Github Link (Part 1): https://github.com/alexrusu77/SYSC4001_A3_P1

Github Link (Part 2): https://github.com/alexrusu77/SYSC4001_A3P2

1. Introduction

The three algorithms compared are EP, which is a non-preemptive, priority-based scheduler (lowest PID is highest priority); RR, which is a preemptive, time-sharing scheduler with a quantum of 100ms in this case; and RR+EP, a combination of both that has external priorities, including preemption, and a 100ms timeout in Round-Robin style.

Metric	EP	RR	RR+EP
Throughput	0.4286	0.4476	0.4476
Average Turnaround Time	17.93	18.60	17.15
Average Wait Time	5.38	7.04	5.60
Average I/O Response Time	0.90	3.52	0.10

Based on the data collected, the RR+EP scheduler appears to have the best performance. It has the highest Throughput (tied with RR), lowest average turnaround time, and a greatly superior response time for I/O tasks, only 0.1 time units.

2. Metric Analysis

2.1 Efficiency and Context Switching Overhead (Average Turnaround and Throughput)

The preemptive algorithms (RR and RR+EP) achieved better overall throughput than the non-preemptive EP. Preemption is important for high throughput because it ensures the system can quickly switch resources to processes that are ready to run, maximizing the utilization of both the CPU and I/O devices concurrently.

However, Round Robin (RR) recorded the worst Average Turnaround Time (18.60). This result is directly attributed to the high context switching overhead that is in its design. In RR, every process, even long CPU-bound ones, is repeatedly interrupted every time quantum (100ms). This constant saving and loading of process states adds time, increasing the overall completion time of jobs.

In contrast, RR+EP achieved the best Average Turnaround Time (17.15). By employing priority (the PID) as the primary scheduling criterion, it minimizes unnecessary context switching, efficiently prioritizing important jobs for completion while retaining preemption for responsiveness.

2.2 Responsiveness and Fairness (Average Wait Time and I/O Response Time)

The non-preemptive EP achieved the worst Average Wait Time (5.38). This success is misleading. In EP, high-priority processes (low PIDs) receive quick service, which reduces the overall average. Conversely, any low-priority, long-running process faces starvation, experiencing excessively high wait times that are obscured by the low average. RR had the highest Average Wait Time (7.04) because its focus on fairness means every process must wait for its turn in the cycle, distributing the wait time more evenly but increasing the average.

As for I/O Response Time,

RR+EP’s performance highlights its advantage, immediate preemptive service for I/O completion. When an I/O-bound job finishes its task and moves from the WAITING state to the READY state, it needs a short, critical CPU burst to process the data before issuing its next I/O request.

The RR+EP scheduler is designed to recognize this urgency. It allows the newly ready I/O-bound process to preempt the currently running job (even if it's a CPU-bound process). This immediate preemption allows the I/O-bound process to quickly finish its short CPU burst and immediately transition back to the WAITING state to continue its I/O operation. By forcing this quick turnaround, RR+EP minimizes the delay on the I/O-bound process, ensuring efficient parallelism by allowing the CPU to handle more important tasks while the slower I/O can perform its own operations without affecting the other processes and potentially decreasing performance.

In contrast,

RR struggles here, recording 3.52. The I/O-bound job must re-enter the general FIFO queue and wait for its quantum, delaying its return to the CPU and hindering efficient I/O device utilization.

Similarly, EP's 0.90 is a direct result of its non-preemptive nature. The I/O-bound job must wait until the current process voluntarily releases the CPU, resulting in a delayed, but often smaller, service time than RR's queue-based delay.

3. Conclusion

EP is efficient for high-priority, CPU-bound tasks but is fundamentally flawed for modern, mixed environments due to starvation and poor I/O management.

RR offers maximum fairness but its strict, blind time-slicing introduces debilitating context switching overhead, leading to the worst overall Turnaround Time.

RR+EP is the optimal design for a general-purpose operating system. By combining the priority-based efficiency of EP with the preemptive responsiveness necessary for I/O-bound processes, it maintains high Throughput and achieves exceptionally low Average Turnaround Time and I/O Response Time. Its results demonstrate that a hybrid, preemptive priority model effectively manages the conflicting demands of both CPU-bound efficiency and I/O-bound responsiveness.