CS 314: Operating Systems lab

Vinay – 210010027

Aryan – 210010005

Lab: 4

Part: 1) SJF – shortest job first

Introduction:

The class-driven shortest-job-first scheduler, which was implemented in C++, treats each process as a process object. The SJF method makes use of multiple data structures, such as priority queues for CPU bursts and process vectors. Parallel increments are made to a clock variable to maintain keeping tabs on the simulation's time.

The job in the ready queue with the lowest CPU burst time is selected for execution by the SJF scheduler while the CPU is idle. It is also important to remember that the SJF scheduler is not a pre-emptive scheduler, meaning that one job executing on the CPU must finish before another can start.

Execution:

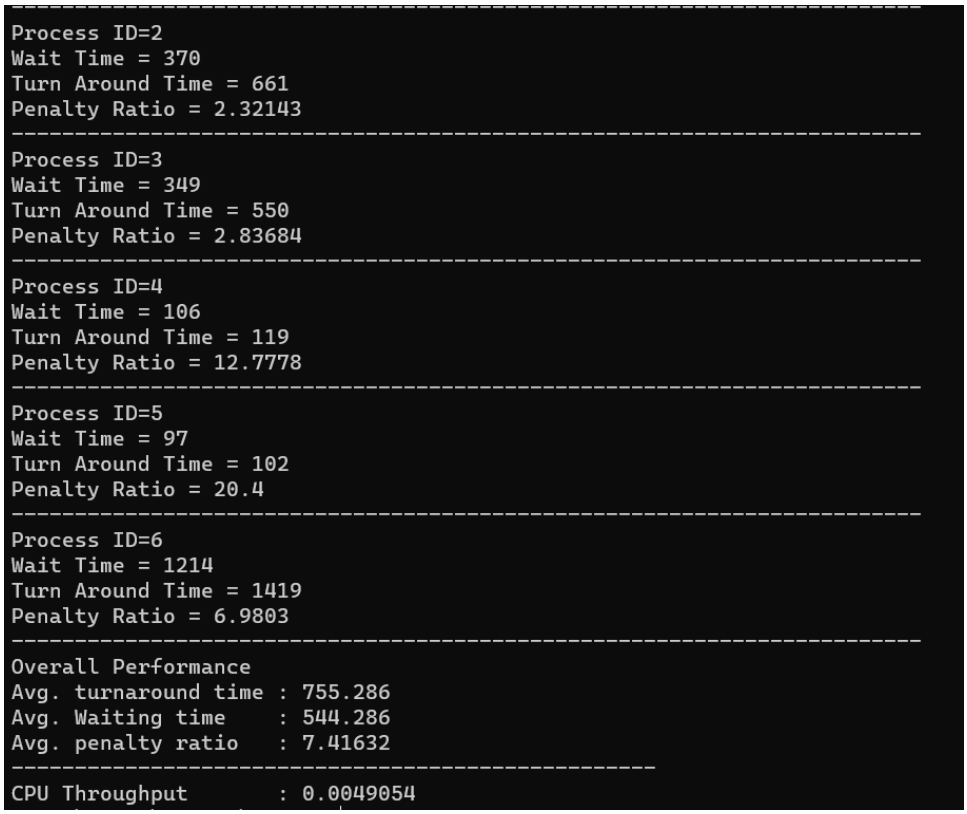
Run: g++ sjf.cpp -o sjf

./sjf process1.dat

Analysis:

- The "convoy effect," in which lengthier processes could make shorter ones wait, can affect non-preemptive SJF and lengthen turnaround times overall.

- Penalty ratio is the ratio of turnaround time to the burst time. In non-preemptive SJF, shorter processes may wait for longer ones, leading to potentially higher penalty ratios.
- Non-preemptive SJF can result in longer waiting times for shorter processes if they arrive while a longer process is already executing.



Part 2) Shortest Remaining Time First

Introduction:

Operating systems use the preemptive scheduling method known as SRTF (Shortest Remaining Time First) to schedule processes. It chooses which process to run next, the one with the shortest burst time left. Reducing the overall processing time and enhancing system responsiveness are the main goals.

Since SRTF is a preemptive algorithm, if a new process with a shorter burst duration comes, the scheduler has the ability to halt the execution of a running process. A preemptive variant of the SJF (Shortest Job First) algorithm is how SRTF is frequently referred to. SRTF makes advantage of the remaining time, whereas SJF schedules depending on the overall burst time of processes. Each process's remaining burst duration dynamically determines its priority. Priority increases with a reduction in remaining time. Because the method is preemptive, it frequently switches contexts, which could result in overhead. Shorter jobs are completed quickly with SRTF, which reduces response time for interactive procedures.

Execution:

Run: g++ srtf.cpp -o srtf

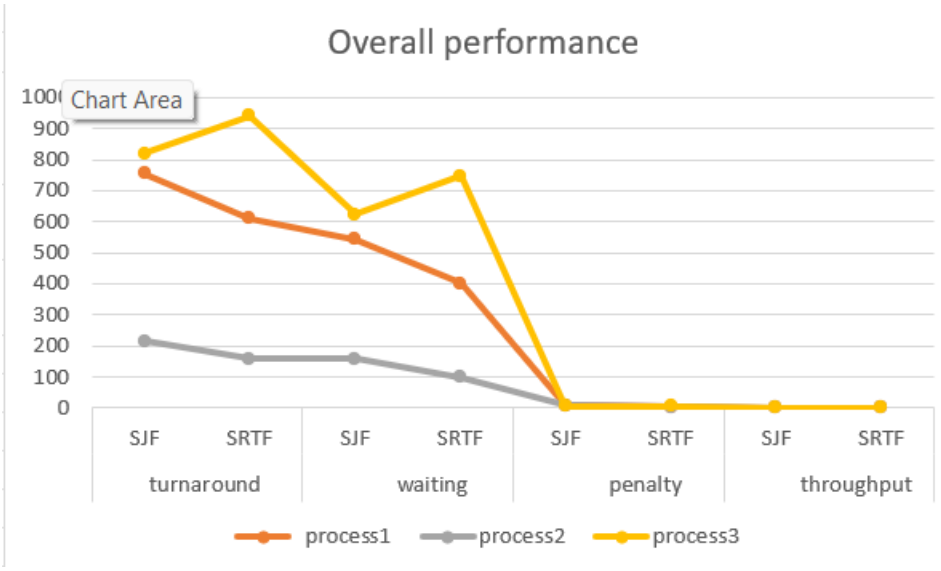
.\srtf process1.dat

Analysis:

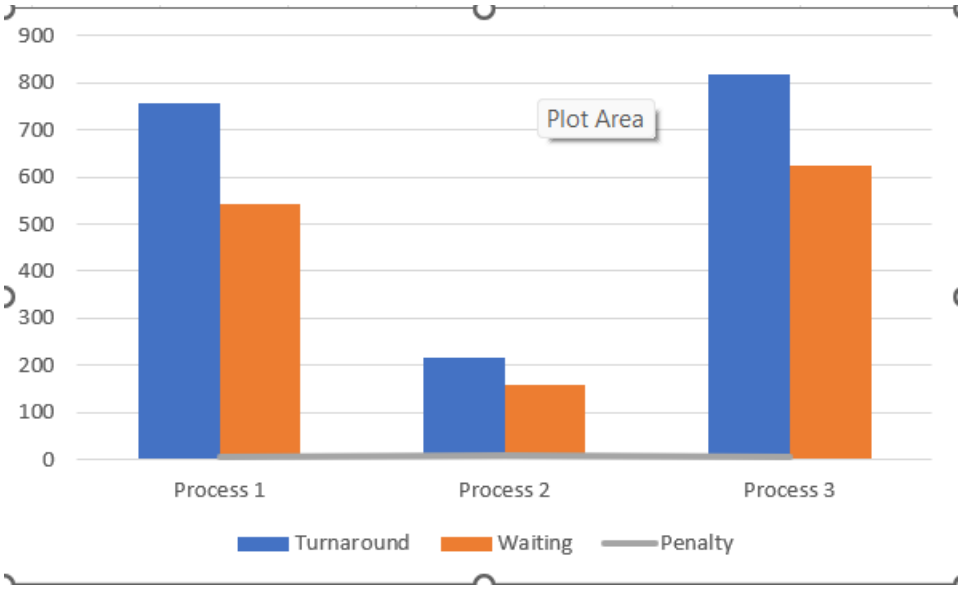
- Preemptive SRTF aims to minimize turnaround time by allowing shorter processes to interrupt longer ones, reducing waiting times.
- Penalty ratios can be lower in SRTF because it allows shorter processes to execute promptly, minimizing waiting times.
- Preemptive SRTF reduces waiting times by allowing the scheduler to switch to shorter processes when they arrive, even if longer processes are running.
- SRTF attempts to mitigate the convoy effect by preempting longer processes when shorter ones arrive, minimizing the impact on overall system responsiveness.

```
Penalty Ratio   : 1.70103
------------------------
Process ID      : 3
Turnaround Time : 278
Waiting Time    : 77
Penalty Ratio   : 1.38308
------------------------
Process ID      : 4
Turnaround Time : 18
Waiting Time    : 5
Penalty Ratio   : 1.38462
------------------------
Process ID      : 5
Turnaround Time : 8
Waiting Time    : 3
Penalty Ratio   : 1.6
------------------------
Process ID      : 6
Turnaround Time : 1420
Waiting Time    : 1215
Penalty Ratio   : 6.92683
------------------------
Overall Performance
-------------------
Avg. Turnaround Time: 611.571
Avg. Waiting Time   : 400.571
Avg. Penalty Ratio  : 2.62196
-------------------
CPU Throughput      : 0.00489168
```

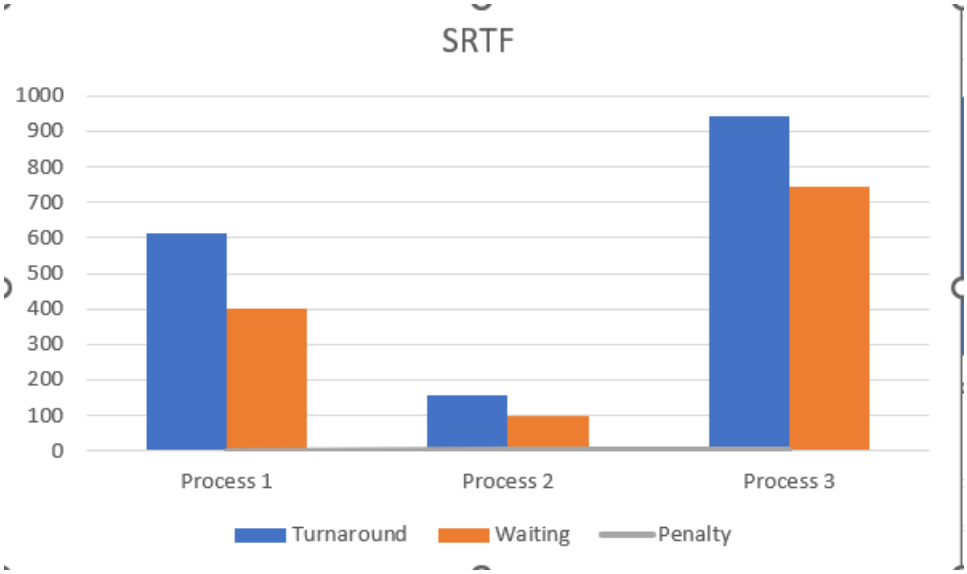
Graphs:


Overall performance
Chart Area
1000
900
800
700
600
500
400
300
200
100
0
SJF
SRTF
SJF
SRTF
SJF
SRTF
SJF
SRTF
turnaround
waiting
penalty
throughput
process1
process2
process3

SJF


900
800
700
600
500
400
300
200
100
0
Plot Area
Process 1
Process 2
Process 3
Turnaround
Waiting
Penalty

SRTF


SRTF
1000
900
800
700
600
500
400
300
200
100
0
Process 1
Process 2
Process 3
Turnaround
Waiting
Penalty