CS 232 Operating Systems

The Vaccine Clinic

Overview

In this assignment you will explore concurrency and semaphores by writing a C program that runs several different threads. Your concurrent threads must use semaphores to coordinate the resources in a simple vaccine clinic. An overview of how the clinic works is as follows:

- the clinic has two kinds of actors: nurses and clients.
- each client moves through these stages:
    - registration desk
    - waiting queue for assignment to a nurse's station
    - nurse's station (to get the vaccination)
    - leave the clinic
- each nurse moves through these stages:
    - walk to get a vial of vaccine
    - wait in line to a vial, and then get the vial
    - walk back to the station
    - give 6 vaccinations from the vial
    - (possibly go get another vial -- i.e., repeat from the beginning).
    - leave the clinic
- a separate thread is used for each instance of a nurse and client

Detailed Algorithm for each Client

A client thread must do these steps:

- after arrival (i.e., thread creation), walk a random amount between 3 and 10 seconds to the registration desk.
- wait for an opening at the registration desk.
- take a random amount of time between 3 and 10 seconds to register.
- walk a random amount of time between 3 and 10 seconds to get to the station-assignment queue.
- wait for station assignment
- walk a random amount of time between 1 and 2 seconds to get to the assigned nurse's station.
- indicate to the nurse that the client is ready for the vaccination
- wait for the nurse to complete the vaccination

- leave the clinic.

Detailed Algorithm for each Nurse

A nurse thread must do these steps: (for simplicity I will use "her" below, but the nurse could be a "he" or a "they")

- start her shift (i.e., thread gets created)
- walk a random amount of time between 1 and 3 seconds to get a vial of vaccine
- If there are no more vials, leave the clinic.
- walk back (again between 1 and 3 seconds)
- do this 6 times:
    - indicate to the queue of clients waiting for a station assignment that you are ready for the next client.
    - wait for the client to indicate that they are ready to be vaccinated
    - give the client a vaccination (which takes 5 seconds to administer)
- Go back to the 2nd step -- walk to get a vial of vaccine.

Algorithm for the main code:

- create the semaphores necessary for the simulation
- create the nurse threads.
- create the client threads, with a 0 to 1 second random delay between each creation.

Other details:

The registration desk is really 4 desks and thus can process up to 4 clients simultaneously.

The queue for waiting for an assignment to a nurse's station must be implemented with a producer-consumer bounded-buffer. Each nurse produces her station id into the buffer, and each client consumes the id from the buffer.

Your benevolent professor will give you a lot of starting code, found by accepting this assignment: https://classroom.github.com/a/0y_0C_jB

Your code must be written in C, not C++. Your code will be compiled and tested on the lab Ubuntu Linux boxes. Note that MacOS does not implement pthreads correctly or completely, so you should just plan on working on the lab machines (as I had to).

The nurses and clients produce voluminous amounts of output to indicate what they are doing. E.g., here is some of my sample output:

Sat Mar 13 13:23:10 2021: client 33 has arrived and is walking to register
Sat Mar 13 13:23:10 2021: client 24 waiting to register
Sat Mar 13 13:23:10 2021: client 24 is registering
Sat Mar 13 13:23:11 2021: client 22 waiting to register
Sat Mar 13 13:23:11 2021: client 9 is at station 0
Sat Mar 13 13:23:11 2021: client 9 is ready for the shot from nurse 0
Sat Mar 13 13:23:11 2021: nurse 0 sees client is ready. Giving shot now.
Sat Mar 13 13:23:11 2021: client 34 has arrived and is walking to register
Sat Mar 13 13:23:11 2021: client 21 done registering. Now walking to next queue.
Sat Mar 13 13:23:11 2021: client 22 is registering
Sat Mar 13 13:23:11 2021: client 23 done registering. Now walking to next queue.
Sat Mar 13 13:23:11 2021: client 20 done registering. Now walking to next queue.
Sat Mar 13 13:23:11 2021: nurse 1 gave client the shot
Sat Mar 13 13:23:11 2021: nurse 1 tells the waiting queue that she/he is available
Sat Mar 13 13:23:11 2021: client 0 got the shot! It hurt, but it is a sacrifice I'm is willing to make!
Sat Mar 13 13:23:11 2021: client 10 got assigned to station 1: walking there now
Sat Mar 13 13:23:11 2021: nurse 1 waiting for a client to arrive.
Sat Mar 13 13:23:11 2021: client 0 leaves the clinic!
Sat Mar 13 13:23:11 2021: client 25 waiting to register
Sat Mar 13 13:23:11 2021: client 25 is registering

Here is a sample of output for one client (which I got by sending all my output to a file and then using grep on it):

vtn2@goldvm13:~/cs232/vacc_clinic$ grep 'client 4 ' output
Sat Mar 13 13:22:58 2021: client 4 has arrived and is walking to register
Sat Mar 13 13:23:03 2021: client 4 waiting to register
Sat Mar 13 13:23:03 2021: client 4 is registering
Sat Mar 13 13:23:04 2021: client 4 done registering. Now walking to the next queue.
Sat Mar 13 13:23:07 2021: client 4 got assigned to station 8: walking there now
Sat Mar 13 13:23:08 2021: client 4 is at station 8
Sat Mar 13 13:23:08 2021: client 4 is ready for the shot from nurse 8
Sat Mar 13 13:23:13 2021: client 4 got the shot! It hurt, but it is a sacrifice I'm willing to make!
Sat Mar 13 13:23:13 2021: client 4 leaves the clinic!

Here is a simple of output for one nurse:

vtn2@goldvm13:~/cs232/vacc_clinic$ grep 'nurse 4 ' output
Sat Mar 13 13:22:56 2021: nurse 4 started
Sat Mar 13 13:22:56 2021: nurse 4 walking to get a vial
Sat Mar 13 13:22:57 2021: nurse 4 got vial. Num vials left = 29
Sat Mar 13 13:22:58 2021: nurse 4 back at station
Sat Mar 13 13:22:58 2021: nurse 4 tells the waiting queue that she/he is available
Sat Mar 13 13:22:58 2021: nurse 4 waiting for a client to arrive.
Sat Mar 13 13:23:07 2021: nurse 4 sees client is ready. Giving shot now.
Sat Mar 13 13:23:12 2021: nurse 4 gave client the shot

Sat Mar 13 13:23:12 2021: nurse 4 tells the waiting queue that she/he is available
Sat Mar 13 13:23:12 2021: nurse 4 waiting for a client to arrive.
Sat Mar 13 13:23:14 2021: nurse 4 sees client is ready. Giving shot now.
Sat Mar 13 13:23:19 2021: nurse 4 gave client the shot
Sat Mar 13 13:23:19 2021: nurse 4 tells the waiting queue that she/he is available
Sat Mar 13 13:23:19 2021: nurse 4 waiting for a client to arrive.
Sat Mar 13 13:23:21 2021: nurse 4 sees client is ready. Giving shot now.
Sat Mar 13 13:23:26 2021: nurse 4 gave client the shot
Sat Mar 13 13:23:26 2021: nurse 4 tells the waiting queue that she/he is available
…. snip ….
Sat Mar 13 13:25:07 2021: nurse 4 walking to get a vial
Sat Mar 13 13:25:08 2021: nurse 4 found all vials are gone: exiting
Sat Mar 13 13:25:08 2021: nurse 4 is done

My implementation has only 4 global variables (besides the `sem_t` and `pthread_t` variables):

- `num_vials_left`
- 3 variables to implement the producer-consumer buffer of station ids.

A good overview and tutorial of pthreads can be found here. Note that your code **does not** have to handle possible error return values from pthread calls. Most of the calls won't return errors, in all practical cases.

Your output messages do not have to exactly match mine, however, you must print out enough information so a user can tell what is going on at the clinic. Note that my output messages all correspond to pthread or semaphore calls -- none of the messages are there to just make it look like something is happening when it really isn't.

You should print out messages to `stderr`, so that your output messages are not buffered (as they could be if printing to `stdout`). This helps ensure that messages come out in the correct order.

Also, don't forget that you have to link your code with the pthread library. So, when compiling, you need to specify **-lpthread** to **gcc**. E.g.,

```
gcc vacc_clinic.c -o vacc_clinic -lpthread
```

A suggested plan of attack

1. create one nurse, one client, and get the client to go through the system. Don't implement the registration table, or assignment to a nurse/station, etc. Just implement the client going to station/nurse 0 and the rendezvous between the client and nurse to

get the vaccination. Set `num_vials` to 1. Don't worry about the walking times between steps.
2. Add code to the nurse function so that the nurse checks if the # of vials is 0. If so, the nurse exits the thread. (difficult to test at this stage)
3. Add code to the client to stand in the registration line and get registered when one of the 4 registration "desks" is free, then move to the waiting queue to be assigned to a station. Implement the producer-consumer buffer so that the client is assigned to a free nurse's station.
4. Add the random walking delays.
5. Pretend that this is due in 1 week. Then, when you have problems, you don't have to ask for an extension…

Submission

Submit to your repo. Don't forget to sync to github so that the grader can see your code.

Grading Rubric

20 pts total:

- 3 pts: threads created and used correctly.
- 5 pts: correct communication between clients and nurses
- 3 pts: client moves through stages correctly.
- 3 pts: nurse moves through stages correctly.
- 4 pts: output is complete and clear.
- 2 pts: code is well-written and hospitable: good variable names, perfect consistent indentation, good documentation, etc.

This page is maintained by Victor Norman.