COMP 3500 Introduction to Operating Systems
Project 2 - CPU Scheduling

Points Possible: 100 due: 11:59 pm Feb 16th, 2024

There should be no collaboration among students. A student shouldn't share any project code with any other student. Students' collaboration will be treated as a serious violation of the University's academic integrity code.

Objectives:
1. Complete the source code to simulate CPU scheduling.
2. Build a binary program on a Linux machine.
3. Run the program and record the result of the simulation.
3. Manage the scheduling algorithms Round Robin with priority used in this project.

Requirements:
- Each student should **independently** accomplish this project assignment. You may discuss with other students to solve the coding problems.
- To embark on this project, you may choose one of the following options.
    - Important! Option 1: For Mac and Linux users, use SSH to connect to a remote Linux server. Please read the files in the "tutorial" on Canvas for details.
    - Important! Option 2: For Windows users, Please read "Win subsys Linux" for details.
- Important! Read the I/O format specification carefully and follow. It is very important to your final grade for this project!

1. Introduction to Round Robin Schedule with a Priority

Round Robin (RR) with a priority is a CPU scheduling algorithm where each process is assigned a fixed time slot in a cyclic way.

Assume there are 5 processes with a structure of a Process Control Block (PCB):

1. Process ID: identify each process.
2. Priority: integer 0-100; the bigger the number is, the higher the priority is. The number will be decreased by 1 once a process has been done for a quantum.
3. Execution time (user input): how many ms a process needs.
4. Status: Ready(initialized), Terminated, and currently running. Represented by letters "R" "F" and "R" respectively.

To simplify our simulation, we assume all 5 processes arrive at the same time 0. Also, we do **NOT** require you to implement a cyclic linked list to connect all 5 processes' structures using pointers. We define the structure of PCB in source code and simply store them in an array, looping each PCB every time we use them.

2. Follow the Format Specification (10 Points)

In the source file "Firstname_Lastname.cpp", you will find the four comment lines:

```
// #0#BEGIN# DO NOT MODIFY THIS COMMENT LINE!
// Firstname
// Lastname
// #0#END# DO NOT MODIFY THIS COMMENT LINE!
```

Your first task is modifying the two lines **between** the beginning line and end line. Change them into your first name and last name. Remember the strings are case-sensitive, so capitalize the first letter of the word and follow exactly the syntax of the example.

You can see lots of similar code blocks in the file. You are free and supposed to fill your answer between those special beginning and ending comment lines. You can insert and edit multiple lines between special comment lines in anyways. However (Important!), as the comment indicated, do not modify the special begin and comment lines **themselves**!

Let's do the second task. Scroll down to the bottom of the file and find those lines (press "shift + g" if you are using vi/vim):

```
// #9#BEGIN# DO NOT MODIFY THIS COMMENT LINE!
banner_id = 903900281;
// #9#END# DO NOT MODIFY THIS COMMENT LINE!
```

Look at your student ID card, and check your banner ID. Again, change the "banner_id" value to **your own ID**. Your unique student id will be compiled into the program, and the input of the experiment also uniquely depends on your ID.

Warning: Since every student has a unique banner id, the later compiled binary file is also unique. Copy binary files from other students will be easily detected!

3. Complete Source Code (80 Points)

Read the source code and rest comments and try to understand the function of each line of code. Follow the instructions in the comments and insert proper code into the rest eight blocks to implement a CPU scheduling with RR/Priority. (C/C++ files are acceptable). Read the requirements carefully because some blocks require exact syntax.

4. Run and Record Simulation Result (5 Points)

Compile your source code into a binary program. For example, use following command:

$ g++ Firstname_Lastname.cpp -o Firstname_Lastname

You can run the program and set-up any condition you wish to simulate.

$ ./Firstname_Lastname

Play the program for a while and observe how it works. Now we run a specific simulation: (1) take the last five digits reversed and set them as the priority (2) set their running time 1, 2, 3, 4, 5 respectively. (3) Gives any five id your like to the 5 processes

For example, the banner id is **903900281,** then the last 5 reversed digits **1,8,2,0,0**. and I call 5 process "AA", "BB", "CC", "DD", "EE", then my entire setup for 5 processes will be:

process id	priority	running time
AA	1	1
BB	8	2
CC	2	3
DD	0	4
EE	0	5

We have already learned instructions on how to use command script recording commands in the terminal in project 1. Here we use the script command to record.

$ script Firstname_Lastname.script
Script started, file is typescript
$ ./Firstname_Lastname
banner_id: 903900281
ID PRIORITY RUNTIME--->PCB[0]
_

The program starts running and waits for your input. In this example, according to my set-up of processes, my first line input is as following:

$ AA 1 1

Press enter and finish the rest lines of input. Following the instructions are given by the program and run the program until the end. Then exit recording and save it into typescript file "Firstname_Lastname.script" by the following command:

$ exit
exit Script done, file is Firstname_Lastname.script

Warning: Since every student has a unique banner id, the result of the simulation is also unique. Copy simulation results from other students will be easily detected!

5. Deliverables (5 Points)

Since you have generated multiple script files, please save all the script files in one directory. Create a folder "Firstname_Lastname" first. Please make sure the **folder name** in correct form. You can use the command mv to rename the folder:

$ mv folder-name Firstname_Lastname

Make sure **all the three files** are in this folder. You should have those following files inside the folder:

1. commands recording file: Firstname_Lastname.script
2. executable binary file: Firstname_Lastname
3. source code file: Firstname_Lastname.cpp

Achieve all the folder into a single tarred and compressed file with a tar command.

tar -zcvf Firstname_Lastname.tar.gz Firstname_Lastname

You need to submit one tarred file with a format: Firstname_Lastname.tar.gz

Grading Criteria:

1. Follow the format specification: 10%
    a. Do not break the special comments.
    b. Input your name properly (mark #0).
    c. Input your banner id properly (mark #9).
2. Complete the source code: 80%
    a. Each blank worth 10, total 80 (mark #1 ~ #8).
    b. See detailed specification for each blank in the source code file.
3. Compiling and running result: 5%
    a. Compile the code successfully.
    b. Running the program properly depends on your ID.
    c. Record the running results.

4. Deliverables: 5%
    a. Contains all the files.
    b. Naming all the files properly.

Rebuttal Period:

You will be given **TWO business days** to read and respond to the comments and grades of your homework or project assignment. The TA may use this opportunity to address any concern and question you have. The TA also may ask for additional information from you regarding your homework or project.