Assignment 1 Report

CS433 Operating Systems

Micah McCarver and Brandon Holman

9/18/17

**Assignment 1 Submitted Files:**

Main.cpp

Main.o

PCB.cpp

PCB.o

PCB.h

Executable

Makefile

Report

**Instructions**: Source code can be compiled and run in a SSH such as Putty and used with the compiler g++. Executable and makefile can be run using the ./ command or make command respectively. The report is in .docx format and can be opened by a word processor.

**Introduction:** The objective of this assignment was to create a Process Control Block (PCB). A process in this assignment can be defined as having one of five states (NEW, RUNNING, READY, WAITING, or TERMINATED), an ID, an a priority number. The highest priority is established by the lowest priority number, where priority number 1 is defined as the highest priority. There are two tests that make us the assignment. The first test is where predetermined processes are added and then removed individually. The second test randomly, where each option has a 50% chance, decides to remove or add an item. The removed item is determined by highest priority and the priority is decided using the random function.

**Design**: The data structure utilized in this assignment was an array. The space complexity is O(n). The time complexity is Θ(n) for search, deletion, and insertion. It was decided by our group to use an array due to it’s reliability and orthogonality. Arrays, as a data structure, provide speed when iterating through all elements (such as when searching for the highest priority) and take up less memory than other data structures such as linked lists. Features implemented include a search function, removal function, output function, and counting function. These function utilize for loops to go through the array and simple return statements to return values.

**Results**: After being compiled and run, the program accurately outputs the specified outputs/processes as well as the execution time. The program first shows the ReadyQueue and displays it after every removal. For the second test, the program accurately removes or adds entries at random. It runs 1,000,000 iterations and displays the final contents of the ReadyQueue. The program also outputs to the total execution time after the program has completed.

**Conclusion:** Future improvements that could be added would be using a singly or doubly linked list. Although reliability was considered more important by our group than speed, it could be important to optimize the program for speed. An important lesson learned was how to utilize certain loops and algorithms to sort through data. Another valuable lesson learned was establishing the benefits and negatives between an array or a linked list. These decisions are crucial to the program and choosing the best option for the requirements affects the whole program.