**OS - Mid exam Question Bank 1 Solution**

**Q.1. Define Process Scheduling in Operating Systems, explain its importance and Types.**

**Definition:**  
Process scheduling is the method by which the operating system decides the order in which processes are executed. It ensures efficient utilization of the CPU and other resources while maximizing system performance.

**Importance:**

* Maximizes CPU utilization.
* Minimizes waiting time, turnaround time, and response time.
* Ensures fairness among processes.
* Helps in achieving a balanced system.

**Types of Process Scheduling:**

1. **Long-Term Scheduling:** Determines which jobs or processes are admitted to the ready queue.
2. **Short-Term Scheduling:** Selects a process from the ready queue for execution on the CPU.
3. **Medium-Term Scheduling:** Swaps processes in and out of memory (used in multitasking systems).

**Q.2. Explain the FCFS Scheduling Algorithm. What is the disadvantage of this scheduling algorithm?**

**Explanation:**  
First-Come-First-Serve (FCFS) is a scheduling algorithm where processes are executed in the order they arrive in the ready queue.

**Characteristics:**

* Non-preemptive.
* Easy to implement.
* Suitable for batch systems.

**Disadvantage:**

1. **Convoy Effect:** Longer processes can block shorter ones, causing increased waiting times.
2. **Poor CPU Utilization:** If a long process occupies the CPU, others may starve.

**Q.3. Discuss the Producer-Consumer Problem. How does it introduce data inconsistency with synchronization?**

**Producer-Consumer Problem:**  
It is a classic synchronization problem where two processes (producer and consumer) share a common buffer.

* The **producer** generates data and stores it in the buffer.
* The **consumer** retrieves data from the buffer.

**Issue:**  
Without synchronization, both the producer and consumer may access the buffer simultaneously, leading to data inconsistency:

* The producer might overwrite a slot before it is read.
* The consumer might read uninitialized or incorrect data.

**Q.4. Define Inter-Process Communication (IPC) and explain its importance in a multi-process system.**

**Definition:**  
IPC is a mechanism that allows processes to communicate and synchronize with each other.

**Importance:**

* Facilitates resource sharing.
* Enables process synchronization.
* Ensures data consistency in concurrent execution.

**Examples of IPC Mechanisms:**

* Pipes, Message Passing, Shared Memory, and Semaphores.

**Q.5. What is Mutual Exclusion, and how does it relate to the Critical Section Problem?**

**Mutual Exclusion:**  
It ensures that only one process accesses the critical section (shared resources) at a time to avoid inconsistencies.

**Relation to Critical Section Problem:**  
The critical section problem involves designing mechanisms to maintain mutual exclusion, progress, and bounded waiting in concurrent systems.

**Q.6. Explain Peterson’s Solution to the Critical Section Problem.**

**Explanation:**  
Peterson's Solution is a software-based approach to achieve mutual exclusion between two processes.

* It uses two shared variables: flag (indicates a process's intention to enter the critical section) and turn (indicates whose turn it is).
* Each process checks the other's flag and turn before entering the critical section.

**Steps:**

1. A process sets its flag to true and assigns the turn to the other process.
2. It waits until the other process's flag is false or the turn is not theirs.
3. Once inside the critical section, it resets its flag to false after exiting.

**Q.7. Explain Strict Alternation Solution for the Critical Section Problem.**

**Explanation:**  
Strict Alternation forces processes to alternate their execution in the critical section.

* A shared variable turn indicates whose turn it is.
* A process enters the critical section only if turn matches its identifier.

**Disadvantage:**  
This approach is inefficient as it may lead to unnecessary waiting when a process is not ready.

**Q.8. Explain the following terms:**

1. **Semaphores:**  
   A synchronization primitive used to control access to shared resources.
   * **Binary Semaphore:** Allows one process at a time.
   * **Counting Semaphore:** Allows a fixed number of processes.
2. **Event Counters:**  
   Generalization of semaphores that includes a counter to track the number of occurrences of specific events.
3. **Monitors in IPC:**  
   High-level synchronization construct that encapsulates shared data, operations, and a mutual exclusion mechanism.
4. **Message Passing in IPC:**  
   Communication between processes by sending and receiving messages.

**Q.9. What is the Critical Section in Inter-Process Communication (IPC)?**

**Definition:**  
The critical section is a code segment where processes access shared resources, such as variables or files. Proper synchronization mechanisms are needed to ensure mutual exclusion and data consistency.

**Q.10. Compare and contrast different synchronization techniques:**

| **Technique** | **Ease of Use** | **Performance** | **Deadlock Prevention** | **Scalability** |
| --- | --- | --- | --- | --- |
| **Semaphores** | Moderate, requires manual coding. | High, but prone to race conditions. | Complex to prevent deadlock. | Scales well with proper design. |
| **Monitors** | Easier, encapsulates synchronization logic. | High, with built-in mutual exclusion. | Easier to avoid deadlock. | Suitable for small-scale systems. |
| **Message Passing** | Simple for distributed systems. | Lower due to overhead. | No deadlock in asynchronous communication. | Highly scalable. |

**Q.11. How do semaphores help in solving the Producer-Consumer problem, and what is the role of the buffer in this solution?**

**Semaphores in Producer-Consumer Problem:**  
Semaphores are used to synchronize access to the shared buffer between the producer and the consumer:

1. **Binary Semaphore (Mutex):** Ensures mutual exclusion while accessing the buffer.
2. **Counting Semaphore (Full):** Tracks the number of filled slots in the buffer.
3. **Counting Semaphore (Empty):** Tracks the number of empty slots in the buffer.

**Role of Buffer:**

* Stores data produced by the producer until it is consumed by the consumer.
* Acts as a shared resource requiring synchronization to prevent data inconsistency.

**Q.12. How can message passing be used to solve the Producer-Consumer problem?**

**Message Passing Solution:**

* The producer sends data as messages to the consumer.
* The consumer receives messages and processes them.

**Steps:**

1. The producer sends a message containing the produced item to a message queue.
2. The consumer retrieves the message from the queue.
3. Synchronization is achieved as the producer waits if the queue is full, and the consumer waits if the queue is empty.

**Q.13. Explain Peterson's solution to the critical section problem and highlight how it differs from the strict alternation approach.**

**Peterson's Solution:**

* Uses two variables, flag (to indicate interest) and turn (to decide whose turn it is).
* Ensures mutual exclusion, progress, and bounded waiting.

**Steps:**

1. A process sets its flag to true and gives the turn to the other process.
2. Waits until the other process is not interested (flag is false) or it is its turn.
3. Exits the critical section and resets its flag to false.

**Difference from Strict Alternation:**

* Peterson's solution allows processes to re-enter the critical section if the other is not interested.
* Strict alternation forces processes to alternate, even if one is idle.

**Q.14. Perform the shortest job first (SJF) preemptive scheduling based on the given data:**

**Processes:**

| **Process** | **Arrival Time** | **Burst Time** |
| --- | --- | --- |
| P1 | 0 | 7 |
| P2 | 2 | 4 |
| P3 | 4 | 1 |
| P4 | 5 | 4 |

**Gantt Chart:**

Time: 0 2 4 5 6 10 14

|P1|P2|P3|P2|P4|P1|

**Calculation Table:**

| **Process** | **Completion Time** | **Turnaround Time (CT-AT)** | **Waiting Time (TAT-BT)** |
| --- | --- | --- | --- |
| P1 | 14 | 14 - 0 = 14 | 14 - 7 = 7 |
| P2 | 6 | 6 - 2 = 4 | 4 - 4 = 0 |
| P3 | 5 | 5 - 4 = 1 | 1 - 1 = 0 |
| P4 | 10 | 10 - 5 = 5 | 5 - 4 = 1 |

**Average Waiting Time:** (7 + 0 + 0 + 1) / 4 = 2  
**Average Turnaround Time:** (14 + 4 + 1 + 5) / 4 = 6

**Q.15. What are the differences between a process and a thread?**

| **Aspect** | **Process** | **Thread** |
| --- | --- | --- |
| **Definition** | A program in execution. | A lightweight unit of a process. |
| **Memory** | Has its own memory space. | Shares memory with other threads. |
| **Context Switching** | Slower due to heavy context switching. | Faster due to minimal context switching. |
| **Communication** | Requires IPC mechanisms. | Easier via shared memory. |
| **Overhead** | Higher resource overhead. | Lower resource overhead. |

**Q.16. What is the difference between a Monolithic Kernel and a Microkernel? Provide examples.**

| **Aspect** | **Monolithic Kernel** | **Microkernel** |
| --- | --- | --- |
| **Structure** | All OS services in one module. | Minimal kernel, services in user space. |
| **Performance** | Faster due to fewer context switches. | Slower due to more context switches. |
| **Stability** | Less stable, bugs affect entire system. | More stable, faults are isolated. |
| **Examples** | Linux, Unix | Minix, QNX |

**Q.17. What is the main difference between preemptive and non-preemptive scheduling?**

| **Aspect** | **Preemptive Scheduling** | **Non-Preemptive Scheduling** |
| --- | --- | --- |
| **Definition** | CPU can be taken away from a process. | CPU is allocated until the process finishes. |
| **Interruptions** | Allows interruptions. | No interruptions. |
| **Examples** | Round Robin, SRTF | FCFS, Priority (Non-Preemptive) |

**Q.18. Explain the concept of mutual exclusion in the context of critical sections. How does mutual exclusion prevent race conditions?**

**Mutual Exclusion:**  
Ensures that only one process can access the critical section at a time.

**Prevention of Race Conditions:**

* By locking shared resources, it ensures consistency and prevents simultaneous access.
* Eliminates situations where multiple processes modify shared data concurrently, causing data inconsistency.

**Q.19. What is strict alternation in the context of process synchronization? Discuss its advantages and disadvantages.**

**Definition:**  
Strict alternation forces two processes to take turns accessing the critical section using a shared variable (turn).

**Advantages:**

* Simple to implement.
* Provides mutual exclusion.

**Disadvantages:**

* Wastes time when one process is idle.
* Can cause unnecessary delays.

**Q.20. Given process arrival times and burst times, calculate the Turnaround Time, Waiting Time, and Completion Time using FCFS scheduling.**

**Processes:**

| **Process** | **Arrival Time** | **Burst Time** |
| --- | --- | --- |
| P1 | 0 | 5 |
| P2 | 1 | 3 |
| P3 | 2 | 2 |
| P4 | 3 | 1 |

**Gantt Chart:**

Time: 0 5 8 10 11

|P1|P2|P3|P4|

**Calculation Table:**

| **Process** | **Completion Time (CT)** | **Turnaround Time (CT - AT)** | **Waiting Time (TAT - BT)** |
| --- | --- | --- | --- |
| P1 | 5 | 5 - 0 = 5 | 5 - 5 = 0 |
| P2 | 8 | 8 - 1 = 7 | 7 - 3 = 4 |
| P3 | 10 | 10 - 2 = 8 | 8 - 2 = 6 |
| P4 | 11 | 11 - 3 = 8 | 8 - 1 = 7 |

**Average Waiting Time:** (0 + 4 + 6 + 7) / 4 = 4.25  
**Average Turnaround Time:** (5 + 7 + 8 + 8) / 4 = 7.

**Q.21. Round Robin Scheduling (Time Quantum = 3 units)**

| **Process** | **Arrival Time** | **Burst Time** |
| --- | --- | --- |
| P1 | 0 | 6 |
| P2 | 1 | 8 |
| P3 | 2 | 7 |
| P4 | 3 | 3 |

**Gantt Chart:**

Time: 0 3 6 9 12 15 17 20 22

|P1|P2|P3|P4|P1|P2|P3|P2|P3|

**Completion Times (CT):**

| **Process** | **Completion Time (CT)** | **Turnaround Time (CT - AT)** | **Waiting Time (TAT - BT)** |
| --- | --- | --- | --- |
| P1 | 12 | 12 - 0 = 12 | 12 - 6 = 6 |
| P2 | 22 | 22 - 1 = 21 | 21 - 8 = 13 |
| P3 | 20 | 20 - 2 = 18 | 18 - 7 = 11 |
| P4 | 9 | 9 - 3 = 6 | 6 - 3 = 3 |

**Average Waiting Time:** (6 + 13 + 11 + 3) / 4 = 8.25  
**Average Turnaround Time:** (12 + 21 + 18 + 6) / 4 = 14.25

**Q.22. Define the Operating System.**

An **Operating System (OS)** is system software that acts as an interface between the hardware and the user. It manages hardware resources, software applications, and provides essential services for program execution.

**Q.23. What are the various main functions of an OS?**

1. **Process Management:** Handles process creation, scheduling, and termination.
2. **Memory Management:** Allocates and deallocates memory for programs.
3. **File System Management:** Organizes, stores, and retrieves data on storage devices.
4. **Device Management:** Controls hardware devices like printers, disks, and displays.
5. **Security and Access Control:** Protects data and resources from unauthorized access.
6. **Error Handling:** Detects and handles errors in the system.

**Q.24. Describe the structure of the Process Control Block (PCB).**

A **Process Control Block (PCB)** is a data structure used to store process-specific information.

**Structure of PCB:**

1. **Process ID (PID):** Unique identifier for each process.
2. **Process State:** Current state (e.g., ready, running, blocked).
3. **Program Counter:** Address of the next instruction to execute.
4. **CPU Registers:** Stores CPU-specific data for context switching.
5. **Memory Management Info:** Includes pointers to memory segments.
6. **I/O Status:** Tracks I/O devices allocated to the process.
7. **Priority:** Priority level of the process.

**Q.25. What do you understand about Context Switching?**

**Context Switching** is the process of saving the state of a currently running process and loading the state of another process.

* Occurs during multitasking, interrupt handling, or preemptive scheduling.
* Saves CPU registers, program counter, and other process data in the PCB.

**Q.26. What do you understand about threads? Explain the concept of multithreading in an Operating System.**

**Threads:**

* Threads are the smallest unit of a process.
* A process can have multiple threads sharing memory and resources but executing independently.

**Multithreading:**

* Enables a single process to execute multiple tasks concurrently using threads.
* **Example:** A web browser can use one thread to render a webpage and another to download files.

**Advantages of Multithreading:**

1. Efficient resource sharing.
2. Faster context switching.
3. Better CPU utilization.

**Q.27. Differentiate Multitasking and Multiprocessing.**

| **Aspect** | **Multitasking** | **Multiprocessing** |
| --- | --- | --- |
| **Definition** | Executes multiple tasks (processes or threads) concurrently on a single CPU. | Uses multiple CPUs to execute processes concurrently. |
| **Resources** | CPU time is shared between tasks. | Processes run on separate CPUs. |
| **Speed** | Slower compared to multiprocessing. | Faster due to parallel execution. |
| **Example** | Running a text editor and music player on the same CPU. | Running scientific simulations on a multi-core system. |