Assignment 4 – Simulation of CPU Scheduling Algorithms (10%)

Introduction

Assignment 4 contributes 10% towards your final course grade and the assignment total is 14 marks. You should begin Assignment 4 in Module Eight; it is due at the end of Module Nine. Check your Course Schedule for the precise due date. Directions for submitting Assignment 4 to your Open Learning Faculty Member for grading can be found in the "Assignment and Project Instructions" document located on your Homepage. An assignment marking criteria and your assignment submission details follows at the end of this document.

In this assignment, you will simulate various CPU scheduling algorithms. The efficiency of Operating System depends on two main factors:

- Algorithms used for scheduling; and
- The environment in which this algorithms are implemented.

One of the methods used in evaluation of CPU scheduling algorithms is simulation on historical logs or samples generated by a random number generator with certain parameters of processes in a given environment. This is more popular, and random number generators generate samples using queuing theory and probability theory.

Instructions

Design and implement a simulation program that will allow evaluation of following CPU scheduling algorithms:

- FCFS without preemption
- SJF without preemption
- Priority **with** preemption
- RR without preemption.

This program should include the above four algorithms, OR you can use one program for each scheduling algorithm.

Input data for simulation could be obtained by using SampleGenerator.java. Use the seven Java files in the Assignment 4 Sample Process Generator Java File Folder located on your Home Page. SampleGenerator asks to enter:

- Minimum priority (with 0 as the highest)
- Maximum CPU burst time in ms
- Average process arrival interval in ms
- Total simulation time in ms, and
- Output file name

It will generate samples (records of **process number**, **arrival time**, **priority**, and **CPU burst time**) and save them in the output file. SampleReader.java could be used to read samples, and SampleReaderTest.java contains sample codes how to read the samples from the sample file generated by SampleGenerator.

You need to generate samples for at least 3600000 ms (1 hr).

Your simulation program will generate the followings at every minute for in total 4 different algorithms:

- Average waiting time for the past 10 minutes
- Average turnaround time for the past 10 minutes

Algorithm could be:

```
    // Ready queue is a sort of priority queue: arrival time for FCFS, priority for
Priority, CPU burst time for SJF
    //     and the entering time into the queue for RR
    For (current = 0; current < SIM_TIME; current++) {
        If there is no sample process
            Read a sample process from the sample file;
        If the arrival time of the sample process == current
            Put the sample process into the priority queue;
        If there is a running process
            If the terminating time of the process == current
                Remove the process from the CPU
            Else
                If RR && the timer expires          // RR scheduling
                    Remove the process from the CPU
                    Put the process into the priority queue
                If Preemptive                       // Preemptive scheduling
                    Get a sample process from the priority queue
                    Compare the priority
                    If the priority of the running process on the CPU is lower
                        Remove the process from the CPU
                        Put the process into the priority queue
                        Dispatch the new sample process to the CPU
                    Else
                        Put the sample process back into the priority queue
        Else
            If the priority queue is not empty
                Get a sample process from the priority queue;
                Dispatch the process to the CPU;
    }
```

Again, the following Java files can be found in the Assignment 4 Sample Process Generator Java File Folder folder on the HomePage in your course management system:

- SampleGenerator.java
- SampleReader.java
- PriorityQueue.java
- QueueItem.java
- ProcessControlBlock.java – you can extend this class to keep some information for your simulation
- KeyboardIn.java
- SampleReaderTest.java

```
public class SampleReaderTest
{
    public static void main(String args[])
    {
            PriorityQueue readyQ = new PriorityQueue();
            ProcessControlBlock pcb;
            SampleReader sr;
            String sampleFile;
            int process, arrival, priority, burst;

            System.out.print("Sample File?  ");
            sampleFile = KeyboardIn.readLineWord();
            sr = new SampleReader(sampleFile);

            while(true) {

                    process = -1; arrival = -1; priority = -1; burst = -1;

                    process = sr.readProcess(); if (process < 0) break;
                    arrival = sr.readArrival(); if (arrival < 0) break;
                    priority = sr.readPriority(); if (priority < 0) break;
                    burst = sr.readBurst(); if (burst < 0) break;
```

```
                        pcb = new ProcessControlBlock(process, arrival, priority,
burst);
                        readyQ.putQueue(pcb, arrival);
                }

                while(!readyQ.isEmpty()) {
                        System.out.println(readyQ.getHighestPriority());
                        pcb = (ProcessControlBlock)readyQ.getQueue();
                        System.out.println(pcb.getProcessNo() + "  " +
pcb.getArrivalTime()
                                + "  " + pcb.getPriority() + "  " +
                            pcb.getCpuBurstTime());
                }
        }
}
```

Report Submission Details

You need to submit a report that consists of:

- Short description and analysis of the simulation result
- Java source files
- Two comparison graphs: one for the average waiting time and the other for the average turnaround time
- Screen shots that show how your programs work.

Assignment Marking Criteria	Weighting
Comprehensive, insightful written analysis of simulation result	/4
Two accurate and properly labeled comparison graphs – one for average waiting time; one for average turnaround time (1 mark for each graph)	/2
Four algorithms that are developed and implemented into program correctly	/4
No syntax error: All requirements are fully implemented without syntax errors. Submitted screen shots will be reviewed with source code.	/4
Total	/14