ITF22519: Introduction to Operating Systems

Fall Semester, 2021

Lab11: Process Scheduling

Submission Deadline: November 11st, 2021 23:59

This lab is mandatory!

In this lab, you will implement two simple CPU scheduling algorithms named **first-come first-served (FCFS)** and **shortest-job-first (SJF)**. These two algorithms are used by OS to decide which process will be run next.

Before you start, remember to `commit` and `push` your previous lab to your git repository. Then, try to `pull` the new lab to have all necessary materials:

```
$ cd Introduction2OS/labs
$ git pull upstream main
$ cd lab11
```

1 Report

Include the followings in the report of your lab assignment:

- GitHub: <yourGitHubAccount> on the top of your report. Otherwise, you will not be graded.
- Source code of two functions you implemented.
- The output of one run with Testcase provided under lab11 repository.

2 Assignment Description

First, please take a look around the provided code in the file ***scheduling.c***. Some parts of the entire program. However, it is good to look through it and understand how the different parts work.

2.1 Process Structure

The file ***scheduling.c*** contains the main function that will call two scheduling functions implemented by you with the same array of process structures. The structure is defined in the scheduling.h file as follow:

```
struct process{
/* Values initialized for each process */
     int id;
     int arrivaltime;  /* Time process arrives and wishes to start */
     int runtime;      /* Time process requires to complete job */
```

```
/* Values algorithm may use to track processes */
     int starttime;
     int endtime;
};
```

The first three values (id, arrivaltime, and runtime) are the ID of a process, the time that the process enters the system, and time needed by the process to complete its job, respectively. They are the inputs to the scheduling algorithms and can be read from *Testcase.txt* file. These values must not be modified by your implementation of the scheduling algorithms.

The last two values (starttime, endtime)are available to the algorithms to store information of each process. You are supposed to print out their values in the terminals. You are also fee to add more values which are neccessary for your implementations. All time values are given in the seconds. In this lab assignement, for simplicity, let's assume that there is no cost to switch from one process to another.

2.2 Input file

The *scheduling.c* program reads the first three values for each process from the *Testcase.txt* file and then stores the input information for each process in a process array. You can take a look at *Testcase.txt* for more information about the format of the file. In general, it contains three columns that are corresponding to the first three values of the process structure. Each line corresponds to one process.

```
1 0 4
2 7 10
3 9 2
4 13 8
5 25 1
6 35 7
7 36 2
```

With simple explaination, the first line means that Process 1 arrives system at time 0 and its running time is 4. The same applies to other lines.

3 Scheduling Algorithms

In this assignement, you are asked to implement two simple process scheduling algorithms in Batch Systems. They are first come first-come, first-served (FCFS) and shortest job first (SJF) algorithms which you already from the lectures.

3.1 void first_come_first_served()

This function in the *scheduling.c* file implements FCFS algorithm. The FCFS algorithm simply selects the process which arrives first and run until the process finishes its job. If there are more than one process arriving at the same time, the process which has lower index will be run first.

3.2 void shortest_job_first()

This function in the *scheduling.c* file implements SJF algorithm. The SJF algorithm simply selects the process which has the shortest running time and allows the process run to its completion. If there are several processes with the same running time arriving to the system at the same time, they will be run in the order of their index in the process array. The one with lowest index will be run first.
Note: For simplicity, we assume that the scheduler can swithch every second and that there is no cost to switch from one process to another.

3.3 Output

For each scheduling algorithm, print out the process ID, the arriving time of the process and its running time. The majority of this part is almost done for you in the *scheduling.c*. Next, print out the time that the process starts and finishes. This is what you have to implement for each algorithm and then print the information you get in the terminals. The sample output for FCFS algorithm is given below:

```
Process   arrival    runtime
1          0          4
2          7          10
3          9          2
4         13          8
5         25          1
6         35          7
7         36          2
First come first served
Process 1 started at time 0 and finished at time 4
Process 2 started at time 7 and finished at time 17
Process 3 started at time 17 and finished at time 19
Process 4 started at time 19 and finished at time 27
Process 5 started at time 27 and finished at time 28
Process 6 started at time 35 and finished at time 42
Process 7 started at time 42 and finished at time 44
```

The same output applies to SJF scheduling algorithm, except its name.

3.4 Gradings

Each algorithm implementation weighs for 50 points. For each algorithm, your implementation will be tested with a number of testcases.

- If your implementation produces the correct outputs for all testcases, you will get 50 points.
- If your implementation does not produce the correct output for any testcase, you will get 0 points.
- If your implementation produces the correct outputs for some testcases but not all, we will look at your code and your report to determine your scores.

4 What To Submit

Complete this lab and put your files into the lab11 directory of your repository. Run git `add .` and `git status` to ensure the files have been added and commit the changes by running `git commit -m Commit Message`. Finally, submit your files to GitHub by running `git push`. Check the GitHub website to make sure all files have been submitted.