Lab exercise 4

Round Robin Scheduler

Due date

Feb 27, 2023

Academic Integrity

The following actions are strictly prohibited and violate the honor code. **The minimum penalty for plagiarism is a grade of zero and a report to the Aggie honor system office.**

- Uploading assignments to external websites or tutoring websites such as Chegg, Coursehero, Stackoverflow, Bartleby, etc.
- Copying code from external websites or tutoring websites such as Chegg, Coursehero, Stackoverflow, Bartleby, etc.
- Copying code from a classmate or unauthorized sources and submitting it as your work.
- Sharing your code with a classmate.

Keywords

CPU scheduling

Introduction

The objective of this exercise is to give you hands-on experience creating a process CPU scheduler.

Expected Functionality

Your code should take as input a CSV file containing a list of processes (or tasks) and time quantum. It should create a CPU trace of which process runs every second and when each job finishes. You will simulate a Round Robin scheduler. The inputs to your program will be the -i and -q flags, input file, and time quantum, respectively.

All input files are provided in the Tasks/ directory. Assume that all times are given in seconds. Also, assume that the arrival time of any given process is given sequentially in the input file. If there are no jobs running at a time, you may specify NOP as the pid. The burst time associated with a process is the amount of CPU time the process needs to complete.

Example: Given the input (Columns are: PID, arrival time, burst time):

(Task1.csv in the Tasks folder)

PID	Arrival Time	Burst Time
1	0	3
2	3	4
3	4	5

4	4	6
5	5	5
6	8	9
7	9	4
8	12	2

The expected output should be:

```
:: ./schedule -i Task1.csv -q 3
System Time [0].........Process[PID=1] is Running
System Time [1].........Process[PID=1] is Running
System Time [2].........Process[PID=1] is Running
System Time [3].........Process[PID=1] finished its job!
System Time [3].........Process[PID=2] is Running
System Time [4].........Process[PID=2] is Running
System Time [5].........Process[PID=2] is Running
System Time [6].........Process[PID=3] is Running
System Time [7].........Process[PID=3] is Running
System Time [8].........Process[PID=3] is Running
System Time [9].........Process[PID=4] is Running
System Time [10].........Process[PID=4] is Running
System Time [11].........Process[PID=4] is Running
System Time [12].........Process[PID=5] is Running
System Time [13].........Process[PID=5] is Running
System Time [14].........Process[PID=5] is Running
System Time [15].........Process[PID=6] is Running
System Time [16].........Process[PID=6] is Running
System Time [17].........Process[PID=6] is Running
System Time [18].........Process[PID=7] is Running
System Time [19].........Process[PID=7] is Running
System Time [20].........Process[PID=7] is Running
System Time [21].........Process[PID=8] is Running
System Time [22].........Process[PID=8] is Running
System Time [23].........Process[PID=8] finished its job!
System Time [23].........Process[PID=2] is Running
System Time [24].........Process[PID=2] finished its job!
System Time [24].........Process[PID=3] is Running
```

```
System Time [25].........Process[PID=3] is Running
System Time [26].........Process[PID=3] finished its job!
System Time [26].........Process[PID=4] is Running
System Time [27].........Process[PID=4] is Running
System Time [28].........Process[PID=4] is Running
System Time [29].........Process[PID=4] finished its job!
System Time [29].........Process[PID=5] is Running
System Time [30].........Process[PID=5] is Running
System Time [31].........Process[PID=5] finished its job!
System Time [31].........Process[PID=6] is Running
System Time [32].........Process[PID=6] is Running
System Time [33].........Process[PID=6] is Running
System Time [34].........Process[PID=7] is Running
System Time [35].........Process[PID=7] finished its job!
System Time [35].........Process[PID=6] is Running
System Time [36].........Process[PID=6] is Running
System Time [37].........Process[PID=6] is Running
System Time [38].........Process[PID=6] finished its job!
```

The code

You only need to edit two functions in the RoundRobin.cpp file: Correctly completing only the schedule_tasks() function and an incomplete constructor in RoundRobin class are necessary for full credit. The scheduler should select which process is active and pre-empt processes once the time quantum has been completed.

You are provided a Process class in Process.cpp, RoundRobin class in RoundRobin.cpp, and a main function in schedule.cpp. The only file to modify is **RoundRobin.cpp**; everything else is already written for you.

The Process class and the main() are already completely implemented. DO NOT MODIFY these files, as it may affect the accuracy of testing scripts.

The Process class represents each process/task that needs to be scheduled. For example, Task1.csv has eight processes, so there should be eight instances of the process class created when running RoundRobin. Data to be stored and potentially beneficial functions to use can be found across the Process.h/.cpp. Particularly important functions include the constructor(s) and Run().

The schedule.cpp includes the main(), which creates an instance of RoundRobin with parameters depending on the flags. The main() also parses inputs to ensure correct formatting of inputs and includes the support of the following flags:

- *-h*, for help
- *-i <filepath>*, for tasks input in CSV format
- *-q <number>*, for quantum length

Your tasks

You need to implement the constructor and schedule_tasks functions of the RoundRobin class. The completion of the constructor can include saving the time quantum, opening and/or parsing the file given. You may also find it helpful to initialize vectors for ready processes or the active process.

The schedule_tasks() function is the bulk of the assignment. It will be in charge of using a queue of Process pointers to run a system simulation where these processes are simulated to be “running”. This function is also responsible for producing print statements, as seen in the example output (this is where the print function of the RoundRobin can come in handy - and feel free to modify it if you like!).

It is recommended you look through the other functions to understand how they work, as you will use some of them. If you choose to modify the Process class and schedule.cpp, you will be doing it at your own risk, so please exercise caution if you must edit them.

Rubric

There are three tests that check the elements of the program which you have been asked to fix, and grades will be assigned as the score you see on GitHub classrooms (under the Actions tab):

1. Compilation (10 pts)
2. Correct output when jobs are shorter than quantum (30 pts)
3. Correct output when jobs are longer than quantum (30 pts)
4. Correct output with NOP (20 pts)
5. No memory leaks(10 pts)

Getting started

1. Go to the assignment’s GitHub classroom: https://classroom.github.com/a/RovVnFs5
2. Create a repository for your project.
3. Watch the Getting Started video: https://youtu.be/XpVJs1j_u38