بسم الله الرحمن الرحيم



**Operating Systems Project 1 Report**

**Instructor:** Dr. Adnan Yahya

**Name:** Mohammed Sehwail, **ID**: 1131253

**Name:** Hussein Dahir , **ID**: 1131138

**Abstract:**

In this project, we implemented a simulator for some CPU scheduling algorithms, which were, SJF, SRTF, RR and RMS, since our task number was (1131138 + 1131253) % 10 = 1. As a bonus, we designed a very simple and friendly use interface, and we also implemented an extra scheduling algorithm, which is FCFS.

**Theory:**

**SJF:**

This algorithm associates with each process the length of the process’s next CPU burst. When the CPU is available, it is assigned to the process that has the smallest next CPU burst. If the next CPU bursts of two processes are the same, FCFS scheduling is used to break the tie.

**SRTF:**

A preemptive version of SJF algorithm, which preempts the currently executing process when a less remaining time process becomes available.

**RR:**

It is similar to FCFS scheduling, but preemption is added to enable the system to switch between processes. A small unit of time, called a time quantum or time slice, is defined. The ready queue is treated as a circular queue. The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of up to 1 time quantum.

**RMS:**

The rate-monotonic scheduling algorithm schedules periodic tasks using a static priority policy with preemption. If a lower-priority process is running and a higher-priority process becomes available to run, it will preempt the

lower-priority process. Upon entering the system, each periodic task is assigned a priority inversely based on its period. The shorter the period, the higher the priority; the longer the period, the lower the priority. The rationale behind this policy is to assign a higher priority to tasks that require the CPU more often.

Furthermore, rate-monotonic scheduling assumes that the processing time of a periodic process is the same for each CPU burst. That is, every time a process acquires the CPU, the duration of its CPU burst is the same.

**FCFS:**

The simplest CPU-scheduling algorithm is the first-come, first-served (FCFS) scheduling algorithm. With this scheme, the process that requests the CPU first is allocated the CPU first. The implementation of the FCFS policy is easily managed with a FIFO queue.

**Program Implementation:**

In our implementation, we used java language. We made 3 java classes only, Interface Class, Process Class and Scheduler Class. Interface Class was for the GUI only, Process class was consisting of the process attributes (burst time, arrival time, start time, finish time, etc…), processes were java objects from this class. Scheduler Class was our main program which consists of all algorithms codes and its functions.

**How to Run Our Program:**

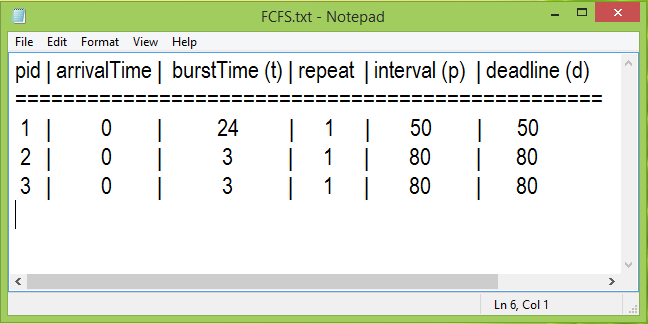
We attached an executable jar file for our program, so that you can run it with no need for an IDE, and we also attached the code and a soft copy of this report.

Our interface is very easy to use, as shown in the following screenshot, you just need to browse to find the input file, and then choose the algorithm, and click “Run”.



**Input File Structure:**

The structure of the input file for our program should be like the following screenshot:



First two lines will be ignored (but they are necessary for our code, since we always skip the first two lines in any input file), they are just to make it easier to enter and read data in the file. The “|” separator is necessary between the numbers in each line (but the spaces are not, you can enter data without spaces). Book examples for all the algorithms we have implemented are created in the correct input file structure and attached with the code.

**Conclusion:**

From this project, one can see and test different scheduling algorithms, compare between them, and so know what is better for a specific purpose, since some algorithms give better average turnaround time and others give better average waiting time (e.g. SJF scheduling algorithm). We also noticed that FCFS is good with long processes (e.g. batch processes). Simulating Algorithm is good, since we can see their efficiency before applying them in real, and it was very nice to have a project train us to do something like this.

**References:**

* Operating System Concepts, 9th Edition.
* <https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/5_CPU_Scheduling.html>
* <https://www.researchgate.net/publication/249645533_A_Comparative_Study_of_CPU_Scheduling_Algorithms>