1. CPU Scheduling Overview

CPU scheduling selects a waiting process from the ready queue and allocates the CPU to it.

- **Dispatcher**: Module that hands CPU control to processes.
- **Objective**: Maximize CPU utilization and process execution efficiency.
- **Types of Scheduling**:
  - **Non-Preemptive**: Process keeps the CPU until it voluntarily releases it.
  - **Preemptive**: CPU can be taken from a process and given to another.

Scheduling Decisions Occur When:

- Process switches from running to waiting state.
- Process switches from running to ready state.
- Process switches from waiting to ready state.
- Process terminates.

2. Scheduling Criteria

- **CPU Utilization**: % of CPU being used (40%-90% in real systems).
- **Throughput**: Number of completed processes per unit time.
- **Turnaround Time**: Total time taken from submission to completion.
- **Waiting Time**: Time spent in the ready queue.
- **Response Time**: Time from request submission to first response.

3. Scheduling Algorithms

A) First-Come, First-Served (FCFS) (Non-Preemptive)

- Simplest algorithm.
- Processes are scheduled in the order they arrive.
- Managed by a queue.
- **Disadvantage**: Long waiting times if CPU bursts vary significantly.

Algorithm:

1. Accept number of processes and burst times.
2. Initialize waiting time of first process as 0.
3. Calculate waiting times iteratively: `WT[i] = WT[i-1] + BT[i-1]`

4. Compute average waiting time.

B) Shortest Job First (SJF) (Preemptive & Non-Preemptive)

- Process with shortest burst time is scheduled first.
- **Preemptive SJF (Shortest Remaining Time First - SRTF)**: Shorter job can preempt the current job.
- **Non-Preemptive SJF**: CPU is assigned until completion.

Algorithm:

1. Accept number of processes and burst times.
2. Sort processes by burst time (ascending order).
3. Compute waiting times iteratively.
4. Compute average waiting time.

C) Priority Scheduling (Preemptive & Non-Preemptive)

- CPU assigned to the process with highest priority.
- **Issue**: Starvation (low-priority processes may never execute).
- **Solution**: Aging (increase priority over time).

Algorithm:

1. Accept number of processes, burst times, and priorities.
2. Sort by priority (ascending order).
3. Compute waiting times iteratively.
4. Compute average waiting time.

D) Round Robin (RR) (Preemptive)

- Time-sharing algorithm.
- Each process gets CPU for a **time quantum**.
- If a process is not finished, it is placed back in the ready queue.

Algorithm:

1. Accept number of processes and burst times.
2. Define a time quantum.
3. Allocate CPU to each process for the time quantum in cycles.
4. If a process still has burst time left, put it back in the queue.
5. Repeat until all processes are done.

4. CPU-OS Simulator

Simulates process and memory management.

Steps:

1. Compile source code.
2. Load program into CPU simulator.
3. Create processes from program.
4. Select scheduling policies and observe execution.
5. Analyze CPU register values and process execution behavior.