



CPU Scheduling

In this lesson

- CPU Scheduler
- Preemptive and Non-preemptive Scheduling
- Dispatcher
- Scheduling Criteria
- First- Come, First-Served (FCFS) Scheduling
- Shortest-Job-First (SJF) Scheduling
- Shortest-Remaining-Time-First (SRTF) Algorithm
- Priority Scheduling Algorithm
- Round Robin (RR) Scheduling



CPU Scheduler

- The **Short-Term Scheduler** is responsible for selecting one process from the ready queue and assigning the CPU to it for execution. This queue can be arranged in several different ways, depending on the scheduling algorithm used.
- **Events that trigger CPU scheduling decisions:**
 1. **Process switches from running to waiting state:** This occurs, for example, during an I/O operation.
 2. **Process switches from running to ready state:** This happens when a process is preempted by a higher-priority process.
 3. **Process switches from waiting to ready state:** This can occur when a waiting process has completed its I/O operation.
 4. **Process terminates:** The process finishes and releases the CPU.

Preemptive vs Non-preemptive Scheduling

- Scheduling triggered by events **1 and 4** is **non-preemptive** (i.e., the process holds the CPU until it finishes or enters a waiting state).
- Scheduling triggered by events **2 and 3** is **preemptive**, where the CPU can be taken away from the running process.

Important Considerations in Preemptive Scheduling:

- **Access to shared data:** Processes must be synchronized to avoid data corruption.
- **Preemption during kernel mode:** Special care must be taken to avoid interrupting the operating system while it handles critical tasks.
- **Interrupts during key OS activities:** Critical sections should not be interrupted to prevent inconsistent system states.

Dispatcher

The **Dispatcher Module** is responsible for:

- **Switching context:** Saving the state of the current process and restoring the state of the next process.
- **Switching to user mode:** Transferring control to the user program.
- **Jumping to the correct program location:** Ensuring the user program resumes correctly from where it was last paused.
- **Dispatch Latency:** This refers to the time taken by the dispatcher to stop one process and start another. It includes the time for context switching and transitioning between user and kernel modes.

Scheduling Criteria

- **CPU utilization:** keep the CPU as busy as possible
- **Throughput** of processes: that complete their execution per time unit
- **Turnaround time:** amount of time to execute a particular process
- **Waiting time:** amount of time a process has been waiting in the ready queue
- **Response time:** amount of time it takes from when a request was submitted until the first response is produced, not output (for time-sharing environment)

Optimization Criteria

- Max CPU utilization
- Max throughput
- Min turnaround time
- Min waiting time
- Min response time

First- Come, First-Served (FCFS) Scheduling

<u>Process</u>	<u>Burst Time</u>
P_1	24
P_2	3
P_3	3

- Suppose that the processes arrive in the order: P_1, P_2, P_3
The Gantt Chart for the schedule is:



- Waiting time for $P_1 = 0$; $P_2 = 24$; $P_3 = 27$
- Average waiting time: $(0 + 24 + 27)/3 = 17$

FCFS Scheduling

Suppose that the processes arrive in the order:

$$P_2, P_3, P_1$$

- The Gantt chart for the schedule is:



- Waiting time for $P_1 = 6$; $P_2 = 0$; $P_3 = 3$
- Average waiting time: $(6 + 0 + 3)/3 = 3$
- Much better than previous case
- Convoy effect** - short process behind long process
 - Consider one CPU-bound and many I/O-bound processes

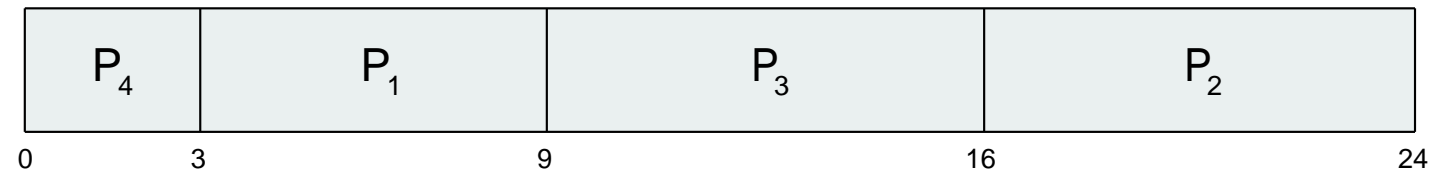
Shortest-Job-First (SJF) Scheduling

- Each process is associated with the estimated length of its next CPU burst, representing how long the process will need the CPU for its next run.
- The system schedules the process with the shortest estimated time remaining for execution next.

Example

<u>Process</u>	<u>Arrival Time</u>	<u>Burst Time</u>
P_1	0.0	6
P_2	2.0	8
P_3	4.0	7
P_4	5.0	3

- SJF scheduling chart



- Average waiting time = $(3 + 16 + 9 + 0) / 4 = 7$

Example - continued

- Order the processes by their burst times:

- P4: 3 ms
- P1: 6 ms
- P3: 7 ms
- P2: 8 ms

- Determine the waiting time for each process:

- **P4** starts immediately at 0 ms. So, waiting time for P4 = 0 ms.
- **P1** starts after P4 finishes. So, waiting time for P1 = 3 ms.
- **P3** starts after P1 finishes. So, waiting time for P3 =
 $3 \text{ ms (P4)} + 6 \text{ ms (P1)} = 9 \text{ ms}.$
- **P2** starts after P3 finishes. So, waiting time for P2 =
 $3 \text{ ms (P4)} + 6 \text{ ms (P1)} + 7 \text{ ms (P3)} = 16 \text{ ms}.$

Example – continued

Calculate the average waiting time:

$$\text{Average waiting time} = \frac{\text{Total waiting time}}{\text{Number of processes}}$$

$$\text{Total waiting time} = 0 + 3 + 9 + 16 = 28 \text{ ms}$$

$$\text{Average waiting time} = \frac{28 \text{ ms}}{7 \text{ ms}}$$

SJF Scheduling Performance

Optimal Performance:

- **Minimizes Average Waiting Time:** SJF provides the lowest average waiting time compared to other scheduling algorithms because it executes the shortest jobs first.
- **Challenges:**
 - **Predicting CPU Burst Time:** Estimating the length of the next CPU burst is not straightforward. While it can be theoretically optimal, predicting the exact length of future bursts is a challenge in practical systems.
- **Approximations:** Historical information (past bursts) or probabilistic approaches may be used to approximate the next burst's length.

Shortest-Remaining-Time-First (SRTF) Algorithm

- This algorithm selects the process with the smallest amount of remaining processing time to execute next. If a new process arrives with a shorter remaining time than the current process, the current process is preempted, and the new process is executed.
- **How it Works**
 - 1. Identify the Process with the Shortest Remaining Time:** At each moment, select the process with the shortest remaining CPU burst time.
 - 2. Preemption:** If a new process arrives with a shorter remaining time than the current running process, preempt the current process and start executing the new process.
 - 3. Repeat:** Continue this process until all processes are completed.

Example

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

- At time 0, P1 starts execution.
- At time 1, P2 arrives with a burst time of 4. Since $4 < 7$ (remaining time of P1), preempt P1 and execute P2.
- At time 2, P3 arrives with a burst time of 9. Continue executing P2 as it has the shortest remaining time.
- At time 3, P4 arrives with a burst time of 5. Continue executing P2. P2 completes at time 5. Now compare P1 (remaining time 7), P3 (burst time 9), and P4 (burst time 5). P4 is selected.
- Continue this process until all processes are completed.

Priority Scheduling Algorithm

- **Priority Scheduling** is a CPU scheduling algorithm where each process is assigned a priority. Processes with higher priority are executed before those with lower priority. If two processes have the same priority, they are scheduled based on their arrival time or another secondary criterion.

Example

Consider five processes with the following priorities and burst times:

Process	Burst Time	Priority
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2

The CPU will schedule the processes in the following order based on their priorities (lower number indicates higher priority):

- 1.P2 (Priority 1)
- 2.P5 (Priority 2)
- 3.P1 (Priority 3)
- 4.P3 (Priority 4)
- 5.P4 (Priority 5)

Example – continued

Calculating Waiting Time and Turnaround Time:

1. Waiting Time (WT): Time process spends waiting in the ready queue.
2. Turnaround Time (TAT): Total time taken from arrival to completion (WT + Burst Time).

For our example:

- P2: WT = 0, TAT = 1
- P5: WT = 1, TAT = 6
- P1: WT = 6, TAT = 16
- P3: WT = 16, TAT = 18
- P4: WT = 18, TAT = 19

- Average Waiting Time (AWT) and Average Turnaround Time (ATAT):
- $AWT = (0 + 1 + 6 + 16 + 18) / 5 = 8.2$
- $ATAT = (1 + 6 + 16 + 18 + 19) / 5 = 12$

Issues with Priority Scheduling:

- **Starvation:** Low-priority processes may never execute if higher-priority processes keep arriving.
- **Aging:** Solution to starvation; increase the priority of a process if it has been waiting for too long.

Round Robin (RR) Scheduling

- Round Robin (RR) is a CPU scheduling algorithm that assigns a fixed time unit per process and cycles through them. It is designed to be simple and fair, ensuring that each process gets an equal share of the CPU time.
- Key Characteristics:
 1. **Time Quantum (Time Slice):** Each process is assigned a small unit of CPU time called a time quantum or time slice. The length of this time quantum is crucial in determining the efficiency of the Round Robin scheduling.
 2. **Fairness:** RR is one of the fairest scheduling algorithms, as all processes get an equal share of the CPU. This helps in avoiding starvation.
 3. **Preemptive:** The CPU scheduler picks the first process in the ready queue, assigns the CPU to that process for the duration of the time slice, and then moves on to the next process in the queue if the process's burst time is longer than one time quantum.
 4. **Performance:** The performance of Round Robin depends heavily on the length of the time quantum. If the time quantum is too short, the system may spend more time context switching between processes, reducing CPU efficiency. If it's too long, it behaves like First-Come, First-Served (FCFS) scheduling.

RR- example

- Consider three processes P1, P2, and P3 with burst times of 24, 3, and 3 milliseconds respectively, and a time quantum of 4 milliseconds:
- P1: 24 ms
- P2: 3 ms
- P3: 3 ms
- Execution Steps:

First Cycle:

1. P1 runs for 4 ms (remaining burst time = 20 ms)
2. P2 runs for 3 ms (completed)
3. P3 runs for 3 ms (completed)

Example - continued

Second Cycle: P1 runs for 4 ms (remaining burst time = 16 ms)


Third Cycle: P1 runs for 4 ms (remaining burst time = 12 ms)

Fourth Cycle: P1 runs for 4 ms (remaining burst time = 8 ms)

Fifth Cycle: P1 runs for 4 ms (remaining burst time = 4 ms)

Sixth Cycle: P1 runs for 4 ms (completed)

Example - continued



- Turnaround Time:
 - P1: $24 \text{ ms} + 5 \text{ cycles of } 4 \text{ ms} = 44 \text{ ms}$
 - P2: 3 ms
 - P3: 3 ms
- Waiting Time:
 - P1: $44 \text{ ms (turnaround time)} - 24 \text{ ms (burst time)} = 20 \text{ ms}$
 - P2: 0 ms (no wait)
 - P3: 0 ms (no wait)
- Average Waiting Time: $(20+0+0)/3=6.67 \text{ ms}$
- Average Turnaround Time: $(44+3+3)/3=16.67 \text{ ms}$

Thank you

