ITF22519: Introduction to Operating Systems

Fall Semester, 2022

Lab8: Process Scheduling

Submission Deadline: November 18th, 2022 23:59

You need to get at least 50 points to pass this lab assignment!

In this lab, you will do a simulation with process scheduling. Process scheduling can also be called *CPU scheduling*. Supposed that there are several processes in the system, each may arrive at different time, the OS has to decide which process to run next. The selected process is determined by on a process scheduling algorithm in the OS. In this assignment, you will explore two simple simple CPU scheduling algorithms named **first-come first-served (FCFS)** and **shortest-remaining-time-next (SRTN)**.

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 OS2022/labs
$ git pull main main
$ cd lab8
```

The details of FCFS and SRTN algorithms are described in the file *FCFS_And_SRTN.pdf*

1 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.

1.1 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 *Testcase0.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 free to add more values which are necessary for your implementations. All time values are given in the seconds. In this lab assignment, for simplicity, let's assume that there is no cost to switch from one process to another.

1.2 Input file

The *scheduling.c* program reads the first three values for each process from the *Testcase0.txt* file and then stores the input information for each process in a process array. You can take a look at *Testcase0.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.

2 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 remaining time next (SRTN) algorithms which you already learned from the lectures.

2.1 void first_come_first_served(char argv[14])

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.

2.2 void shortest_remaining_time_next(char argv[14])

This function in the *scheduling.c* file implements SRTN algorithm. At a certain time, the SJF algorithm checks status of all processes in the system and selects the one which has the shortest remaining run time. 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. Unlike FCFS, the SRTN algorithm allows the CPU to switch from one process to another. **Note:** For simplicity, we assume that the scheduler can switch every second and that there is no cost to switch from one process to another.

2.3 int main (int argc, char *argv[])

This is the main function in the *scheduling.c*. The function takes the name of the input file, explained in subsection 1.2 as its argument. The input file is then passed to the `first_come_first_served()` and `shortest_remaining_time_next()` when they are called in the `main()`. Supposed that the input file is *Testcase0.txt* and that `out` is the bin file when you compile your code in *scheduling.c*, you need to run your code with the following commands:

```
./out Testcase0.txt
```

2.4 Output

Each scheduling algorithm will print its output to an output file. This part is already provided for you in the *scheduling.c*. Your task is to print out the time each process starts and finishes for process scheduling algorithm.
Example:

- Sample output for FCFS implementation given that *Testcase0.txt* is the input file

```
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
```

- Sample output for SRTN implementation given that *Testcase0.txt* is the input file

```
Sortest remaining time next...
Process 1 started at time 0 and finished at time 4
Process 2 started at time 7 and finished at time 19
Process 3 started at time 9 and finished at time 11
Process 4 started at time 19 and finished at time 28
Process 5 started at time 25 and finished at time 26
Process 6 started at time 35 and finished at time 44
Process 7 started at time 36 and finished at time 38
```

2.5 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.

3 What To Submit

Complete this lab and put your files into the lab8 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.