Simon Owens
Operating System CS470
Process Scheduling

High-level Design

This program is a process scheduler class which demonstrates how a preemptive round robin queue works. The purpose of this assignment is to gain appreciation of the work involved in managing processes. There are two files, the process-management driver, and the other is the input file. Output is written to a file. The usage for the program is as follows:

"python3 process-scheduler.py input-file q output-file"

This program is combined into three large pieces: the "process" class, the "process object block" class, and the main program. The "process_obj" class acts as a simple process for us: it holds its PID, initial burst time, state, and the time left till it is complete. This class also has functions to easily change and get values residing inside this class. The process control block "PCB" class manages all of the processes by keep track of the ready queue, wait queue, current running process, the parent process, q, and how much time q has left to run. This class was created so that users could easily operate with processes however they want: it is very flexible. You just give it commands, and it manages everything for you. That is what the main program does, it defines all of the commands, opens an input file, and executes those commands in the file.

The processes control block is where all of the design is for this project. It can create processes, destroy them, interrupt them, place them in a wait queue, take them out of the wait queue, print out the status of all processes, and exit. The user does not need to be concerned with how processes change state or values: everything is taken care of in the "update" function. This function is called after every operation to ensure that the correct amount of time has been taken, and that everything is in the right state. Python lists are used so that processes can easily be added or removed. They are also excellent because they are ordered, I only need to take the 0 index when trying to find the next process in the ready queue. A dictionary was used for passing commands to the class because this reduces the amount of conditional statements. If operations were to be added, they could either be stuck into the existing conditional statements, or one more could be easily made.

Additional Questions

1) The RR scheduling algorithm seems to be moderately efficient at completing processes. However, if the quantum time is much smaller, than any processes burst time, then it takes forever to finish any process. If the quantum time is much larger than any processes burst time, then some processes wait a tremendous amount of time.
2) It was very very difficult making sure everything was in the right state. I had to make several conditionals and functions just to keep track of everything. There were several edge cases to consider.
3) It was very easy adding, removing, and searching for processes from the ready queue and waiting queue. This was mostly because of python.
4) I would refactor a TON of stuff. I think my design is decent and very flexible but many things could be cleaned up. I did not refactor because lack of time.
5) I was unsure if the quantum time decrement on a event wait or signal. This project was awesome, it improved my process scheduling knowledge as well as my design skills. It was hard keeping everything clean, simple, and reusable.