Carleton University

Assignment 1: Group Submission
Part 1 – Report

SYSC 4001A - L3 - GROUP 8: Operating Systems

Student 1: Teereza Allaham (101289630)

Student 2: Sahil Todeti (101259541)

Due: December 1st, 2025

https://github.com/TeerezaAllaham/SYSC4001_A3P2.git

https://github.com/TeerezaAllaham/SYSC4001_A3P1.git

1. Introduction

This report presents an analysis of CPU scheduling algorithms implemented in an Operating System environment.
The schedulers evaluated include:

- First Come First Serve (FCFS),
- Round Robin (RR),
- External Priority (EP),
- External Priority with Round Robin (EP-RR).

Each scheduling algorithm has been tested with twenty simulation scenarios containing CPU-bound processes, I/O-bound processes, and mixed workloads. Performance was evaluated using the key metrics: throughput, average waiting time, average turnaround time, and average response time. These metrics allow us to compare how each algorithm behaves under different system conditions and identify which workloads each scheduler favours.

2. Simulation Methodology

Each scenario consists of multiple processes led by arrival times, CPU burst lengths, I/O frequency, I/O duration, and priority levels (EP and EP-RR). The simulator produced detailed execution traces indicating state transitions, including READY, RUNNING, WAITING, and TERMINATED. Metrics were derived from these traces. To ensure broad coverage, the scenarios included: CPU-bound workloads (no I/O operations), I/O-bound workloads (frequent I/O interruptions,) Mixed workloads (combination of CPU- and I/O-intensive processes), Workloads with skewed priorities (EP and EP-RR only). The resulting dataset allowed for consistent comparisons across all algorithms.

3. Results and Metric Evaluation

- The Throughput:

Throughput measures how many processes complete per unit of time. RR consistently delivered the highest throughput in I/O-bound scenarios because it ensures continuous CPU utilization while processes perform I/O. FCFS performed well in CPU-bound workloads but poorly with I/O-bound tasks due to the convoy effect. EP-RR demonstrated stable throughput, benefiting from priority awareness while avoiding starvation.
Q: How to calculate?

- Throughput: Completed Processes / Total Time

- The Average Waiting Time:

Waiting time varied significantly depending on the workload. FCFS produced reasonable results for CPU-bound tasks but extremely high waiting times for I/O-bound tasks. RR increased waiting time for CPU-heavy processes due to frequent context switching but significantly improved fairness under mixed workloads. EP favored high-priority tasks, resulting in very low waiting times for them and high waiting times for low-priority processes. EP-RR balanced these effects and reduced starvation while still acknowledging priority.
Q: How to calculate?

- Average Waiting Time: Total Ready Queue Time / Number of Processes

- The Average Turnaround Time:

Turnaround time followed similar trends to waiting time. FCFS performed well only when long-running jobs did not block short tasks. RR provided improved turnaround for I/O-bound processes but worsened it for long CPU bursts. EP reduced turnaround time for high-priority tasks, while EP-RR achieved the most consistent turnaround times across diverse workloads.

Q: How to calculate?

- Average Turnaround Time: (Completion Time - Arrival Time) / Number of Processes

- The Average Response Time:

RR demonstrated the best response time because every process received CPU attention within one quantum of arrival. FCFS had poor response time when long processes arrived early. EP gave excellent response time to high-priority processes, but EP-RR offered the best overall experience by combining responsiveness with fairness.

Q: How to calculate?

- Average Response Time: (First Run Time - Arrival Time) / Number of Processes

4. Workload Analysis

- CPU-Bound Workloads:

FCFS delivered exceptional performance for long-running CPU-bound tasks since the lack of context switching minimized overhead. RR performed poorly due to quantum expiration.

EP improved performance when the highest priority was assigned to CPU-bound processes.

EP-RR performed moderately well but introduced additional overhead due to round-robin cycling.

- I/O-Bound Workloads:

In scenarios dominated by I/O operations, RR was the strongest performer. The algorithm ensured that while one process waited for I/O, the CPU remained active with other processes. While FCFS dropped significantly there as a blocking process delayed all others. EP's performance depended heavily on priority configuration, while EP-RR provided more fairness and stable results.

- Mixed Workloads:

EP-RR performed the best under mixed workloads, balancing priority management and fairness.

RR also performed well, especially for I/O-heavy tasks, but EP-RR prevented starvation and ensured that high-priority tasks were completed more quickly. FCFS consistently underperformed in mixed scenarios.

5. Summary

FCFS is ideal for simple batch systems or CPU-bound workloads but unsuitable for interactive or I/O-heavy environments.

RR is best for interactive, time-sharing systems due to low response time and excellent performance with I/O-bound tasks.

EP is highly effective when clear priority levels must be forced but risks starving lower-priority processes.

EP-RR combines the strengths of RR and EP, offering balanced fairness, responsiveness, and priority enforcement. Overall, EP-RR is the most robust general-purpose scheduler. RR excels in I/O-heavy or interactive workloads, while FCFS remains limited to sequential batch-style processing.

This analysis demonstrates how different scheduling algorithms vary in performance depending on workload characteristics and system goals. Choosing an appropriate scheduling strategy requires understanding both the nature of the processes and the objectives of the operating system.

- Summary Table:

Algorithm	Strengths	Weakness	Best use case
FCFS	Simple, great for CPU-bound workloads	Terrible for interactive/I/O-bound	Batch processing, long jobs
RR	Best response time, best throughput for I/O-bound	High overhead, poor for long CPU-bound jobs	Interactive, time-sharing OS
EP	Excellent performance for high-priority tasks	Starvation, unfairness	Real-time where priority matters
EP-RR	Balanced, prevents starvation, respects priority	More complex	Mixed workloads, real systems