**EECS 2011 Assignment 2**

**System for Deterministic Modeling of CPU Scheduling Algorithms**

**Date:** November 8, 2017

**Course:** EECS 2011

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Name** | **Student ID** | **EECS ID** |
| 1 | Haseeb Ahmad | 213579891 | ahmadhas |
| 2 | Jason Skinner | 215115678 | skinner1 |

**Report**

CPU scheduling is nothing but picking up the process from the ready state and sending it to the CPU. The short-term scheduler and the dispatcher (sometimes part of the scheduler) do this task from ready state to running state. The process can move from running to termination (process completed), or to wait state or ready state (ready for the new process). Now there are many ways to assess the situation of the process. For example, if the process the priority queue algorithm is used, then we set the priority of the process and send to queue. Whichever has the higher priority, it’s state change from to ready to be running or waiting to running (if the process has already been started.)

For this assignment, first we needed to create processes and define their states. We made a processes class, which defines what each process has and the states are defined using ENUM. The Test class creates an ArrayList of the processes to test for the algorithms.

The First-come, First-serve algorithm goes through all the processes until all the processes are done. It will print the wait time and start time and increment time and process time for each process CPU Time. The process gets removed as it outputs the data of the process.

The Shortest-Job First gets the index of the process which has the shortest time, then it will go through all the processes and checks if the process time is less than the time of the shortest process time, then increment the time and process time, it will at the end remove that process and again gets the index of the process which has the shortest time.

In the Priority Scheduling, we first had to create a queue, in which has a process is added if the priority of that process is greater than all other, also we had to check for whether it is done and whether it is CPU or IO process as well.

Round-Robin Scheduling goes through all the processes and whatever the process is it will execute that process for a particular amount of time and after the execution, it will pre-empt it and onto the next process.

Multi-Level Scheduling combines all the above scheduling and gives the priority to the highest-level process. For example, the highest-level process is System Process (using FCFS) and then Intermediate (SJF) and then Round Robin and finally lowest-level process is using priority. In this case System process will be executed first and then the next and so on. The advantage in this is that for various types of processes, we can apply various types of scheduling.

Multi-Level Feedback Scheduling is very similar to Multi-Level Scheduling, but the difference is that the process at the lowest-level can be moved to a higher priority and will not be executing for a long time. The implementation of it can be done by checking which process has important and which is not and then move the priority of that process and which algorithm to use.

Testing was done with random processes of random lengths. The processes were created once the Algorithm class was initiated. Each time the algorithm run a new set of processes are created and analysed. These processes were tested and results were compared by calculations by hand.