, but it also requires a lot of overhead and may result in loss of data.
- **Kill one or more processes:** This method involves killing one or more processes to free up their resources and break the deadlock. This method is straightforward, but it may result in loss of data or work.
- **Timeout:** This method involves waiting for a specified amount of time for a process to release its resources. If the process does not release its resources within the specified time, it is killed, freeing up its resources and breaking the deadlock.

The most appropriate method for recovery from a deadlock depends on the specific requirements of the system and the resources involved. In general, the best approach is to prevent deadlocks in the first place, by using algorithms such as the Banker's algorithm.

(G)

1. **Explain logical address and physical address.;**

Ans.

In an operating system, a logical address and a physical address are two different types of memory addresses.

A logical address, also known as a virtual address, is an address generated by the CPU and used by a process to access memory. The logical address is translated by the operating system into a physical address, which is the actual location of the data in memory. The logical address space of a process is virtual, meaning that it is an abstraction of the physical memory, and the operating system maps logical addresses to physical addresses.

A physical address, also known as a real address, is the actual location of the data in memory. The physical address space of the system is the actual memory available to the system.

The use of logical addresses and physical addresses provides several benefits. It allows the operating system to allocate memory dynamically to processes, to share memory among processes, and to provide memory protection. The use of logical addresses also allows multiple processes to run concurrently, each with its own virtual address space, without interfering with each other's memory.

In summary, a logical address is a virtual address generated by the CPU and used by a process to access memory, while a physical address is the actual location of the data in memory. The operating system translates logical addresses into physical addresses for access to memory.

2. **How do you convert logical address to physical address?**

Ans.

The conversion of a logical address to a physical address is performed by the operating system. The operating system uses a data structure called the page table to store the mapping between logical addresses and physical addresses. The page table is stored in memory and is used by the operating system to translate logical addresses into physical addresses.

Here's a high-level explanation of how the conversion process works:

i. The CPU generates a logical address and sends it to the memory management unit (MMU).
ii. The MMU uses the page table to translate the logical address into a physical address. The page table is indexed using the page number part of the logical address, and the physical frame number is obtained from the page table entry.
iii. The physical address is formed by combining the physical frame number with the offset part of the logical address.
iv. The memory unit accesses the physical address and retrieves the data stored at that location.
v. The data is returned to the CPU for processing.

Note that the page table is managed by the operating system and can be updated dynamically to reflect changes in the mapping between logical and physical addresses. The operating system

also uses the page table to enforce memory protection and to manage page replacement algorithms.

3. Explain multiple-partition allocation technique of memory.

Ans.

Multiple-partition allocation is a memory management technique used in operating systems to allocate memory to processes. In this technique, the main memory is divided into a number of fixed-sized partitions. Each partition can be assigned to a single process, and the process can use all the memory in that partition.

There are two main types of multiple-partition allocation:

- **Static Partitioning:** In this technique, the memory partitions are created during system initialization and remain unchanged throughout the execution of the system. The size of each partition is fixed, and the operating system assigns a partition to a process when the process is created. The disadvantage of this technique is that it is not flexible, and a process may not be able to allocate all the memory it needs if the partitions are not of the right size.
- **Dynamic Partitioning:** In this technique, the memory partitions are created and merged dynamically as processes are created and terminated. The operating system can adjust the size of the partitions to allocate memory efficiently. The advantage of this technique is that it is flexible and can allocate memory to processes more efficiently.

In both techniques, the operating system uses a memory allocation algorithm, such as first-fit, best-fit, or worst-fit, to allocate memory to processes. The goal of the memory allocation algorithm is to minimize fragmentation and maximize memory utilization.

In summary, multiple-partition allocation is a memory management technique used in operating systems to allocate memory to processes. The main memory is divided into a number of fixed-sized partitions, and the operating system uses a memory allocation algorithm to allocate memory to processes. The technique can be either static or dynamic, depending on the needs of the system.

4. Explain paging system with proper do example.

Ans.

Paging is a basic function in memory management for a computer's operating system (OS) as well -- this includes Windows, Unix, Linux and macOSs.

Paging works by writing data to, and reading it from, secondary storage for use in primary storage.In a memory management system that takes advantage of paging, the OS reads data from secondary storage in blocks called pages, all of which have identical size. The physical region of memory containing a single page is called a frame. When paging is used, a frame does not have

to comprise a single physically contiguous region in secondary storage. This approach offers an advantage over earlier memory management methods, because it facilitates more efficient and faster use of storage.

5**. What is segmentation.**

Ans.

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

6. **Describe how logical address maps physical address under segmentation scheme with proper diagram.**

Ans.

Under the segmentation scheme, a logical address is divided into two parts: a segment number and an offset within the segment. The segment number identifies a segment in the program, and the offset identifies a specific location within the segment.

To map a logical address to a physical address, the operating system maintains a segment table, which contains the base address and length of each segment in physical memory. The operating system uses the segment number from the logical address to index into the segment table and find the corresponding segment. It then adds the offset from the logical address to the base address of the segment to determine the physical address.

Here's a diagram that shows how a logical address maps to a physical address under the segmentation scheme:

Code:

Logical Address: [Segment Number] [Offset]

Physical Address: [Base Address of Segment] + [Offset]

```
        +---------------+
```

| Segment Table |

+---------------+

| [Segment 1] |

| [Base Address]|

| [Length] |

+---------------+

| [Segment 2] |

| [Base Address]|

| [Length] |

+---------------+

| ... |

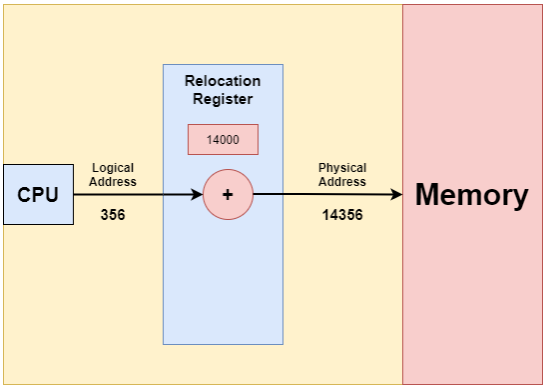
+---------------+

| [Segment n] |

| [Base Address]|

| [Length] |

+---------------+



In this diagram, the logical address [Segment Number] [Offset] is used to index into the segment table and find the corresponding segment. The base address of the segment is then added to the offset to determine the physical address.

In summary, under the segmentation scheme, a logical address is divided into two parts: a segment number and an offset within the segment. The operating system uses the segment number to index into the segment table and find the corresponding segment, and then adds the offset to the base address of the segment to determine the physical address.

7. **What do you mean by swaping.**

Ans.

Swapping is a memory management technique used in operating systems to temporarily transfer data from main memory to disk storage. The process of transferring data from main memory to disk storage is known as "swapping out," and the process of transferring data from disk storage to main memory is known as "swapping in."

Swapping is used when the operating system needs to free up space in main memory for a new process or for data for an existing process. The operating system selects a process that is currently in memory and swaps out its data to disk storage. The operating system then frees up the space in main memory occupied by the swapped-out process, making it available for use by other processes.

When the swapped-out process needs to continue execution, the operating system swaps its data back into main memory and resumes its execution. This process of swapping in and out continues throughout the lifetime of the process, with the operating system swapping out the process's data whenever it needs to free up space in main memory.

Swapping has the advantage of allowing the operating system to allocate memory to processes dynamically, based on their current memory needs. However, swapping can be slow and can degrade performance, as disk I/O is slower than main memory access. To minimize the performance impact of swapping, operating systems use a variety of techniques, such as demand paging and page replacement algorithms, to optimize the use of memory and disk storage.

In summary, swapping is a memory management technique used in operating systems to temporarily transfer data from main memory to disk storage. The operating system uses swapping to free up space in main memory for new processes or data, and to manage memory allocation dynamically. However, swapping can be slow and can degrade performance, so operating systems use a variety of techniques to optimize its use.

8.**What do you mean by virtual memory technique? What are the advantages of it.**

Ans.

Virtual memory is a memory management technique used in operating systems to provide the illusion of a large, continuous memory space to processes, even if the actual physical memory is

limited. Virtual memory allows processes to allocate more memory than is physically available in the system.

The operating system uses a combination of main memory and disk storage to implement virtual memory. When a process requests memory, the operating system allocates memory from the main memory. If the main memory is full, the operating system can transfer some data from main memory to disk storage, freeing up space in main memory for the new data. When a process needs data that has been transferred to disk storage, the operating system transfers the data back to main memory. This transfer of data between main memory and disk storage is transparent to the process, and the process is unaware that its data is stored on disk.

The advantages of virtual memory are:

i. **Increased Memory Capacity:** Virtual memory allows processes to allocate more memory than is physically available, increasing the effective memory capacity of the system.
ii. **Improved Resource Utilization:** By transferring data between main memory and disk storage as needed, virtual memory ensures that memory resources are used efficiently and optimally.
iii. **Simplified Memory Allocation:** Virtual memory provides a uniform and simplified view of memory to processes, eliminating the need for processes to manage memory allocation and freeing them to focus on their core tasks.
iv. **Enhanced Memory Protection:** Virtual memory provides a mechanism for isolating processes from each other and protecting their memory space, reducing the risk of one process interfering with the memory of another process.

In summary, virtual memory is a memory management technique used in operating systems to provide the illusion of a large, continuous memory space to processes. The operating system uses a combination of main memory and disk storage to implement virtual memory, and its advantages include increased memory capacity,