ITEC 371: Operating Systems

Project 4: RUIN– the RU virtual memory simulatIoN

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

Description

For this project, you will build upon your job simulator from project 3 to create an executable called RUIN. You will add the ability for the simulator to use virtual memory. Virtual memory is unlimited in this project. A page table will not be used for this project. A job is either in memory or it is not. The Most Recently Used algorithm will be used to determine which job will be switched into virtual memory if the system memory is full when a job is started or when the running job needs memory. The aspect measured by MRU will be the job that has been waiting the shortest amount of time to run. If there is a tie, choose the job that is farther away from the CPU from running. The running job can never be swapped to disk.

When a job is added to the system it is allocated memory. This means that VM may be allocated before the first time unit has been executed in the simulation.

When a job's memory is stored on disk, it is required to fetch its contents in 2 time units before it can execute. This time is subtracted from the burst size which is guaranteed to be at least 2 in this project. Note: this may cause another job to swap its memory to disk.

A program must fit in memory to run, if it uses more memory than the system has available it cannot be added to the run queue.

Output for the RUIN shell commands

The major commands for output are step and run from the previous project. You should add whether or not a job is in memory to your output.

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

```
./RUIN sample.bin
EnterCommand>setMemory 2
EnterCommand>addProgram first 3 1
EnterCommand>addProgram second 3 1
EnterCommand>addProgram third 3 1
EnterCommand>start first
EnterCommand>start second
EnterCommand>start third
EnterCommand>setBurst 3
EnterCommand>run
Advancing the system until all jobs finished
```

```
Current time <0>
Running job first has 3 time left and is using 1 memory
     resources.
The queue is:
     Position 1:  job second has 3 units left and is using 1
     resource on disk.
     Position 2:  job third has 3 units left and is using 1
     memory resources.
Current time <3>
Running job second has 3 time left and is using 1
     resource on disk.
The queue is:
     Position 1:  job third has 3 units left and is using 1
     memory resources.
Finished jobs are:
     first 3 3
Current time <6>
Running job third has 3 time left and is using 1 memory
     resources.
The queue is:
     Position 1:  job second has 2 units left and is using 1
     memory resources.
Finished jobs are:
     first 3 3
Current time <9>
Running job second has 2 time left and is using 1
     memory resources.
The queue is:  empty
Finished jobs are:
     first 3 3
     third 3 9
Current time <11>
Running job is empty
The queue is:  empty
Finished jobs are:
     first 3 3
     third 3 9
     second 3 11
```

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 named project4_RUEmailLogin along with a Makefile that can build your program and produce the RUIN executable. Tar and gzip this directory into the file project4_ RUEMailLogin.tar.gz and submit it to D2L.

When I untar the file with tar xfzv RUEMailLogin_project4.tar.gz I expect one directory to be produced (project4_ 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 60 on the assignment.

Rubric

Your program will be graded in the following areas. Your assignment is graded on its capabilities.

Adherence to submission requirements

- 3 – Meets all requirements in the assignment
- 1 – Does not meet all requirements

Functionality

- 3 – Works on input not provided at time of assignment
- 2 – Works on input provided in the assignment
- 1 – Does not compile or work on input provided in the assignment

Code quality

- 3 – Good design and formatting
- 2 – Design has minor flaws or minor formatting issues
- 1 – Major design flaws or low code formatting quality

Comments

- 3 – Well commented, good coverage, functions, classes, the program itself, readme files
- 2 – Code is mostly commented, some features from 3 level work left out
- 1 – Little to no commenting

This assignment is worth 100 points and are assigned as follows.

A – Is rated as a 3 in all areas

B – Has a 3 in functionality and adherence to submission requirements and at most one two in the other areas

C - Has a 2 in functionality, a three in adherence to submission requirements, and no lower than twos in the other areas

D – Has a 2 in functionality
F – Does not meet the above requirements