



## **Department of Computer Science & Engineering**

**Course Title: Operating System Lab**

**Course Code: CSE 406**

**Lab Report No: 02**

**Lab Report: Shortest Job First (SJF) Scheduling**

**Submission Date: 07-03-2025**

**Submitted To:**

**Atia Rahman Orthi**

**Lecturer,**

**Department of CSE, UAP**

**Submitted By:**

**Name: Amirul Islam Papon**

**Reg No: 21201076**

**Sec: B**

**Problem Statement:** The Shortest Job First (SJF) scheduling algorithm selects the process with the shortest CPU burst time first, reducing the overall waiting time and improving system efficiency. We aim to implement the Non-Preemptive SJF Scheduling Algorithm, calculate the completion, turnaround, and waiting times, and analyze its performance.

## Steps:

1. **Define the Process Data:**
  - Input consists of process ID, arrival time, and burst time.
2. **Sort the Processes:**
  - Select the process with the shortest burst time among available processes (those that have arrived at or before the current time).
  - If multiple processes have the same burst time, select the one that arrived first.
3. **Schedule Execution:**
  - Assign the selected process to the CPU and update the current time.
  - Record the completion time, then calculate the turnaround and waiting times.
4. **Repeat Until All Processes Are Scheduled:**
  - Continue selecting and executing the shortest available job until all processes are completed.
5. **Display Results:**
  - Print process scheduling order.
  - Compute and display completion time, turnaround time, and waiting time for each process.

## Source Code:

```
shortest_first_job.py > calculate_sjf
1 def calculate_sjf():
2     n = len(process_data)
3
4     current_time = 0
5     completed = []
6     execution_order = []
7     available = []
8
9     while len(completed) < n:
10        # Add newly arrived processes
11        for process in process_data:
12            if process[1] <= current_time and process not in completed and process not in available:
13                available.append(process)
14
15        if available:
16            (variable) execution_order: list # Sort by burst time
17            execution_order.append(current_process[0])
18            current_time += current_process[2]
19            completed.append(current_process)
20        else:
21            current_time += 1 # If no process is available, increment time
22
23    # Calculate Completion, Turnaround, and Waiting times
24    completion_times = {}
25    waiting_times = {}
26    turnaround_times = {}
27
28    current_time = 0
29    for process in execution_order:
30        for p, at, bt in process_data:
31            if p == process:
32                if current_time < at:
33                    current_time = at
34                completion_times[p] = current_time + bt
35                turnaround_times[p] = completion_times[p] - at
36                waiting_times[p] = turnaround_times[p] - bt
37                current_time = completion_times[p]
38                break
39
40    # Print results
41    print("Process\tArrival\tBurst\tCompletion\tTurnaround\tWaiting")
42    for p, at, bt in process_data:
43        print(f"{p}\t{at}\t{bt}\t{completion_times[p]}\t{turnaround_times[p]}\t{waiting_times[p]}")
44
45    # Input data
46    process_data = [
47        ('P1', 2, 6),
48        ('P2', 5, 2),
49        ('P3', 1, 8),
50        ('P4', 0, 3),
51        ('P5', 4, 4)
52    ]
53 ]
54
```

## Output:

Process	Arrival	Burst	Completion	Turnaround	Waiting
P1	2	6	9	7	1
P2	5	2	11	6	4
P3	1	8	23	22	14
P4	0	3	3	3	0
P5	4	4	15	11	7

[Done] exited with code=0 in 0.06 seconds

**Discussion and Conclusion:** The SJF scheduling algorithm efficiently reduces the average waiting time and turnaround time, making it an optimal scheduling technique for batch processing systems. However, SJF has limitations:

- **Starvation Issue:** Longer processes may suffer from indefinite waiting if shorter jobs keep arriving.
- **Non-Preemptive Limitation:** Once a process starts execution, it cannot be interrupted, which may not be ideal for real-time systems.