Drew Goin

Operating Systems Concepts (undergraduate)

Project 2: Theater Simulation

**SUMMARY**

I began project 2 without really understanding how semaphores work, so it seemed fairly daunting to say the least. I spent a chunk of an afternoon studying the barbershop example from the book and drawing out different diagrams of possible scenarios before I felt confidant enough to get started.

Since the design section of this project was a whopping 40% of the grade, I decided to start off with the pseudo code. I'm grateful for this however, it ended up saving me a lot of time in the end. I studied the barbershop example and made my design similar in structure and layout. This took a decent amount of time to make, but I'm glad I did this phase first.

I had decided to start this project with java as I did with project 1, but soon realized I had several problems unique to this project that weren't easily solved with java. One thing was that I needed to make my code significantly different from my pseudo code due to both java requiring classes and the fact that threads and semaphores are different from the pseudo code. I was also having difficulties with was finding a way for the separate threads to all have access to modifying the shared variables (the list of movies, list of customer choices, etc...) Since I already knew that c++ allowed functions to alter global variables intuitively and it let me code similarly to my pseudo code, I ended up switching to c++ before I got too far with java.

That doesn't mean I didn't have slight difficulties outside of java however. The first time I ran my code with a couple threads that would print output simultaneously, I quickly found out that output could be corrupted when multiple threads tried printing at the same time. To remedy this I added a new function that would print a string, but used a new semaphore to provide mutual exclusion for output. This let me send a string to print and be confidant that the string will not be corrupted during output.

I had a small issue when I was trying to get the threads to sleep for an appropriate amount of time. I originally used sleep(), but sleep doesn't allow for time that is less than one second, so I moved to usleep(). I thought this used milliseconds, but after a bunch of testing I realized it seemed like things moved much faster than they should. It was an easy change once I found that usleep() uses microseconds instead of milliseconds however.

Another problem that I had was getting the input from the file and storing it into the proper variables. At first, I didn't realize that the number of available tickets were tabbed over, so I had a tough time finding a way to parse the strings. It wasn't too bad after I discovered that there is a tab character that breaks up the string.

I had to make a few changes to the design document after I finished my code, including the new print function and it's semaphore.

I had a lot of fun with this project, it was interesting and was a very interactive way to learn the ins and outs of semaphores. If I could make any recommendations to later classes, I would suggest setting an earlier due date for the Design section and reduce it's grading weight. I found it very helpful to be “forced” to the design first, but I think I would prefer having more weight on the actual results/code.