Homework 3 Report - CSE344

Djuro Radusinovic
Student no. 171044095

April 2021

1 Composition

For this project I used multiple files for readability purposes. There is main.c, parsearg.c, parsearg.h, utils.c, utils.h, types.h

2 Program logic

First important thing to note here is that we will have three processes: Nurse, Vaccinator and the Citizen
Problems
Nurse communicates only with the Vacciantors and is not able to communicate with the citizen directly. For Vaccinator to give the vaccination to the citizen waiting, Vaccinator has to call the citizen in. This way we know that only Nurse communicates with both processes directly. With Vaccinator through the shared buffer (shared memory or similar in order to solve first producer/consumer problem) and with Citizen by performing synchronization (also form of producer / consumer).
To solve this I will focus independently to all the interactions at first and later on combine them in order to solve the complete problem. Of course, some hurdles will pop up, but this way it will be much easier to debug when we know independent problems work properly.
I will separate our three different processes as:
Nurses - producers
Vaccinators - consumer
Citizen - producer
Semaphores implemented with shared memory is the most straightforward approach to solve this problem.

2.1 Nurses

Nurses are producers here. There will be more than 2 of them. This means that there are multiple producers of vaccines in our situation. Synchronization needs to be established between multiple nurses (producers) and multiple vaccinators (consumers). To do this shared memory will be used between these two processes and of course an unnamed semaphore.
In order to make this synchronization possible, I will use 3 semaphores. One will be our semaphore for access to the buffer so that we can avoid multiple processes accessing the same shared memory segment (buffer will be represented as our shared memory). The other 2 semaphores are used to avoid overflow and underflow. This buffer semaphore would at the same time solve the problem of having multiple nurses accessing this memory segment. So this way we kill 2 fly with one shot.

Another problem here is that there are multiple nurses that share the same offset. To avoid any misleading communication between them the semaphore already mentioned for protecting the buffer will be enough for now.

2.2 Vaccinators

Vaccinators are the main accots here because they communicate with both nurses and citizens. In a sense they can be regarded as the main consumer in this problem. To solve nurse/vaccinator problem we used the classical approach to producer/consumer solution.
On the other hand the communication between vaccinators and citizens is a bit more different but the logic is similar. In order to implement this I used 4 semaphores. One similar to producer/cosumer for this case. One protecting the shared memory segment needed for the communication with citizen. Citizen is writing to this shared memory segment its ID and vaccinators is taking it cautiously and then progresses in order to give the vaccine to the citizen

2.3 Citizens

Citizens are here sort of a producer and They are communicating with the vaccinators of course. They are writing to vaccinators they PID which is used to perform the vaccination. After the citizen is vaccinated a semaphore light turns green for citizen to check if it will finish its execution or continue with the next dose.

2.4 Semaphores

I used a tootal of 7 semaphores. 4 are used for citizen/vaccinator communication and 3 are used for nurse/vaccinator communication

3 Testing

I tested this program using the example file provided

- Test 1: Checking the program using the example file provided

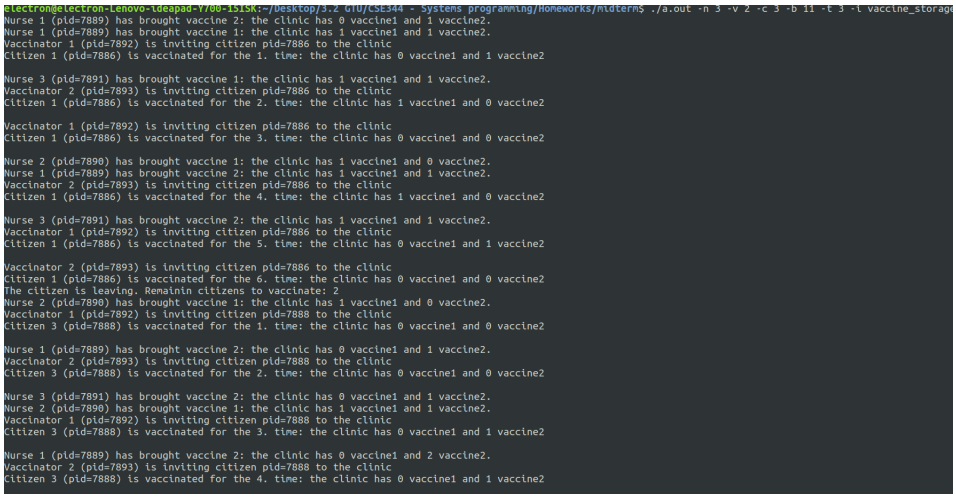

Figure 1: Test1

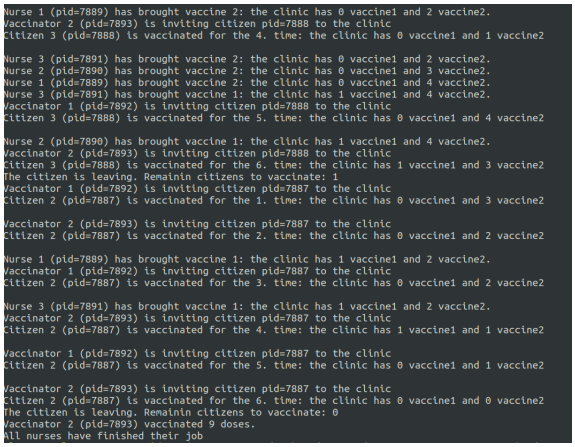

Figure 2: Test1