Name: Jacob Picardat
V-number: V00906806

GITHUB

https://github.com/picardatjp/os_sim

REQUIREMENTS

There is a sim.exe you can run. If you wish to compile it, either run the makefile or the following command:

```
g++ -o sim.exe ./*.hpp ./*.cpp
```

Note: The sim.exe in the project folder was compiled on windows and I have not tested it on any other OS.

DESCRIPTION

Part1: This is an operating system simulator in the terminal written in C++. It loads text files that contain commands along with their cycle operation limits. Just about all of the functionality is in the os.cpp file, and some struct definitions in the os.hpp. I implemented a critical section which I put a mutex lock on, just a bool next to the very first operation of the first process. The process runs until the operation is over so other processes can "access" the memory. I implemented first come first serve scheduling as well as round robin, which you can pick one at the beginning of program. It is a bit buggy at times and I am looking to find and fix them. I tried to add comments when I remembered too, sorry if the readability is bad. CALCULATE just counts down cycles when running, IO counts down when it is the first operation regardless of running, and fork creates duplicate process from when it was called, minus the fork command.

Part 2: There is 512MB of main memory available between all processes. If there is not enough memory available, the remaining instructions will not be moved to the ready queue. Every instruction corresponds to 1 page, and 1 page is 1MB (I just made all memory sizes denominations of MB to make things easier. Normally a 1MB page would be bad). All, or as many can fit, operations get set to a page in "virtual memory". Then, when a process' state is NEW, the page gets put into a frame and this relation gets set into the page table. Page replacement is done via a queue and removes the first page of the queue from its frame and adds the new page to the back of the queue and attaching it to the newly freed frame. I did not make it any easier to read or modular this time around, sorry about that. For part 3 I will try to refactor to help make grading this less of a nightmare.

USER GUIDE

When you run the simulator, you will be given prompts of what you can do. Select an option by typing the number at the beginning and pressing enter. Prompts may result in more prompts, just type in the needed information and press enter. To run a file, make sure it's a text file, and put the command in all caps, followed by a space and the minimum number of cycles followed

Name: Jacob Picardat
V-number: V00906806

by a space and the maximum number of cycles. For the Fork command put no spaces or numbers behind it. Every command should be on its own line. I didn't test putting in wrong input so it might not work if the data and inputs are typed correctly. The files used to run on the simulator should be in the same folder as the exe. You can just type "file.txt" instead of figuring out relative path. And you have to make the file, I only provided test.txt.