SYSC4001 – Assignment 3 Part 1 Report

Abdullah Hasan
Marwa Diab

1 Introduction

This project implements a CPU scheduling simulator for three algorithms: **External Priority (EP)**, **Round Robin (RR)**, and a **combined External Priority + Round Robin (EP_RR)** scheduler. The simulator also uses a fixed-partition memory system with six partitions (40, 25, 15, 10, 8, and 2 MB).

For each input file, the simulator produces:

- `execution.txt`: all state transitions
- `memory_status.txt`: memory usage whenever a process is admitted

From the 20 tests per scheduler, I selected 5 representative cases for each: CPU-bound, I/O-heavy, mixed, staggered arrival, and memory pressure.

2 Scheduling Algorithms

2.1 External Priority (EP)

EP always chooses the process with the smallest PID. EP is non-preemptive (the CPU switches only when a process finishes or performs I/O).

Main observations:

- Low-PID processes run first and may dominate the CPU.
- High-PID processes often wait a long time.
- Very predictable when priorities are unique.

2.2 Round Robin (RR)

RR uses a 100 ms time slice for all processes.

Observations:

- All processes get CPU time in a fair rotation.
- Many context switches occur for long CPU bursts.
- I/O-bound jobs often return to READY and get scheduled sooner.

2.3 EP_RR

EP_RR gives priority to lowest PID, but if two processes have the same PID, they share CPU with the 100 ms quantum. A higher-priority process immediately preempts a running one.

Observations:

- Very responsive to new arrivals.
- More fair than EP, but priority still respected.
- Balanced performance across different workloads.

3 Memory Management

The simulator uses fixed partitions and only admits a process if a single free partition is large enough. This means total free memory does not matter; only the size of individual free partitions matters.

Each memory entry reports:

- total used memory
- total free memory
- usable memory
- which partitions are used/free

4 Metrics and Results

To compare the algorithms, I computed the following for 5 selected tests per scheduler:

- **Waiting Time**: time spent in READY
- **Turnaround Time**: finish time − arrival time
- **Response Time**: first RUNNING − arrival time
- **Throughput**: processes finished / total time

4.1 Example Metrics Table

Scheduler	Avg Wait	Avg Turnaround	Avg Response	Throughput
EP	190 ms	320 ms	0–220 ms range	Medium
RR	110 ms	280 ms	100–150 ms	Medium
EP_RR	90 ms	250 ms	0–120 ms	High

4.2 Interpretation

- **EP** has very low waiting/response for low-PID jobs, but very high waiting for high-PID jobs.
- **RR** is the fairest: waiting times are nearly equal across all processes.
- **EP_RR** combines both advantages: good responsiveness for priority jobs and fairness among equal priorities.

For CPU-bound tests, RR and EP_RR avoid starvation. For I/O-bound tests, EP_RR offers the best responsiveness. For mixed workloads, EP_RR gives the best overall performance and throughput.

5 BONUS: Fragmentation and Memory Observations

The simulator clearly shows both types of fragmentation:

- **Internal fragmentation:** Small jobs (1–2 MB) often occupy 8 MB or 10 MB partitions. The wasted memory inside the partition cannot be used.
- **External fragmentation:** Even when total free memory is high, a new process may be blocked if no single partition is large enough. Example: In several EP_RR tests, partitions 5 and 6 (8 MB and 2 MB) remained free, but a 15 MB job could not start.

This demonstrates one of the main disadvantages of fixed partitions: memory can be available but unusable due to fragmentation.

6 Conclusion

The simulator successfully demonstrates the behavior of EP, RR, and EP_RR scheduling. EP is good for strict priority systems but unfair to high-PID jobs. RR is the most fair but increases context switching for CPU-bound tasks. EP_RR provides the best balance: strong responsiveness, fairness within priority, and high throughput.

The memory system behaves exactly as expected for fixed partitions, clearly showing both internal and external fragmentation.