

CPU Scheduling: Examples and Solved Problems

Let's look at some practical **examples** of CPU scheduling using several algorithms: FCFS (First Come First Serve), SJF (Shortest Job First), SRTF (Shortest Remaining Time First), and Round Robin. We'll walk through real problems and outline key steps, then check your understanding with a guiding question.

1. FCFS (First Come First Serve)

Example:

Process	Arrival Time	Burst Time
P1	0	5
P2	0	3
P3	0	8

- **Order of execution:** P1 → P2 → P3 (since all arrive at same time)

- **Gantt Chart:**

| P1 | P2 | P3 |

- P1: 0 → 5
- P2: 5 → 8
- P3: 8 → 16

- **Calculations:**

- Completion Time (CT): End time for each process.
- Turnaround Time (TAT) = CT - Arrival Time
- Waiting Time (WT) = TAT - Burst Time

Process	AT	BT	CT	TAT	WT
P1	0	5	5	5	0
P2	0	3	8	8	5
P3	0	8	16	16	8

- **Average Waiting Time:** $(0 + 5 + 8) / 3 = 4.33$
- **Average Turnaround Time:** $(5 + 8 + 16) / 3 = 9.67$ ^[1]

2. SJF (Shortest Job First; non-preemptive)

Example:

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

- At time 0: P1 is the only process, so it starts.
- As new processes arrive, always pick the one with **shortest remaining burst** among ready processes.
- (Try working through the order; what happens at time 7?)

3. SRTF (Shortest Remaining Time First; preemptive)

Example:

Suppose three processes:

Process	Arrival Time	Burst Time
P1	0	6
P2	1	4
P3	2	2

- At time 0: Only P1 runs.
- At time 1: P2 arrives. Now, choose between P1 (5 left) and P2 (4 left); P2 starts.
- At time 2: P3 arrives (burst 2), so it starts.
- Continue in this way.

Try it yourself: What is the average waiting time?

4. Round Robin (RR)

Example: (Time Quantum = 3 units)

Process	Arrival Time	Burst Time
P1	0	5
P2	1	3
P3	2	1

- At time 0: P1 runs (uses 3 units), 2 units left after quantum.
- At time 3: P2 & P3 in queue. Next, P2 runs (3 units, finishes).
- At time 6: P3 runs (1 unit, finishes).
- P1 runs again for its remaining 2 units.

Try tracing the order and completion times.

Your Turn!

Pick one algorithm above, and try to:

1. **List the order the processes will execute.**
2. **Draw the Gantt chart.**
3. **Calculate waiting and turnaround times for each process.**



shortest remianing job first

Shortest Remaining Time First (SRTF), also known as preemptive Shortest Job First (SJF), is a CPU scheduling algorithm that always selects the process with the least remaining processing time for execution. It is preemptive, meaning if a new process arrives with a shorter remaining burst time than the currently running process, the CPU switches to the new process immediately. [\[20\]](#) [\[21\]](#) [\[22\]](#) [\[23\]](#)

How SRTF Works

- At any point, the scheduler checks all ready processes.
- If a process arrives with less remaining time than the current process, preemption occurs.
- The algorithm continues making these decisions until all processes finish.

Practical Example

Consider the following processes:

Process	Arrival Time	Burst Time
P1	0	7
P2	1	3
P3	3	4

- At time 0, only P1 has arrived, so it runs.
- At time 1, P2 arrives; since P2 needs only 3 units and P1 needs 6 more, P1 is preempted and P2 runs.
- At time 3, P3 arrives; P2 needs 1 more unit, while P3 requires 4, so P2 continues.
- After P2 finishes, P3 runs (since it needs less than P1's remaining time).
- After P3, P1 resumes and finishes.

Completion and Waiting Times:

Process	Arrival	Burst	Completion	Turnaround	Waiting
P1	0	7	14	14	7
P2	1	3	4	3	0
P3	3	4	8	5	1

- **Average waiting time:** $(7+0+1)/3 = 2.67$ \$ units. [\[21\]](#) [\[22\]](#)

Advantages and Disadvantages

Advantages:

- Very efficient for short processes. [\[22\]](#) [\[23\]](#) [\[20\]](#)
- Minimizes average waiting time, compared to other basic algorithms.

Disadvantages:

- May cause starvation of long processes if short jobs continuously arrive. [\[23\]](#) [\[20\]](#)
- Context switching overhead may become significant. [\[22\]](#)

SRTF is mostly used in batch systems and academic settings because it requires advance knowledge of process burst times to work optimally. [\[20\]](#) [\[23\]](#)

✱

another solved problems on shortest remaining job first

Here is a fully solved problem using the Shortest Remaining Time First (SRTF) algorithm:

Problem Statement

Given four processes:

Process	Arrival Time	Burst Time
P1	0	8
P2	1	2

Process	Arrival Time	Burst Time
P3	2	4
P4	4	3

SRTF Scheduling Steps

- At time 0: Only P1 is available. It starts executing.
- At time 1: P2 arrives with burst time 2, which is shorter than P1's remaining burst (7), so P2 starts executing and preempts P1.
- At time 3: P2 finishes, now P1 (7 left) and P3 (burst 4) are ready. P3 runs because it has the shortest burst.
- At time 4: P4 arrives with burst time 3, which is shorter than both P1 (7) and P3 (still 3 left). P4 runs.
- After P4 finishes (at time 7), now choose shortest among the rest (P3 with 3 left, P1 with 7 left). P3 resumes.
- When P3 finishes (at time 10), P1 resumes.
- When P1 finishes (at time 17), all processes are done.

Gantt Chart

```

| P1 | P2 | P3 | P4 | P3 | P1 |
|----|----|----|----|----|----|
0-1  1-3  3-4  4-7  7-10 10-17

```

Completion Times

Process	Completion Time
P1	17
P2	3
P3	10
P4	7

Calculating Turnaround and Waiting Time

- Turnaround Time (TAT) = Completion Time - Arrival Time
- Waiting Time (WT) = TAT - Burst Time

Process	Arrival	Burst	Completion	Turnaround	Waiting
P1	0	8	17	17	9
P2	1	2	3	2	0

Process	Arrival	Burst	Completion	Turnaround	Waiting
P3	2	4	10	8	4
P4	4	3	7	3	0

- **Average Waiting Time:** $\$ (9 + 0 + 4 + 0) / 4 = 3.25 \$$
- **Average Turnaround Time:** $\$ (17 + 2 + 8 + 3) / 4 = 7.5 \$$

Key Highlights

- SRTF preempts the CPU whenever a new process arrives with a shorter remaining time than the running process. [\[27\]](#) [\[28\]](#) [\[29\]](#)
- This example demonstrates minimizing process waiting and turnaround time, as short jobs finish quickly. [\[29\]](#) [\[27\]](#)