CS 3502 Project 2
CPU Scheduling and Deadlock Management

1. Project Overview

Purpose: This project focuses on implementing and analyzing CPU scheduling algorithms to explore how different scheduling strategies affect process efficiency and overall system performance. Two algorithms were implemented in C++: Shortest Remaining Time First (SRTF) and Highest Response Ratio Next (HRRN). These represent both preemptive and non-preemptive approaches.

Implementation Details: The scheduler was developed using standard C++ with no external libraries. The simulation reads a list of processes, then executes either SRTF or HRRN based on user selection. The system tracks waiting time, turnaround time, throughput, and CPU utilization. Gantt chart output is printed to illustrate execution order.

Key Considerations & Challenges: The SRTF algorithm required step-by-step simulation of CPU time, tracking remaining burst times for preemptive execution. HRRN required dynamic response ratio calculations each cycle. To simplify testing, all processes were hard-coded during test runs. Screenshots of the output were submitted as a substitute for a video demonstration due to technical issues.

Execution Output: The system produces accurate performance metrics and a readable Gantt chart for both algorithms. Screenshots of these results were submitted alongside the codebase and report.

2. SRTF Scheduling Implementation

Purpose: The Shortest Remaining Time First (SRTF) algorithm is a preemptive scheduling method that always selects the process with the smallest remaining CPU burst time. It aims to minimize average waiting time by prioritizing shorter tasks and interrupting longer ones when necessary. This implementation serves to demonstrate preemptive CPU scheduling and its effects on system performance.

Implementation Details: The scheduler continuously scans all available (arrived) processes to find the one with the shortest remaining time. If a process with a shorter remaining time arrives while another is running, a preemption occurs. A Gantt chart is generated by

recording process IDs for each unit of simulated time. Metrics like waiting time, turnaround time, throughput, and CPU utilization are calculated after all processes complete.

The algorithm tracks:

- Remaining burst times for each process
- Completion, waiting, and turnaround times
- CPU idle periods
- Total time and process count for throughput and utilization

Key Considerations & Challenges: The primary challenge in SRTF is handling preemption efficiently. This requires simulating CPU execution one time unit at a time and checking all available processes on each cycle. The implementation avoids starvation by assuming all processes are guaranteed to arrive in a bounded time range. CPU idle time is also accounted for when no processes are ready.

Execution Output: A set of test inputs (3 processes with varying arrival and burst times) was used. The system printed a full Gantt chart showing the preemptive execution order and final performance metrics. Screenshots of this output are included in the submission folder as a substitute for a live video.

3. HRRN Scheduling Implementation

Purpose: The Highest Response Ratio Next (HRRN) algorithm is a non-preemptive scheduling method designed to minimize starvation while improving overall efficiency. It selects the process with the highest calculated response ratio, favoring processes that have waited longer relative to their burst times. HRRN balances fairness and efficiency without the complexity of preemptive context switches.

Implementation Details: Each scheduling cycle, the system calculates the response ratio for all available (arrived but incomplete) processes. The response ratio formula is:

$$\text{Response Ratio} = \frac{\text{Waiting Time} + \text{Burst Time}}{\text{Burst Time}}$$

The process with the highest response ratio is selected for execution. Because HRRN is non-preemptive, once a process starts, it runs to completion without interruption. After completion, the scheduler recalculates response ratios and selects the next best candidate. Like SRTF, the implementation outputs a Gantt chart and calculates average waiting time, turnaround time, CPU utilization, and throughput.

Key Considerations & Challenges: Correctly computing the dynamic response ratio each scheduling cycle was a key requirement for HRRN. Care was taken to avoid errors when multiple processes were waiting. HRRN naturally prevents starvation without needing extra mechanisms. As with SRTF, CPU idle time was handled gracefully when no processes were ready.

Execution Output: A test case with three processes was run, and the scheduler produced a Gantt chart illustrating non-preemptive execution. The output metrics aligned with expectations for HRRN behavior. Screenshots of the Gantt chart and results were captured and submitted with the project.

4. Testing and Results

Purpose: Testing was conducted to validate the correctness of both the SRTF and HRRN scheduling implementations. Small process sets with different arrival times and burst times were used to generate predictable and verifiable outcomes.

Implementation Details: The testing process involved manually inputting arrival and burst times for three processes during each run. The system then executed either SRTF or HRRN based on user selection and printed the corresponding Gantt chart and calculated performance metrics.

Test Input Example:

- Process 0: Arrival Time = 0, Burst Time = 5
- Process 1: Arrival Time = 1, Burst Time = 3
- Process 2: Arrival Time = 2, Burst Time = 1

For HRRN testing, slightly longer burst times were used to better observe the effect of aging on response ratios.

Execution Output: Screenshots of the terminal output after running both algorithms are provided as evidence. These include:

- The initial prompts and process input
- The printed Gantt charts
- Average Waiting Time, Average Turnaround Time, CPU Utilization, and Throughput for each test run

Notes on Demonstration: Due to system performance issues and time constraints with other concurrent assignments, a full video demonstration could not be recorded. Instead, detailed screenshots were captured at each major execution stage and submitted to fully document the project results.

5. Analysis and Discussion

Purpose: This section provides a comparative analysis of the two implemented scheduling algorithms based on the collected metrics and observations from testing.

Implementation Details: Both SRTF and HRRN were tested under identical conditions to ensure a fair comparison. Key metrics analyzed included average waiting time, average

turnaround time, CPU utilization, and throughput.

Observations:

- **SRTF Performance:** SRTF consistently produced the lowest average waiting time and turnaround time due to its aggressive preemption of longer processes. However, it required more complex logic to manage time unit-by-unit scheduling and handle CPU idle periods effectively.
- **HRRN Performance:** HRRN provided a good balance between fairness and efficiency. Although it occasionally resulted in slightly higher waiting times compared to SRTF, it naturally prevented starvation and was simpler to implement since it did not require preemptions.
- **CPU Utilization and Throughput:** Both algorithms demonstrated nearly 100% CPU utilization during testing, with throughput varying slightly based on total execution time. Idle periods were minimized effectively in both cases.

Key Takeaways: SRTF is more optimal when pure efficiency (lower waiting times) is the goal, but it introduces additional complexity. HRRN offers a practical alternative that handles a wider variety of workloads fairly without significant overhead. The choice between algorithms depends on system goals: strict efficiency vs. fairness and simplicity.

6. Conclusion

Summary: This project successfully implemented two CPU scheduling algorithms, Shortest Remaining Time First (SRTF) and Highest Response Ratio Next (HRRN), to explore different scheduling strategies and their impacts on system performance. Both implementations demonstrated key scheduling principles and produced accurate performance metrics across multiple tests.

Challenges and Solutions: Handling preemption in SRTF required simulating CPU time unit-by-unit, while HRRN required dynamic response ratio calculations at each decision point. Both challenges were addressed with careful process tracking and iterative testing. Due to technical limitations, a full video demonstration was not provided; however, detailed screenshots were captured to ensure full transparency and documentation of results.

Final Remarks: The project deepened understanding of core operating system concepts related to CPU scheduling, resource management, and performance analysis. It also reinforced the importance of balancing system complexity against fairness and efficiency in scheduler design. All deliverables — source code, documentation, and screenshots — have been submitted to comprehensively showcase the project work.