CSEN 383: Advanced Operating Systems

Group 3 Project 2 Report

Members: Guru Sushma Kasa, Joshua Kindarara, Samantha Lee, Praful Mamidi, and Lydia Myla

Objective: We implemented the various process scheduling algorithms in C.

- 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) [non-preemptive and preemptive]

Final Output Over 5 Runs:

- **First-Come First Serve (Non-Preemptive):**
  - Avg of Avg TAT: 27.899
  - Avg of Avg WT : 21.264
  - Avg of Avg RPT: 21.264
  - Avg Throughput: 0.1469
- **Shortest Job First (Non-Preemptive):**
  - Avg of Avg TAT: 12.960
  - Avg of Avg WT : 8.023
  - Avg of Avg RPT: 8.023
  - Avg Throughput: 0.1957
- **Shortest Remaining Time First (Preemptive):**
  - Avg of Avg TAT: 12.179
  - Avg of Avg WT : 7.242
  - Avg of Avg RPT: 6.235
  - Avg Throughput: 0.1957
- **Round Robin (Preemptive):**
  - Avg of Avg TAT: 52.162
  - Avg of Avg WT : 46.285
  - Avg of Avg RPT: 7.992
  - Avg Throughput: 0.1645
- **Highest Priority First (Non-Preemptive):**
  - Avg of Avg TAT: 17.233
  - Avg of Avg WT : 11.188
  - Avg of Avg RPT: 11.188
  - Avg Throughput: 0.1603
- **Highest Priority First (Preemptive):**
  - Avg of Avg TAT: 26.203
  - Avg of Avg WT : 20.205

    - Avg of Avg RPT: 4.761
    - Avg Throughput: 0.1594

<u>Observations:</u>

- **First Come First Serve (Non-Preemptive):**
    - The FCFS algorithm executes processes in the order of their arrival, irrespective of other processes' execution time or wait time. Its performance depends heavily on the order in which processes arrive, resulting in a larger response time as seen in our output. If a bigger process arrives first, smaller processes might have to wait a long time for the CPU to run them, resulting in poor performance. FCFS is not suitable for systems where response time is critical, since it doesn't prioritize processes.
- **Shortest Job First (Non-Preemptive):**
    - The SJF algorithm selects, at each dispatch, the job with the smallest service time among those that have arrived and then runs it to completion (non-preemptive). It typically lowers average waiting/response time, but can starve long jobs if short ones keep arriving, so it's best when job lengths are known and fairness is less critical.
- **Shortest Remaining Time First (Preemptive):**
    - The SRT algorithm always runs the job with the least remaining time, preempting the current job if a shorter one arrives (preemptive SJF). It offers excellent average response/wait times and high interactivity, but may cause more context switches and starvation of long jobs under a stream of short tasks.
- **Round Robin (Preemptive):**
    - The RR algorithm uses a fixed time slice to provide a fair share of CPU time to each ready process. It cycles through a queue of processes, running each process for the specified time slice before moving it to the back of the line. This algorithm ensures that no process is starved. However, the algorithm's performance in terms of turnaround time and waiting time can be poor, since each job, regardless of whether it's short or long, has to wait for every other process in the queue to finish its turn.
- **Highest Priority First (Non-Preemptive):**
    - The HPF (non-prememptive) algorithm assigns a priority to each process and prioritizes the processes with a higher priority. Once a process begins, it will run uninterrupted until completion. Then, the next process of the highest priority that arrived first will run. So within a priority queue, the process is chosen by FCFS. This algorithm has a poor response time, turnaround time, and wait time because lower-priority processes might have to wait longer until they actually start. Its reasons for a poor performance metrics are similar to FCFS since it uses that algorithm within each priority queue.

- **Highest Priority First (Preemptive):**
    - The HPF (preemptive) algorithm assigns a priority to each process and prioritizes the process with a higher priority as well. However, a running process can be interrupted by a process of equal priority. When multiple processes of the same priority have already arrived and are at the queue, this algorithm uses RR to choose between them in a given fixed time slice. This algorithm has a good response time because of the fact that higher-priority processes don't have to wait for lower-priority processes to finish. However, the turnaround time and waiting time are poor because those lower-priority processes have to wait for all higher-priority processes to complete. Additionally, since tasks of the same priority are switching between each other (RR), this results in a higher turnaround time and waiting time.

Conclusion:

Overall, the Shortest Remaining Time First algorithm is the best algorithm when it comes to turnaround time, waiting time, and throughput. It also has a relatively low response time, making it the best and most balanced algorithm in general. For specifically targeting a low response time, Round Robin and Highest Priority First (Preemptive) would be the best algorithms to use, coming at the cost of high turnaround and waiting times.