Introduction

CPU Scheduling Simulator project, featuring Round Robin with quantum numbers 3 and 5, FCFS, and SJF algorithms. This simulation provides a practical exploration of diverse CPU scheduling strategies, offering users a hands-on experience to understand the nuances of task management within a CPU. Explore the impact of quantum numbers on Round Robin scheduling and gain insights into FCFS and SJF algorithm performance. An accessible and educational tool for grasping fundamental concepts in operating systems.

This report is to answer set of questions and it will cover the following questions:

- Software and hardware tools we used.
- Strength and weakness of program
- Why do we prefer to read Gantt chart as an output of a processor schedule?
- If we need to simulate the whole operating system (instead of just the CPU Scheduler)

Software and Hardware Tools Used

We used Eclipse as the primary development environment for this project. The program was tested on an Intel Core i7 Gen 2.8GHz processor, ensuring compatibility and performance on a robust computing platform.

Strengths and Weaknesses:

Strengths:

The program excels in simulating the CPU scheduler, providing a comprehensive understanding of various scheduling algorithms. The chosen development environment and testing on a high-performance processor contribute to its reliability.

Weaknesses:

However, the project does have limitations, notably a constrained memory size of 8192 and a cap of 30 jobs. Future iterations could explore ways to overcome these constraints for more scalable simulations.

Preference for Gantt Chart Output

In terms of visualizing the output of the processor schedule, a Gantt chart is the preferred method. Its simplicity effectively conveys the scheduling details without overwhelming the user. While tables are useful in certain contexts, the Gantt chart aligns with the project's goal of providing a clear and straightforward representation of the scheduling process.

Expanding to Simulate the Whole Operating System

To broaden the scope and simulate the entire operating system, several enhancements are necessary. The program should be modified to incorporate I/O functionality, file management, process synchronization, inter-process communication, and improved security measures. Performance optimization, achieved through code reduction and input validation, would further enhance the overall functionality of the simulated operating system.

Conclusion

The CPU Scheduling Simulator project has laid a strong foundation, offering a practical exploration of CPU scheduling algorithms while paving the way for future improvements and extensions. This endeavor not only deepened our understanding of operating system concepts but also sparked an ongoing curiosity for further exploration and refinement in the realm of system simulation.