Nicholas Everekyan, Kevin Gallagher
CS433: Operating Systems
Professor Zhang
October 25th, 2024

Programming Assignment 3 Report

Problem Description:

Our problem involves implementing multiple CPU scheduling algorithms to manage the execution of processes. We were tasked with creating four different scheduling algorithms:

- First-Come, First-Served (FCFS): Schedules processes in the order they request the CPU.
- Shortest-Job-First (SJF): Prioritizes processes based on the length of their next CPU burst.
- Priority Scheduling: Schedules processes based on a predefined priority level, where higher numbers indicate higher priority.
- Round-Robin (RR): Executes processes for a given time quantum in a cyclic manner.
- Priority-Based Round-Robin: A hybrid of Priority Scheduling and Round-Robin, where processes are grouped by priority and handled within their priority level using a round-robin scheme.

The programs read a list of tasks from an input file and calculate the turnaround and waiting times for each process, along with their respective averages. The results of the simulation needed to match the output format provided in the example file and handle various task inputs robustly.

Program Design:

The scheduling algorithms were implemented in C++ with the use of inheritance and polymorphism to extend a base Scheduler class. Each algorithm subclass (e.g., SchedulerFCFS, SchedulerSJF) had to override functions for initialization, simulation, and printing results.

FCFS Implementation:

In the FCFS scheduler, tasks were executed in the order they arrived (assumed at time 0). The turnaround and waiting times were calculated by keeping a running total of CPU time and recording how long each process waited before it was executed.

SJF Implementation:

For the SJF scheduler, tasks were sorted by their burst time, and the one with the shortest burst time was executed next. Since all processes arrive at the same time, the sorting of tasks was based only on their burst lengths. Ensuring that the process selection and calculations for waiting and turnaround times were done in the correct order without reordering tasks incorrectly was the challenge.

Priority Scheduling Implementation:

The Priority Scheduling algorithm assigned a higher priority to processes with a larger priority value. Tasks were selected based on priority, and if two processes shared the same priority, they were scheduled in the order they appeared in the input file. This required maintaining a sorted list of processes, dynamically rearranged as processes finished execution.

Round-Robin (RR) Implementation:

Round-Robin Scheduling uses a time quantum to divide CPU time among processes. Each process would run for the specified time quantum, and then it would be added back to the queue if not finished. This approach ensured that no task monopolized CPU time.

Analysis:

FCFS: With a time complexity of O(n) as tasks are processed in the order they arrive. It is simple, but can lead to poor performance under certain workloads, specifically when short tasks are blocked by longer ones.

SJF: This algorithm, when implemented non-preemptively, required sorting the tasks by burst time, making it slightly more complex with a time complexity of O(n^2). SJF tends to yield lower waiting and turnaround times on average compared to FCFS. SJF consistently outperformed FCFS in terms of both metrics, especially for workloads with a mix of short and long tasks.

Priority Scheduling: The time complexity for the priority queue operations was O(n log n) due to the need to maintain a sorted order. The algorithm efficiently handled scenarios where some tasks needed higher priority, but it risked starving low-priority processes.

Round-Robin (RR): Performance was highly dependent on the chosen time quantum. A smaller quantum led to higher responsiveness but also increased context-switching overhead. In scenarios with a more interactive workload, Round-Robin offered better user experience, although average turnaround and waiting times increased with smaller quanta.

Priority-Based Round-Robin: This hybrid algorithm showed a balance between fairness and priority management. High-priority tasks were executed more promptly, while round-robin with priority levels ensured that no process starved. The overall performance was a compromise between priority scheduling and the responsiveness of round-robin, with a slightly higher overhead due to managing multiple queues.

Timing Results:

Scheduling Algorithm	Avg Turnaround Time	Avg Waiting Time
FCFS	94.375	73.125
SJF	82.5	61.25
Priority Scheduling	96.25	75
Round-Robin (Q = 6)	143.125	121.875
Priority Round-Robin	106.75	85.5

Conclusion:

Our program successfully implements and solves the problem of managing CPU scheduling using various algorithms, including FCFS, SJF, Priority Scheduling, Round-Robin, and Priority-Based Round-Robin. The lessons we learned from this assignment include the importance of selecting appropriate data structures (like priority queues for Priority Scheduling) and properly managing time quantum values in Round-Robin scheduling for optimal performance. One area for improvement in the future could be exploring more scalable solutions and properly learning how to manage a larger set of processes.