Cameron Meyer
CS 4348.001

Project #2 Summary

In Project #2, I was tasked with simulating several patients' visits to a clinic. By creating a multithreaded program, I was able to simulate the actions of patients, doctors, nurses, and a receptionist for the clinic. These threads coordinated their actions through the use of semaphores as opposed to mutexes, busy waiting, etc. This implementation required plenty of trial and error as I constantly tweaked my code to ensure proper output was always guaranteed. This project was a neat exercise in practicing semaphores via a real-world example, where the coordinated actions of each thread corresponded to a real-world event one might see in a clinic. This made the project feel much more manageable, since it was relatively easy to wrap my head around the problem once I had an understanding of how semaphores worked.

Relative to the previous project, project 2 felt much more straightforward, and more achievable within the allotted time. Since I spent a great deal of time sorting out my development environment with the last project, I used that knowledge to get up and running much more quickly this time around. Once again, I was able to complete all my development on the UTD Linux server by utilizing VS Code's ability to SSH directly into the cs1 server. As development progressed, I periodically saved my work and pushed updates to a public git repository via the VS Code terminal.

I chose to implement this project in C++, which involved using POSIX pthreads and semaphores to create threads and coordinate their actions. My program expected command line input from the user in order to determine how many doctor, nurse, and patient threads to create at runtime. I implemented checks to ensure the program is given the proper amount of valid arguments. Once the threads were created, they each executed their own appropriate code for their assigned function. By waiting and signaling with semaphores at strategic moments, each thread coordinates its actions with the others to effectively recreate the behaviors of people in a real clinic.

During development, I luckily didn't encounter too many bugs. There was one strange issue with thread output via the printf function where statements could occasionally be printed twice from the same thread, but I was able to resolve that by clearing the stdout buffer before creating the threads. Most of the development issues were related to ensuring proper output--despite starting with a thoroughly pseudocoded program, I occasionally had to make some revisions, adding new semaphores to keep my output consistent. There were a couple segmentation fault issues that cropped up along the way, but I was able to solve those by either shifting an existing semaphore around in code or simply adding a new one. The biggest issue I had towards the end of development involved my program aborting itself whenever patientCount exceeded doctorCount in the program. I spend several hours debugging this before emailing the professor and being advised that a semaphore was created with incorrect bounds. Once that was solved, the rest was smooth sailing.

Obviously, the project cannot perfectly simulate doctor's appointments at a clinic. After all, there's no distinction between doctors' skills or abilities, no specific medical conditions the patients are seeking help for, and no patient personal information kept on file after registration.

However, on a broader level, this project does allow us to see the general interactions between patients, reception, nurses, and doctors. From a distant viewpoint, it is clear that patients always need to check in with a receptionist before they can be taken to a doctor's office for a check up. Additionally, it makes sense that receptionists can only check in one patient at a time, and doctors can only interact with one patient at a time. The project fidelity breaks down a bit when you start to consider that the nurse doesn't really do anything once the patient is interacting with the doctor, or that the patient doesn't ever ask questions about the advice they were given, but that's not really the point here. The overall, most important interactions simulated in this project are fairly consistent with the actions I might encounter in a real clinic. For that, this project seems much more interesting as it can be broadly applied to a real world example and easily understood.