Operating System

CPU Schedular

CSC-227

Team Members

|  |  |
| --- | --- |
| Abdulmajid Alromaih | 442101425 |
| Raed Alsaedi | **438106740** |
| Abdullah Assab | **441105054** |

Due Date:

27/11/2023

Table of Contents

[Software Tools 3](#_Toc152012122)

[Hardware Tools 3](#_Toc152012123)

[Strength of our program 3](#_Toc152012124)

[Weakness of our Program 3](#_Toc152012125)

[How do we prefer to read the output ? 3](#_Toc152012126)

[How would we simulate the whole operating system ? 4](#_Toc152012127)

[Test Run Image 4](#_Toc152012128)

# Software Tools

We used Eclipse text editor to write the program.

# Hardware Tools

The only hardware devices that were used in the process of completing the project were our own personal laptops

# Strength of our program

Modularity: The program is organized into distinct functions and classes, making it modular and easy to understand. Each scheduling algorithm is implemented in its own method, promoting code readability and maintainability.

Thread Usage: The program effectively utilizes threads for concurrent execution of file reading, job loading, and scheduling. This allows for efficient handling of multiple tasks concurrently, enhancing overall performance.

Dynamic Memory Management: The program checks and ensures that jobs are loaded into the ready queue only if there is enough available memory. This dynamic memory management is crucial for simulating realistic operating system behavior.

# Weakness of our Program

Code Redundancy:

There might be areas of redundancy or duplicated code across different scheduling algorithms.

# How do we prefer to read the output ?

We did not use a Gantt chart. We went with a clear simple text output that was easy to read. As for what we prefer, we don’t really have a preference on how to read the output of the process schedule as long as it is clear and easy to read

# How would we simulate the whole operating system ?

To turn the program into a whole operating system simulation, we need to add more features. Make the Process Control Block (PCB) hold extra info about processes, like priority and counters. Also, we need to make memory management handle memory changes and introduce a basic file system for file operations. We also need to add functions for input/output tasks and create a simple way for users to interact with the system. We also need to provide ways for processes to communicate, improve how errors are handled, and think about basic security. We need to make sure to manage time-related stuff like clocks and timers. Lastly, we assume that we need to split the system into two parts: the kernel (for critical tasks) and user space (for running applications). We understand that these changes require careful planning and rearranging to create a complete and working operating system simulation but this is what we as group assume the process would look like in a vague manner.

# Test Run Image