Question Bank Solution for CAT – 1: Operating System

By **JAAT**

FOR MORE INFO ABOUT US VISIT: https://youtu.be/aIC3suP4rZM

Q1. Define Operating Systems and list out the various types of Operating Systems.

Ans An Operating System (OS) is a system software that manages computer hardware, software resources, and provides services for computer programs. It acts as an intermediary between computer applications and hardware. Its main functions include managing the system's resources, scheduling processes, managing memory, and providing a user interface.

There are several types of Operating Systems, including:

1. Batch Operating System
2. Time-Sharing or Multitasking Operating System
3. Real-Time Operating System
4. Network Operating System
5. Distributed Operating System
6. Mobile Operating System
7. Embedded Operating System
8. Virtualization Operating System
9. Cloud Operating System
10. Hypervisor Operating System

Q2. What are the various types of operating systems?

ANS There are several types of operating systems, including:

Batch operating system

Time-sharing operating system

Distributed operating system

Network operating system

Real-time operating system

Multi-user operating system

Mobile operating system

Q3. Classify the various types of the operating system.

ANS Operating systems can be classified into several types based on various criteria. Here are some common ways of classifying operating systems:

By Purpose: a. General Purpose Operating Systems (GPOS) - These are designed for a wide range of applications and hardware platforms, such as Windows, Linux, and macOS. b. Special Purpose Operating Systems (SPOS) - These are designed for specific applications or devices, such as real-time systems, embedded systems, mobile devices, and gaming consoles.

By User Interface: a. Command Line Interface (CLI) - The user interacts with the system by typing commands. b. Graphical User Interface (GUI) - The user interacts with the system using visual elements such as icons, menus, and windows. c. Web-based User Interface (WUI) - The user interacts with the system through a web browser.

By Number of Users: a. Single-user Operating System - These are designed for a single user, such as personal computers and smartphones. b. Multi-user Operating System - These are designed for multiple users to share system resources, such as servers and mainframes.

By Processing Method: a. Batch Processing Operating System - These are designed to process a large number of similar jobs in batches, without human interaction. b. Time-sharing Operating System - These are designed to share the CPU time among multiple users or processes, providing the illusion of concurrent execution. c. Real-time Operating System - These are designed to process tasks with strict timing constraints, such as aerospace systems and medical devices.

By Memory Management: a. Monolithic Operating System - These are designed to run the entire operating system in a single address space, such as MS-DOS and UNIX. b. Microkernel Operating System - These are designed to run only essential kernel functions in the privileged mode, while the rest of the system services run in the user mode, such as MINIX and QNX. c. Hybrid Operating System - These are designed to combine the advantages of both monolithic and microkernel architectures, such as Linux and Windows NT.

Q4. Explain different types of the operating system.

ANS Batch Operating System: This type of operating system is designed to execute a series of programs, known as a batch, without any human interaction. The batch is submitted to the system as a job, and the operating system executes the job without any further user intervention.

Real-Time Operating System (RTOS): This type of operating system is designed to respond to events or data input within a predetermined time frame. RTOS is used in systems that require a quick and predictable response time, such as robotics, aviation, and medical equipment.

Network Operating System (NOS): This type of operating system is designed to manage and coordinate the resources of a network. The NOS provides services such as file and print sharing, application sharing, and security.

Distributed Operating System: This type of operating system manages a group of networked computers and makes them appear to be a single computer. It provides transparent access to shared resources and allows processes to communicate with each other across the network.

Time-sharing Operating System: This type of operating system allows multiple users to use the computer simultaneously by dividing the CPU time into time slices. Each user gets a small time slice to use the CPU and then the operating system switches to another user.

Mobile Operating System: This type of operating system is designed for mobile devices such as smartphones and tablets. The mobile operating system provides features such as touch input, GPS, and multimedia support.

Embedded Operating System: This type of operating system is designed to run on embedded systems such as digital cameras, set-top boxes, and gaming consoles. The embedded operating system is designed to be small and efficient, with minimal hardware requirements.

Q5. Show the following terminologies associated with the operating system and explain each in detail.

a) Multiprogramming systems

b) Multitasking systems

c) Multiprocessor systems.

ANS. a) Multiprogramming systems: Multiprogramming systems refer to the technique of running multiple programs concurrently by the operating system. The operating system assigns CPU time to each program, allowing them to run simultaneously. It is achieved by keeping multiple programs in main memory and switching between them, keeping the CPU busy at all times. This technique improves CPU utilization and increases the throughput of the system.

b) Multitasking systems: Multitasking systems are an extension of multiprogramming systems. In multitasking systems, the CPU switches between different tasks rapidly to give the illusion of concurrent execution. Each task gets a small amount of CPU time before the operating system

switches to the next task. This technique is used to improve system responsiveness and allows users to run multiple applications simultaneously.

c) Multiprocessor systems: Multiprocessor systems are those that have multiple CPUs working together to execute tasks. These systems can be categorized as symmetric multiprocessing (SMP) or asymmetric multiprocessing (AMP) systems. In SMP systems, all CPUs have equal access to the memory and I/O subsystem, and the operating system distributes the workloads evenly among the CPUs. In AMP systems, one CPU acts as the master processor and controls the other CPUs, which are called slave processors. AMP systems are used in real-time applications and high-performance computing. Multiprocessor systems offer increased processing power, improved system reliability, and better fault tolerance.

Q6. Illustrate the distinguishing features of i). Real time system ii) Multiprocessor system Identify the difference between mainframe and desktop operating system.

ANS

i) Real-time systems: Real-time systems are designed to handle real-time applications that require a response to an event within a certain period of time. The distinguishing features of real-time systems are:

- Predictability: Real-time systems have to be predictable in terms of response time, and the response time has to be guaranteed.
- Deadline: Real-time systems must meet the deadline of the event, which means the system must provide the response within a specified time frame.
- Determinism: Real-time systems have to be deterministic, which means that they should be able to respond to a request within a predictable time frame.

ii) Multiprocessor systems: Multiprocessor systems are designed to handle tasks in parallel by using multiple processors. The distinguishing features of multiprocessor systems are:

- Scalability: Multiprocessor systems can be scaled up or down based on the requirement of the system.
- Performance: Multiprocessor systems provide better performance by parallelizing the tasks.
- Fault-tolerance: Multiprocessor systems have a higher level of fault-tolerance, as if one processor fails, the system can still function using the remaining processors.

Difference between mainframe and desktop operating system:

Mainframe operating systems are designed for large centralized computers that are used for enterprise-level computing. Mainframe operating systems are designed to handle multiple users and large amounts of data.

Desktop operating systems, on the other hand, are designed for personal computers used by individual users. Desktop operating systems are designed to handle single users and small amounts of data.

The main differences between mainframe and desktop operating systems are:

- Size: Mainframe operating systems are designed for large centralized computers, while desktop operating systems are designed for personal computers.
- Multi-user support: Mainframe operating systems are designed to support multiple users, while desktop operating systems are designed for single users.
- Data handling: Mainframe operating systems are designed to handle large amounts of data, while desktop operating systems are designed for small amounts of data.

Q7. Describe the distinguishing features of

i) Real-time system ii) Multiprocessor system.

i) Real-time systems are designed to respond to events or inputs within a specified time frame, known as a deadline. They are used in applications where timing is critical, such as in industrial control systems, flight control systems, and medical devices. Some of the distinguishing features of real-time systems are:

- Predictability: Real-time systems are designed to operate with predictable behavior, so that the response time to a given input can be guaranteed.
- Deadline-driven: The system must complete a task within a specific time frame, called a deadline, otherwise the result is considered useless or incorrect.
- Deterministic: The system must behave in a predictable manner, without unexpected delays or variations in timing.
- Priority-based scheduling: Tasks with higher priority are given preference over lower priority tasks.
- High reliability and fault tolerance: Real-time systems often operate in safety-critical environments where any failure can have serious consequences.

ii) Multiprocessor systems are computer systems that have more than one processor or CPU. They can be used for parallel processing, distributed processing, or to provide redundancy in case of hardware failures. Some of the distinguishing features of multiprocessor systems are:

- Parallelism: Multiple processors can work together to execute tasks in parallel, which can improve performance and reduce response times.

- Scalability: Multiprocessor systems can be scaled up by adding more processors to handle increasing workloads.
- Shared memory: Multiple processors can access the same memory, which simplifies programming and makes it easier to share data between processes.
- Load balancing: The workload can be distributed among the processors to avoid overloading any one processor.
- Interprocessor communication: The processors must communicate with each other to coordinate

Q8. Justify the statement "Operating System can be viewed as a government, resource allocator and a control program"

ANS The statement "Operating System can be viewed as a government, resource allocator, and a control program" is a widely accepted metaphor for describing the role and functions of an operating system. Here is a justification for this statement:

Government: An operating system can be compared to a government in that it establishes and enforces policies and rules for the use of computer resources. Just as a government sets rules for the allocation and use of resources such as land, water, and energy, an operating system sets rules for the allocation and use of computer resources such as memory, CPU time, and disk space. The operating system ensures that these resources are used efficiently and fairly among all the processes running on the computer system.

Resource Allocator: An operating system can be compared to a resource allocator in that it manages and allocates computer resources to various processes running on the system. Just as a resource allocator manages the allocation of resources such as food, water, and shelter to different people or groups, an operating system manages the allocation of computer resources such as memory, CPU time, and disk space to different processes. The operating system ensures that these resources are allocated optimally to meet the needs of all the processes running on the system.

Control Program: An operating system can be compared to a control program in that it manages the execution of other programs and provides a uniform interface for accessing computer resources. Just as a control program manages the execution of other programs and ensures that they run correctly and efficiently, an operating system manages the execution of processes and ensures that they run correctly and efficiently. The operating system also provides a uniform interface for accessing computer resources, such as files and devices, which makes it easier for programs to interact with the computer system.

In conclusion, the metaphor of an operating system as a government, resource allocator, and control program helps to illustrate the important functions that an operating system performs in managing and coordinating the use of computer resources. It highlights the complex and

essential role that an operating system plays in ensuring that computer systems run efficiently, reliably, and securely.

Q9. Classify the Critical Section problem. Give the conditions that a solution to the critical section problem must satisfy.

The critical section problem is a synchronization problem that arises when multiple processes or threads access a shared resource concurrently. The problem can lead to race conditions and inconsistent results. A solution to the critical section problem must satisfy three conditions: mutual exclusion, progress, and bounded waiting. Mutual exclusion ensures that only one process can access the shared resource at a time. Progress guarantees that processes waiting to enter their critical section are allowed to do so, while bounded waiting ensures that no process is indefinitely blocked from accessing its critical section. Synchronization mechanisms such as semaphores, monitors, and locks are used to enforce these conditions.

Q10. Distinguish between hard real time systems and soft real time systems.

ANS Real-time systems are designed to process and respond to events within a specific time constraint. They are commonly used in applications such as control systems, medical devices, and aerospace systems. Real-time systems can be classified into two categories: hard real-time systems and soft real-time systems.

Hard real-time systems are systems in which the response to a specific event must be guaranteed within a very strict and deterministic time frame. Any delay beyond the specified deadline can lead to catastrophic consequences, such as the failure of the system or loss of life. Examples of hard real-time systems include air traffic control systems, nuclear power plant control systems, and autonomous vehicle control systems.

Soft real-time systems, on the other hand, have a more relaxed response time constraint. They can tolerate occasional delays in the processing and response to events without catastrophic consequences. Examples of soft real-time systems include multimedia systems, online gaming, and e-commerce systems.

In summary, the main difference between hard real-time systems and soft real-time systems lies in the severity of the consequences of missing a deadline. Hard real-time systems have strict and deterministic time constraints, and missing a deadline can lead to catastrophic consequences. Soft real-time systems have more relaxed time constraints, and occasional delays in meeting deadlines are tolerable.

Q11. Distinguish among the following terminologies associated with the operating system and explain each of them in detail. I) Multiprogramming systems, ii) Multitasking systems, iii) Multiprocessor systems.

ANS I) Multiprogramming systems: Multiprogramming systems refer to operating systems that can simultaneously execute multiple programs or processes in memory. In a multiprogramming system, the CPU is rapidly switched between different processes, allowing multiple programs to share the CPU and other resources. The goal of multiprogramming is to maximize the use of system resources and improve overall system performance. It is achieved through techniques like time-sharing, where each program is given a time slice to execute on the CPU, and virtual memory, where processes can use more memory than is physically available.

II) Multitasking systems: Multitasking systems are similar to multiprogramming systems, but they go a step further by allowing multiple processes to run simultaneously on a single CPU. In a multitasking system, each process is given a slice of CPU time, and the system rapidly switches between processes, giving the illusion that they are all running simultaneously. Multitasking systems can improve system responsiveness and allow users to run multiple applications at the same time, increasing productivity. They can be found in desktop and mobile operating systems like Windows, macOS, iOS, and Android.

III) Multiprocessor systems: Multiprocessor systems refer to computers that have multiple CPUs or processors working together to execute programs. These systems can be symmetric, where each CPU is identical and can execute any process, or asymmetric, where one CPU is designated as the master and controls the other CPUs. Multiprocessor systems can provide significant performance improvements over single-processor systems, but they also require specialized hardware and software to manage the distribution of processes across multiple CPUs. Multiprocessor systems can be found in high-performance computing environments, data centers, and supercomputers.

Q12. Distinguish between batch systems and time-sharing systems.

ANS Batch systems and time-sharing systems are two different types of operating systems used to manage computing resources.

Batch systems were one of the earliest types of operating systems, and they were designed to optimize the use of computer resources in a non-interactive environment. Batch systems work by grouping a set of similar jobs together into a batch, and then running them one after the other without any user intervention. The main advantage of batch systems is that they can process large amounts of data efficiently, but they offer little interactivity and are not suitable for real-time or interactive applications.

Time-sharing systems, on the other hand, are designed for interactive computing. In a time-sharing system, multiple users can access the same computer simultaneously, with each user being allocated a slice of CPU time to run their applications. Time-sharing systems provide a much more interactive experience for users, allowing them to run multiple applications concurrently and providing real-time responsiveness. The main disadvantage of time-sharing

systems is that they require a lot of CPU time to manage multiple users, which can reduce overall system performance.

Q 13. Describe the Operating system. Also, discover the Operating-System Functions.

Ans An operating system (OS) is a software program that manages computer hardware and software resources and provides common services for computer programs. It acts as a communication bridge between the hardware and the software components, enabling applications to interact with the system hardware.

The functions of an operating system include:

Process management: The OS manages processes, which are instances of programs running on a computer. It schedules processes, allocates resources, and ensures that processes are running efficiently.

Memory management: The OS manages the computer's memory resources, allocating and deallocating memory as needed by running programs.

File system management: The OS manages the file system, which stores and organizes files on the computer's storage devices.

Device management: The OS manages the computer's input/output devices, such as printers, keyboards, and displays, and provides a way for programs to interact with them.

Security: The OS provides security features such as user authentication, access control, and encryption to protect the system and its data from unauthorized access.

Network management: The OS manages network connections and provides services such as file sharing and remote access.

Overall, the operating system is the fundamental software layer that enables other programs to run and utilize the computer's resources.

Q14 Differentiate among the following types of OS by defining their essential properties. a) Time sharing system b) Parallel system c) Distributed system d) Real time system

ANS a) Time-sharing system: It allows multiple users to share a single processor by switching between them quickly, giving the appearance of simultaneous execution. Each user gets a small time slice of CPU time to execute their program. This type of system is designed to maximize the use of the processor, and to provide a responsive computing experience for users.

b) Parallel system: A parallel system has multiple processors or cores that work together to execute a single task, with the goal of achieving higher performance or throughput than a single processor could achieve alone. Parallel systems can be shared-memory or distributed-memory architectures.

c) Distributed system: A distributed system is a collection of independent computers that work together to provide a unified computing service. The computers communicate with each other through a network, and they typically have their own local resources, such as memory, storage, and processors.

d) Real-time system: A real-time system is designed to respond to events or input within a fixed time frame, typically in milliseconds or microseconds. This type of system is used in applications where timing is critical, such as in robotics, aviation, and medical equipment. Real-time systems are often designed to be deterministic, meaning that their behavior can be predicted and controlled.

In summary, time-sharing systems enable multiple users to share a single processor, parallel systems use multiple processors to execute a task, distributed systems are a collection of independent computers that work together, and real-time systems respond to events within a fixed time frame.

Q15. Explain Process Control Block (PCB)

ANS A Process Control Block (PCB) is a data structure used by an operating system to store information about a running process. The PCB contains information about the process, such as its process ID, state, priority, CPU usage, and memory allocation.

When a process is scheduled for execution, its PCB is loaded into memory and used by the operating system to manage the process's execution. The PCB is updated by the operating system as the process executes, with changes made to the PCB reflecting changes in the process's state.

The PCB is an essential component of process management in modern operating systems. It allows the operating system to manage the execution of multiple processes simultaneously, ensuring that each process is given its fair share of system resources, and preventing processes from interfering with one another.

Q16 Compare the essential properties of the following types of Operating system: i) Batch operating system ii) Interactive operating system iii) Time sharing operating system iv) Real time operating system v) Distributed operating system.

ANS i) Batch operating system: A batch operating system processes a set of tasks in batch mode, without any user interaction. Jobs are submitted to the system as a batch, and the operating system processes them in order. The essential property of a batch operating system is that it maximizes the use of system resources by processing multiple jobs in sequence.

ii) Interactive operating system: An interactive operating system allows users to interact with the system through a user interface, such as a command line or graphical interface. Users can issue commands, run applications, and interact with the system in real-time. The essential

property of an interactive operating system is that it provides a responsive and interactive user experience.

iii) Time-sharing operating system: A time-sharing operating system allows multiple users to share a single processor by switching between them quickly, giving the appearance of simultaneous execution. Each user gets a small time slice of CPU time to execute their program. The essential property of a time-sharing operating system is that it maximizes the use of the processor, and provides a responsive computing experience for users.

iv) Real-time operating system: A real-time operating system is designed to respond to events or input within a fixed time frame, typically in milliseconds or microseconds. This type of system is used in applications where timing is critical, such as in robotics, aviation, and medical equipment. The essential property of a real-time operating system is that it provides a predictable and deterministic execution environment.

v) Distributed operating system: A distributed operating system is a collection of independent computers that work together to provide a unified computing service. The essential property of a distributed operating system is that it provides a scalable and fault-tolerant computing environment by allowing the distribution of workloads across multiple machines.

In summary, the essential properties of different types of operating systems are determined by their primary use case and the requirements of their intended applications. Batch operating systems maximize the use of resources, interactive operating systems provide a responsive user experience, time-sharing operating systems enable multiple users to share a single processor, real-time operating systems provide a predictable and deterministic execution environment, and distributed operating systems provide a scalable and fault-tolerant computing environment.

Q17. Associate a Relationship between FCFS and Round Robin Scheduling Algorithm.

ANS First-Come-First-Serve (FCFS) and Round Robin (RR) are two popular scheduling algorithms used in operating systems.

FCFS is a non-preemptive scheduling algorithm that processes tasks in the order in which they arrive. Once a task is allocated the CPU, it runs to completion before the next task is selected for execution.

Round Robin, on the other hand, is a preemptive scheduling algorithm that allocates CPU time to tasks in equal time slices. Each task runs for a predefined time slice, called a quantum, before being preempted and replaced with the next task in the queue.

Although FCFS and RR are different scheduling algorithms, they are related in that RR can be seen as a variant of FCFS with preemption. In RR, each process is given a fixed time quantum, and if it doesn't complete within that time, it is preempted and moved to the back of the queue, allowing the next process to run.

Therefore, RR can be viewed as FCFS with preemption and a fixed time quantum. Both algorithms have their strengths and weaknesses, and their suitability depends on the specific use case and system requirements.

Q18. Define semaphore and its types.

ANS In operating systems, a semaphore is a synchronization mechanism that controls access to shared resources. It is essentially a variable that can be accessed by multiple processes, and its value is used to coordinate access to shared resources.

There are two types of semaphores:

1. Binary semaphore: Also known as a mutex, a binary semaphore is a type of semaphore that has only two values: 0 and 1. It is used to control access to a single shared resource, where only one process can access the resource at a time.
2. Counting semaphore: A counting semaphore is a type of semaphore that can have a range of values greater than 1. It is used to control access to multiple instances of a shared resource, where multiple processes can access the resource at a time, up to a maximum number defined by the semaphore's value.

Semaphores are used to prevent conflicts and race conditions that can occur when multiple processes attempt to access a shared resource simultaneously. They are an essential tool for synchronization in concurrent systems, helping to ensure that multiple processes can access shared resources in an orderly and controlled manner.

Q19. How will you implement mutual exclusion with test_and_set () instruction?

ANS. The test_and_set() instruction is a synchronization primitive that can be used to implement mutual exclusion in a multi-threaded or multi-process environment. Here's how mutual exclusion can be implemented using test_and_set():

3. Declare a boolean variable, say "lock", and initialize it to false.
4. Before accessing the shared resource, call the test_and_set() instruction with the address of the "lock" variable.
5. If the value returned by the test_and_set() instruction is true, it means that the lock is already held by another thread/process, so the current thread/process should spin and try again later.
6. If the value returned by the test_and_set() instruction is false, it means that the lock is available, so the current thread/process can access the shared resource.
7. When the thread/process is done accessing the shared resource, it should release the lock by setting the value of the "lock" variable back to false.

This way, only one thread/process can hold the lock at a time, ensuring mutual exclusion and preventing race conditions.

Q20. A Counting Semaphore was initialized to 12. then 10P (wait) and 4V (Signal) operations were computed on this semaphore. Explain the result?

ANS. A counting semaphore is a synchronization mechanism that can have a range of values greater than 1. In this case, the semaphore was initialized to a value of 12, which means that there were 12 available resources that could be accessed by processes.

The 10P (wait) operations on the semaphore mean that 10 processes were trying to access the shared resource. Each time a process attempts to access the resource, it decrements the value of the semaphore. If the semaphore value becomes negative, the process is blocked until another process signals (increments) the semaphore.

In this case, if all 10 processes attempted to access the resource simultaneously, the semaphore value would become -8, indicating that there were 8 processes waiting for the resource.

The 4V (signal) operations on the semaphore mean that 4 processes signaled (incremented) the semaphore, indicating that they were done accessing the shared resource and releasing it for others to use. Each signal operation increments the value of the semaphore.

Assuming that no processes were blocked while waiting to access the resource, after the 10P and 4V operations, the semaphore value would be 6, indicating that there are 6 resources still available for other processes to use.

Q21. Discuss the concept of Process. Explain various states of process with Process transition diagram.

ANS. In operating systems, a process is an instance of a running program, with its own memory space and state. A process can be in one of several states at any given time, depending on its activity and whether it is waiting for resources or other events.

The different states of a process are:

New: The process is being created.

Ready: The process is waiting for a processor to be assigned.

Running: The process is currently being executed by the processor.

Blocked: The process is waiting for a resource or event before it can continue.

Terminated: The process has completed its execution.

A process can transition between these states depending on various events, such as receiving an interrupt, waiting for I/O, or being preempted by a higher-priority process.

The process transition diagram is a visual representation of the different states of a process and the events that can cause it to transition between these states. It is a useful tool for understanding the lifecycle of a process and the events that affect its behavior.

In the diagram, each state is represented by a node, and the transitions between the states are represented by arrows. The diagram shows how a process can move from one state to another, depending on the events that occur.

Overall, understanding the concept of a process and its various states is crucial for understanding how an operating system manages the execution of programs and resources in a multi-tasking environment.

Q22. Q22. Compare and contrast between short term and long-term Scheduler.

The short-term scheduler (also known as the CPU scheduler) selects which process in the ready queue should execute next and allocates CPU time for that process. It is responsible for making efficient use of the CPU resources available to the system, ensuring that no process monopolizes the CPU for too long. Its main goal is to minimize the turnaround time of processes and increase system throughput.

In contrast, the long-term scheduler (also known as the job scheduler) selects which new processes should be admitted into the system and adds them to the ready queue. It decides which processes should be brought into memory from the pool of available processes waiting on the disk. Its main goal is to maintain a balance between the number of processes in the system and the available system resources, avoiding system overload and ensuring that new processes have sufficient resources to execute.

In summary, the short-term scheduler manages CPU time allocation for processes already in the system, while the long-term scheduler manages the admission of new processes to the system.

Q23. Establish Relationship between FCFS and Round Robin Scheduling Algorithm.

First-Come-First-Serve (FCFS) and Round Robin (RR) are both scheduling algorithms used in operating systems. FCFS is a non-preemptive scheduling algorithm, meaning that once a process is assigned the CPU, it will hold it until it completes. RR, on the other hand, is a preemptive scheduling algorithm, where each process is assigned a fixed time slice or quantum, and it will be preempted by the scheduler once its quantum expires.

One way to establish a relationship between these two algorithms is by noting that RR is a variation of FCFS with preemption. When the quantum of a process is larger than the remaining time required to complete its task, the RR scheduler behaves like the FCFS scheduler. However, if the process does not complete within its quantum, it is preempted, and the next process in the ready queue is scheduled. Therefore, RR can be seen as a compromise between the

benefits of FCFS and the advantages of preemptive scheduling, allowing for better responsiveness and fairness among processes.

Q24. Define CPU scheduling. Explain pre-emptive and no pre-emptive scheduling. Give example.

CPU scheduling is the process of selecting which process should be executed next on the CPU. In an operating system, multiple processes may be ready to execute, but the CPU can only execute one process at a time. The CPU scheduler manages the allocation of CPU resources to ensure efficient and fair use of the CPU.

Preemptive scheduling is a scheduling algorithm where a process can be interrupted and stopped before it completes its task. An example of preemptive scheduling is the Round Robin algorithm, where each process is given a fixed time slice to execute before being preempted and moved to the back of the ready queue.

Non-preemptive scheduling is a scheduling algorithm where a process holds the CPU until it completes its task or voluntarily yields the CPU. An example of non-preemptive scheduling is the First-Come-First-Serve (FCFS) algorithm, where processes are executed in the order they arrive in the ready queue.

In summary, preemptive scheduling allows for better responsiveness and fairness among processes, while non-preemptive scheduling may lead to longer waiting times for some processes but may also reduce overhead and complexity.

Q25. Explain Round robin algorithm using an example.

Round robin is a scheduling algorithm commonly used in computer systems to distribute processing time among multiple processes or threads. It works by allocating a fixed amount of processing time to each process in a cyclical manner. For example, consider a system with three processes, P1, P2, and P3, and a time quantum of 2 units. Initially, all processes are added to a ready queue. The first process, P1, is allocated 2 units of processing time. After 2 units, P1 is preempted, and P2 is allocated 2 units. The same process is repeated for P3. Then, the cycle starts again with P1. This continues until all processes are completed. Round robin ensures that no process monopolizes the CPU and provides a fair distribution of processing time.

Q26. Discover various CPU scheduling algorithm. Explain pre-emptive and no pre-emptive scheduling. Give example.

There are various CPU scheduling algorithms, including:

First-Come-First-Serve (FCFS): The process that arrives first in the ready queue is executed first.

Shortest Job First (SJF): The process with the shortest burst time is executed first.

Priority Scheduling: The process with the highest priority is executed first.

Round-Robin Scheduling: Each process is allocated a fixed time slice or quantum. The process that finishes its quantum is preempted, and the next process in the ready queue is executed.

Multilevel Queue Scheduling: The ready queue is divided into several queues, and each queue has a different priority.

Multilevel Feedback Queue Scheduling: Processes can move between different queues based on their performance.

Preemptive scheduling is when a process can be interrupted and moved out of the CPU, and another process can take its place. Round-robin scheduling is an example of preemptive scheduling.

Non-preemptive scheduling is when a process can't be interrupted once it starts executing, and it runs to completion. FCFS and SJF scheduling are examples of non-preemptive scheduling.

Q27. List the various scheduling criteria for CPU scheduling?

The CPU scheduling criteria determine which process should be selected next for execution from the ready queue. Some of the commonly used scheduling criteria include:

First-Come-First-Serve (FCFS): The process that arrives first in the ready queue is executed first.

Shortest Job First (SJF): The process with the shortest burst time is executed first.

Priority Scheduling: The process with the highest priority is executed first. Priority can be determined based on various factors such as the amount of CPU time required, memory requirements, I/O requirements, etc.

Round-Robin Scheduling: Each process is allocated a fixed time slice or quantum. The process that finishes its quantum is preempted, and the next process in the ready queue is executed.

Multilevel Queue Scheduling: The ready queue is divided into several queues, and each queue has a different priority. Processes are assigned to different queues based on their characteristics such as CPU utilization, I/O requirements, memory requirements, etc.

Multilevel Feedback Queue Scheduling: This is an extension of multilevel queue scheduling, where processes can move between different queues based on their performance. The scheduling algorithm dynamically adjusts the priority of a process based on its behavior in the system.

Q28. Consider the following process.

Process Id	**Burst Time**
P0	**5**

P1	**6**
P2	**1**
P3	**2**
P4	**6**

Point out the average waiting time and average turnaround time for this process with

(i)FCFS Scheduling

(ii)SJF Scheduling

(iii) Round Robin with Time Quantum = 2

ANS (i) FCFS Scheduling: The order of the processes in the ready queue is P0, P1, P2, P3, P4. The waiting time for P0 = 0, P1 = 5, P2 = 11, P3 = 12, P4 = 14. The turnaround time for P0 = 5, P1 = 11, P2 = 12, P3 = 14, P4 = 20. The average waiting time = (0 + 5 + 11 + 12 + 14)/5 = 8.4 The average turnaround time = (5 + 11 + 12 + 14 + 20)/5 = 12.4

(ii) SJF Scheduling: The order of the processes in the ready queue is P2, P3, P0, P1, P4. The waiting time for P2 = 0, P3 = 1, P0 = 2, P1 = 8, P4 = 14. The turnaround time for P2 = 1, P3 = 3, P0 = 7, P1 = 14, P4 = 20. The average waiting time = (0 + 1 + 2 + 8 + 14)/5 = 5 The average turnaround time = (1 + 3 + 7 + 14 + 20)/5 = 9

(iii) Round Robin with Time Quantum = 2: The order of the processes in the ready queue is P0, P1, P2, P3, P4. The waiting time for P0 = 2, P1 = 12, P2 = 1, P3 = 2, P4 = 15. The turnaround time for P0 = 7, P1 = 20, P2 = 2, P3 = 4, P4 = 22. The average waiting time = (2 + 12 + 1 + 2 + 15)/5 = 6.4 The average turnaround time = (7 + 20 + 2 + 4 + 22)/5 = 11.4

Note: In Round Robin scheduling, if a process completes its execution within the time quantum, its waiting time and turnaround time are calculated accordingly. Otherwise, the process is preempted, and the next process in the queue is executed.

Q29. Consider the set of 5 processes whose arrival time and burst time are given below-

Process Id	**Arrival time**	**Burst time**	**Priority**
P1	**0**	**4**	**2**
P2	**1**	**3**	**3**
P3	**2**	**1**	**4**
P4	**3**	**5**	**5**
P5	**4**	**2**	**5**

If the CPU scheduling policy is priority non-preemptive, calculate the average waiting time and average turnaround time.

ANS The order of the processes in the ready queue according to the given priority non-preemptive scheduling policy is P1, P2, P3, P4, P5.

The waiting time for P1 = 0, P2 = 4, P3 = 7, P4 = 8, P5 = 13. The turnaround time for P1 = 4, P2 = 7, P3 = 8, P4 = 13, P5 = 15.

The average waiting time = (0 + 4 + 7 + 8 + 13)/5 = 6.4 The average turnaround time = (4 + 7 + 8 + 13 + 15)/5 = 9.4

Therefore, the average waiting time is 6.4 and the average turnaround time is 9.4 for the given set of processes according to the priority non-preemptive scheduling policy.

Q30. Consider the set of 6 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	0	4
P2	1	5
P3	2	2
P4	3	1
P5	4	6
P6	6	3

If the CPU scheduling policy is Round Robin with time quantum = 2, calculate the average waiting time and average turnaround time.

Consider the following process.

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

Draw a Gantt chart and point out the average waiting time and average turnaround time:

(i) FCFS Scheduling

(ii) SRTF Scheduling

ANS For the set of 6 processes with round-robin scheduling and time quantum = 2:

The Gantt chart is:

|P1|P2|P3|P4|P5|P2|P5|P6|P5|P5| 0 2 4 6 7 9 11 14 16 18

The waiting time for P1 = 0, P2 = 3, P3 = 2, P4 = 4, P5 = 9, P6 = 8 The turnaround time for P1 = 6, P2 = 8, P3 = 9, P4 = 5, P5 = 15, P6 = 11

The average waiting time = (0 + 3 + 2 + 4 + 9 + 8)/6 = 4 The average turnaround time = (6 + 8 + 9 + 5 + 15 + 11)/6 = 8.33 (rounded to 2 decimal places)

Therefore, the average waiting time is 4 and the average turnaround time is 8.33 for the given set of processes according to round-robin scheduling with time quantum = 2.

For the given set of processes:

(i) FCFS Scheduling:

The Gantt chart is:

|P1|P2|P3|P4| 0 8 12 21

The waiting time for P1 = 0, P2 = 7, P3 = 10, P4 = 13 The turnaround time for P1 = 8, P2 = 11, P3 = 19, P4 = 18

The average waiting time = (0 + 7 + 10 + 13)/4 = 7.5 The average turnaround time = (8 + 11 + 19 + 18)/4 = 14

Therefore, the average waiting time is 7.5 and the average turnaround time is 14 for the given set of processes according to FCFS scheduling.

(ii) SRTF Scheduling:

The Gantt chart is:

|P1|P2|P4|P2|P3|P1|P3|P3|P3| 0 1 3 8 9 13 16 18 27

The waiting time for P1 = 0, P2 = 7, P3 = 15, P4 = 2 The turnaround time for P1 = 8, P2 = 12, P3 = 24, P4 = 3

The average waiting time = (0 + 7 + 15 + 2)/4 = 6 The average turnaround time = (8 + 12 + 24 + 3)/4 = 11.75 (rounded to 2 decimal places)

Therefore, the average waiting time is 6 and the average turnaround time is 11.75 for the given set of processes according to SRTF scheduling.

Q31. Consider the set of 5 processes whose arrival time and burst time are given below-

Process Id	Arrival Time	Burst time
P1	**0**	**5**
P2	**1**	**3**
P3	**2**	**1**
P4	**3**	**2**
P5	**4**	**3**

If the CPU scheduling policy is Round Robin with time quantum = 2 unit, calculate the average waiting time and average turnaround time.

ANS To calculate the average waiting time and average turnaround time for the given set of processes using Round Robin scheduling with time quantum = 2, we can use the following table:

Process	Arrival Time	Burst Time	Completion Time	Turnaround Time	Waiting Time
P1	0	5	13	13 - 0 = 13	13 - 5 = 8
P2	1	3	7	7 - 1 = 6	6 - 3 = 3
P3	2	1	3	3 - 2 = 1	1 - 1 = 0
P4	3	2	11	11 - 3 = 8	8 - 2 = 6
P5	4	3	10	10 - 4 = 6	6 - 3 = 3

The completion time for each process is calculated as follows:

- Process P1: The first time slice is from time 0 to 2, the second time slice is from time 2 to 4, and the third time slice is from time 4 to 9. The process completes at time 9+5 = 14.
- Process P2: The first time slice is from time 1 to 2, and the second time slice is from time 2 to 5. The process completes at time 5+3 = 8.
- Process P3: The process completes after one time slice from time 2 to 3.
- Process P4: The first time slice is from time 3 to 4, and the second time slice is from time 4 to 7. The process completes at time 7+2 = 9.
- Process P5: The first time slice is from time 4 to 6, and the second time slice is from time 6 to 9. The process completes at time 9+3 = 12.

Using this table, we can calculate the average waiting time and average turnaround time as follows:

- Average waiting time = (8+3+0+6+3)/5 = 4
- Average turnaround time = (13+6+1+8+6)/5 = 6.8

Therefore, the average waiting time and average turnaround time for the given set of processes using Round Robin scheduling with time quantum = 2 are 4 and 6.8, respectively.

Q32. Analyze the kernel mode & user mode of an operating system. Write steps to switch between the two.

ANS The kernel mode and user mode are two distinct execution modes of an operating system. The kernel mode is a privileged mode in which the operating system executes its core functions, including memory management, process scheduling, and I/O operations. In contrast, the user mode is a non-privileged mode in which user applications run. In the user mode, applications can only access limited resources and cannot execute privileged instructions.

To switch between kernel mode and user mode, the following steps are typically followed:

8. When a user application executes a system call or requests a kernel service, the CPU enters kernel mode.
9. In kernel mode, the operating system executes the requested kernel service and performs any necessary privileged operations.
10. Once the kernel service completes, the operating system returns to user mode.
11. The user application resumes execution and continues with its processing.

The switch from user mode to kernel mode is typically done through a system call instruction, which transfers control from the user mode to the operating system kernel. When the operating system receives the system call, it switches to kernel mode and executes the requested service. The switch from kernel mode to user mode is done by returning from the system call, which transfers control back to the user application. The operating system sets the CPU back to user mode, and the application continues to execute.

Q33. Illustrate briefly the Layered Operating system structure with a neat sketch.

ANS The Layered Operating System Structure is a common architecture used for building operating systems. It is organized as a set of layers, where each layer provides services to the layer above it and receives services from the layer below it. The layers are typically arranged in a hierarchical structure, where the higher layers provide more abstract and user-friendly services, while the lower layers provide more fundamental and hardware-oriented services.

The layered operating system structure typically consists of the following layers:

Hardware Layer: This is the lowest layer of the operating system, which interacts directly with the computer hardware. It provides basic services such as memory management, interrupt handling, and device drivers.

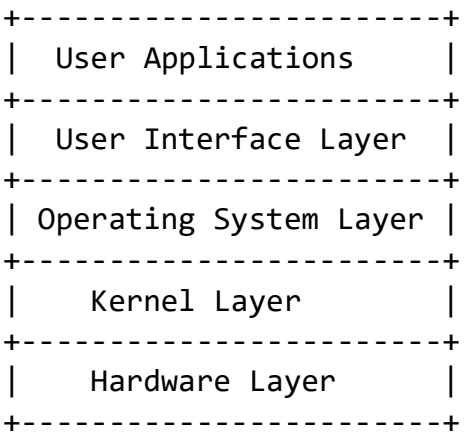
Kernel Layer: This layer provides essential services to the operating system and is responsible for managing resources such as memory, CPU, and input/output devices.

Operating System Services Layer: This layer provides services that are commonly used by applications, such as file management, process management, and network services.

User Interface Layer: This layer provides a user-friendly interface to interact with the operating system. It includes components such as the graphical user interface (GUI) and the command-line interface (CLI).

The following is a diagrammatic representation of the Layered Operating System Structure:

Sql



In this structure, each layer provides services to the layer above it and receives services from the layer below it. The user applications interact with the operating system through the user interface layer, which provides a user-friendly interface. The operating system layer provides services such as process management and file management to the user interface layer. The kernel layer manages system resources such as memory, CPU, and input/output devices, and the hardware layer interacts directly with the computer hardware.

Q34. Consider a system running two processes. Process A starts to read from the keyboard, and while it is waiting for input from the keyboard, system context switches happen and start running process B in user mode. Then, a key is pressed and the corresponding interrupt happens. Where will process B's registers be saved while the interrupt is happening? Justify your answer.

ANS When an interrupt occurs, the current execution context of the interrupted process must be saved so that it can be restored once the interrupt handler has finished. This includes saving the contents of the CPU registers that were being used by the interrupted process.

In the scenario described, Process A is waiting for input from the keyboard, while Process B is running in user mode. When a key is pressed, the corresponding interrupt occurs, and the system switches to kernel mode to handle the interrupt. At this point, the CPU registers that were being used by Process B will be saved on the kernel stack.

This is because the kernel mode has full access to the hardware, and it is the only mode that can execute the interrupt service routine (ISR) that handles the keyboard interrupt. Process B, being in user mode, does not have the privilege to directly access the hardware or handle interrupts. Therefore, its register values must be saved in the kernel stack before the interrupt is handled.

Once the interrupt is handled, the system context switches back to user mode and resumes the execution of Process B, restoring the saved register values from the kernel stack.

Q35. Differentiate between process and program.

ANS A process and a program are related concepts in the context of an operating system, but they are not the same thing. Here are the differences between the two:

Definition: A program is a set of instructions or code written in a specific programming language that can be executed by a computer. A process, on the other hand, is an instance of a program that is running on the computer.

Execution: A program is a static entity that exists on a storage device, such as a hard disk or flash drive, until it is loaded into memory and executed. A process, on the other hand, is a dynamic entity that is created when the operating system loads a program into memory and begins executing it.

Resources: A program does not consume system resources until it is executed. In contrast, a process consumes system resources such as CPU time, memory, and I/O devices while it is running.

Memory: A program is stored in non-volatile memory, while a process is stored in volatile memory.

States: A program is always in the "ready" state, waiting to be executed. A process, however, can be in various states, such as "running", "waiting", or "blocked".

In summary, a program is a set of instructions that is executed by a computer, while a process is an instance of a program that is currently running and consuming system resources.

Q36. Illustrate the rules for the critical section.

ANS In operating systems, a critical section is a section of code that accesses shared resources such as variables or data structures. Only one process can execute its critical section at a time to prevent race conditions and ensure data consistency. The following are the rules that must be followed to ensure the correct execution of critical sections:

Mutual Exclusion: Only one process can execute its critical section at a time. If one process is executing its critical section, other processes must wait for it to finish before executing their own critical sections.

Progress: A process that does not want to enter its critical section should not block other processes from entering their critical sections. If no process is executing its critical section and some processes want to enter their critical sections, then the selection of which process will enter its critical section next cannot be postponed indefinitely.

Bounded Waiting: There must be a bound on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted.

No Assumption of Speed or Number of CPUs: The critical section problem should not depend on assumptions about the speeds or the number of CPUs in the system.

These rules ensure that processes execute their critical sections correctly and that data consistency is maintained in the system. Violating any of these rules can lead to race conditions, deadlocks, and other synchronization problems.

Q37. What are the different states of the process with the help of neat and clean diagram?

In an operating system, a process is a program in execution. The process goes through various states during its execution, and the state transitions are controlled by the operating system scheduler. The following are the different states of a process:

New: This is the initial state of a process. In this state, the operating system creates a new process and allocates the necessary resources such as memory, CPU time, and other system resources.

Ready: In this state, the process is waiting to be assigned the CPU by the scheduler. The process has all the necessary resources, but the CPU is currently executing another process.

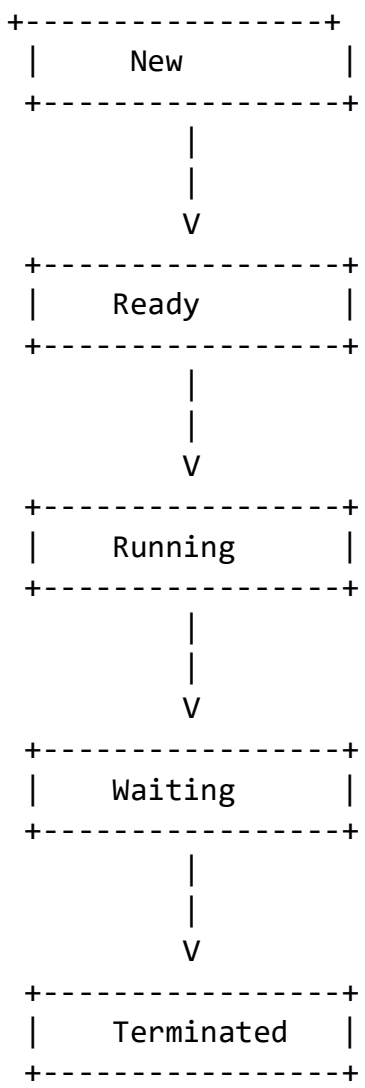
Running: In this state, the process is currently executing on the CPU. The process remains in this state until it is interrupted by an interrupt or it voluntarily releases the CPU.

Waiting: In this state, the process is waiting for a specific event such as input/output (I/O) operations or a signal from another process or the user.

Terminated: In this state, the process has finished its execution or has been terminated by the operating system.

The following diagram shows the different states of a process:

sql



The operating system transitions a process from one state to another depending on the status of the process and the availability of system resources. By managing the process states effectively, the operating system can ensure that the system is running efficiently and that all processes have fair access to system resources.

Q38. What is Long-Term, Short-Term, and Medium-Term schedulers?

ANS Long-term, Short-term, and Medium-term schedulers are three different types of process scheduling algorithms used by an operating system.

Long-term scheduler or Job scheduler selects which processes should be brought into the ready queue from the pool of available processes. It determines the degree of multiprogramming, i.e., how many processes should be in the ready queue at any given point in time. Its goal is to increase system performance by keeping the CPU busy with the appropriate number of processes.

Short-term scheduler or CPU scheduler selects which process should be executed next from the pool of processes present in the ready queue. It selects the process from the ready queue and assigns the CPU to that process. Its goal is to keep the CPU busy by selecting processes quickly and efficiently.

Medium-term scheduler or Swapper swaps processes in and out of the memory to increase the degree of multiprogramming. It removes the processes from memory that are not being used frequently and places them on the disk to free up memory for other processes. When a process is required again, it is brought back into memory from the disk.

In summary, Long-term scheduler determines the degree of multiprogramming, Medium-term scheduler manages memory by swapping processes, and Short-term scheduler selects which process should be executed next from the ready queue.

Q39. Explain how a process is being created and terminated?

A process is created and terminated by the operating system in the following ways:

Process Creation:

The first step in creating a process is to allocate memory for the process's code and data. This is typically done by the operating system's memory manager.

Next, the process's PCB (Process Control Block) is created, which contains information about the process such as its process ID, state, priority, memory allocation, etc.

The process is then assigned a unique process ID and added to the list of processes that the operating system is managing.

Finally, the process is set to the ready state and added to the ready queue so that it can be executed by the CPU.

Process Termination:

When a process completes its execution or needs to be terminated due to an error or user request, the operating system sets the process state to "terminated".

The operating system then releases all the resources that were allocated to the process, such as memory, open files, and any other resources.

The PCB of the process is removed from the list of active processes and the process is marked as terminated in the process table.

If the terminated process has any child processes, the parent process is notified of the child process's termination.

Finally, any remaining system resources held by the terminated process are released, and the CPU is freed up to execute other processes.

Q40. Explain Process Control Block in Operating System.

ANS In an operating system, a Process Control Block (PCB) is a data structure that stores information about a running process. A PCB is created by the operating system when a process is created, and it contains various pieces of information about the process, including:

Process state: This indicates whether the process is running, waiting, or ready.

Process ID (PID): A unique identifier that the operating system assigns to each process.

CPU registers: This includes the contents of the processor's registers and the program counter.

Memory management information: This includes the base and limit registers that define the process's memory space.

Process scheduling information: This includes the priority of the process and its position in the scheduling queue.

I/O status information: This includes a list of I/O devices that the process is using, their status, and any pending I/O requests.

The PCB is used by the operating system to manage the execution of a process. Whenever a context switch occurs, the operating system saves the state of the currently running process in its PCB, and then loads the state of the next process to run from its PCB. The PCB is also used by the operating system to manage process synchronization, scheduling, and termination.

When a process terminates, the operating system deallocates the process's resources and removes its PCB from the system.

Q41. What is context switching? Discuss how context switching happens in OS?

ANS Context switching is the process of saving and restoring the state of a process or thread so that execution can be resumed from where it was previously left off. It is an essential mechanism in operating systems to allow for multitasking and time-sharing.

When a process is running, it has its own set of registers, stack, and program counter that are used to execute its code. When an interrupt or a system call occurs, the operating system saves the current process's state in a data structure called the Process Control Block (PCB) and selects another process to run. This selection is made by the scheduler, which decides which process to run next based on some scheduling algorithm.

The scheduler then restores the new process's state from its PCB and sets its program counter to the location where it was last executing. This process of saving the state of the current process, selecting a new process to run, and restoring its state is called a context switch.

Context switching involves a significant overhead in terms of time and resources, as the operating system must save and restore the entire process state. To minimize this overhead, modern operating systems use various techniques, such as prioritizing processes based on their CPU and I/O requirements, optimizing the scheduling algorithms, and reducing the size of the PCBs.

Q42. Discuss different types of schedulers in OS. Which Schedular is a part of Time-sharing systems?

ANS There are three types of schedulers in operating systems:

Long-Term Scheduler: It selects processes from the job pool and loads them into the ready queue for execution. Its primary goal is to maintain a balanced mix of I/O bound and CPU-bound processes in the system.

Short-Term Scheduler: It selects processes from the ready queue and assigns the CPU to one of them. It is also known as the CPU scheduler and its primary goal is to minimize the average response time for the system.

Medium-Term Scheduler: It selects processes from the ready queue and removes them from the main memory to prevent memory overflow. It is mainly used in systems that support virtual memory.

The short-term scheduler is a part of time-sharing systems. Its goal is to maximize CPU utilization by allowing each process to use the CPU for a small amount of time before switching to another process. This results in the illusion of multiple processes executing simultaneously. Time-sharing systems are commonly used in interactive systems such as personal computers and servers, where multiple users access the system simultaneously.

Q43. What is the purpose of a Process Control Block? What is the function of program counter in PCB?

ANS The Process Control Block (PCB) is a data structure used by the operating system to manage and keep track of each process's state and essential information. It is created when a

process is initiated and deleted when the process is terminated. The primary purpose of a PCB is to hold essential process information so that the operating system can manage and control each process's execution.

The program counter (PC) is a register that keeps track of the current instruction's address that is being executed by the CPU. The program counter value is also saved in the PCB when the process is interrupted or its execution is paused. When the process is resumed, the program counter value is retrieved from the PCB, and the execution of the process is resumed from where it left off. Thus, the program counter in the PCB ensures that the process can resume execution from the point at which it was interrupted or paused.

Q44. Explain Dining Philosophers' problem? Discuss the solution to the Dining philosopher's problem using semaphores.

ANS The Dining Philosophers problem is a classic synchronization problem in computer science. It involves a group of philosophers who sit around a table with a bowl of food and a fork on each side of the bowl. The philosophers spend their time thinking and eating. To eat, a philosopher needs two forks, one on each side of the bowl. However, there are only as many forks as there are philosophers, and a fork can only be held by one philosopher at a time.

The problem arises when all philosophers pick up a fork at the same time and hold onto it, waiting for the other fork to become available. This situation leads to a deadlock where no philosopher can eat.

The solution to the Dining Philosophers problem involves using semaphores to control access to the forks. One possible solution involves assigning a semaphore to each fork. Each philosopher must acquire both forks simultaneously, which can be done by acquiring the semaphore for each fork. Once a philosopher has finished eating, they release both forks by releasing the semaphores.

A pseudocode for a solution using semaphores is:

scssCopy code

```
// initialize an array of semaphores, one for each fork
Semaphore forks[5] = {1, 1, 1, 1, 1};

// function to represent a philosopher's actions
void philosopher(int id) {
    while (true) {
        // philosopher thinks for a while
        think(id);

        // acquire the left fork
        forks[id].wait();
```

```
        // acquire the right fork
        forks[(id + 1) % 5].wait();

        // philosopher eats for a while
        eat(id);

        // release the right fork
        forks[(id + 1) % 5].signal();

        // release the left fork
        forks[id].signal();
    }
}
```

In this solution, each philosopher tries to acquire the left fork first and then the right fork. If a fork is already held by another philosopher, the semaphore associated with that fork will be blocked, and the philosopher will be blocked until the fork becomes available.

By using semaphores to control access to the forks, the Dining Philosophers problem can be solved without deadlocks or starvation.

Q45. What is degree of multiprogramming? How this is controlled? What happens when the degree of multiprogramming is very low?

ANS Degree of multiprogramming refers to the number of programs that can be present in the main memory of a computer system at a given time. It is controlled by the operating system and varies depending on the amount of available memory, the size of the programs being executed, and the scheduling algorithm being used.

When the degree of multiprogramming is very low, it means that only a small number of programs can be present in the main memory at any given time. This can lead to poor resource utilization, longer waiting times for program execution, and decreased system throughput. Processes may spend more time waiting for resources to become available, resulting in a longer turnaround time.

To increase the degree of multiprogramming, the operating system may use various techniques such as swapping, paging, or virtual memory. Swapping involves moving a process from main memory to secondary storage, such as a hard disk, when it is not being executed. Paging involves dividing a process into small fixed-sized blocks called pages and loading only the required pages into main memory. Virtual memory involves the use of a portion of the hard disk as an extension of the main memory, allowing more programs to be executed simultaneously.

In summary, controlling the degree of multiprogramming is essential for efficient utilization of resources and better system performance.