CSE 344 – SYSTEM PROGRAMMING -MIDTERM

PROCESSES AND SEMAPHORES

OZAN GEÇKİN

1801042103

Systems Programming midterm project report

Problem:

This is basically a (compound) producer/consumer paradigm. You will emulate a covid19 vaccination flow. We'll keep this simple: nurses bring vaccine products to a clinic/buffer, vaccinators vaccinate the citizens by taking two of them (both the first and second shots) and the citizens leave the clinic. Each nurse, vaccinator and citizen will be represented by a distinct process. Each citizen needs to get the two shots t times.

In this problem, what is required from us is to create processes according to the number of processes we receive from the command line and to make them work synchronously.

Solution:

I analyzed the project and divided it into headers in order not to look complicated. I used semaphore, shared memory and signal. To prevent many processes from reaching the places I want to reach in the Semaphore shared memory at the same time. I used it to keep share memory to common data Signal, on the other hand, is used to communicate between processes. I was able to solve my synchronization problems with Semaphores. I explained how I use it in the design decisions section.

Design Decisions:

helper.h analys :

I have defined the semaphorees, structs and functions that I will use in common in different classes.

extern sem_t * bufmutex; => To block the buffer shared by the Vaccinator and nurse class.
extern sem_t *citmutex; => To block when they reach the citizen data of the vaccinator and citizen.
extern sem_t * filemutex; => To make a file read, for nursen reading in the nurse class to block the file.
extern sem_t * vacmutex; => I used it when trading with the number of vaccines between citizens.
extern sem_t empty;=> I used it to make the vaccinator, the consumer, wait while the producer, the nursing, was filling the buffer.
extern sem_t full; => I used it to make the consumer, the vaccunator, receive it and the producer, the nurse, wait.

I used a struct to hold information about citizens.

```
typedef struct{
        int pid;                // To keep the pid of every citizen process
        int shot_number; // To keep the number of vaccines each citizen process has.
}citizen_struct;
```

I have kept a struct so that the buffer can communicate between processes.

```
typedef struct{
        char * buffer;      //This is the data buffer shared between nurse processes and vaccinator
processes
        int buffer_size;    //Buffer size
        int v1_ctr;         //Vaccine 1 counter in the buffer
        int v2_ctr;         //Vaccine 2 counter in the buffer
        int file_read_all_flag;
}buf_ipc_data_struct;
```

I have kept a struct to enable citizens to communicate between processes.

```
typedef struct{
        citizen_struct * citizens;
        int citizens_buffer_len;  // Length of citizen buffer
        int vaccinated_citizens; // Number of inscribed citizen

        int number_of_citizens; // Total number of citizen
        int number_of_shots;    // Total number of vaccines
}cit_ipc_data_struct;
```

I kept a struct to allow vaccinators to communicate between processes.

```
typedef struct{
        int no_vaccination;
        int kill_process;
}vac_ipc_data_struct;
```

Main.c analysis:

I initlized the extern semaphores and structs that I defined in the header class. I used a struct to get the command line arguments and defined this struct globally.And I wrote the necessary functions in main.

main classı function analysis:

void program_exit_function(int status ,void *args) = This fuction provide each process exit we clean the memory in this function.

Void print_program_usage() = This function is the helper function. Command shows how to enter arguments when they are incorrect.

Int validate_command_line_args(int argc, const char *argv[],command_line_arg_struct * prg_args)= It enables commandline arguments to be taken as desired.

void sig_handler(int sig) = CTRL-C I'm control. And I return all my used places.

Int main(int argc,char const *argv[])= This is the main function of my program. I open the semaphorees I use here. I reserve space for structs with mmap. After doing both semaphore and mmap, I check if there is a problem. Here I have done value assignments to the variables I use in my structs. I open the file with the filename I got from the command line. If it doesn't open, I return error. I'm freezing the filename I reserved with malloc. Then I create the processes again according to the numbers I get from the commandline. I do it in separate loops in 3 types of processes. And I call my functions. At the end of the main function, after all processes have finished their work, I exit the program by clearing the places I bought.

nurse.c analysis:

In my project, The nurse processes have a simple job. They read the input file, one character at a time , and place the vaccine they read into the clinic. Imagine this as bringing one vaccine from cold storage to the clinic .The nurse processes terminate after having carried all the vaccines into the clinic. If there is not enough room in the buffer for vaccines, the nurses are supposed to wait.

The nurse function is using one more of the counter in the for loop as parameters, file discreapter,the command line argument is t "number_of_shots", the command line argument is c “number_of_citizens" the number of vaccines that should be int for communication between nurses”nurses_shared_counter".

True, the process starts within a while loop. I keep a flag to check if you can read it. If there is no reading, I lock the file with sem_wait (filemutex), ie file semaphore. I unlock the file with sem_post (filemutex) after reading a character from the file. I used sem_wait (empty) to lock this buffer when it is full, which is necessary to control the buffera write job, and sem_wait (bufmutex) to not write another process while writing the buffera. If the character I read is 1 or 2, these are the types of vaccines. Since we are writing a single buffer, I assign "1" s to the first half of the buffer and "2" to the second half. This assignment process makes my job easy when vaccinating vaccinator citizens. If I received the initial vaccine number for all citizens, the nurse process is over. When the process ends, I first remove the birth lock with sem_post (bufmutex) so that other processes can read. Keeps that the sem_post (full) procedure increases as it adds elements to the buffer.

vaccinator.c analysis:

In my project, The vaccinators are medical personel, and if there is at least one ‘1’, and one ‘2’ at the buffer/clinic, then a vaccinator invites a citizen into the clinic, applies both vaccines , by removing the two characters from the buffer, and sends the citizen away. If there are not sufficient shots in the buffer (or only ‘1’s or only ‘2’s), they wait until there is. The vaccinators terminate after having vaccinated all citizens t times. The vaccinators must not prevent the maximal use of the buffer by the nurses.

The vaccinator function takes as parameter the index of the vaccinator process, the command line argument is t "number_of_shots", the command line argument is c "number_of_citizens".
I am using 3 local variables, their purpose is to keep the number of vaccines local_vaccination_ctr has made. Has_two_vaccines is the flag that checks whether 2 vaccines are available. I use it for for loop. True, the process starts within a while loop. At first, I check to see if I have 2 vaccines. If not, I'll get the vaccines from the buffer with my vaccinator process. For running into some problems here. I put a flag called kill_process inside the vaccinator processes sturct. Semaphore called vacmutex in one. These check vanicator processes. If there is no problem, the vanicator process takes the vaccines from the

buffer. Here he acts as a consumer. First, it closes the critical region input with sem_wait (full) and sem_wait (bufmutex). It checks if there is a vaccine1 and vaccine2. If available, he assigns the vaccines to a variable. Here, there is another critical zone between vaccinators. In this region, if the number of vaccination process is equal to number_of_citizens * number_of_shots, it gives a kill signal. It leaves the vaccine back by doing sem_post (full). If the vaccination is not over, he / she receives the vaccinations. Buffer opens semaphoren with sem_post (bufmutex).

Vaccinator, now with the vaccine. By taking the citizen in the program for loop. When the number of vaccines is less than the number of vaccines, it sends the SIGUSR1 signal with the pid of the citizen to vaccinate. And it increases the number of vaccines of that citizen. If citizen got the vaccine that should have been with that vaccine. Vaccinated_citizen is also incremented. The local vaccines of the vaccinator are increased 2 times because we have vaccinated two times with v1 and v2. Then the vaccinator flag in the hand of the vaccinator is set to 0. Increase the Citizen mutex with sem_post (citmutex).

citizen.c analysis:

In my project, Each citizen process waits for a vaccinator to call her inside the clinic. If that happens she gets vaccinated, and waits to be called again for a total of t times. Once the citizen receives both shots t times, she leaves and lives happily ever after.

Bonus part: However, the citizen processes must be vaccinated according to their age, oldest first. Their pid will be related to their age; so that the citizen process with the smallest pid will be considered the oldest. Don’t let vaccinators sit idle. While a citizen x receives her n-th dose, a younger citizen y should still be able to receive her n-1 dose.

In the Citizen class, there are functions that I use for the bonus part, in addition to the citizen function.

int pid_comparator(const void * a, const void * b)= Citizenların pid lerini karşılaştırıyorum.

void order_citizens_by_age() = Citizenleri pid lerine göre sıralamamı sağlıyor.

citizen_struct * find_citizen_pointer_by_pid(int pid)= Citrizenleri Search in array and return its pointer, call it after entring the critical region. Eğer bulamazsa NULL dönüyor.

void citizen(const int no, const int t) = Citizen function takes process index and number of shots as parameters. It initializes the signal to get a signal from the clinic. I enter the critical region by making the Citizen mutex as sem_wait (citmutex). I'm setting your Citizen. And I call the function of sorting citizens. I'm waiting for a signal from the vaccinator in the true while loop. If the signal comes, I search for citizen according to the pid. And I'm getting over. If the citizen has completed the number of shot from the command line, I remove the citizen. I used citmutex semaphore for synchronization in this while loop.

Some running examples:

1)

```
cse312@ubuntu:~/Desktop/456/1801042103$ ./program -n 3 -v 2 -c 3 -b 100 -t 4 -i example_file.txt
Welcome to the GTU344 clinic. Number of citizens to vaccinate c=3 with t=4 doses.
Nurse 1 (pid=2414) has brought vaccine 1: the clinic has 1 vaccine1 and 0 vaccine2.
Nurse 1 (pid=2414) has brought vaccine 2: the clinic has 1 vaccine1 and 1 vaccine2.
Nurse 1 (pid=2414) has brought vaccine 2: the clinic has 1 vaccine1 and 2 vaccine2.
Vaccinator 1 (pid=2417) put vaccines in his bag
Vaccinator 1 (pid=2417) is inviting citizen pid=2419 to the clinic.
Nurse 2 (pid=2415) has brought vaccine 1: the clinic has 1 vaccine1 and 1 vaccine2.
Citizen 1 (pid=2419) is vaccinated for 1 times: the clinic has 1 vaccine1 and 1 vaccine2.
Vaccinator 2 (pid=2418) put vaccines in his bag
Vaccinator 2 (pid=2418) is inviting citizen pid=2419 to the clinic.
Citizen 1 (pid=2419) is vaccinated for 2 times: the clinic has 1 vaccine1 and 0 vaccine2.
Nurse 3 (pid=2416) has brought vaccine 1: the clinic has 1 vaccine1 and 0 vaccine2.
Nurse 3 (pid=2416) has brought vaccine 2: the clinic has 1 vaccine1 and 1 vaccine2.
Nurse 3 (pid=2416) has brought vaccine 1: the clinic has 2 vaccine1 and 1 vaccine2.
Vaccinator 1 (pid=2417) put vaccines in his bag
Vaccinator 1 (pid=2417) is inviting citizen pid=2419 to the clinic.
Nurse 3 (pid=2416) has brought vaccine 2: the clinic has 1 vaccine1 and 1 vaccine2.
Citizen 1 (pid=2419) is vaccinated for 3 times: the clinic has 1 vaccine1 and 1 vaccine2.
Nurse 3 (pid=2416) has brought vaccine 1: the clinic has 2 vaccine1 and 1 vaccine2.
Nurse 1 (pid=2414) has brought vaccine 2: the clinic has 2 vaccine1 and 2 vaccine2.
Nurse 1 (pid=2414) has brought vaccine 1: the clinic has 3 vaccine1 and 2 vaccine2.
Nurse 2 (pid=2415) has brought vaccine 1: the clinic has 4 vaccine1 and 2 vaccine2.
Nurse 2 (pid=2415) has brought vaccine 2: the clinic has 4 vaccine1 and 3 vaccine2.
Nurse 1 (pid=2414) has brought vaccine 1: the clinic has 5 vaccine1 and 3 vaccine2.
Nurse 1 (pid=2414) has brought vaccine 1: the clinic has 6 vaccine1 and 3 vaccine2.
Nurse 1 (pid=2414) has brought vaccine 2: the clinic has 6 vaccine1 and 4 vaccine2.
Nurse 1 (pid=2414) has brought vaccine 1: the clinic has 7 vaccine1 and 4 vaccine2.
Nurse 3 (pid=2416) has brought vaccine 2: the clinic has 7 vaccine1 and 5 vaccine2.
Nurse 3 (pid=2416) has brought vaccine 2: the clinic has 7 vaccine1 and 6 vaccine2.
Vaccinator 1 (pid=2417) put vaccines in his bag
Nurse 2 (pid=2415) has brought vaccine 2: the clinic has 6 vaccine1 and 6 vaccine2.
Vaccinator 1 (pid=2417) is inviting citizen pid=2419 to the clinic.
Vaccinator 1 (pid=2417) put vaccines in his bag
Vaccinator 1 (pid=2417) is inviting citizen pid=2420 to the clinic.
Nurse 2 (pid=2415) has brought vaccine 1: the clinic has 6 vaccine1 and 5 vaccine2.
Vaccinator 2 (pid=2418) put vaccines in his bag
Nurse 2 (pid=2415) has brought vaccine 2: the clinic has 5 vaccine1 and 5 vaccine2.
Vaccinator 1 (pid=2417) put vaccines in his bag
Nurse 3 (pid=2416) has brought vaccine 1: the clinic has 5 vaccine1 and 4 vaccine2.
Citizen 2 (pid=2420) is vaccinated for 1 times: the clinic has 5 vaccine1 and 5 vaccine2.
Nurse 2 (pid=2415) has brought vaccine 1: the clinic has 6 vaccine1 and 4 vaccine2.
Nurses have carried all vaccines to the buffer, terminating.
Nurse 1 (pid=2414) has brought vaccine 2: the clinic has 6 vaccine1 and 5 vaccine2.
Citizen 1 (pid=2419) is vaccinated for 4 times: the clinic has 6 vaccine1 and 5 vaccine2.
Citizen 1 (pid=2419) is leaving. Remaining 2 citizens.
Vaccinator 1 (pid=2417) is inviting citizen pid=2420 to the clinic.
Vaccinator 1 (pid=2417) put vaccines in his bag
Vaccinator 1 (pid=2417) is inviting citizen pid=2420 to the clinic.
```

```
Vaccinator 1 (pid=2417) put vaccines in his bag
Vaccinator 1 (pid=2417) is inviting citizen pid=2420 to the clinic.
Vaccinator 1 (pid=2417) put vaccines in his bag
Vaccinator 1 (pid=2417) is inviting citizen pid=2421 to the clinic.
Vaccinator 1 (pid=2417) put vaccines in his bag
Vaccinator 1 (pid=2417) is inviting citizen pid=2421 to the clinic.
------> Giving the kill signal to vaccinator process i am 1
Vaccinator 1 (pid=2417) put vaccines in his bag
Vaccinator 1 (pid=2417) is inviting citizen pid=2421 to the clinic.
Vaccinator 1 vaccinated 20 times
Citizen 3 (pid=2421) is vaccinated for 3 times: the clinic has 1 vaccine1 and 0 vaccine2.
Citizen 3 (pid=2421) is vaccinated for 3 times: the clinic has 1 vaccine1 and 0 vaccine2.
Nurse 3 (pid=2416) has brought vaccine 2: the clinic has 1 vaccine1 and 1 vaccine2.
Citizen 2 (pid=2420) is vaccinated for 4 times: the clinic has 1 vaccine1 and 1 vaccine2.
Citizen 2 (pid=2420) is leaving. Remaining 1 citizens.
Vaccinator 2 (pid=2418) is inviting citizen pid=2421 to the clinic.
Vaccinator 2 vaccinated 4 times
Citizen 3 (pid=2421) is vaccinated for 4 times: the clinic has 1 vaccine1 and 1 vaccine2.
Citizen 3 (pid=2421) is leaving. Remaining 0 citizens.
All citizens have been vaccinated.
The clinic is now closed. Stay healthy.
cse312@ubuntu:~/Desktop/456/1801042103$
```

2)

```
cse312@ubuntu:~/Desktop/lastdance/project$ ./program -n 3 -v 2 -c 3 -b 100 -t 2 -i example_file.txt
Welcome to the GTU344 clinic. Number of citizens to vaccinate c=3 with t=2 doses.
Nurse 1 (pid=20342) has brought vaccine 1: the clinic has 1 vaccine1 and 0 vaccine2.
Nurse 1 (pid=20342) has brought vaccine 2: the clinic has 1 vaccine1 and 1 vaccine2.
Nurse 1 (pid=20342) has brought vaccine 1: the clinic has 2 vaccine1 and 1 vaccine2.
Nurse 1 (pid=20342) has brought vaccine 2: the clinic has 2 vaccine1 and 2 vaccine2.
Nurse 1 (pid=20342) has brought vaccine 1: the clinic has 3 vaccine1 and 2 vaccine2.
Nurse 1 (pid=20342) has brought vaccine 2: the clinic has 3 vaccine1 and 3 vaccine2.
Nurse 1 (pid=20342) has brought vaccine 1: the clinic has 4 vaccine1 and 3 vaccine2.
Nurse 1 (pid=20342) has brought vaccine 2: the clinic has 4 vaccine1 and 4 vaccine2.
Nurse 1 (pid=20342) has brought vaccine 1: the clinic has 5 vaccine1 and 4 vaccine2.
Nurse 1 (pid=20342) has brought vaccine 2: the clinic has 5 vaccine1 and 5 vaccine2.
Nurse 1 (pid=20342) has brought vaccine 1: the clinic has 6 vaccine1 and 5 vaccine2.
Nurse 1 (pid=20342) has brought vaccine 2: the clinic has 6 vaccine1 and 6 vaccine2.
Nurses have carried all vaccines to the buffer, terminating.
Nurse 2 (pid=20343) has brought vaccine 1: the clinic has 7 vaccine1 and 6 vaccine2.
Nurse 3 (pid=20344) has brought vaccine 2: the clinic has 7 vaccine1 and 7 vaccine2.
Vaccinator 1 (pid=20345) put vaccines in his bag
Vaccinator 2 (pid=20346) put vaccines in his bag
Vaccinator 1 (pid=20345) is inviting citizen pid=20349 to the clinic.
Vaccinator 1 (pid=20345) put vaccines in his bag
Vaccinator 1 (pid=20345) is inviting citizen pid=20349 to the clinic.
Vaccinator 1 (pid=20345) put vaccines in his bag
Citizen 3 (pid=20349) is vaccinated for 2 times: the clinic has 3 vaccine1 and 3 vaccine2.
Citizen 3 (pid=20349) is leaving. Remaining 2 citizens.
Vaccinator 1 (pid=20345) is inviting citizen pid=20348 to the clinic.
Vaccinator 1 (pid=20345) put vaccines in his bag
Vaccinator 1 (pid=20345) is inviting citizen pid=20348 to the clinic.
------> Giving the kill signal to vaccinator process i am 1
Vaccinator 1 (pid=20345) put vaccines in his bag
Citizen 2 (pid=20348) is vaccinated for 2 times: the clinic has 1 vaccine1 and 1 vaccine2.
Citizen 2 (pid=20348) is leaving. Remaining 1 citizens.
Vaccinator 1 (pid=20345) is inviting citizen pid=20347 to the clinic.
Vaccinator 1 vaccinated 10 times
Vaccinator 2 (pid=20346) is inviting citizen pid=20347 to the clinic.
Vaccinator 2 vaccinated 2 times
Citizen 1 (pid=20347) is vaccinated for 2 times: the clinic has 1 vaccine1 and 1 vaccine2.
Citizen 1 (pid=20347) is leaving. Remaining 0 citizens.
All citizens have been vaccinated.
The clinic is now closed. Stay healthy.
```

3)

```
cse312@ubuntu:~/Desktop/lastdance/project$ ./program -n 3 -v 2 -c 3 -b 9 -t 2 -i example_file.t
Welcome to the GTU344 clinic. Number of citizens to vaccinate c=3 with t=2 doses.
Nurse 1 (pid=20174) has brought vaccine 1: the clinic has 1 vaccine1 and 0 vaccine2.
Nurse 3 (pid=20176) has brought vaccine 2: the clinic has 1 vaccine1 and 1 vaccine2.
Nurse 3 (pid=20176) has brought vaccine 1: the clinic has 2 vaccine1 and 1 vaccine2.
Nurse 3 (pid=20176) has brought vaccine 2: the clinic has 2 vaccine1 and 2 vaccine2.
Nurse 3 (pid=20176) has brought vaccine 1: the clinic has 3 vaccine1 and 2 vaccine2.
Nurse 3 (pid=20176) has brought vaccine 2: the clinic has 3 vaccine1 and 3 vaccine2.
Nurse 3 (pid=20176) has brought vaccine 1: the clinic has 4 vaccine1 and 3 vaccine2.
Nurse 3 (pid=20176) has brought vaccine 2: the clinic has 4 vaccine1 and 4 vaccine2.
Nurse 3 (pid=20176) has brought vaccine 1: the clinic has 5 vaccine1 and 4 vaccine2.
Vaccinator 1 (pid=20177) put vaccines in his bag
Nurse 3 (pid=20176) has brought vaccine 2: the clinic has 4 vaccine1 and 4 vaccine2.
Vaccinator 2 (pid=20178) put vaccines in his bag
Nurse 1 (pid=20174) has brought vaccine 1: the clinic has 4 vaccine1 and 3 vaccine2.
Vaccinator 1 (pid=20177) is inviting citizen pid=20179 to the clinic.
Vaccinator 1 (pid=20177) put vaccines in his bag
Nurse 3 (pid=20176) has brought vaccine 2: the clinic has 3 vaccine1 and 3 vaccine2.
Nurses have carried all vaccines to the buffer, terminating.
Citizen 1 (pid=20179) is vaccinated for 1 times: the clinic has 3 vaccine1 and 3 vaccine2.
Vaccinator 1 (pid=20177) is inviting citizen pid=20179 to the clinic.
Vaccinator 1 (pid=20177) put vaccines in his bag
Nurse 1 (pid=20174) has brought vaccine 1: the clinic has 3 vaccine1 and 2 vaccine2.
Vaccinator 1 (pid=20177) is inviting citizen pid=20180 to the clinic.
Vaccinator 1 (pid=20177) put vaccines in his bag
Vaccinator 1 (pid=20177) is inviting citizen pid=20180 to the clinic.
------> Giving the kill signal to vaccinator process i am 1
Vaccinator 1 (pid=20177) put vaccines in his bag
Vaccinator 1 (pid=20177) is inviting citizen pid=20181 to the clinic.
Vaccinator 1 vaccinated 10 times
Nurse 2 (pid=20175) has brought vaccine 2: the clinic has 1 vaccine1 and 1 vaccine2.
Vaccinator 2 (pid=20178) is inviting citizen pid=20181 to the clinic.
Vaccinator 2 vaccinated 2 times
Citizen 2 (pid=20180) is vaccinated for 2 times: the clinic has 1 vaccine1 and 1 vaccine2.
Citizen 2 (pid=20180) is leaving. Remaining 0 citizens.
Citizen 1 (pid=20179) is vaccinated for 2 times: the clinic has 1 vaccine1 and 1 vaccine2.
Citizen 1 (pid=20179) is leaving. Remaining 0 citizens.
Citizen 3 (pid=20181) is vaccinated for 2 times: the clinic has 1 vaccine1 and 1 vaccine2.
Citizen 3 (pid=20181) is leaving. Remaining 0 citizens.
All citizens have been vaccinated.
The clinic is now closed. Stay healthy.
cse312@ubuntu:~/Desktop/lastdance/project$
```

4)

```
cse312@ubuntu:~/Desktop/lastdance/project$ ./program -n 3 -v 2 -c 3 -b 11 -t 3 -i example_file.txt
Welcome to the GTU344 clinic. Number of citizens to vaccinate c=3 with t=3 doses.
Nurse 1 (pid=20164) has brought vaccine 1: the clinic has 1 vaccine1 and 0 vaccine2.
Nurse 3 (pid=20166) has brought vaccine 2: the clinic has 1 vaccine1 and 1 vaccine2.
Nurse 3 (pid=20166) has brought vaccine 1: the clinic has 2 vaccine1 and 1 vaccine2.
Nurse 3 (pid=20166) has brought vaccine 2: the clinic has 2 vaccine1 and 2 vaccine2.
Nurse 3 (pid=20166) has brought vaccine 1: the clinic has 3 vaccine1 and 2 vaccine2.
Nurse 3 (pid=20166) has brought vaccine 2: the clinic has 3 vaccine1 and 3 vaccine2.
Nurse 3 (pid=20166) has brought vaccine 1: the clinic has 4 vaccine1 and 3 vaccine2.
Nurse 3 (pid=20166) has brought vaccine 2: the clinic has 4 vaccine1 and 4 vaccine2.
Nurse 3 (pid=20166) has brought vaccine 1: the clinic has 5 vaccine1 and 4 vaccine2.
Nurse 3 (pid=20166) has brought vaccine 2: the clinic has 5 vaccine1 and 5 vaccine2.
Nurse 3 (pid=20166) has brought vaccine 1: the clinic has 6 vaccine1 and 5 vaccine2.
Vaccinator 1 (pid=20167) put vaccines in his bag
Nurse 3 (pid=20166) has brought vaccine 2: the clinic has 5 vaccine1 and 5 vaccine2.
Vaccinator 2 (pid=20168) put vaccines in his bag
Nurse 1 (pid=20164) has brought vaccine 1: the clinic has 5 vaccine1 and 4 vaccine2.
Vaccinator 2 (pid=20168) is inviting citizen pid=20169 to the clinic.
Vaccinator 2 (pid=20168) put vaccines in his bag
Nurse 3 (pid=20166) has brought vaccine 2: the clinic has 4 vaccine1 and 4 vaccine2.
Citizen 1 (pid=20169) is vaccinated for 1 times: the clinic has 4 vaccine1 and 4 vaccine2.
Vaccinator 2 (pid=20168) is inviting citizen pid=20169 to the clinic.
Vaccinator 2 (pid=20168) put vaccines in his bag
Nurse 1 (pid=20164) has brought vaccine 1: the clinic has 4 vaccine1 and 3 vaccine2.
Citizen 1 (pid=20169) is vaccinated for 2 times: the clinic has 4 vaccine1 and 3 vaccine2.
Vaccinator 1 (pid=20167) is inviting citizen pid=20169 to the clinic.
Vaccinator 1 (pid=20167) put vaccines in his bag
Nurse 3 (pid=20166) has brought vaccine 2: the clinic has 3 vaccine1 and 3 vaccine2.
Citizen 1 (pid=20169) is vaccinated for 3 times: the clinic has 3 vaccine1 and 3 vaccine2.
Citizen 1 (pid=20169) is leaving. Remaining 2 citizens.
Vaccinator 2 (pid=20168) is inviting citizen pid=20170 to the clinic.
Vaccinator 2 (pid=20168) put vaccines in his bag
Nurse 1 (pid=20164) has brought vaccine 1: the clinic has 3 vaccine1 and 2 vaccine2.
Citizen 2 (pid=20170) is vaccinated for 1 times: the clinic has 2 vaccine1 and 2 vaccine2.
Vaccinator 1 (pid=20167) is inviting citizen pid=20170 to the clinic.
Vaccinator 1 (pid=20167) put vaccines in his bag
Nurse 3 (pid=20166) has brought vaccine 2: the clinic has 2 vaccine1 and 2 vaccine2.
Nurses have carried all vaccines to the buffer, terminating.
Citizen 2 (pid=20170) is vaccinated for 2 times: the clinic has 2 vaccine1 and 2 vaccine2.
Vaccinator 1 (pid=20167) is inviting citizen pid=20170 to the clinic.
Vaccinator 1 (pid=20167) put vaccines in his bag
Vaccinator 1 (pid=20167) is inviting citizen pid=20171 to the clinic.
------> Giving the kill signal to vaccinator process i am 1
Vaccinator 1 (pid=20167) put vaccines in his bag
Nurse 1 (pid=20164) has brought vaccine 2: the clinic has 0 vaccine1 and 1 vaccine2.
Nurse 2 (pid=20165) has brought vaccine 1: the clinic has 1 vaccine1 and 1 vaccine2.
Vaccinator 1 (pid=20167) is inviting citizen pid=20171 to the clinic.
Vaccinator 1 vaccinated 10 times
Vaccinator 2 (pid=20168) is inviting citizen pid=20171 to the clinic.
Vaccinator 2 vaccinated 8 times
Citizen 3 (pid=20171) is vaccinated for 3 times: the clinic has 1 vaccine1 and 1 vaccine2.
Citizen 3 (pid=20171) is leaving. Remaining 0 citizens.
Citizen 2 (pid=20170) is vaccinated for 3 times: the clinic has 1 vaccine1 and 1 vaccine2.
Citizen 2 (pid=20170) is leaving. Remaining 0 citizens.
All citizens have been vaccinated.
The clinic is now closed. Stay healthy.
cse312@ubuntu:~/Desktop/lastdance/project$
```