

**Jaypee University of Information Technology**

**Department of Computer Science and Engineering**

**REPORT**

**Course Code: 18B17CI471**

**Course Name: OPERATING SYSTEM LAB**

Submitted by: Submitted to:

Name: Pranjal Bansal Name: Dr. MONIKA BHARTI

Roll no:  211449

Batch: CS 48

Submitted by:

Name: Tushar Bhardwaj

Roll no:  211457

Batch: CS 48

**Problem Statement:**

The objective of this project is to implement various CPU scheduling algorithms and evaluate their performance in a simulated operating system environment. The project will focus on implementing the following scheduling algorithms in C programming language:

1. First-Come, First-Serve (FCFS)

2. Shortest Job First with preemption (SJF-P)

3. Shortest Job First without preemption (SJF-NP)

4. Round Robin (RR)

5. Priority Scheduling

The project aims to compare the performance of each scheduling algorithm and analyze their strengths and weaknesses. The project will also evaluate the impact of different parameters such as the quantum time in RR, context switching time, and preemption time on the system's performance.

**Project** **Description**:

The objective of this project is to implement five different scheduling algorithms in a simulated operating system environment using the C programming language. The scheduling algorithms are First Come First Serve (FCFS), Shortest Job First (SJF) with preemption, Shortest Job First (SJF) without preemption, Round Robin (RR), and Priority scheduling.

The project will simulate a CPU scheduling algorithm in a multitasking environment with multiple processes running on the CPU. The simulated environment will consist of a set of processes with different arrival times, burst times, and priorities. The project will implement a scheduling algorithm that assigns a priority to each process based on its attributes, and the CPU will execute the process with the highest priority.

The project will have the following main components:

1. **Process Generation Module:** This module will generate a set of processes with random arrival time, burst time, and priority. The module will generate a specific number of processes based on the user's input.

2. **Scheduling Module:** This module will implement the five scheduling algorithms. The module will determine which process should be executed next based on the scheduling algorithm used. The module will also calculate the waiting time, turnaround time, and response time for each process.

3. **Execution Module:** This module will simulate the execution of the processes on the CPU. It will keep track of the time each process spends on the CPU and the waiting time.

4. **Results Module:** This module will display the results of the simulation, including the average waiting time, average turnaround time, and the order in which the processes were executed.

The project will test and compare the performance of the five scheduling algorithms. The performance of the algorithms will be evaluated based on the average waiting time and the average turnaround time of the processes.

The project will help the user understand the CPU scheduling algorithms and their impact on the performance of the system. The project will also demonstrate the importance of selecting an appropriate scheduling algorithm based on the system's requirements.

The project will include the following expansions for each scheduling algorithm:

1. **First Come First Serve (FCFS**): This algorithm is the simplest scheduling algorithm that executes the processes in the order they arrive. In this expansion, the project will demonstrate how the FCFS algorithm can result in long waiting times for processes with longer burst times.

2. **Shortest Job First (SJF) with preemption**: This algorithm selects the process with the shortest burst time to execute next. In this expansion, the project will demonstrate how preemption can improve the performance of the SJF algorithm by allowing a shorter process to execute before a longer one.

3. **Shortest Job First (SJF) without preemption**: This algorithm selects the process with the shortest burst time to execute next but does not allow preemption. In this expansion, the project will demonstrate how the SJF without preemption algorithm can result in longer waiting times for longer processes.

4. **Round Robin (RR):** This algorithm assigns a fixed time slice or quantum to each process and switches to the next process after the quantum expires. In this expansion, the project will demonstrate how the quantum size can affect the performance of the RR algorithm and how a larger quantum can lead to longer waiting times.

5. **Priority Scheduling**: This algorithm assigns a priority to each process, with higher priority processes being executed first. In this expansion, the project will demonstrate how assigning priorities can impact the performance of the system and how different priority levels can affect the scheduling of processes.

Overall, this project will provide the user with a hands-on experience of implementing various scheduling algorithms in a simulated operating system environment. The project will help the user understand the strengths and weaknesses of each algorithm and how to select an appropriate algorithm based on the system's requirements. The project will also help the user develop their skills in C programming, data structures, and algorithm implementation.

**METHODOLOGY:**

To implement the project in which FCFS, SJF with preemption, SJF without preemption, Round Robin, and Priority Scheduling are performed in C programming language, the following methods can be used:

**Implementing the Scheduling Algorithms:**

Once the data structures are implemented, the next step is to implement the various scheduling algorithms. Each algorithm will take input from the data structures and process the data according to its algorithm. For example, the FCFS algorithm will process the processes in the order they arrive, while the SJF algorithm will process the shortest jobs first. The Round Robin algorithm will process the processes for a fixed time slice and then switch to the next process. The Priority Scheduling algorithm will process the processes according to their priority.

**Testing and Debugging:**

Testing is an essential part of software development. The system should be tested for functionality, stability, and performance. The system should be tested with various test cases and inputs to ensure that it is working correctly. Any bugs or errors should be identified and fixed before the system is deployed.

**Performance Analysis:**

Once the system is tested and verified, the next step is to analyze its performance. The performance of the various scheduling algorithms should be evaluated using metrics such as average waiting time, turnaround time, and response time. The performance of each algorithm should be compared, and the impact of different parameters such as quantum time, context switching time, and preemption time should be evaluated.

**CONCLUSION:**

In conclusion, this project demonstrates the implementation of five different process scheduling algorithms in C: First-Come-First-Serve (FCFS), Shortest Job First (SJF) with and without preemption, Priority Scheduling, and Round Robin. The goal of this project is to simulate the performance of these scheduling algorithms.

The results of the simulation show that each algorithm has its advantages and disadvantages depending on the characteristics of the input processes. For example, FCFS is simple and fair, but it can lead to long waiting times for processes with longer burst times. SJF without preemption can minimize waiting times but may cause starvation for longer processes. SJF with preemption and Priority Scheduling can handle dynamic priority processes and balance resource allocation but may cause overhead due to context switching. Round Robin can ensure fairness and response time for processes but can cause a higher average waiting time for processes.

Overall, the project provides a practical understanding of process scheduling algorithms and their impact on system performance. It also emphasizes the importance of choosing the right scheduling algorithm based on the requirements and characteristics of the system and the processes running on it.