

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

# OPERATING SYSTEMS - CS235AI

**REPORT**

# Submitted by

**PALLAVI BG 1RV22CS134**

**SANJANA S 1RV22CS177**

**Computer Science and Engineering 2023-2024**

**INTRODUCTION**

~CPU scheduling in operating systems is a crucial component that manages the allocation of CPU resources among processes.

~The primary goal of CPU scheduling is to maximise CPU utilisation, ensure fairness among processes, and optimise system performance.

~When a process is ready to execute, it is placed in a queue called the ready queue. The operating system uses scheduling algorithms to select the next process from the ready queue for execution.

~Common scheduling algorithms include First-Come, First-Served (FCFS), Shortest Job Next (SJN) or Shortest Job First (SJF), Priority Scheduling, Round Robin (RR), and Multilevel Queue Scheduling.

~The selected process is then dispatched, and its context is loaded into the CPU for execution. Context switching occurs when the CPU switches from executing one process to another, involving saving the current process's state and loading the state of the new process.

~CPU scheduling algorithms are evaluated based on criteria such as CPU utilisation, throughput, turnaround time, waiting time, and response time, with the aim of optimising system performance and resource utilisation.

**PROBLEM STATEMENT**

* In a computer system with multiple processes competing for CPU time, the CPU scheduling algorithm plays a critical role in determining the order in which processes are executed.
* The goal of CPU scheduling is to maximize CPU utilization, ensure fairness among processes, minimize response time, and optimize system throughput.
* CPU scheduling algorithms are evaluated based on criteria such as CPU utilization, throughput, turnaround time, waiting time, and response time, with the aim of optimizing system performance and resource utilization
* However, designing an efficient CPU scheduling algorithm is challenging due to the trade-offs between various scheduling criteria such as throughput, turnaround time, waiting time, and response time.
* Therefore, the problem statement is to develop a CPU scheduling algorithm that balances these criteria to improve system performance and resource utilization.

**APPLICATIONS**

1. **Multiprogramming**: CPU scheduling allows the OS to manage multiple processes concurrently, even on a single-core CPU. By switching between processes quickly, the OS gives the illusion of parallel execution, improving overall system efficiency.
2. **Time Sharing Systems**: In time-sharing systems, CPU scheduling ensures that each user gets a fair share of the CPU time. It allows users to interact with the system simultaneously, providing a responsive and interactive user experience.
3. **Real-time Systems**: In real-time systems, CPU scheduling is critical for meeting strict timing requirements. The scheduler must prioritize tasks based on their deadlines to ensure that time-critical processes are executed on time.
4. **Batch Processing Systems**: In batch processing, CPU scheduling helps in efficient execution of large jobs by interleaving the execution of multiple jobs. The scheduler decides which job to run next based on criteria such as job priority, arrival time, and execution time.
5. **Interactive Systems**: In interactive systems, CPU scheduling aims to provide a responsive user interface by quickly switching between processes. The scheduler needs to balance between providing quick response times and ensuring efficient resource utilization.
6. **Distributed Systems**: In distributed systems, CPU scheduling plays a role in task allocation among multiple nodes. The scheduler decides where to execute a task based on factors like load balancing, network latency, and resource availability.
7. **Power Management**: Modern CPUs have power-saving features that can adjust the CPU frequency and voltage based on the workload. CPU scheduling can influence these decisions by managing the CPU workload to achieve a balance between performance and power consumption.

**METHODOLOGY**

1. Initialization: Set up the initial frame values and counters.
2. Iteration: Iterate through the pages, checking for existing values in the frames.
3. Handling Hits and Faults: Manage page hits and faults based on whether the page is already in the frames or needs replacement
4. Output Display: Conclude the process by displaying the final frame array and counters.
5. This flow ensures a systematic implementation of the FIFO algorithm, visualized through a block diagram or flow chart.

**TOOLS**

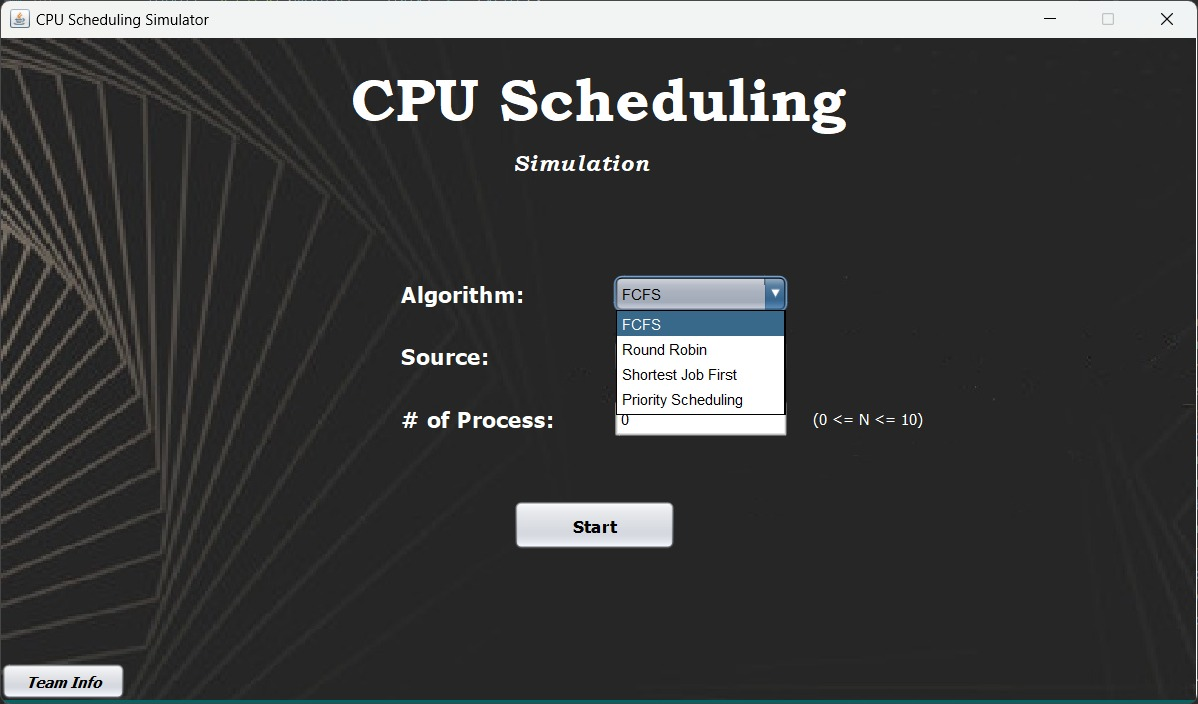
1. **Java Programming Language**: The core language used to write the simulator. Java is a popular, high-level programming language known for its portability, robustness, and ease of use.

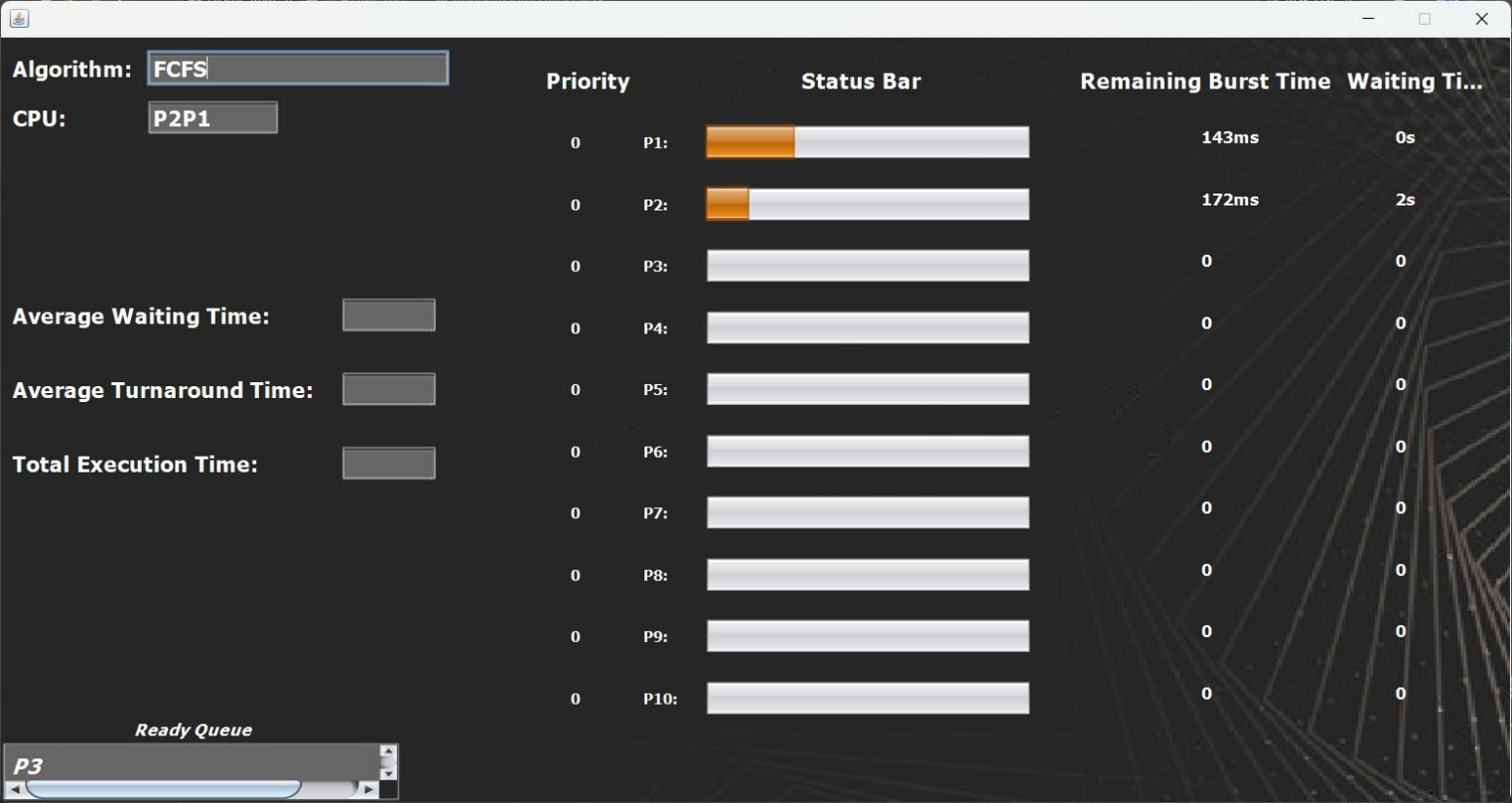
2. **Java Swing (javax.swing):** A set of GUI (Graphical User Interface) components provided by Java for building desktop applications. In this simulator, Java Swing is used to create the GUI components like labels, buttons, and combo boxes.

3. **Integrated Development Environment (IDE):** While not explicitly mentioned in the code, an IDE such as Eclipse, is commonly used for Java development. IDEs provide features like code editing, debugging, and project management, making development more efficient. These are the main tools used in developing the Java CPU Scheduling Simulator. Additionally, standard Java libraries and utilities are also utilized, such as Arrays and Comparator for sorting processes, and ActionListener for handling GUI events.

4. **JDK:** JDK stands for Java Development Kit. It's a software development kit used by Java developers to create Java applications. The JDK includes tools for compiling, debugging, and running Java code, as well as libraries and documentation. It's essential for anyone who wants to develop Java applications.

**OUTPUT**

****



**CONCLUSION**

CPU Scheduling is vital since it enhances the following functions:

1. Resource Utilization: Efficient CPU scheduling ensures that the CPU is utilized to its fullest potential, minimizing idle time and maximizing throughput.
2. Response Time: Effective scheduling algorithms reduce the response time for interactive tasks, improving user experience and system responsiveness.
3. Fairness: Fair CPU scheduling ensures that all processes get a fair share of CPU time, preventing starvation and ensuring equitable resource allocation.
4. Priority Management: Scheduling allows for prioritizing critical tasks over non- critical ones, ensuring that important processes are executed in a timely manner.
5. Throughput :By efficiently scheduling processes, the system can achieve higher throughput, processing more tasks in a given time period.
6. Predictability :Scheduling algorithms provide predictability in task execution, which is crucial for real-time systems and applications with strict timing requirements.
7. Resource Management: CPU scheduling plays a vital role in managing system resources, coordinating CPU access among multiple processes and preventing resource contention.
8. Multiprogramming: In a multiprogramming environment, scheduling ensures that multiple processes can run concurrently on a single CPU, maximizing system efficiency.
9. Adaptability: Modern scheduling algorithms can adapt to varying workloads and system conditions, dynamically adjusting scheduling parameters to optimize performance.
10. Overall System Performance: Effective CPU scheduling contributes significantly to the overall performance and stability of the operating system, influencing factors such as throughput, latency, and resource utilization.