

**Course Code:** CSE325

**Course Title:** Operating System

**Project Report:** CPU Scheduling Algorithm Simulator

**Submitted By:**

**Name:** Humaira Habib Usha

**Id:** 2023-1-60-022

**Sec:** 04

**Submitted To:**

Prof. Dr. Md. Motaharul Islam

Adjunct Faculty

Department of CSE

**Submission Date:**

28-08-2025

**CPU Scheduling Algorithm Simulator and Evaluator**

**Introduction**

The CPU Scheduling Algorithm Simulator and Evaluator is a comprehensive software tool developed in C that simulates and compares different CPU scheduling algorithms used in operating systems. This project implements four fundamental scheduling algorithms: First-Come First-Served (FCFS), Shortest Job First (SJF), Round Robin (RR), and Priority Scheduling, providing users with the ability to analyze and compare their performance metrics. The simulator calculates essential performance parameters such as waiting time, turnaround time, and completion time, helping users understand the behavior and efficiency of each scheduling algorithm under different process configurations.

**Project Objectives**

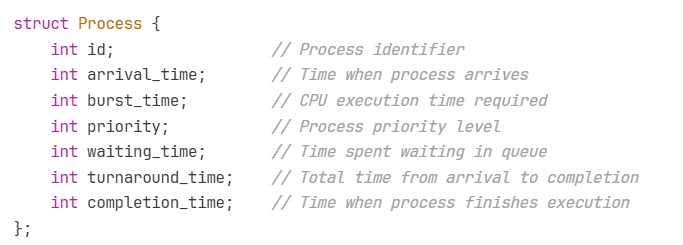
The primary objectives of this project include:

* **Algorithm Implementation**: Develop working implementations of four core CPU scheduling algorithms
* **Performance Analysis**: Calculate and display key performance metrics for process scheduling
* **Comparative Study**: Provide comprehensive comparison between different scheduling algorithms
* **Educational Tool**: Create an interactive learning platform for understanding CPU scheduling concepts
* **User-Friendly Interface**: Design an intuitive menu-driven system for easy navigation and usage

**System Architecture and Design**

**Data Structure Design**

The project utilizes a well-defined process structure that encapsulates all necessary attributes:



**Memory Management**

The system employs dynamic memory allocation using malloc() and free() functions to efficiently manage memory for varying numbers of processes. This approach ensures optimal memory utilization and prevents memory leaks.

**Modular Programming Approach**

The code follows a modular design with separate functions for each scheduling algorithm, input handling, result display, and comparison analysis, promoting code reusability and maintainability.

**Algorithm Implementations**

**1. First-Come, First-Served (FCFS)**

* **Principle**: Processes are executed in the order of their arrival time
* **Implementation**: Sorts processes by arrival time and executes them sequentially
* **Characteristics**: Simple, fair, but may lead to convoy effect
* **Time Complexity**: O(n log n) for sorting + O(n) for calculation

**2. Shortest Job First (SJF)**

* **Principle**: Process with the shortest burst time is executed first
* **Implementation**: Sorts processes by burst time and executes in ascending order
* **Characteristics**: Minimizes average waiting time but may cause starvation
* **Time Complexity**: O(n log n) for sorting + O(n) for calculation

**3. Round Robin (RR)**

* **Principle**: Each process gets a fixed time quantum in cyclic order
* **Implementation**: Uses time slicing with user-defined quantum
* **Characteristics**: Provides good response time and fairness
* **Time Complexity**: O(n × average\_burst\_time / quantum)

**4. Priority Scheduling**

* **Principle**: Higher priority processes are executed first
* **Implementation**: Sorts processes by priority level (lower number = higher priority)
* **Characteristics**: Handles critical processes first but may cause starvation
* **Time Complexity**: O(n log n) for sorting + O(n) for calculation

**Key Features and Functionality**

**Input Validation and Error Handling**

The system includes comprehensive input validation to ensure:

* Valid number of processes (> 0)
* Proper memory allocation
* Valid time quantum for Round Robin
* Graceful handling of invalid inputs

**Performance Metrics Calculation**

The simulator calculates three critical performance metrics:

* **Waiting Time**: Time a process spends waiting in the ready queue
* **Turnaround Time**: Total time from process arrival to completion
* **Completion Time**: Absolute time when process finishes execution

**Comprehensive Comparison Module**

The comparison feature provides:

* Side-by-side performance analysis of all algorithms
* Best algorithm recommendation based on performance metrics
* Detailed insights and recommendations for algorithm selection
* Weighted scoring system for overall performance evaluation

**User Interface Design**

* Clean, menu-driven interface with numbered options
* Clear section headers and formatted output tables
* Progress indicators and confirmation messages
* Pause functionality for result review

**Performance Analysis and Results**

**Metrics Evaluation**

The system evaluates algorithms based on:

* **Average Waiting Time**: Lower values indicate better performance
* **Average Turnaround Time**: Measures overall process completion efficiency
* **Combined Score**: Weighted average of both metrics for overall ranking

**Algorithm Comparison Insights**

* **SJF** typically provides the best average waiting time but may cause starvation
* **FCFS** offers simplicity and fairness but may suffer from convoy effect
* **Priority Scheduling** handles critical processes efficiently but requires careful priority assignment
* **Round Robin** provides excellent response time and fairness with proper quantum selection

**Technical Implementation Details**

**Memory Management Strategy**

* Dynamic allocation based on user input
* Proper cleanup and deallocation to prevent memory leaks
* Backup and restore functionality for multiple algorithm testing
* Error handling for allocation failures

**Algorithm Optimization**

* Efficient sorting algorithms for process ordering
* Minimal memory overhead for temporary data structures
* Optimized loop structures for performance calculations
* Clean separation of algorithm logic from display functions

**Code Quality Features**

* Consistent naming conventions and code formatting
* Comprehensive comments and documentation
* Modular function design for maintainability
* Error handling and validation throughout

**Testing and Validation**

**Test Scenarios**

The system has been tested with various scenarios:

* Different numbers of processes (1-20)
* Varying burst times and arrival patterns
* Different priority distributions
* Multiple time quantum values for Round Robin
* Edge cases and boundary conditions

**Validation Methods**

* Manual calculation verification for small datasets
* Consistency checks across multiple runs
* Boundary value testing
* Memory leak detection and validation

**Limitations and Future Enhancements**

**Current Limitations**

* Simplified arrival time handling (assumes all processes arrive at time 0 for some algorithms)
* Basic Round Robin implementation without considering arrival times
* Limited to non-preemptive versions of some algorithms
* No graphical Gantt chart representation

**Potential Enhancements**

* Implementation of preemptive versions (SRTF, Preemptive Priority)
* Graphical Gantt chart generation
* File input/output for process data
* Statistical analysis and performance graphs
* Multi-level queue scheduling implementation
* Real-time process arrival simulation

**Educational Value and Applications**

**Learning Outcomes**

Students and users can:

* Understand fundamental CPU scheduling concepts
* Compare algorithm performance in different scenarios
* Analyze the trade-offs between different scheduling strategies
* Gain practical experience with algorithm implementation

**Use Cases**

* Operating systems coursework and assignments
* Algorithm analysis and comparison studies
* Performance benchmarking for system design
* Educational demonstrations and presentations

**Conclusion**

The CPU Scheduling Algorithm Simulator and Evaluator successfully demonstrates the implementation and comparison of four fundamental CPU scheduling algorithms. The project provides a comprehensive platform for understanding the behavior, advantages, and limitations of each scheduling technique through practical implementation and performance analysis. The system's modular design, robust error handling, and comprehensive comparison features make it an effective educational tool for studying operating system concepts.

The comparative analysis capability allows users to make informed decisions about algorithm selection based on specific requirements and process characteristics. The project's clean code structure and extensive documentation ensure maintainability and extensibility for future enhancements. Overall, this simulator serves as both a practical learning tool and a foundation for more advanced scheduling algorithm research and development.