Advanced Operating Systems
COEN 383 Project 2

Group 3
(Andrew Knaus, Dylan Hoover, Venkata Sai Srikar Jilla, Sai Preeti Kakuru, Stuti Jani)

Objective: The objective of this project is to understand and implement the various process scheduling algorithms taught in class. We have written a C program that runs the following algorithms:

- First-come first-serve (FCFS) [non-preemptive]
- Shortest job first (SJF) [non-preemptive]
- Shortest remaining time (SRT) [preemptive]
- Round robin (RR) [preemptive]
- Highest priority first (HPF) [both non-preemptive and preemptive]

The following is the output obtained from the 6 algorithms implemented on 52 processes generated based on the following constraints:

- Arrival time for each process is less than 100 quanta
- Arrival time, expected run time and priority are randomly generated
- There is only one process queue
- No I/O time
- RR time slice is 1 quantum
- 4 queues are used for HPF

First-Come First Serve (Non-Preemptive):
Average response time: 18.6
Average Wait Time: 18.6
Average Turnaround Time: 23.7
Average Throughput: 0.5 processes per quanta unit of time

Shortest Job First (Non-Preemptive):
Average response time: 7.8
Average Wait Time: 8.3
Average Turnaround Time: 11.5
Average Throughput: 0.3 processes per quanta unit of time

Shortest Remaining Time First (Preemptive):
Average response time: 4.4
Average Wait Time: 5.7
Average Turnaround Time: 8.2
Average Throughput: 0.4 processes per quanta unit of time

Round Robin (Preemptive):
Average response time: 29.1
Average Wait Time: 64.8
Average Turnaround Time: 59.8
Average Throughput: 0.2 processes per quanta unit of time

Highest Priority First (Non-Preemptive):
Average response time: 10.5
Average Wait Time: 10.5
Average Turnaround Time: 15.6
Average Throughput: 0.2 processes per quanta unit of time

Highest Priority First (Preemptive):
Average response time: 6.0
Average Wait Time: 32.3
Average Turnaround Time: 37.4
Average Throughput: 0.2 processes per quanta unit of time

Observations:

1. **First Come First Serve (Non-Preemptive):**
   In First Come First Service(FCFS) process which arrives first will be executed irrespective of other processes execution time or wait time. The performance of FCFS depends heavily on the order in which processes arrive in the ready queue. If long-running processes arrive first, short-running processes may have to wait a long time for the CPU, resulting in poor performance. It is not suitable for real-time systems, where response time is critical, as it does not guarantee a timely response to high-priority processes.
2. **Shortest Job First (Non-Preemptive):**
   The Shortest Job First algorithm selects the next process based on the shortest execution time of all the available processes leaving the processes with high execution times to execute at last. This helps in reducing the wait time.

3. **Shortest Remaining Time First (Preemptive):**
   In Shortest Remaining Time First (Preemptive), jobs that require a short time for completion get executed first. This also includes jobs that have a short execution time. The results obtained for these particular observations are pretty much similar to the results of the Shortest Job First (Non-Preemptive)

4. **Round Robin (Preemptive):**
   The round robin algorithm gives all processes an equal time slice for execution. Due to this, each process in the queue only gets on the CPU for a limited amount of time so longer processes do not complete in a single turn. This increases the turnaround time of all the processes and increases the response and wait time as well. This results in less throughput than other algorithms.

5. **Highest Priority First (Non-Preemptive First Come First Serve):**

   In the Highest Priority First (HPF) algorithm, the process with the highest priority is executed first. This involves dividing the processes into queues, each with a different priority. When looking for a process to run, it first checks the highest priority queue and runs the process that arrived first. If there is no available process, then it moves to the next priority queue. Once a process is selected it will continue to execute until it has finished at which point it will give up the CPU for the next process. This algorithm ensures that processes with high priority are executed quickly without needing to wait for lower priority processes. This algorithm results in good performance for high priority processes, but causes the performance of lower priority processes to suffer. While we may not care about bad performance for low priority processes, there are other problems that could arise. For example, if a low priority process that has a long execution time is given the CPU, it could cause future high priority processes to be delayed as they wait for the process to finish. Alternatively, if a high priority process is mistakenly given a low priority, its execution could be delayed causing poor performance for the system.

6. **Highest Priority First (Preemptive Round Robin):**

   The HPF Preemptive with Round Robin scheduling is similar to normal HPF in that it divides the processes into queues. However, after each time segment it has the possibility of switching from the currently running process to a new process. This could happen in two situations. If a process with a higher priority arrives, the current process is stopped and the higher priority process is run. Or, if a process with the same priority arrives, it begins to alternate which process it is running in a round robin fashion. By introducing preemptive scheduling into HPF, we address some of the issues with that we had non-preemptive HPF. Specifically, when a low priority process with a long execution

time is given control of the CPU, it can not be interrupted when a higher priority process comes along. By allowing interruptions, preemptive HPF scheduling results in a faster average response time. However, this also causes the average waiting time as well as turnaround time to increase due to the constant context switching amongst processes.

Conclusion:

Based on response time and waiting time the best performance comes from shortest remaining time first. The worst performing algorithm is round robin and the highest throughput based on the data comes from