Jiashuai Lu
JXL173630
Department of Computer Science

March 30, 2018

Project2 of Operating Systems Concepts

Project Summary

My clinic simulator uses three types of threads and nine different types of semaphores. In this clinic program, receptionist and doctor_and_nurse threads are most likely as daemon threads which will keep on running in loop for serving incoming patients once started. Correspondingly, the patients do not belong to our clinic, so each of them would just come and go after they get their advice from doctors.

The most difficult part of this project to me is designing the coordination between different threads to make the whole clinic is in operation regularly and each thread to perform their responsibilities successfully. At the very first I try to use more than thirteen types of semaphores in the design of my clinic. However, with the process of coding going, I realize that some of them are indeed redundant or meaningless or just incorrect. For example, I define the patients semaphore to denote how many patients are waiting for the receptionist to register for them. But I also define receptionist semaphore to let patients know a receptionist is available. In fact, this pair of semaphores are incorrect design in this project under the project description. The clinic does not has a hard limit for number of patients could be accommodated in clinic. Since the receptionist is a daemon thread, once it finishes register service for a patient, it will automatically be available for serving next patient's request. Thus, as long as we have a semaphore for waiting patients, we do not need to have another semaphore to let a patient know if a receptionist is able to serve her. Instead, we can just notice the patient that her registering is finished by signal another semaphore, finishing_registering.

I will use a patient's view to go through the whole process of visiting a doctor. In the following explanation, we will assume a three doctor_and_nurse threads (0, 1, 2), three patient threads (0, 1, 2), and one receptionist thread scenario. Suppose we are observing patient 1, when she just enter the clinic, she will en-queue her patient id into total patient queue. Due to multithreads' non-deterministic feature, even we start patients in the order 0, 1, 2, their patient ids can be insert in the queue in arbitrary order such as 1, 2, 0. Each of these patients will signal the semaphore patients by increment it by 1. We also know the receptionist is waiting on this semaphore, after that the receptionist will dequeue the total patient queue and get patient id 1. Then we assume receptionist en-queue this id to doctor 2's (randomly choice) waiting_patients queue, and signal finish_registering[patient_id]. The patient will then signal waitingfordoctor[doctor_id] to notice doctor_and_nurse thread and wait for nursecall[doctor_id]. After that her will follow the nurse to doctor 2's office, signal nursetell[doctor_id] to let the doctor know she is in the office. Doctor 2 will then listen to her symptoms and giving his advice to her. After she leave the office, patient 1 thread signals leave[patient_id] and doctor 2 waiting for this semaphore to finish this loop for serving next patient.

By doing this project, I learn a lot about concurrency control and how to solve producer/consumer problems by coordinating these threads with proper semaphores. For instance, which threads should be started early and which threads need to be started after specific threads. In this clinic, daemon threads obviously need to be started before you issue any patients threads because you can imagine that patient only attend into the hospital when the hospital is open. The open state of a clinic needs all staff are ready to work. Another typical problem I encounter is as the barbershop example. When the receptionist registers a patient to specific doctor. We need to let the nurse of that doctor, i.e, the specific thread knows there is a new patient request its service. In this case, for all semaphore which we want to use it to coordinate threads respect to specific patient id, we need a unique semaphore for every patient, just as what we solve the problem in barbershop example2.

Below is a result of example by running my clinic simulator with three doctor threads and three patient threads.

```
Output:
# src/main 3 3
Patient 0 enters waiting room, waits for receptionist
Patient 1 enters waiting room, waits for receptionist
Patient 2 enters waiting room, waits for receptionist
Receptionist registers patient 0
Receptionist registers patient 2
Patient 0 leaves receptionist and sits in waiting room
Nurse 1 takes patient 0 to doctor's office
Receptionist registers patient 1
Patient 2 leaves receptionist and sits in waiting room
Nurse 2 takes patient 2 to doctor's office
Patient 1 leaves receptionist and sits in waiting room
Patient 0 enters doctor 1's office
Patient 2 enters doctor 2's office
Doctor 1 listens to symptoms from patient 0
Doctor 2 listens to symptoms from patient 2
Patient 0 receives advice from doctor 1
Patient 2 receives advice from doctor 2
Patient 0 leaves
Patient 2 leaves
Nurse 1 takes patient 1 to doctor's office
Patient 1 enters doctor 1's office
Doctor 1 listens to symptoms from patient 1
Patient 1 receives advice from doctor 1
Patient 1 leaves
```