Assignment 3: Concurrency, Shared Memory, Virtual Memory, Files

Emeka Anonyei (101209704)

Steven Arvanitis (101303797)

December 1, 2025

SYSC 4001 - L1

Repo: https://github.com/stevenarvanitis-maker/SYSC-4001-Assignment-3.git

Overview

In this assignment, we built on the work from Assignments 1 and 2 and created a full CPU scheduling simulator. We implemented three scheduling methods: External Priorities, Round Robin, and a combined EP_RR version. The simulator keeps track of memory partitions and moves each process through the usual operating system states such as New, Ready, Running, Waiting, and Terminated. Every change is written into the execution file.

Results and Analysis

External Priorities

EP always runs the process with the smallest process ID first. This makes the behavior more predictable, but it often leads to worse turnaround times for lower priority processes. In CPU heavy workloads, EP finishes the highest priority job very quickly and leaves the others waiting for a long time. Lower priority processes can wait so long that they are almost ignored. Processes that rely on I/O also struggle because when they return from I/O usually they don't run right away. EP is only helpful when strict priority rules are needed.

Round Robin

RR gives every process a quantum of CPU time before switching to the next one. This makes the system feel much more fair and keeps all processes moving forward, leading to better turnaround times as a whole. Long processes will have longer turnaround but on average it will be better. Throughput stays good because no single process holds the CPU for too long. Waiting time becomes shorter for long and low priority jobs, and response time improves for I/O heavy processes since they regularly get another chance to run. The only drawback is that long CPU jobs finish slower because they are constantly being interrupted.

EP_RR Hybrid

The RR and EP scheduler mixes priority rules with assigned quantums of time. Processes with lower process IDs still go first, but they cannot hold the CPU forever. This makes the system more balanced. Throughput becomes better than pure EP. Waiting times for many processes become more reasonable, and turnaround times stop being extreme. Response time improves as well and becomes close to what RR provides. In general, the hybrid scheduler handles a wide range of workloads in a stable way.

Comparative Discussion

Across all simulations, each scheduler worked best under different conditions as expected. CPU-bound workloads tended to favor EP because it quickly serves the highest-priority process, while RR and the hybrid scheduler provided better overall fairness and average turnaround times. I/O-bound workloads performed best with RR and the hybrid approach because both allow processes to resume quickly after an I/O event. In mixed workloads, the hybrid scheduler consistently delivered the most balanced results. Only EP showed signs of starvation, and although RR produced the most context switches, this also made it the most responsive.

Conclusion

This assignment showed the differences between common CPU scheduling methods. EP focuses on priority but sacrifices on average turnaround time. RR has better turnaround times, on average, and quick reactions but slows down large CPU tasks. The EP_RR hybrid finds a middle point by respecting priority without ignoring other processes. After running many simulations and reviewing the results for throughput, waiting time, turnaround time, and response time, we gained a clearer understanding of why each scheduler performs better for certain types of workloads.