Coby Cockrell
10/23/2022
OS Part1

Currently, My implementation is reset as it's original definition I had made it nearly impossible to access the members inside the PCB. I redefined the structure of the Process and PCB classes. I could not properly complete part one, I am still having issues messing with the state machine, I want to implement a state machine class outside the rest, and inside have a single cycle implemented controlling the process's next state. However, In doing so, multiple times I come across either multiple definition errors, or I run out of data in the stack. This new version has mostly resolved that issue, however, in some runs of testing it appears.

My next step as I had in the previous version was to create a linked list from the processes. I would initialize the process class as the node, and make a list class to link them. This would then be used in main, or in the OS which could add to the list of processes. Moving forward after figuring out how each process will cycle from run to ready without being in the state machine, a linked list of the processes is next.

In order to use what is left of my Operating System, in the main, there are displayed two example prints of the XML files. My UI had originally been an interrupt menu that would stall until the next user input (The options included –help, –log, –RunCycles, –loadFile). In order to run a process, after initializing an object of the StateMachine class, all that is needed would be to call the RunSim function to run a single cycle of the operating system. I have not yet implemented a scheduler, as I spent most of the time rebuilding the Process's connection to the State Machine.

Note, My implementation uses the "rapidxml.hpp" library, this was a standard library included at this source: https://rapidxml.sourceforge.net/rapidxml.hpp

The Two XML files I have included have the following input format:

```
<TemplateData>
    <Operation>... Being (CALCULATE, I/O, or FORK)
        <name>...  Being(Calc, IO, Fork)
        <min>...   Being(any positive integer)
        <max>...  Being(any positive integer)
    </Operation>
    …
</TemplateData>
```

That being said I have more implementation questions rather than conceptual questions coming away from the first part, I suppose my first biggest question is what would it look like to have my processes auto init to the state new, place them all into ready a cycle after the creation, and then maybe only in the run state put them on the “cpu” instead of having the process only active one at a time in my current implementation. I also spent a good amount of time looking at how to task user input without affecting the rest of the program running. In order for the state machine to run and only respond to user input I would need to implement a sort of interrupt. I have a good idea on how I can implement a scheduler if it is simply assigning the next process based on the techniques given in class, and for a critical section I would include a nested class within the process.