May 03, 2021

Gebze Technical University
Computer Engineering
CSE344 - Spring 2021
MIDTERM

Nevzat Seferoglu
171044024

1 Problem Definition

The problem was applying classical **producer-consumer** pattern. There is a clinic. There are nurses to bring vaccines to clinic. There are vaccinators which vaccinates the citizens which are wating for to be vaccinated for certain amount of doses. The paradign was clear. Nurses correspond to producer, vaccinator correspond to consumer. Each citizen must be vaccinated at least one pair dose which means that vaccinating for two times (specified before).

2 Solution

I exploited certain type of structures in that project. These are **unnamed semaphores**, **shared memory**. Each nurses, vaccinators and citizens are actual processes and need to communicate each other for synchronization and also sharing some resource all together.

2.1 Design and Solutions

2.1.1 Shared Memory and Buffer Structure

Shared memory is a well-known structure for communication between processes. There are common resources in that sturucture and each has to be altered synchronously. In that moment, semaphores step in. Semaphores are our main gun for synchronization between processes. I used **unnamed semaphores** intentionally. Unnamed semaphores are either private, inherited through fork(), or are protected by access protections of the regular file where they are allocated and mapped.

```
typedef struct Buffer {
    sig_atomic_t sigint;    /* Reaching signal notification among processes. */
    int totalAppliedDose;
    int deadCitizen;
    long currCitizen;
    int nurseTermination;
    long fNurse, fVaccinator, fCitizen;
    int firstDose;
    int secDose;
    int fd;
    sem_t empty;        /* Synchronization purpose. (Producer get locks) */
    sem_t full;         /* Synchronization purpose. (Consumer get locks) */
    sem_t m;            /* Shared memory protection. */
    sem_t mConsumer;    /* Protecting for not to be blocked while consuming */
    sem_t mCitizen;     /* Citizen information sharing with vaccinator */
    sem_t mSafe;        /* Citizen information sharing with vaccinator */
} Buffer;
```

Shared memory is diretly returned to that structure. Every single variable that is kept in this structure are being used somewhere in code by more than two unrelated or related processes. The design makes me more comfortable to reach out some resources while multiprocessing.

2.1.2 Nurse Design (the Producer)

Nurses bring the vaccines one by one from the clinic until the vaccines are over. We were taking doses from file and each nurse needs to reach file synchronously. For that purpose I use inherited file descriptor which is opened in parent process. Each nurse takes the vaccines by protecting with mutex, bring them to clinic. The file could also include newline character so I designed a structure that ignores unrelated character which is taken from file.

```
if (gfile->isEof) {
    ++gbuffer->nurseTermination;
    if (gbuffer->nurseTermination == numOfNurses)
        fprintf(stdout, "Nurses have carried all vaccines to the buffer, terminating.\n");
    x_sem_post(&(gbuffer->m));
    x_sem_post(&(gbuffer->full));
    break;
}
if (gfile->c != '1' && gfile->c != '2') {
    x_sem_post(&(gbuffer->m));
    x_sem_post(&(gbuffer->empty));
    continue;
}
/* Taken does is recorded to shared memory to be used by vaccinator. */
if (gfile->c == '1') {
```

```
    ++(gbuffer->firstDose);
    fprintf(stdout, "Nurse %2d (pid=%ld) has brought "
    "vacinne 1: the clinic has %2d vaccine1 and %2d vacinne 2.\n",
    ((int)(((long)getpid())-(gbuffer->fNurse)+1)), (long)getpid(),
    gbuffer->firstDose, gbuffer->secDose);
}
else if (gfile->c == '2') {
    ++(gbuffer->secDose);
    fprintf(stdout, "Nurse %2d (pid=%ld) has brought "
    "vacinne 2: the clinic has %2d vaccine1 and %2d vacinne 2.\n",
    ((int)(((long)getpid())-(gbuffer->fNurse)+1)), (long)getpid(),
    gbuffer->firstDose, gbuffer->secDose);
}
```

Nurses are actual producer. So I exploited producer solution for that type.

```
x_sem_wait(&(gbuffer->empty));
if (gbuffer->sigint == 1) {
    x_sem_post(&(gbuffer->empty));
    exit(EXIT_FAILURE);
}
x_sem_wait(&(gbuffer->m));
/* Nurse Operation */
x_sem_post(&(gbuffer->m));
x_sem_post(&(gbuffer->full));
```

Tricky Fragment
The tricky part for the nurse is designing a structure to be ignored unrelated file input (without ruining general semaphore mechanism). And also all nurses need to be exited when at least one nurse encountered with EOF.

2.1.3 Vaccinator Design (the Consumer)

Vaccinators are our main consumer. They do more than one job for vaccination. Their first job is vaccination, second job is inviting citizen to vaccinate. They consumes vaccines which are located in common buffer stucture and if there are enough doses for vaccination, they simple invite citizen by using semaphore by using contractual mechanism with citizen processes. Other than these.

```
/* Guarded by other semaphore */
if (!isThereAnyDose()) {
    x_sem_post(&(gbuffer->m));
    x_sem_post(&(gbuffer->full));
    x_sem_post(&(gbuffer->full));
    x_sem_post(&(gbuffer->mConsumer));
    continue;
}
/* Guarded by other semaphore */
```

It is checking the dose amount. If the doses do not enough for vaccination. It will wait for existence of enough vaccines. And then call the citizen by using 2 semaphores. These two semaphores are assure that each citizen will take vaccines and record his id in shared buffer structure one by one without ruining each other.

```
/* Guarded by other semaphore */
x_sem_wait(&(gbuffer->mCitizen));
x_sem_wait(&(gbuffer->mSafe));
--(gbuffer->firstDose);
--(gbuffer->secDose);
++appliedDoses;
++(gbuffer->totalAppliedDose);
fprintf(stdout, "Vaccinator %d (pid=%ld) is inviting citizen pid=%ld to the clinic\n",
    ((int)((long)getpid()-(gbuffer->fVaccinator)+1)),
    ((long)getpid()),
    (gbuffer->currCitizen));
x_sem_post(&(gbuffer->mSafe));
/* Guarded by other semaphore */
```

When vaccinator is ready, he waits (invitation) for a citizen (CPU choice) to vaccinate. Alter the some value in comman buffer. Citizens are also using these two semaphores to record their PID in an instant structure for giving them to vaccinator. Vaccinators are actual consumer. But while producer is producing an item. In this project, consumer needs to consume two items (if exists).

```
/* Warning: This is not an actual code. It is just general overview of solution. */
x_sem_wait(&(gbuffer->mConsumer));
x_sem_wait(&(gbuffer->full));
x_sem_wait(&(gbuffer->full));
x_sem_wait(&(gbuffer->m));
/* Vaccinator Operation */
x_sem_post(&(gbuffer->m));
x_sem_post(&(gbuffer->empty));
x_sem_post(&(gbuffer->empty));
x_sem_post(&(gbuffer->mConsumer));
```

Tricky Fragment

The tricky part for the vaccinator is consuming two doses by waiting nurses in a synchronize way. I did not use extra semaphores for each dose. I needed to design a structure which consume doses while only these two doses exists for both type (1,2) at the same time at least one. Otherwise it needs to skip its own turn to be filled with that both doses (1,2). If all doses are applied to citizens it needs to be exited with precise message.

```
if (gbuffer->totalAppliedDose == numOfCitizens*numOfDose) {
    fprintf(stdout, "Vaccinator %d (pid=%ld) vaccinated %d doses. ",
    ((int)((long)getpid()-(gbuffer->fVaccinator)+1)),
    ((long)getpid()),
    appliedDoses);
    x_sem_post(&(gbuffer->m));
    x_sem_post(&(gbuffer->full));
    x_sem_post(&(gbuffer->full));
    x_sem_post(&(gbuffer->mConsumer));
    break;
}
```

2.1.4 Citizen Design (Intermediary Processes)

Citizens are processes which are invited by vaccinators. In my design, they are constantly ready for to be vaccinated. If the vaccinator invite the citizen with certain semaphores, they have some kind of **protocol** which collaborate on not to be interleaved by another citizen and also by another vaccinator.

The Protocol

Actually, the citizen is trying to be vaccinated constantly but will not be vaccinated until the vaccinator is ready to vaccinate. By using that protocol, any others cannot attempt to be vaccinated by current processing vaccinator until he release protocol semaphores.

Note: This protocol causes an output might seen not as expected. CPU choices also affect output. **But all citizens vaccinated by certain vaccinator and that vaccinator invites correct citizens with correct PID in correct time.**

```
/* Warning: This is not an actual code. It is just general overview of solution. */
x_sem_wait(&(gbuffer->mSafe));
x_sem_post(&(gbuffer->mCitizen));
gbuffer->currCitizen = (long)getpid();
fprintf(stdout, "Citizen %d (pid=%ld) is vaccinated for the %dst time:"
" the clinic has %d vaccine1 and %d vaccine2\n",
((int)(((long)getpid())-(gbuffer->fCitizen)+1)),
(long)(getpid()),
(numOfDose-i+1),
gbuffer->firstDose,
gbuffer->secDose);
x_sem_post(&(gbuffer->mSafe));
```

Counting Vaccinated Citizen

```
/* Warning: This is not an actual code. It is just general overview of solution. */
x_sem_wait(&(gbuffer->mSafe));
++(gbuffer->deadCitizen);
if (gbuffer->deadCitizen == numOfCitizens) {
    fprintf(stdout, "All citizens have been vaccinated.\n");
}
else {
    fprintf(stdout, "citizen is leaving."
    " Remaining citizens to vaccinate: %d\n",(numOfCitizens-(gbuffer->deadCitizen)));
}
x_sem_post(&(gbuffer->mSafe));
```

2.1.5 CTRL+C Handling

In parent, there is an handler for entire program. I also put variable to detect SIGINT across processes that are member of shared memory. When one process encountered with SIGINT, any other processes are being informed atomically. They make their own clean up then exited immediately to parent process.

```
/* Warning: This is not an actual code. It is just general overview of solution. */
void handler_SIGINT() {
    /* Detection */
    gbuffer->sigint = 1;
}
memset(&sa_SIGINT, 0, sizeof(sa_SIGINT));
sa_SIGINT.sa_handler = handler_SIGINT;
x_sigaction(SIGINT, &sa_SIGINT, NULL);

/* Example escaping from vaccinator */
if (gbuffer->sigint == 1) {
    x_sem_post(&(gbuffer->full));
    x_sem_post(&(gbuffer->full));
    x_sem_post(&(gbuffer->mConsumer));
    exit(EXIT_FAILURE);
}
```

3 Which requirements I achieved ?

I tested my project with several scripts. It works properly.

```
ubuntu@ubuntu1604:~/Desktop/Midterm$ ./program -n 2 -v 2 -c 5 -b 21 -t 4 -i input.txt
Welcome to the GTU344 clinic. Number of citizens to vaccinate c=5 with t=4 doses.
Nurse  1 (pid=5695) has brought vacinne 1: the clinic has  1 vaccinel and  0 vacinne 2.
Nurse  1 (pid=5695) has brought vacinne 1: the clinic has  2 vaccinel and  0 vacinne 2.
Nurse  1 (pid=5695) has brought vacinne 1: the clinic has  3 vaccinel and  0 vacinne 2.
Nurse  1 (pid=5695) has brought vacinne 1: the clinic has  4 vaccinel and  0 vacinne 2.
Nurse  1 (pid=5695) has brought vacinne 1: the clinic has  5 vaccinel and  0 vacinne 2.
Nurse  1 (pid=5695) has brought vacinne 1: the clinic has  6 vaccinel and  0 vacinne 2.
Nurse  1 (pid=5695) has brought vacinne 1: the clinic has  7 vaccinel and  0 vacinne 2.
Nurse  1 (pid=5695) has brought vacinne 1: the clinic has  8 vaccinel and  0 vacinne 2.
Nurse  1 (pid=5695) has brought vacinne 1: the clinic has  9 vaccinel and  0 vacinne 2.
Nurse  1 (pid=5695) has brought vacinne 1: the clinic has 10 vaccinel and  0 vacinne 2.
Nurse  2 (pid=5696) has brought vacinne 1: the clinic has 11 vaccinel and  0 vacinne 2.
Nurse  2 (pid=5696) has brought vacinne 1: the clinic has 12 vaccinel and  0 vacinne 2.
Nurse  2 (pid=5696) has brought vacinne 1: the clinic has 13 vaccinel and  0 vacinne 2.
Nurse  2 (pid=5696) has brought vacinne 1: the clinic has 14 vaccinel and  0 vacinne 2.
Nurse  2 (pid=5696) has brought vacinne 1: the clinic has 15 vaccinel and  0 vacinne 2.
Nurse  2 (pid=5696) has brought vacinne 1: the clinic has 16 vaccinel and  0 vacinne 2.
Nurse  2 (pid=5696) has brought vacinne 1: the clinic has 17 vaccinel and  0 vacinne 2.
Nurse  2 (pid=5696) has brought vacinne 1: the clinic has 18 vaccinel and  0 vacinne 2.
Nurse  2 (pid=5696) has brought vacinne 1: the clinic has 19 vaccinel and  0 vacinne 2.
Nurse  2 (pid=5696) has brought vacinne 1: the clinic has 20 vaccinel and  0 vacinne 2.
Nurse  1 (pid=5695) has brought vacinne 2: the clinic has 20 vaccinel and  1 vacinne 2.
Citizen 1 (pid=5699) is vaccinated for the 1st time: the clinic has 20 vaccinel and 0 vaccine2
Citizen 3 (pid=5701) is vaccinated for the 1st time: the clinic has 20 vaccinel and 1 vaccine2
Citizen 3 (pid=5701) is vaccinated for the 2st time: the clinic has 20 vaccinel and 1 vaccine2
Citizen 1 (pid=5699) is vaccinated for the 2st time: the clinic has 20 vaccinel and 1 vaccine2
Citizen 1 (pid=5699) is vaccinated for the 3st time: the clinic has 20 vaccinel and 1 vaccine2
Citizen 2 (pid=5700) is vaccinated for the 1st time: the clinic has 20 vaccinel and 1 vaccine2
Citizen 2 (pid=5700) is vaccinated for the 2st time: the clinic has 20 vaccinel and 1 vaccine2
Citizen 3 (pid=5701) is vaccinated for the 3st time: the clinic has 20 vaccinel and 1 vaccine2
Citizen 3 (pid=5701) is vaccinated for the 4st time: the clinic has 20 vaccinel and 1 vaccine2
Vaccinator 1 (pid=5697) is inviting citizen pid=5701 to the clinic
Citizen 2 (pid=5700) is vaccinated for the 3st time: the clinic has 19 vaccinel and 0 vaccine2
Nurse  2 (pid=5696) has brought vacinne 2: the clinic has 19 vaccinel and  1 vacinne 2.
Citizen 2 (pid=5700) is vaccinated for the 4st time: the clinic has 19 vaccinel and 1 vaccine2
Nurse  2 (pid=5696) has brought vacinne 2: the clinic has 19 vaccinel and  2 vacinne 2.
citizen is leaving. Remaining citizens to vaccinate: 4
citizen is leaving. Remaining citizens to vaccinate: 3
Citizen 5 (pid=5703) is vaccinated for the 1st time: the clinic has 19 vaccinel and 2 vaccine2
Citizen 4 (pid=5702) is vaccinated for the 1st time: the clinic has 19 vaccinel and 2 vaccine2
Vaccinator 1 (pid=5697) is inviting citizen pid=5702 to the clinic
Citizen 4 (pid=5702) is vaccinated for the 2st time: the clinic has 18 vaccinel and 1 vaccine2
Citizen 4 (pid=5702) is vaccinated for the 3st time: the clinic has 18 vaccinel and 1 vaccine2
Citizen 4 (pid=5702) is vaccinated for the 4st time: the clinic has 18 vaccinel and 1 vaccine2
citizen is leaving. Remaining citizens to vaccinate: 2
Vaccinator 1 (pid=5697) is inviting citizen pid=5702 to the clinic
Nurse  2 (pid=5696) has brought vacinne 2: the clinic has 17 vaccinel and  1 vacinne 2.
Citizen 5 (pid=5703) is vaccinated for the 2st time: the clinic has 17 vaccinel and 1 vaccine2
Nurse  2 (pid=5696) has brought vacinne 2: the clinic has 17 vaccinel and  2 vacinne 2.
Citizen 5 (pid=5703) is vaccinated for the 3st time: the clinic has 17 vaccinel and 2 vaccine2
Nurse  2 (pid=5696) has brought vacinne 2: the clinic has 17 vaccinel and  3 vacinne 2.
Citizen 5 (pid=5703) is vaccinated for the 4st time: the clinic has 17 vaccinel and 3 vaccine2
Nurse  2 (pid=5696) has brought vacinne 2: the clinic has 17 vaccinel and  4 vacinne 2.
citizen is leaving. Remaining citizens to vaccinate: 1
Vaccinator 1 (pid=5697) is inviting citizen pid=5703 to the clinic
Vaccinator 1 (pid=5697) is inviting citizen pid=5703 to the clinic
Vaccinator 1 (pid=5697) is inviting citizen pid=5703 to the clinic
Vaccinator 1 (pid=5697) is inviting citizen pid=5703 to the clinic
Nurse  2 (pid=5696) has brought vacinne 2: the clinic has 13 vaccinel and  1 vacinne 2.
Nurse  1 (pid=5695) has brought vacinne 2: the clinic has 13 vaccinel and  2 vacinne 2.
Nurse  1 (pid=5695) has brought vacinne 2: the clinic has 13 vaccinel and  3 vacinne 2.
Nurse  1 (pid=5695) has brought vacinne 2: the clinic has 13 vaccinel and  4 vacinne 2.
Nurse  1 (pid=5695) has brought vacinne 2: the clinic has 13 vaccinel and  5 vacinne 2.
Nurse  1 (pid=5695) has brought vacinne 2: the clinic has 13 vaccinel and  6 vacinne 2.
Nurse  1 (pid=5695) has brought vacinne 2: the clinic has 13 vaccinel and  7 vacinne 2.
Citizen 1 (pid=5699) is vaccinated for the 4st time: the clinic has 13 vaccinel and 7 vaccine2
All citizens have been vaccinated.
Nurse  2 (pid=5696) has brought vacinne 2: the clinic has 13 vaccinel and  8 vacinne 2.
Vaccinator 1 (pid=5697) is inviting citizen pid=5699 to the clinic
Vaccinator 1 (pid=5697) is inviting citizen pid=5699 to the clinic
Vaccinator 1 (pid=5697) is inviting citizen pid=5699 to the clinic
Vaccinator 1 (pid=5697) is inviting citizen pid=5699 to the clinic
Vaccinator 1 (pid=5697) is inviting citizen pid=5699 to the clinic
Vaccinator 1 (pid=5697) is inviting citizen pid=5699 to the clinic
Vaccinator 1 (pid=5697) is inviting citizen pid=5699 to the clinic
Vaccinator 1 (pid=5697) is inviting citizen pid=5699 to the clinic
Nurse  2 (pid=5696) has brought vacinne 2: the clinic has  5 vaccinel and  1 vacinne 2.
Nurse  2 (pid=5696) has brought vacinne 2: the clinic has  5 vaccinel and  2 vacinne 2.
Nurse  2 (pid=5696) has brought vacinne 2: the clinic has  5 vaccinel and  3 vacinne 2.
Nurse  1 (pid=5695) has brought vacinne 2: the clinic has  5 vaccinel and  4 vacinne 2.
Nurse  1 (pid=5695) has brought vacinne 2: the clinic has  5 vaccinel and  5 vacinne 2.
Vaccinator 2 (pid=5698) is inviting citizen pid=5699 to the clinic
Vaccinator 2 (pid=5698) is inviting citizen pid=5699 to the clinic
Vaccinator 2 (pid=5698) is inviting citizen pid=5699 to the clinic
Vaccinator 2 (pid=5698) is inviting citizen pid=5699 to the clinic
Vaccinator 2 (pid=5698) is inviting citizen pid=5699 to the clinic
Nurses have carried all vaccines to the buffer, terminating.
Vaccinator 2 (pid=5698) vaccinated 5 doses. Vaccinator 1 (pid=5697) vaccinated 15 doses. The clinic is now closed. Stay healthy.
ubuntu@ubuntu1604:~/Desktop/Midterm$
```

Figure 1