CS 314 - OS Laboratory Assignment 4

Cebajel Tanan (210010055)
Tanishq Trivedi (210010056)

February 4th, 2024

Contents

Introduction

We have implemented two scheduling schemes, namely "Shortest Job First"(SJF) and "Round Robin" (RR).

- Shortest Job First (SJF) is a scheduling algorithm that prioritizes tasks based on their expected execution time. The goal is to assign the processor to the task with the shortest burst time first, aiming to minimize waiting times and improve overall system efficiency. SJF can be preemptive or non-preemptive, depending on whether it allows tasks to be interrupted if a shorter task arrives while another is running. We have implemented non-preemptive version of SJF. Overall, SJF tries to minimize turnaround time.
- In a Round Robin (RR) scheduling system, each task is assigned a fixed time quantum or time slice, and the processor cycles through these tasks in a circular order. When a task's time quantum expires, it is moved to the back of the queue, and the next task in line gets a chance to execute. This continues in a round-robin fashion until all tasks are completed or until a predefined condition is met. The algorithm is simple, easy to implement, and ensures that each task gets an equal share of the processor's time, promoting fairness in resource utilization.

Characteristics

SJF

- **Objective:** Minimize waiting times and turnaround times.
- **Criterion:** Prioritizes tasks based on their expected or actual CPU burst time.
- **Non-preemptive SJF:** Selects the next task only when the current task completes its execution.
- **Optimality:** Non-preemptive SJF is optimal in minimizing waiting times, assuming burst times are accurately predicted (which can be challenging in practice).
- **Efficiency:** Aims for efficient resource utilization by quickly processing short-duration tasks.

RR

- **Objective:** Equally distribute processing time among multiple tasks.
- **Mechanism:** Assigns a fixed time quantum to each task, cycling through them in a circular order.

- **Fairness:** Ensures each task gets an equal share of the processor's time.
- **Execution:** A task runs for its allotted time quantum and is moved to the back of the queue if not completed.
- **Simple:** Simple and easy to implement scheduling algorithm.
- **Resource Utilization:** Promotes fairness in resource utilization.
- **Preemption:** It is preemptive in nature.
- **Context Switching:** Involves frequent context switching between tasks, hence, significant overhead on the scheduler.

Test Cases For SJF

Suitability of SJF

```
0 5 -1
0 5 -1
0 5 -1
0 200 -1
```

Output:

```
System's Overall Performance:
        System Throughput: 0.0186047
        Total Cycles: 215
        Average Turnaround Time: 61.25
        Average Waiting Time: 7.5
        Average Response Time: 7.5
        Average Penalty Ratio: 0.690892
```

We can see that when all processes arrived together, SJF choose shortest job ensuring high availability of resources and minimize waiting times of processes and thus minimizing average turnaround time.

Shortcoming of SJF

```
0 200 -1
1 5 -1
2 5 -1
3 5 -1
```

Output:

```
System's Overall Performance:
        System Throughput: 0.0186047
        Total Cycles: 215
        Average Turnaround Time: 206
        Average Waiting Time: 152.25
        Average Response Time: 152.25
        Average Penalty Ratio: 0.268033
```

Even though all processes are same as in previous example, average turnaround time is not same for both the examples. Here, SJF failed in minimizing average turnaround time as shorter jobs arrived later and SJF could not preempt current running task for shorter jobs, resulting in high average turnaround times.

Test Cases For RR

Suitability of RR

```
0 5 -1
3 5 -1
6 5 -1
9 5 -1
```

Output:

```
System Averages and Throughput:
        Average Turnaround Time: 10.5
        Average Waiting Time: 5.5
        Average Response Time: 0.5
        Average Burst Time: 5
        Average I/O Time: 0
        System Throughput: 0.2
        Total Time Taken: 20
```

We can clearly see that the average response time is 0.5, even though there were four processes arriving at different times with equal burst times.

Shortcoming of RR

```
0 5 -1
0 5 -1
0 5 -1
0 5 -1
```

Output:

```
System Averages and Throughput:
        Average Turnaround Time: 18.5
```

```
        Average Waiting Time: 12.75
        Average Response Time: 1.5
        Average Burst Time: 5
        Average I/O Time: 0
        System Throughput: 0.2
        Total Time Taken: 20
```

We can see that RR did not produce as effective average response times as it did in previous example, even though we had same processes. As processes arrived together and all had same burst times, average response time skyrocketed.

Test Cases Performance

process1.txt

Output for SJF:

```
System's Overall Performance:
        System Throughput: 0.00489853
        Total Cycles: 1429
        Average Turnaround Time: 755.286
        Average Waiting Time: 544.286
        Average Response Time: 323.714
        Average Penalty Ratio: 0.238084
```

Output for RR:

```
System Averages and Throughput:
        Average Turnaround Time: 870.429
        Average Waiting Time: 659.428571
        Average Response Time: 2.14285714
        Average Burst Time: 203.857143
        Average I/O Time: 7.14285714
        Average Penalty Ratio: 0.21151197
        System Throughput: 0.00489168414
        Total Time Taken: 1431
```

process2.txt

Output for SJF:

```
System's Overall Performance:
        System Throughput: 0.0186722
        Total Cycles: 964
        Average Turnaround Time: 219.944
        Average Waiting Time: 161.944
```

```
        Average Response Time: 113.333
        Average Penalty Ratio: 0.186231
```

Output for RR:

```
System Averages and Throughput:
        Average Turnaround Time: 240
        Average Waiting Time: 181.833333
        Average Response Time: 5.72222222
        Average Burst Time: 53.1666667
        Average I/O Time: 4.83333333
        Average Penalty Ratio: 0.172215912
        System Throughput: 0.01875
        Total Time Taken: 960
```

process3.txt

Output for SJF:

```
System's Overall Performance:
        System Throughput: 0.00574438
        Total Cycles: 2089
        Average Turnaround Time: 960.917
        Average Waiting Time: 760.5
        Average Response Time: 447.833
        Average Penalty Ratio: 0.30702
```

Output for RR:

```
System Averages and Throughput:
        Average Turnaround Time: 1230.5
        Average Waiting Time: 1023.83333
        Average Response Time: 4.58333333
        Average Burst Time: 174.083333
        Average I/O Time: 21.3333333
        Average Penalty Ratio: 0.128598054
        System Throughput: 0.00573888092
        Total Time Taken: 2091
```

rr_strength.txt

Output for SJF:

```
System's Overall Performance:
        System Throughput: 0.2
        Total Cycles: 20
        Average Turnaround Time: 8
        Average Waiting Time: 3
```

```
        Average Response Time: 3
        Average Penalty Ratio: 0.681097
```

Output for RR:

```
System Averages and Throughput:
        Average Turnaround Time: 10.5
        Average Waiting Time: 5.5
        Average Response Time: 0.5
        Average Burst Time: 5
        Average I/O Time: 0
        Average Penalty Ratio: 0.500541091
        System Throughput: 0.2
        Total Time Taken: 20
0.5
```

rr_shortcoming.txt

Ouptut for SJF:

```
System's Overall Performance:
        System Throughput: 0.2
        Total Cycles: 20
        Average Turnaround Time: 12.5
        Average Waiting Time: 7.5
        Average Response Time: 7.5
        Average Penalty Ratio: 0.520833
```

Output for RR:

```
System Averages and Throughput:
        Average Turnaround Time: 18.5
        Average Waiting Time: 12.75
        Average Response Time: 1.5
        Average Burst Time: 5
        Average I/O Time: 0
        Average Penalty Ratio: 0.282292753
        System Throughput: 0.2
        Total Time Taken: 20
```

sjf_strength.txt

Output for SJF:

```
System's Overall Performance:
        System Throughput: 0.0186047
        Total Cycles: 215
        Average Turnaround Time: 61.25
```

```
        Average Waiting Time: 7.5
        Average Response Time: 7.5
        Average Penalty Ratio: 0.690892
```

Output for RR:

```
System Averages and Throughput:
        Average Turnaround Time: 67.25
        Average Waiting Time: 12.75
        Average Response Time: 1.5
        Average Burst Time: 53.75
        Average I/O Time: 0
        Average Penalty Ratio: 0.450148135
        System Throughput: 0.0186046512
        Total Time Taken: 215
```

sjf_shortcoming.txt

Output for SJF:

```
System's Overall Performance:
        System Throughput: 0.0186047
        Total Cycles: 215
        Average Turnaround Time: 206
        Average Waiting Time: 152.25
        Average Response Time: 152.25
        Average Penalty Ratio: 0.268033
```

Output for RR:

```
System Averages and Throughput:
        Average Turnaround Time: 66.75
        Average Waiting Time: 13
        Average Response Time: 0.75
        Average Burst Time: 53.75
        Average I/O Time: 0
        Average Penalty Ratio: 0.449572027
        System Throughput: 0.0186046512
        Total Time Taken: 215
```

Analysis

Average Response Time

Here clearly, RR has lower response time since it is preemptive, on the other hand, since SJF is non preemptive it completes one job and then only moves to next job.

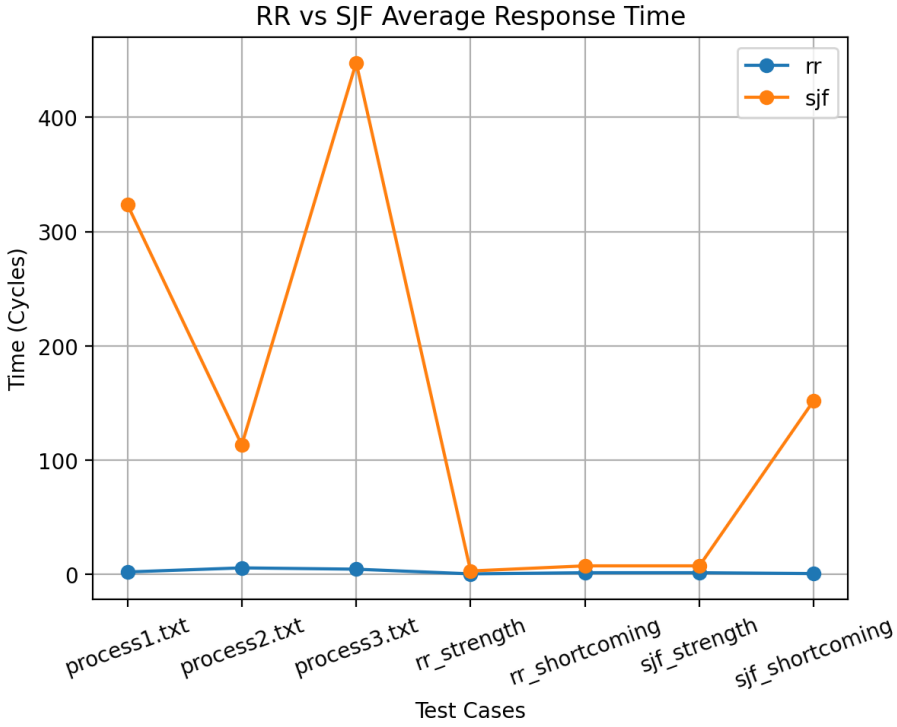

Figure 1: Average Response Time

Average Turnaround Time

SJF has a lower turnaround time than RR because SJF prioritizes the execution of jobs with the shortest burst times. As a result, shorter jobs get scheduled and completed quickly, leading to lower average turnaround times.

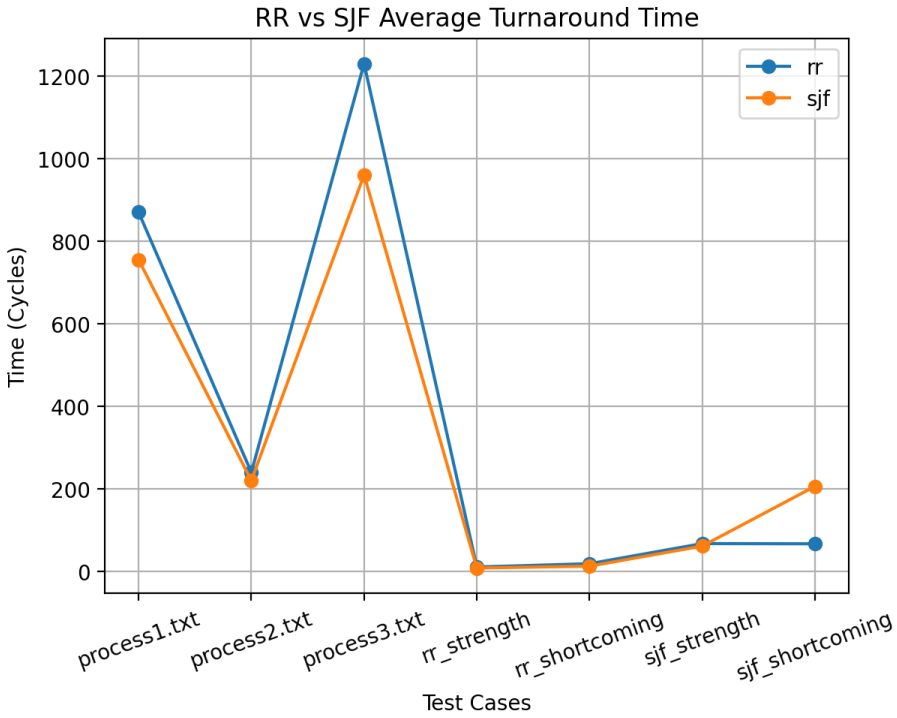

Figure 2: Average Turnaround Time

Average Waiting Time

SJF selects the job with the shortest burst time next for execution. This optimal selection minimizes the time jobs spend waiting in the ready queue, as shorter jobs are given priority.

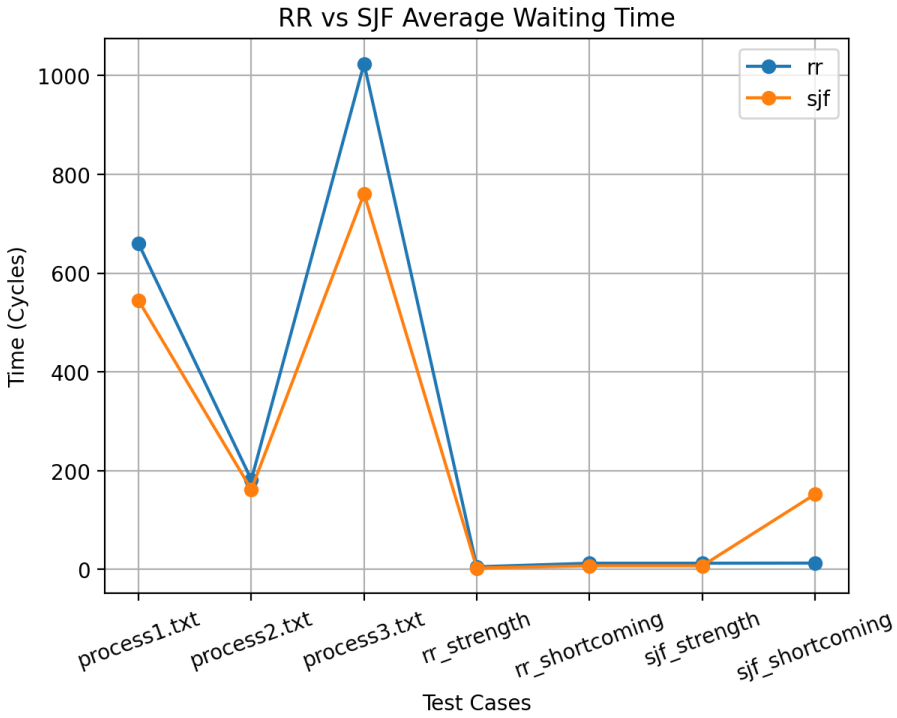

Figure 3: Average Waiting Time

Average Penalty Ratio

SJF may suffer from the starvation problem, where longer jobs may be delayed indefinitely if there are consistently shorter jobs arriving in the system. This can result in higher penalty ratios for the longer jobs that are waiting for execution.

$$\textbf{Penalty Ratio} = \frac{total\_burst\_time}{total\_burst\_time + total\_waiting\_time}$$

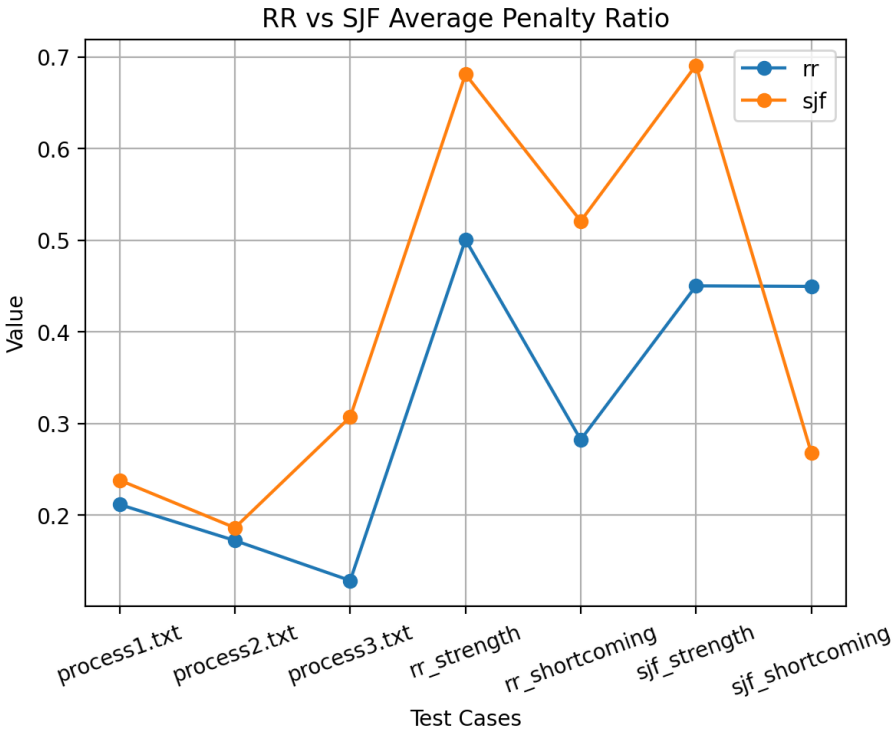

Figure 4: Average Penalty Ratio

Throughput

Here throughput for both SJF and RR overlap, this may be attributed to the goal of keeping the CPU busy as much as possible in both cases.

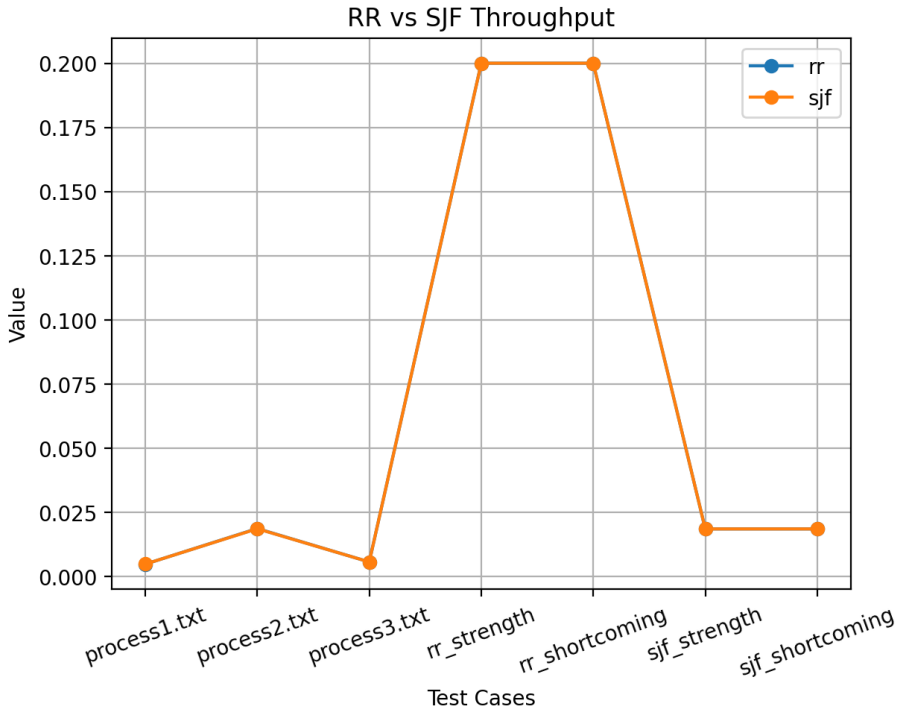

Figure 5: Throughput

Conclusion

In conclusion, Shortest Job First (SJF) excels in minimizing turnaround times by prioritizing shorter jobs, while Round Robin (RR) ensures fairness by allocating fixed time slices to each process. SJF may achieve lower turnaround times for short jobs, but RR prevents starvation and provides a balanced approach in resource allocation, making the choice between them dependent on specific system requirements and characteristics.

Appendix

process1.txt

```
0 100 2 90 2 80 3 70 2 60 2 10 -1
2 80 2 80 2 50 3 70 2 40 2 10 -1
3 70 2 70 2 40 3 70 2 20 2 10 -1
4 10 2 60 2 30 3 70 2 10 2 10 -1
5 3 2 3 2 3 -1
```

```
6 5 -1
10 200 2 3 -1
```

process2.txt

```
0 5 -1
1 3 2 3 2 3 -1
6 10 2 60 2 30 3 70 2 10 2 10 -1
23 3 2 3 2 3 -1
24 70 2 70 2 40 3 70 2 20 2 10 -1
25 3 2 3 2 3 -1
26 80 2 80 2 50 3 70 2 40 2 10 -1
27 3 2 3 2 3 -1
28 25 2 10 -1
29 3 2 3 2 3 -1
31 3 2 3 2 3 -1
33 3 2 3 2 3 -1
35 3 2 3 2 3 -1
40 3 2 3 2 3 -1
40 3 2 3 2 3 -1
42 3 2 3 2 3 -1
43 3 2 3 2 3 -1
45 3 2 3 2 3 -1
```

process3.txt

```
0 15 2 5 2 15 2 5 2 15 2 5 2 15 2 5 2 15 2 5 2 15 2 5 2 15 2 5 2 5 2 15
2 100 2 180 2 10 -1
5 15 2 15 2 15 2 15 2 15 2 15 2 15 2 15 2 15 2 15 2 15  2 15 2 15 2 15 2 15 2 15
8 50 6 100 2 10 -1
12 15 2 5 2 5 2 5 2 5 2 5 2 5 2 5 2 5 2 5 2 5  2 5 2 5 2 5 2 5 2 5 2 5 2 5 2 5 2
20 80 2 80 2 201 2 15 -1
30 15 2 15 2 15 2 15 2 15 2 15 2 15 2 15 2 15 2 15 2 15  2 15 2 15 2 15 -1
35 6 -1
36 8 50 2 -1
37 100 -1
38 5 2 2 -1
40 15 2 15 2 15 2 15 2 15 2 15 2 15 2 15 2 15 2 15 2 15  2 15 2 15 2 15 2 15 2 1
```

rr_strength.txt

```
0 5 -1
3 5 -1
6 5 -1
9 5 -1
```

rr_shortcoming.txt

```
0  5  -1
0  5  -1
0  5  -1
0  5  -1
```

sjf_strength.txt

```
0  5  -1
0  5  -1
0  5  -1
0  200  -1
```

sjf_shortcoming.txt

```
0  200  -1
1  5  -1
2  5  -1
3  5  -1
```
