CS 421

Spring 2023

Program 3

Due: Wednesday, March 22nd by beginning of class.

Objectives:

Write a program to compare the performance of the First Come First Serve (FCFS), Round Robin, and SRTF scheduling algorithms.

Details:

Input: A file of processes and related CPU burst times and arrival times. There will be no header in the file and each line will be of the form, "<processID> <burst time> <arrival time>" with spaces between each field.
Example:

A 8 0
B 1 1
C 2 3
D 1 0
E 5 1

You should read the data into a data structure. Then you will simulate the behavior of the 3 scheduling algorithms on the data, one at a time. For each algorithm, your program will need to print out a sort of vertical Gantt chart followed by some summary statistics.

Gantt chart:
First print the name of the scheduling algorithm, then each time a process is scheduled, print out time of the scheduling decision, the processID, and the reason for the context switch. When the last process completes, print the end time, and "Complete". The 3 possible reasons for a context switch are:
Process terminated
Quantum expired (also give remaining time here)
Process preempted by process with shorter burst time (again give remaining time here)

Example: (assuming quantum of 3 ms)

RR Scheduling
0 A Quantum expired – 5 ms remaining
3 D Process terminated

4 B Process terminated
5 E Quantum expired – 2 ms remaining
8 A Quantum expired – 2 ms remaining
11 C Process terminated
13 E Process terminated
15 A Process terminated
17 Complete

Summary statistics: Then print out the turnaround time and waiting time for each process and the average turnaround time and waiting time.
Example:

Process ID	**Turnaround Time**	**Waiting Time**
A	17	9
B	3	2
C	10	8
D	4	3
E	14	9
Average	48/5 = 9.6	31/5 = 6.2

Then, repeat for the other 2 scheduling algorithms.

If 2 processes have the same remaining burst length (in SRTF) or arrive at the same time (in RR), handle them in alphabetical order.
If a process A is preempted at the same time a new process B arrives, put process A in the queue before B (essentially giving running processes a bit higher priority than new processes.)
You might use an ordered queue to handle prioritization in SRTF.
Use a **quantum of 3** for the Round-Robin scheduler.
Assume no preemption in the first come first serve scheduler. Of course, there is preemption in the RR and SRTF schedulers.

How to get started:

Note that this program is essentially a simulation and hence is supposed to simulate the behavior of a real system implementing the scheduling algorithms on a set of processes. As a result, I would suggest implementing versions of the real data structures that an OS would use here, namely a job queue (a queue of all processes in the system now and the future) and a ready queue (the set of jobs ready to run.) You could then update the ready queue every (simulated) second by moving any jobs which have arrived from the job queue to the ready queue. Then your scheduling algorithm would examine the ready queue to choose the next process to run. You might maintain a process' burst time or remaining burst time as part of the process' record in the event queue as well so that your scheduler has all the information it needs right there.

Remember to do one piece at a time and make sure that you always have something to turn in before moving on to the next piece. Note that this would include the corresponding output.

How can you check if your output is correct? (Because you would want to check, right?) You can easily work out the scheduling order by hand.

Notes:

The inputfile is posted separately. While you will only hand in output for one input file, your program must work for any input file (specifying no more than 8 processes.)

It's always a good idea to use a makefile. You put all your file dependencies in the file and type "make" on the command line to build the project. Here is an example called "makefile":

```
# This is a comment. OS Program 3
# The 2nd line below must begin with a tab.
kbscheduler: kbscheduler.c
    gcc -Wall kbscheduler.c -o kbscheduler
```

Requirements:

You may write your program in any language you choose, and it can run on any platform (Windows, Mac, Linux) you choose. Your program must run as a stand-alone program. You cannot run it through your IDE. If you haven't done that before, this ancient post will get you on the right track: http://www.cplusplus.com/forum/beginner/11397/

Your program must be well-commented including function headers and explanations of complex code. Proper indentation is expected.

You must write one program which simulates all 3 scheduling algorithms, not 3 separate programs.

To submit:

- Your commented code
- Your program's output **including the program invocation command** (ex. xxscheduler Lab3inputfileS23.) This would be a typescript file on a Linux machine, or you could write to a file. It could be also be screenshots on a Windows/Mac machine although this is the worst option. In this case, you must be very careful to ensure that the output is legible and shows all output and program invocation. If they are illegible, then you will not receive credit.
- A paragraph in which you choose one of the three scheduling algorithms and argue why it is better than the others based on the output of your program. For instance, if you

choose RR, you must argue why it is better than FCFS, and then also argue why it is better than SRTF as the reasons may differ.

How to generate a typescript file:

1. Log on to you Linux account.
2. Start up the script command. "script"
3. Run your program on the input file. "xxscheduler inputfile"
4. Exit from script program. "exit"

The resulting script file is called "typescript". Please use the script2txt utility to clean up your output file.
