process control block.

Process Control Block (PCB):

- The PCB is a data structure that holds all information about a process.

- **Contents of PCB:**

1. **Process ID:** Unique identifier.

2. **Process State:** Current state (Ready, Running, etc.).

3. **Program Counter:** Address of the next instruction.

4. **CPU Registers:** Values of the processor registers.

5. **Memory Management Info:** Details about allocated memory (e.g., base and limit registers).

6. **I/O Information:** Devices and files being used by the process.

Methods of IPC:

1. **Message Passing:**
   - Processes communicate by exchanging messages.
   - Direct Communication:
   - Indirect Communication:

Advantages of Message Passing:

- Simpler to implement in distributed systems.
- No need for shared memory.

2. **Shared Memory:**
   - Processes share a common memory segment.
   - Requires synchronization mechanisms like semaphores or monitors to avoid race conditions.

Advantages of Shared Memory:

- Faster communication as no kernel intervention is required.
- Suitable for large amounts of data transfer.

Multithreading

It refers to the ability of a CPU to provide multiple threads of execution within a single process. A thread is the smallest unit of execution within a process.

- **Multithreading** allows a CPU to perform multiple tasks simultaneously, improving the efficiency of a program, particularly in applications that need to perform tasks like data processing, I/O operations, or handling multiple user requests.

Advantages of Multithreading:

1. Efficient use of CPU.
2. Reduced context switching time.
3. Scalability for multi-core processors.
4. Allows concurrent execution of tasks.

Disadvantages of Multithreading:

1. Requires synchronization to avoid race conditions.
2. Debugging multithreaded programs is challenging.
3. Increased overhead due to thread management.

4. Explain five different scheduling criteria used in the computing scheduling mechanism.

Introduction to Scheduling:

- CPU Scheduling is the process of determining which process in the ready queue should be executed by the CPU. The goal is to maximize CPU utilization, throughput, and fairness.

Scheduling Criteria:

1. **CPU Utilization:**
   - Measures the percentage of time the CPU is actively executing a process.
   - Goal: Maximize CPU utilization (ideally 70-80%).

2. **Throughput:**
   - Throughput refers to the number of processes that complete execution per unit of time.
   - Goal: Maximize throughput by reducing waiting time and turnaround time.

3. **Turnaround Time:**
   - Turnaround time is the total time from submission of a process to its completion.
   - Goal: Minimize turnaround time to improve process completion rate.

4. **Waiting Time:**
   - Waiting time is the total time a process spends waiting in the ready queue before it gets executed.
   - Goal: Minimize waiting time to reduce idle periods for processes.

5. **Response Time:**
   - Response time is the time from submitting a request to the first response.
   - Goal: Minimize response time to improve user interaction with processes.

5. Explain context switching.

- **Context Switching** refers to the process where the CPU switches from executing one process to another. It involves saving the state of the current process and loading the state of the next process.

Steps in Context Switching:

1. **Saving the State of the Current Process:**
   - Save the Program Counter (PC), CPU registers, and other process-specific information.

2. **Loading the State of the Next Process:**
   - Load the saved state (PC, registers) of the next process from its Process Control Block (PCB).

3. **Resuming Execution:**
   - The new process resumes execution from where it left off.

Overhead:

- Context switching introduces overhead due to the time spent in saving and loading process states, which does not contribute to the actual execution of processes.

```
+-----------------+     +-----------------+
| Process A       | ----> | Context Switch  |
+-----------------+     +-----------------+
        |
        v
+-----------------+     +-----------------+
| Process B       | <---- | Context Switch  |
+-----------------+     +-----------------+
```

6. Explain with a neat diagram multi-queue scheduling.

Definition of Multi-Queue Scheduling:

- Multi-Queue Scheduling is a scheduling mechanism that uses multiple queues based on process priority or other criteria (e.g., CPU-bound vs I/O-bound processes).

Working of Multi-Queue Scheduling:

1. Processes are classified into multiple queues based on their type or priority.

2. Each queue may use a different scheduling algorithm, such as Round Robin for foreground processes and First-Come-First-Served (FCFS) for background processes.

3. The highest priority queue is given more CPU time.

```
+-----------------+      +-----------------+
| Queue 1 (RR)    |      | Queue 2 (FCFS)  |
+--------+--------+      +--------+--------+
         |                        |
         v                        v
 +---------------+       +---------------+
 | Process A     |       | Process B     |
 +---------------+       +---------------+
         |                      |
         v                      v
 +---------------+       +---------------+
 | Process C     |       | Process D     |
 +---------------+       +---------------+
```

1. Shared Memory Approach

Definition:

In the shared memory approach, multiple processes communicate by sharing a **common region of memory.** This memory is allocated by the operating system and can be accessed by multiple processes simultaneously.

Working:

- A shared **memory segment** is created and mapped into the address space of the participating processes.

- Processes can read from and write to the **shared memory** region directly, allowing for **efficient** data exchange.

- **Synchronization mechanisms** (like semaphores or mutexes) are often needed to ensure that multiple processes do not access the shared memory concurrently, which could lead to data corruption.

Advantages:

- **High-speed communication:** Direct access to shared memory leads to fast data transfer without the need for copying.
- **Efficient data exchange:** Suitable for large amounts of data that need to be exchanged between processes.

Disadvantages:

- **Synchronization complexity:** Ensuring safe access to shared memory can be challenging, requiring careful use of synchronization mechanisms.
- **Security risks:** Shared memory can be vulnerable to unauthorized access if not properly managed.

Diagram:

```
 +---------+      +-----------+      +---------+
| Process A| ----> | Mailbox | <----> | Process B|
 +---------+      +-----------+      +---------+
```