CS 314 – Operating Systems Lab
Lab4 Report

Name: Sahaja Nandyala **Roll-No:** 200010032

1 Shortest Job First

1.1

This is a non-pre-emptive scheduling algorithm that runs the shortest process in CPU ready queue until the process goes to wait or is terminated state.

The processes in this case have numerous CPU and IO burst cycles. The process with the smallest CPU burst time in ready queue is performed after a running job enters the waiting state for I/O or is terminated.

Processes are executed sequentially in I/O. I have used an queue for I/O to store the processes waiting. This follows the first come first serve scheme. After a process finishes its CPU burst cycle, it goes into the IO queue there is any I/O. Whenever a process' IO burst ends, it goes back to the ready queue if it still has to run. I sorted the array of process in ready queue on basis of current CPU burst time and took the first process to run.

To compile :

```
g++ SJF.cpp
./a.out <file name>
```

1.2 Characteristics

The expected job characteristics of my scheme are the following:

- Average waiting time of a process is reduced
- Average turnaround time of a process is reduced
- Shorter jobs that arrive after longer jobs have a greater waiting time and completion time since they execute after the lengthy job has been completed, implying that these long jobs may suffer from starving. and if only short jobs keeps coming it can cause longer jobs to starve.

1.3 Test process data 1

Test process data for when the Shortest Job First scheme is suitable:
test1.dat

```
<html>
<body>
<pre>
0 3 2 1 -1
2 20 3 1 -1
3 1 -1
</pre></body></html>
```

Here, the processes with short CPU bursts executes first which reduces the average waiting time and turn around time. Here if job 3 executes first than job 2 reducing the turn around time.

1.4 Test Process data 2

Test process data for when the Shortest Job First scheme is not favorable or has shortcomings:

test2.dat

```
<html>
<body>
<pre>
0 200 -1
1 1 1 3 -1
2 5 1 2 -1
5 3 -1

</pre></body></html>
```

The initial process arrives at time 0 in this case. It is a lengthy task, with the initial CPU burst lasting 200 seconds. It executes first, consuming the CPU and preventing the execution of shorter jobs that come just a few seconds later.

1.5 Graphs

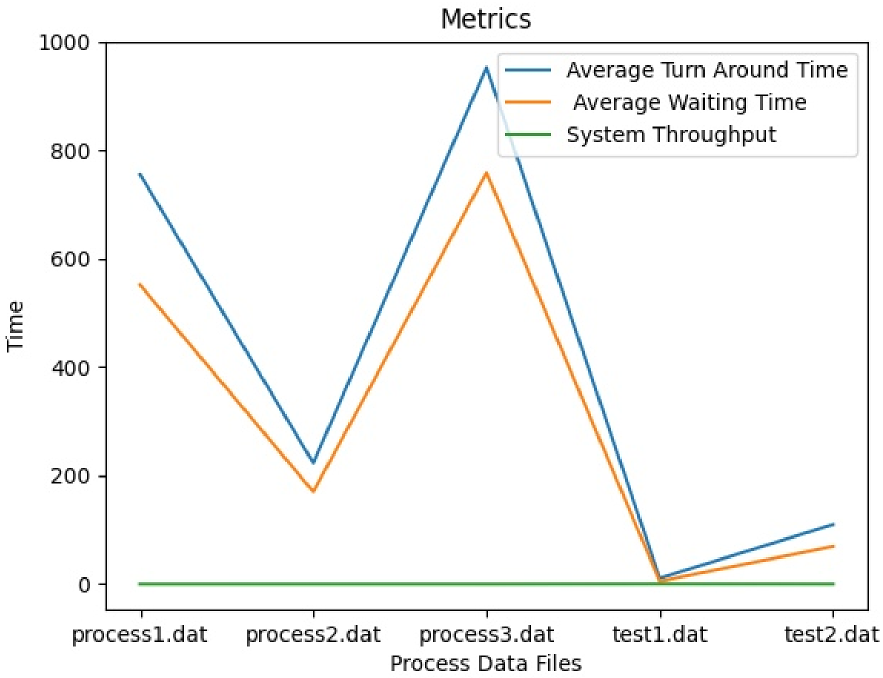

Figure 1: The process data files and their metrics for analysis

2 Shortest Remaining Time First

2.1 T

his is a pre-emptive scheduling algorithm that executes the work with the least amount of time remaining first. If a job with a shorter remaining time comes while a job is running on the CPU, the existing job is preempted and this task begins running. The preempted is added back to the ready queue. In this case, the I/O device runs processes in the same order as in the previous scheduling scheme. An IO queue that follows the FCFS (first come, first served) scheme has been built, and the CPU ready queue is sorted every time to get the work with the least amount of time remaining to perform.

To compile:

```
g++ SRTF.cpp
./a.out <file name>
```

2.2 Characteristics

The expected job characteristics of my scheme are the following:

- The average turnaround time is reduced
- Compared to Shortest Job First, context switches are more.
- Throughput is increased.
- can result in starvation for long jobs if only short jobs keep coming in.

2.3 Test process 1 data

Test process data for when the Shortest Remaining Time First scheme is suitable:
test3.dat

```
<html>
<body>
<pre>
0 2 3 2 -1
0 10 2 3 -1
2 1 1 2 -1
</pre></body></html>
```

The processes in this case are all small jobs that run fast, increasing throughput. In addition, we can see that the average turnaround time is less.

Test process 2 data

Test process data for when the Shortest Remaining Time First scheme is not favorable or has shortcomings:

test4.dat

```
 <html>
<body>
<pre>
0 4 2 3 -1
1 300 4 3 -1
2 1 -1
3 4 3 1 -1
3 1 -1
</pre></body></html>
```

The lengthy task, which comes in after the short ones, is finished last since it always has more time left than the short jobs do. So, the Waiting times and average turnaround times are impacted.

2.4 Graphs

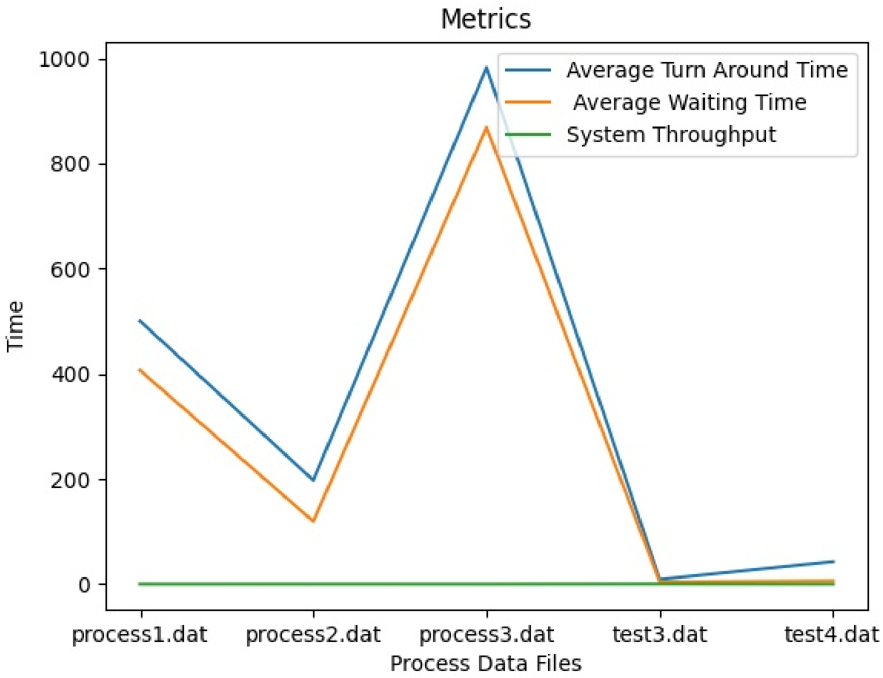

Figure 2: The process data files and their metrics for analysis