Kevin Roa
CS 4348.501
Prof. Ozbirn
4/30/2022

Project 2 Summary

Summary of Simulation

The theater simulation uses various threads to model customer and employee behavior. The threads each use semaphores to coordinate activities smoothly. The code first creates the 2 box office agent threads, 1 ticket taker thread, and 1 concession stand thread. From that point, the theater is able to open and allow customers to enter. The 50 customer threads are each created when there is an available box office agent to print their ticket. The customers go through the workflow in parallel, interacting with the various employee threads when appropriate. Once the customer enters the theater to watch their desired movie, their thread exits and joins back to the main program. Once all 50 customer threads have exited, then the program finishes execution.

Difficulties Encountered

The biggest challenge I faced was the initial conceptualization of the problem. I wasn't sure how to tackle the project and only understood it when I looked at the barbershop example in the book. Handling multiple threads in such a manner, I feel, requires an entirely different way of thinking about programming; you need to consider how each thread will interact with eachother and write the code to ensure it acts property while still being independent of the rest. I had trouble figuring out how/where to place each semaphore to ensure everything would function properly. This was the point when I decided to write the pseudocode to dump everything from my mind to an actual theoretical implementation. From that point onwards, the project went by relatively smoothly, with only a few minor hiccups from writing improper c++ code. Thankfully, the overall logic from the pseudocode was all correct which saved me from miserably debugging a multithreaded program.

What was Learned

The most major thing I learned was how to utilize semaphores to ensure properly ordered execution across threads as well as enforcing mutual exclusion where necessary. Semaphores can actually be utilized in a variety of different manners to achieve certain results. This project was also somewhat of a review for me on how pthreads worked in c++. It had been quite a while since I last used pthreads so I'm glad I got the quick refresher.

Results

The console outputs the current action of the various threads as it progresses. I went through the output and confirmed that everything lined up with each other and that the various threads executed in parallel properly. The overall result is a functioning simulation of a movie theater environment where 50 customers interact with the 2 box office agents, 1 ticket taker, and potentially the 1 concession stand worker.