Amogh Yatnatti

Professor Ozbirn

CS 4348.001

10 April 2021

**Project 2 Summary**

The purpose of this project was to get familiar with semaphores and coordinating threads. The use of semaphores is vital in making sure that the threads do not cause race conditions and that concurrency is maximized. Before this project I had a vague understanding of semaphores, but I learned so much more on how to utilize them effectively.

When I worked on the project, I started by coordinating my patient and receptionist threads. The project constraints allowed only for one receptionist and that receptionist could only deal with one patient at a time (mutual exclusion). To solve this, I had to use a semaphore with an initial value of 1 so that I could have 1 thread initially continue while the others wait. Once that thread is done talking to the receptionist, I would signal that semaphore and allow another thread to proceed. During this communication period, the patient is added to the receptionist’s buffer and the receptionist semaphore is signaled. The receptionist thread is waiting for this and would access the patient’s information. The receptionist assigns the patient a random doctor (doctor ID). A difficulty that I had was coordinating the patient’s ID with the assigned doctor/nurse. To resolve this, I made a HashMap where the key would be the doctor/nurse’s ID and it would store a LinkedList that would serve as a buffer. So, each doctor/nurse would have their own buffer where they could access their patient’s information. The receptionist would place the patient into the designated buffer after assignment. Another difficulty I had encountered was figuring out how to coordinate the doctor/nurse’s ID with the patient. In order to resolve this, I created an array where each index represents a patient’s ID. The value at the patient’s index would be their assigned doctor/nurse. The receptionist would signal a register semaphore after it is done, and the patient would access the array to get information about their assigned doctor. The patient then signifies that they are leaving the room and the receptionist signals the designated nurse to come for the patient. Another difficulty is making sure that the correct doctor/nurses semaphores are communicating with the patient. To resolve this, I made an array of semaphores where each index represents a doctor/nurse and their ID. The patient calls the correct doctor/nurse by using the doctor’s ID. I learned that this makes the communication simple and easy to implement since only 1 patient will be communicating with each nurse/doctor at a time. Inside the nurse threads, there is a semaphore array with values all initialized to 1 so that each nurse/doctor can handle one patient at a time (mutual exclusion). The nurse thread would then peek at the patient file (not remove it so the doctor can access it) and then take the patient to the doctor’s office. The nurse will signal a designated office semaphore to the patient so that the patient can then enter the doctor’s office. The nurse then signals the designated doctor semaphore. The doctor thread will then remove and look at the patient’s information from the buffer and listen to symptoms. The doctor will then signal an advice semaphore for the patient so the patient can receive the advice and leave. The patient signals their advice receival and the doctor signals the mutual exclusion semaphore so the nurse/doctor can move to the next patient.

My results were fairly odd. While my output was printing in the correct order every time, it seemed like it would take a long time for the receptionist and nurse threads to initially communicate with the patient threads. I thought that I had mis-coordinated the semaphores in a way where the receptionist would wait for everyone to enter the waiting room before registering and nurse would wait for everyone to be registered before taking them. However, I noticed that if I used a large sample size like 30 patients, the receptionist would start registering patients after about 20 of them enter the waiting room. This confused me because that means the receptionist would simply be slow to register initially. I noticed that the nurse would take patients to the office after the receptionist was nearly done registering everyone. I am not exactly sure why the receptionist and nurse threads are slow to respond initially, but the order of the output is correct, and the threads do work at the correct times.