**Programming Assignment 3 Report**

**Names: Andrew Ortiz and Mitchell Curtis**

# Problem Description

For this programming assignment we are implementing scheduling algorithms for processes within an operating system. These algorithms include: FCFS (First Come First Served), SJF (Shortest Job First), RR (Round Robin), and a Priority scheduling algorithm that does shortest job first based on each priority level.

# Program Design

Our program consists of two main parts for each scheduling algorithm, the first being the driver.cpp files that take input from a .txt file containing various processes and contains the main function. The second main part is the actual scheduling algorithm that both simulates and prints the results for easy viewing. Both the driver and scheduling portions use the PCB (process control block) class that stores the process id, name, priority, burst time, and arrival time. A black screen with blue text

Description automatically generated

# System Implementation

The logic implementation varies between scheduling algorithms, but the basics stay the same, the driver.cpp loads the processes.txt into a vector of PCB elements to be used in the scheduler.cpp file. This file (regardless of scheduler method) does some basic things, one being an initialization that can include copying the vector of PCB elements so that the original doesn’t have to be modified. A black background with white text

Description automatically generated

The next step is to simulate the scheduling method, this logic varies between methods, but the basics include which process should run when, and recording waiting/turnaround time. For methods like RR (round robin) this gets more complicated as you have to track the remaining burst times for each process and which process(es) have completed.

A screen shot of a computer program

Description automatically generated

The final important system implementation is the print\_results() function that prints the processes through the simulation as well as the average waiting and turnaround time once the simulation portion is completed.

A computer screen shot of a program code

Description automatically generated

**Results**

A screenshot of a computer program

Description automatically generated

# Conclusion

Different scheduling methods have different purposes and are useful for different things. For systems where response time isn’t uber-important, methods like FCFS could be preferred as it tends to have less overhead processing. Consumer desktop operating systems like Windows/MacOS might chose round robin scheduling as it allows many processes to make progress together, which could increase the users perceived response time. For systems where the raw number of processes completed (total throughput) over a given timeframe shortest job first might make most sense. Choosing the correct scheduling method requires knowledge of what’s the expected workload and output for a given system, sometimes this isn’t concrete knowledge and compromises have to be made so that important processes don’t get stalled for less-important processes.