Department Of Computer Science and Engineering

Course Title: Operating System Lab

Course Code: CSE 406

Title: CPU Scheduling SJF Algorithm

Submitted To	Submitted By
Atia Rahman Orthi	Md. Atikul Islam Atik
Lecturer	Reg No:21201063
Department Of Computer Science & Engineering	Sec:B1

Submission Date: 20.02.2025

1)Problem Statement:

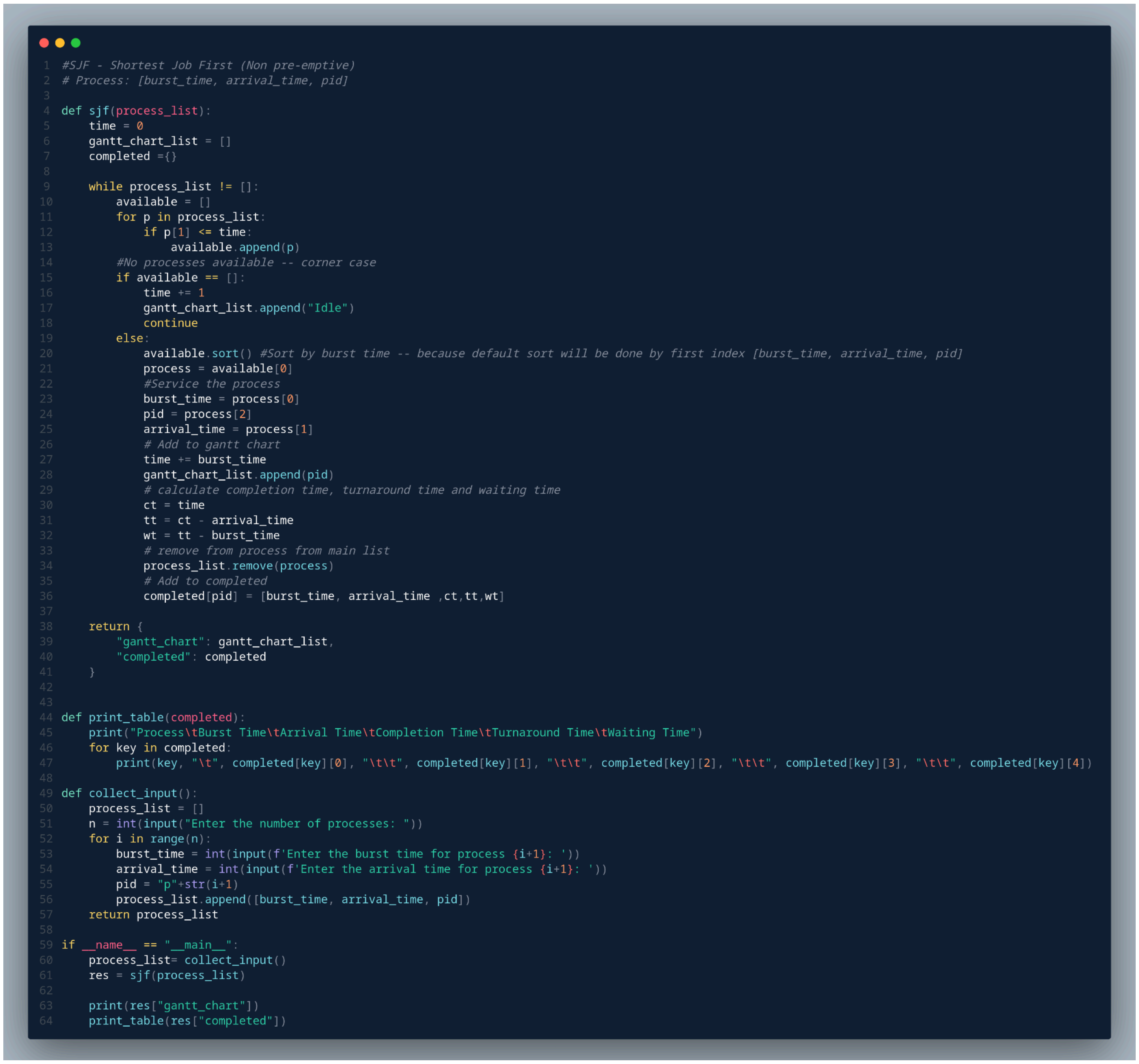
Input:

Process	Burst Time	Arrival Time
P1	6	2
P2	2	2
P3	8	1
P4	3	0
P5	4	4

Output:

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

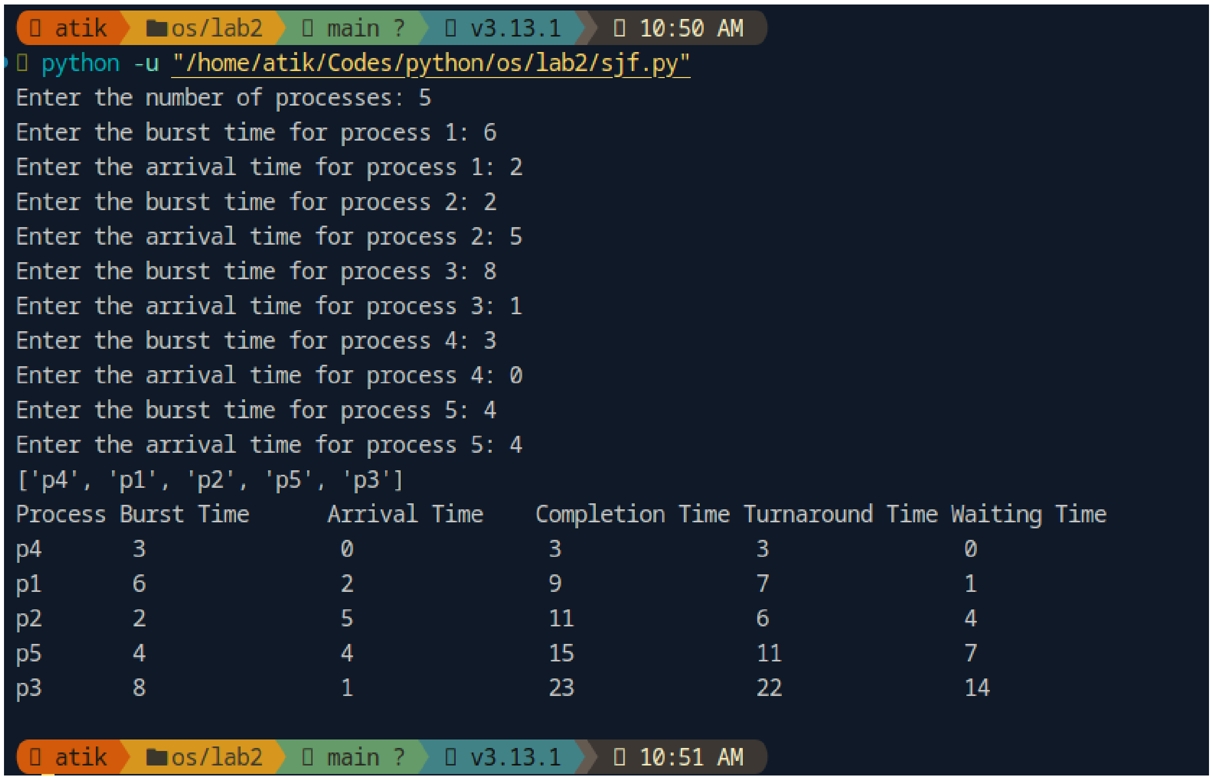
2)Source Code:



```python
#SJF - Shortest Job First (Non pre-emptive)
# Process: [burst_time, arrival_time, pid]

def sjf(process_list):
    time = 0
    gantt_chart_list = []
    completed ={}

    while process_list != []:
        available = []
        for p in process_list:
            if p[1] <= time:
                available.append(p)
        #No processes available -- corner case
        if available == []:
            time += 1
            gantt_chart_list.append("Idle")
            continue
        else:
            available.sort() #Sort by burst time -- because default sort will be done by first index [burst_time, arrival_time, pid]
            process = available[0]
            #Service the process
            burst_time = process[0]
            pid = process[2]
            arrival_time = process[1]
            # Add to gantt chart
            time += burst_time
            gantt_chart_list.append(pid)
            # calculate completion time, turnaround time and waiting time
            ct = time
            tt = ct - arrival_time
            wt = tt - burst_time
            # remove from process from main list
            process_list.remove(process)
            # Add to completed
            completed[pid] = [burst_time, arrival_time ,ct,tt,wt]

    return {
        "gantt_chart": gantt_chart_list,
        "completed": completed
    }


def print_table(completed):
    print("Process\tBurst Time\tArrival Time\tCompletion Time\tTurnaround Time\tWaiting Time")
    for key in completed:
        print(key, "\t", completed[key][0], "\t\t", completed[key][1], "\t\t", completed[key][2], "\t\t", completed[key][3], "\t\t", completed[key][4])

def collect_input():
    process_list = []
    n = int(input("Enter the number of processes: "))
    for i in range(n):
        burst_time = int(input(f'Enter the burst time for process {i+1}: '))
        arrival_time = int(input(f'Enter the arrival time for process {i+1}: '))
        pid = "p"+str(i+1)
        process_list.append([burst_time, arrival_time, pid])
    return process_list

if __name__ == "__main__":
    process_list= collect_input()
    res = sjf(process_list)

    print(res["gantt_chart"])
    print_table(res["completed"])
```

Live Link of Code

3)Output Consular Picture:



```
 atik  os/lab2  main ?  v3.13.1  10:50 AM
 python -u "/home/atik/Codes/python/os/lab2/sjf.py"
Enter the number of processes: 5
Enter the burst time for process 1: 6
Enter the arrival time for process 1: 2
Enter the burst time for process 2: 2
Enter the arrival time for process 2: 5
Enter the burst time for process 3: 8
Enter the arrival time for process 3: 1
Enter the burst time for process 4: 3
Enter the arrival time for process 4: 0
Enter the burst time for process 5: 4
Enter the arrival time for process 5: 4
['p4', 'p1', 'p2', 'p5', 'p3']
Process Burst Time      Arrival Time    Completion Time Turnaround Time Waiting Time
p4       3               0               3               3               0
p1       6               2               9               7               1
p2       2               5               11              6               4
p5       4               4               15              11              7
p3       8               1               23              22              14

 atik  os/lab2  main ?  v3.13.1  10:51 AM
```

4) SJF (Shortest Job First) Algorithm:

Steps:

1. **Input Collection:**
   - The program collects the number of processes, their burst times, and arrival times. Each process is assigned a unique ID (`p1`, `p2`, ..., `pn`).
2. **Process Execution:**
   - Processes are executed in the order of their shortest burst time among those that have arrived by the current time.
   - If no processes are available at the current time, the CPU remains idle until the next process arrives.
3. **Metrics Calculation:**
   - Completion Time (CT): When the process finishes execution.
   - Turnaround Time (TAT): `CT - Arrival Time`.
   - Waiting Time (WT): `TAT - Burst Time`.

4. **Output:**
    - A tabular representation of each process's details (Burst Time, Arrival Time, Completion Time, Turnaround Time, Waiting Time).
    - A Gantt chart showing the execution order.

5) Conclusion:

The Shortest Job First (SJF) scheduling algorithm was successfully implemented in this assignment. The program takes input for the number of processes along with their arrival times and burst times. It then sorts the processes based on their arrival times and reorders them based on their burst times to ensure that the process with the shortest burst time executes first.
The key metrics calculated include:

- Completion Time (CT): The time at which a process completes its execution.
- Turnaround Time (TAT): The total time taken from the arrival of the process to its completion.
- Waiting Time (WT): The time a process spends waiting in the queue before it starts executing.

The output clearly shows the details of each process, including its arrival time, burst time, completion time, turnaround time, and waiting time. The results demonstrate that the SJF algorithm reduces the average waiting time compared to FCFS. However, it can lead to potential starvation for longer processes if shorter jobs keep arriving frequently.
By prioritizing shorter jobs, SJF ensures efficient CPU utilization and minimizes the overall waiting time, making it a better choice than FCFS in scenarios where minimizing waiting time is critical. However, care must be taken to avoid indefinite postponement of longer processes.