ITEC 371: Operating Systems

Project 3: RATCH – The Ru bATCH job simulator

In doing this project, abide by the RU Honor Code or you will fail the assignment and possibly the course. This is not a group assignment

<u>Description</u>

For this project, you will build upon your interactive shell from project 2, i.e. all of the commands and functionality from Project 2 is expected to exist and work. You will add the ability to start programs and simulate their execution similar to what has been done in class. The system will have a set amount of memory, a variable burst time, and uses a round robin queue for keeping track of running processes.

<u>Commands the RUSH shell will implement for this project</u>

start <program> - Add the program to the simulator
step <number> - Advance the system X units of time
setBurst <number> - Set the amount of time a process gets on the CPU before being moved to the back of the queue
setMemory <integer> - Set the amount of memory available for the system
getMemory – Print out the amount of memory for the system
run – Advance the simulator until no jobs are left to run
*addProgram <name> <timeRequirement> <memoryRequirement> <pair>**

<pair> is
<timeTodoIO> <AmountOfIOTime> // Both are integers and pairs will be
// added in ascending order
This command adds a program to the filesystem in the current directory
timeTodoIO is relative to the process not the system time

<u>Notes on simulation:</u>
When a job goes to run, the system must have memory to support it otherwise it will not run.
When a job asks for I/O, it must be removed from the queue until it finishes the I/O request.
Once it is finished it will go to the back of the ready queue.
Whenever the current job is finished or leaves the queue, an updated system state should be printed out.

<u>Output for the RUSH shell commands</u>

The major commands for output are step and run. Both should output the follow information. This information can be presented in any way, but it should be easily understandable / readable.
step <number> or run should respond with the following:

```
Advancing the system for <number> units or all jobs finished
Current time <X>
Running job <name> has <Y> units left and is using <Z>
```

```
     memory resources.
The queue is:
     Position 1:  job <name> has <Y> units left and is using <Z>
     memory resources.
     Position 2:  job <name> has <Y> units left and is using <Z>
     memory resources.
Finished jobs are:
     <name> <timeRequirement> <totalTimeToExecute>
Next burst time <Y>
Running job <name> has <Y> units left and is using <Z>
     memory resources.
The queue is:
     Position 1:  job <name> has <Y> units left and is using <Z>
     memory resources.
     Position 2:  job <name> has <Y> units left and is using <Z>
     memory resources.
Finished jobs are:
     <name> <timeRequirement> <totalTimeToExecute>
```

Sample usage scenario (on an empty binary file called sample.bin)

```
./RATCH sample.bin
EnterCommand>setMemory 3
EnterCommand>addProgram first 4 1 1 1
EnterCommand>addProgram second 4 1
EnterCommand>addProgram third 2 1
EnterCommand>setBurst 4
EnterCommand>start first
EnterCommand>start second
EnterCommand>start third
EnterCommand>run
Advancing the system until all jobs finished
Current time <0>
Running job first has 4 time left and is using 1 memory
     resources.
The queue is:
     Position 1:  job second has 4 units left and is using 1
     memory resources.
     Position 2:  job third has 2 units left and is using 1
     memory resources.
Current time <1>
Running job second has 4 time left and is using 1
     memory resources.
The queue is:
     Position 2:  job third has 2 units left and is using 1
     memory resources.
The process first is obtaining IO and will be back in 1 unit.
Current time <5>
Running job third has 2 time left and is using 1 memory
     resources.
The queue is:
     Position 1:  job first has 3 units left and is using 1
     memory resources.
Finished jobs are:
     second 4 5
Current time <7>
Running job first has 3 time left and is using 1 memory
     resources.
The queue is:  empty
Finished jobs are:
     second 4 5
     third 2 7
Current time <10>
Running job is empty
The queue is:  empty
Finished jobs are:
     second 4 5
     third 2 7
     first 4 10
```

Note: your code must account for extraneous spaces or newlines. If it does not handle these input cases, a grade deduction will be made. Your program will be tested by inputting the commands listed earlier into a file, then redirected into your program via ./NAME < inputFile . If your program does not work in this manner, you will not receive a passing grade on the project.

Submission requirements:

Include all of your files in a directory name project3_ RUEmailLogin along with a Makefile that can build your program and produce the RATCH executable. Tar and gzip this directory into the file project3_ RUEMailLogin.tar.gz and submit it to D2L.

When I untar the file with tar xfzv project3_ RUEMailLogin.tar.gz I expect one directory to be produced (project3_ RUEMailLogin) that contains your source code and a Makefile that will build your program.

I use auto-grading on these assignments. If your assignment does not follow them you will receive at most a 0 on the assignment.

Rubric

Submission requirements: If not met, you receive a 0 for your submission. Homework/Project 1/Project 2 should have provided feedback necessary to successfully submit this project.

Runtime:

To get a C on your submission it must take the input from the specification and produce the output from the specification (albeit it cannot be hardcoded).
To get a B on your submission it must produce correct output for at least ½ of the additional test cases I will use on your program.
To get an A on your submission it must produce correct output for all of the additional test cases I will use on your program.

Design:

Can lower your grade by one letter. Deductions are made for choices such as all of your code in one function. Use what you have learned from previous courses and you should have no issues with this deduction.

Comments:

Can lower your grade by one letter if your submission doesn't contain comments on:

Variables – purpose, type, scope
Function headers (purpose, parameters, return type, called by / calls)
Explanation for blocks of code (inline comments)
Readme file for program containing name, what it is, how to use it.
Header for every source file containing purpose and functions / classes contained / where to start looking.