CSE 344
SYSTEM PROGRAMMING
MIDTERM REPORT

OĞUZHAN SEZGİN
1801042005

GENERAL PROCESS

The program is running as desired. The general flow diagram is as follows;
First of all, the clinic shared memory, where vaccinators will be exchanged vaccine between nurse then vaccine room shared memory, which vaccinators will call citizens to vaccinate, are created.
More citizens are created. A pipe is created for each citizen created to communicate with the vaccinator. After the citizens are created, they wait to read the information coming to their pipes. Then vaccinators are created. Pipes are created so that vaccinators can also get feedback from the citizen. Finally, nurses are created. After the nurses are created, they read the vaccines from the file and write to the buffer. When any of the vaccinators understands that there is a vaccine couple in the buffer (1st and 2nd vaccine), he receives the vaccine and calls the available oldest citizen. He makes the call by writing his own id on the pipe of the relevant citizen and waits to read his pipe for information from the citizen. The relevant citizen who reads this indicates that it is vaccine and writes to the pipe of the vaccinator that is waiting for him and allows him to continue the process. This process continues until all vaccines are over and all citizens are vaccinated.

Problem 1

Status of nurses reading the file at the same time. Nurses were prevented from reading the file at the same time in two ways. First, each nurse must enter the critical section to read the file, and there is only one process here. Second, each nurse locks the file before reading it and unlocks it after reading it.

Problem 2

Nurses writing to the buffer at the same time or vaccinators trying to read the buffer simultaneously while the nurses are writing.This situation has been solved with the producer consumer algorithm. Nurses initially wait and pick up the "empty" semaphore whose value is the buffer size. Then they wait for the "mutex" semaphore to write to the buffer, and when they get it, they write it to the buffer. After they write, they check to see if there is a new vaccinator pair that the vaccinator can take, and if a new vaccine pair is formed after the vaccine they brought, they increase the "full" semaphore and thus allow one of the vaccinators to receive the vaccine. Any vaccinator waiting for "full" semaphore flour will enter the critical section and get the vaccine.

Problem 3

How cooks citizens call. A shared memory is created (cit_vac_num), in which the pids of the citizens and the remaining number of vaccines are kept.Vaccinators looks at this memory and they call the appropriate citizen. To access this memory, the vaccinators must be in the critical section and this is possible with the semaphores in the "VacRoom" shared memory."Mutex" and "full" semaphore s are used for the critical section. The initial value of "full" semaphore is the number of citizens. The reason for this is to make sure that the appropriate citizen is called for vaccination, otherwise the vaccinator will wait for this semaphore.If all citizens are getting vaccinated and other

vaccinators have the vaccine, they should wait for one or more of the citizens to end the vaccination process.

Problem 4

How do the processes end? If one of the nurses reads something different from the character '1' or '2' except for the '\ n' character from the file, it realizes that it has reached the end of the file and makes the "is_end" value in clinic shared memory 1 and increases the semohore values, decreases the value of "rem_nur_num", which indicates the remaining nurse value, by 1 and exits and ends the loop. The nurses who come after that check the "is_end" value and if it is 1, they realize that the file is not vaccinated and terminates itself in the same way.
When vaccianators enter the critical section, they check if there is vaccine in the buffer and whether the "is_end" value is 1. If there is no vaccine in the buffer and "is_end" is 1, that is, if the vaccine will not come yet, it increases the sempahores to allow the next ones to enter the critical section and terminate itself.
Citizens wait for the number of cycles they need to be vaccinated and end when they reach that number.

Problem 5

Where to keep the number of doses made by vaccinators. These numbers are kept in shared memory called "vac_dose_num". Here, the pid and the number of vaccines made by each vaccinator ('1' and '2' vaccine pair is counted as a single dose) respectively. Vaccinatos increase the number in their index by 1 when they are in the critical section when they make each vaccine pair. At the end, the parent process prints these numbers.

Problem 6

Citizens printing out the number of vaccines in the clinic while being vaccinated. This situation is not solved exactly as desired, because when the citizens are called by vaccinators, they take the "mutex" in the "clinic" and save the instant vaccine values to print, but other processes change this value until they come to the printing part. In other words, the citizen prints the number of vaccines when they are first called, but these numbers may change until they come to the printing section.

Problem 7 - Bonus Part

Calling the oldest citizen. Citizens' pids and remaining number of vaccines are kept in "cit_vac_num" shared memory, as stated before. The first index pid the second index shows the number of vaccines remaining ([pid1, rem1, pid2, rem2, ...]). The first pid is the first to be created and therefore the oldest. When a process calls a citizen, it starts searching from the beginning and when rem is in the first index with a value greater than 0, it calls. It sets the rem value of the citizen -1 so that other processes cannot invoke the citizen while vaccinated. Then, he writes his own id on the pipe of the chosen citizen and waits for the reply. Meanwhile, other processes are able to call the next oldest and most suitable process. The relevant citizen he wrote on the vaccinator's pipe wakes up and writes the number of vaccines on the screen and writes how many vaccines are left on the "cit_vac_num" memory and writes his id to continue the pipe of the vaccinator he reads. If the citizen has

had his last vaccine, the writer writes pipe '1' and writes '0' if he is not his last vaccine, and the vaccinator reading this increases his "full" semaphore according to this message and continues to receive another vaccine. Thus, the oldest and most suitable citizen is called and other vaccinators are not prevented.

Requirements

There should be exactly as much vaccine as needed in the file. If it is missing or excessive, it will cause a deadlock.