OPERATING SYSTEM – TIME (CPU) SCHEDULING

---

1. Introduction to CPU Scheduling

In a multiprogramming operating system, multiple processes reside in memory at the same time. Since a system has only **one CPU**, it must decide **which process should use the CPU and for how long**.
This decision-making process is known as **CPU Scheduling**.

CPU scheduling is a core function of the **Operating System** and is responsible for:

- Efficient CPU utilization
- Fairness among processes
- Reduced waiting time
- Improved system performance

The OS scheduler selects one process from the **ready queue** and allocates the CPU to it.

---

2. Need for CPU Scheduling

CPU scheduling is required because:

1. Some processes perform I/O operations and leave the CPU idle.
2. Multiple processes compete for CPU time.
3. Without scheduling, CPU utilization would be poor.
4. It helps in achieving multitasking and time-sharing.

Example:
While one process waits for I/O, another process can use the CPU, increasing efficiency.

---

3. CPU Scheduler and Dispatcher

CPU Scheduler

The CPU scheduler:

- Selects a process from the ready queue
- Decides which process gets CPU next

Dispatcher

The dispatcher:

- Switches context
- Switches to user mode

- Starts execution of selected process

Dispatcher latency is the time taken to switch from one process to another.

---

4. Scheduling Criteria

To evaluate CPU scheduling algorithms, the following criteria are used:

1. **CPU Utilization** – Keep CPU as busy as possible
2. **Throughput** – Number of processes completed per unit time
3. **Turnaround Time** – Completion time − Arrival time
4. **Waiting Time** – Time spent waiting in ready queue
5. **Response Time** – Time from request submission to first response

---

5. Types of CPU Scheduling

5.1 Preemptive Scheduling

- CPU can be taken away from a process
- Used in time-sharing systems
- Example: Round Robin, Preemptive SJF

5.2 Non-Preemptive Scheduling

- Process keeps CPU until it finishes or blocks
- Simple but less responsive
- Example: FCFS, Non-preemptive SJF

---

6. CPU Scheduling Algorithms

6.1 First Come First Serve (FCFS)

Definition

FCFS is the simplest scheduling algorithm where processes are executed in the **order of their arrival**.

Characteristics

- Non-preemptive
- Uses FIFO (First In First Out) queue

Advantages

- Simple and easy to implement

- No starvation

Disadvantages

- High waiting time
- Poor response time
- Suffers from **Convoy Effect**

Example

If a long process arrives first, shorter processes must wait, reducing efficiency.

6.2 Shortest Job First (SJF)

Definition

The process with the **shortest CPU burst time** is scheduled first.

Types

1. **Non-Preemptive SJF**
2. **Preemptive SJF (Shortest Remaining Time First – SRTF)**

Advantages

- Minimum average waiting time
- Optimal scheduling algorithm

Disadvantages

- Difficult to predict burst time
- Starvation of long processes

Use Case

Batch processing systems.

6.3 Priority Scheduling

Definition

Each process is assigned a priority.
The process with the **highest priority** gets the CPU first.

Types

- Preemptive Priority Scheduling
- Non-Preemptive Priority Scheduling

Advantages

- Important tasks get CPU earlier
- Flexible scheduling

Disadvantages

- Starvation of low-priority processes
- Priority inversion problem

Solution

Aging – gradually increase priority of waiting processes.

6.4 Round Robin (RR) Scheduling

Definition

Each process is given a **fixed time quantum** and CPU is shared cyclically.

Characteristics

- Preemptive
- Time-sharing algorithm

Advantages

- Fair to all processes
- Good response time
- Suitable for interactive systems

Disadvantages

- Too small time quantum → overhead
- Too large time quantum → behaves like FCFS

Applications

Time-sharing systems such as multi-user environments.

---

7. Comparison of Scheduling Algorithms

Algorithm	Preemptive	Waiting Time	Response Time	Starvation
FCFS	No	High	Poor	No
SJF	Yes/No	Minimum	Good	Yes
Priority	Yes/No	Medium	Medium	Yes
Round Robin	Yes	Medium	Excellent	No

---

8. Turnaround, Waiting & Response Time

Turnaround Time

Time from process submission to completion.

Waiting Time

Total time process waits in ready queue.

Response Time

Time between request submission and first response.

These metrics help in evaluating scheduling performance.

---

9. Time Scheduling in Modern Operating Systems

Modern OS like Linux and Windows use:

- Multilevel queue scheduling
- Multilevel feedback queue scheduling
- Dynamic priorities

These methods combine fairness and efficiency.

---

10. Conclusion

CPU time scheduling is a fundamental concept in Operating Systems.
Different algorithms are used depending on system requirements.

- **FCFS** is simple but inefficient
- **SJF** gives minimum waiting time
- **Priority scheduling** favors important tasks
- **Round Robin** is best for time-sharing systems

A good scheduler balances **efficiency, fairness, and responsiveness**.