Chaya Chrein

CS 340- Prof. Svadlenka

Project 2 Writeup

For my project design, I used global variables to serve as shared memory to create synchronization. I started with creating a miniPCB class to create a process control block for each "process" to be executed. I then had a global ready queue which was an array and a global message queue which was also an array. The ready queue was rearranged after each time the scheduler function was called so that the returned process would essentially be removed from the ready queue and there would be no empty indexes except for the ones at the end which did not need to be used. To keep track of how many indexes of the array were ready processes, the global variable queueSize started with the total number of processes and then shrunk with every call of the scheduler function.

To maintain the "first in first out" principle of the message queue, the processesRcv global variable kept track of how many processes were received by the logger thread. It started at 0 and then increased reading in the respective indexes in the message queue so as to read them in the order they were inserted. Additionally, there was also a processesSent global variable to ensure that the logger thread never got ahead of the schedule-dispatcher thread. Also, for synchronization, whenever the send function was called, the lock was put on and then removed when it was completed, and prior to the rcv function being called, the lock was put on and then only removed after the data from the miniPCB was written out the the file so that neither process would be disrupted when accessing the message queue.

One of the issues I encountered with this project was writing integer values to the output file. Since I had the same issue in project 1 as well, I used the same toAsc() function that I wrote for project 1 which separated each digit and added 48 to it, then turned it into its ascii character, and then put it back together to give me a string. Also, I had to call the write function quite a number of times to write the commas and spaces after each digit was written since concatenating a string in c is not as simple as it is in java or other languages. I also struggled with finding where to put the create function for the output file, for at first I put it in main and had the file descriptor as a global variable which didn't work. Once I moved it inside the logger function, my program wrote to the file correctly.

Another issue I had to figure out was solved through my design of global variables. When working with threads that rely on other processes being done first and need to know where the other thread is holding, there needs to be a way for each thread to keep track of the other. Through using global variables, the ready queue was able to be accessed by the main thread and s-d thread and the message queue was able to be accessed by the s-d thread and the logger thread. In addition, both threads were able to know exactly where the other was up to in the message queue since there were global counter variables for each one.