May 3, 2021

GTU Department of Computer Engineering
CSE344 - Spring 2021
Midterm Report

Akif Kartal
171044098

1 Problem Definition

The problem is to implement **producer-consumer** problem by simulating covid19 vaccination flow by using shared memory and posix semaphore.

2 Solution

The homework was finished as expected in homework pdf file.

2.1 Some Problems and Solutions

2.1.1 Semaphore Usage

In order to provide synchronization between processes I used **7 posix named semaphore**. Followings are my semaphores;

```
    /*create named semphores*/
    sem_mutex = sem_open("mutex344", O_CREAT, 0666, 1);
    if (sem_mutex == SEM_FAILED)
        errExit("sem_open error!");

    sem_vac = sem_open("wait_vaccinator", O_CREAT, 0666, 0);
    if (sem_vac == SEM_FAILED)
        errExit("sem_open error!");

    sem_cit = sem_open("wait_citizen", O_CREAT, 0666, 0);
    if (sem_cit == SEM_FAILED)
        errExit("sem_open error!");

    sem_full1 = sem_open("full344", O_CREAT, 0666, 0);
    if (sem_full1 == SEM_FAILED)
        errExit("sem_open error!");
    sem_empty = sem_open("empty344", O_CREAT, 0666, bufferSize);
    if (sem_empty == SEM_FAILED)
        errExit("sem_open error!");
    sem_full2 = sem_open("full3442", O_CREAT, 0666, 0);
    if (sem_full2 == SEM_FAILED)
        errExit("sem_open error!");
    sem_run = sem_open("whorunfirst", O_CREAT, 0666, 0);
    if (sem_run == SEM_FAILED)
        errExit("sem_open error!");
```

Usage purpose of each semaphore will be explained in detail coming pages.

2.1.2 Shared Memory Usage

In order to use shared memory correct way between process I made following struct to keep in shared memory for each process. Note that this memory is generic between all process.

```
typedef struct GTU344
{
    args givenParams;
    int dose1;
    int dose2;
    int totalLeft;
    int isRead;
    int fd;
    int leftCiti;
}clinic;
```

In this struct **dose1** and **dose2** denotes number of vaccine 1 and vaccine 2 in buffer(clinic), **isRead** denotes reading is done or not **totalLeft** denotes number of left vaccined to be made.

Usage:

```c
    static char memoryName[50];
    static clinic *biontech;
    mode_t mode = S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH | S_IRWXU;
    strcpy(memoryName, "clinic_sinovac344");
    int memFd = shm_open(memoryName, O_CREAT | O_RDWR, mode);
    if (memFd == -1)
        errExit("shm_open error!");
    if (ftruncate(memFd, sizeof(*biontech)) == -1)
        errExit("ftruncate error");

    clinic *gata = (clinic *)mmap(NULL, sizeof(*biontech), PROT_READ | PROT_WRITE, MAP_SHARED, memFd, 0);
    if (gata == MAP_FAILED)
        errExit("mmap");

    gata->givenParams = givenArgs;
    gata->dose1 = 0;
    gata->dose2 = 0;
    gata->totalLeft = 2 * (givenArgs.tArg * givenArgs.cArg);
    gata->isRead = 0;
    gata->leftCiti = givenArgs.cArg;
    gata->fd = safeOpen(givenArgs.iArg, O_RDONLY);
```

2.1.3 Communication between Nurses and Vaccinators

This problem is nothing but simply producer consumer problem.

2.1.4 Solution

In classical solution of this problem we are using 3 semaphore which are **full, empty, mutex** but these are not enough to solve our problem because vaccinator needs to wait until there is at least 1 shot(1 and 2) in buffer. In order to solve this problem I used 1 extra **full** semaphore, in order to wait on both vaccine1 and vaccine2. This means that consumer in order to consume needs to get both full1 and full2 semaphore. Normally, producer(nurse) post full semaphore directly but here it will post full1 or full2 semaphore not both of them. Simple example is following;

Producer(Nurse)

```c
    if (sem_wait(sem_empty) == -1)
        errExit("sem_wait");
    if (sem_wait(sem_mutex) == -1)
        errExit("sem_wait");
    if (!biontech->isRead)
    {
        char vaccine = readOneChar(biontech->fd);
        if (vaccine == '1')
        {
            biontech->dose1 = biontech->dose1 + 1;
            printNurseMsg(process->index, process->pid, '1', biontech);
            if (sem_post(sem_full1) == -1)
                errExit("sem_post");
        }
        else if (vaccine == '2')
        {
            biontech->dose2 = biontech->dose2 + 1;
            printNurseMsg(process->index, process->pid, '2', biontech);
            if (sem_post(sem_full2) == -1)
                errExit("sem_post");
        }
        else if (vaccine == 'x')
        {
            nurseLeaveMsg();
            biontech->isRead = 1;
            break;
        }
        else
            errExit("vaccine is wrong!!");
        if (sem_post(sem_mutex) == -1)
            errExit("sem_post");
```

Consumer(Vaccinator)

```c
        if (sem_wait(sem_full1) == -1)
            errExit("sem_wait");
        if (sem_wait(sem_full2) == -1)
            errExit("sem_wait");
        if (sem_wait(sem_mutex) == -1)
            errExit("sem_wait");
        if (biontech->totalLeft > 0)
        {
            if (sem_post(sem_vac) == -1)
                errExit("sem_post");
            if (sem_wait(sem_cit) == -1) //wait for cit update its pid
                errExit("sem_wait");
            printVaccinatorMsg(process->index, process->pid, *currentCitPid);
            biontech->dose1 = biontech->dose1 - 1;
            biontech->dose2 = biontech->dose2 - 1;
            biontech->totalLeft = biontech->totalLeft - 2;
            counter++;
            if (sem_post(sem_run) == -1)
                errExit("sem_post");
            if (sem_wait(sem_cit) == -1) //wait for cit give its message
                errExit("sem_wait");
            if (sem_post(sem_mutex) == -1)
                errExit("sem_post");
            if (sem_post(sem_empty) == -1)
                errExit("sem_post");
            if (sem_post(sem_empty) == -1)
                errExit("sem_post");
        }
        else
        {
            break;
        }
```

In this code piece most semaphores belongs to vaccinator and citizen communication. Also, We are making 2 empty semaphore post successive because we are deleting two element from buffer.

2.1.5 Communication between Vaccinators and Citizens

This proble again simple producer consumer problem

2.1.6 Checking Last Potato

To check if receiving potato is last one I used shared memory and if it is then I send a message all other processes making message pid -1, so that if the pid is -1 they will finish their jobs.

2.1.7 Removing Resources and Exiting

In order to run a cleaner method after finish, I used **atexit()** function with a cleaner method like this;

```c
/*global names to remove*/
static char myFifoName[50];
static char memoryName[50];
static char semphoreName[50];

/*remove function before exit*/
static void removeAll(void)
{
    unlink(myFifoName);
    sem_unlink(semphoreName);
    sem_unlink("counter");
    sem_unlink("barrier");
    sem_unlink("fifo_barrier");
    shm_unlink(memoryName);
}
/*save remove method*/
if (atexit(removeAll) != 0)
   errExit("atexit");
```

2.1.8 CTRL-C Handling

In order to give a message on CTRL-C interrupt, I used **sigaction** function from **signal.h** library. Also, I used a **global variable** to set if an interrupt has occur.

```c
volatile __sig_atomic_t exitSignal = 0;
void exitHandler(int signal)
{
    if (signal == SIGINT)
    {
        exitSignal = 1;
    }
}
```

In each loop in code, signal flag was checked, on interrupt signal, resources was given back and exited elegantly.

3 System Requirements

In order to run program you need an ascii file with N rows. Each row must contain a unique fifo name(path). **Note that** number of row in file and number of processes must be same. Also, file must contain an empty row at the end of it.

4 References that was used

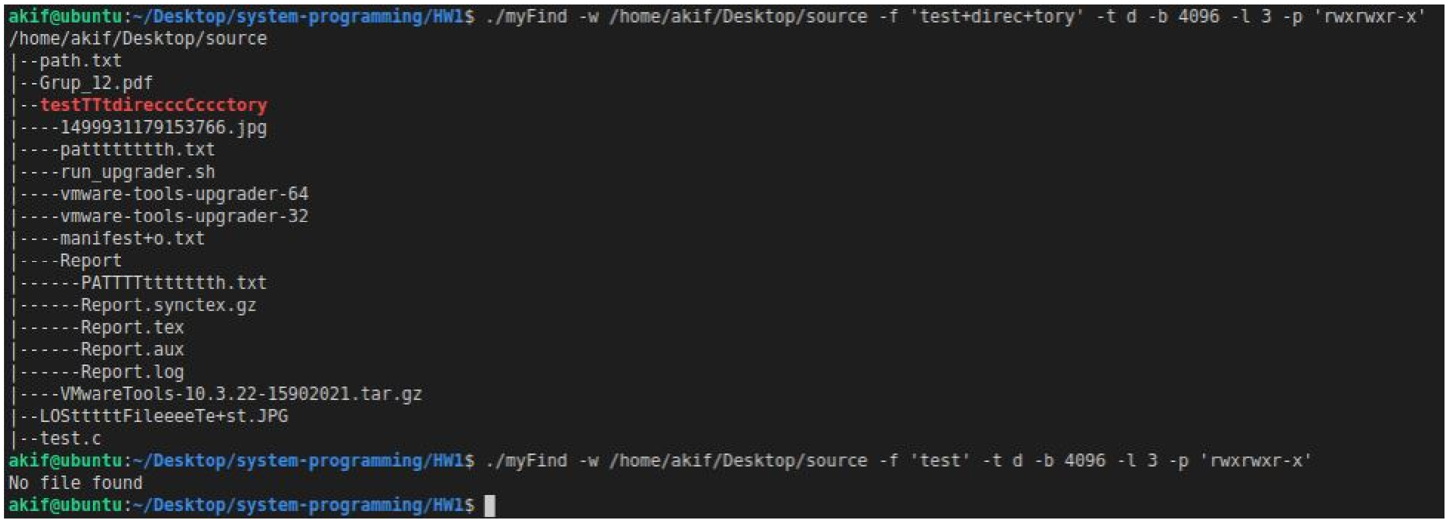
While doing this howemork following references was used;

- Course Textbook Listing 44-7: An iterative server using FIFOs
- Week-8 slides synchronization barrier problem with POSIX semaphores.

5 Test Result

Note: I understand that from pdf it doesn't have to be at least 1 process with zero potatoes. I think there is no such a condition because even though all of them has potato they are sending each other immediately.

A simple test result is following;



```
akif@ubuntu:~/Desktop/system-programming/HW1$ ./myFind -w /home/akif/Desktop/source -f 'test+direc+tory' -t d -b 4096 -l 3 -p 'rwxrwxr-x'
/home/akif/Desktop/source
|--path.txt
|--Grup_12.pdf
|--testTTtdirecccCccctory
|----1499931179153766.jpg
|----pattttttth.txt
|----run_upgrader.sh
|----vmware-tools-upgrader-64
|----vmware-tools-upgrader-32
|----manifest+o.txt
|----Report
|------PATTTTtttttth.txt
|------Report.synctex.gz
|------Report.tex
|------Report.aux
|------Report.log
|----VMwareTools-10.3.22-15902021.tar.gz
|--LOStttttFileeeeTe+st.JPG
|--test.c
akif@ubuntu:~/Desktop/system-programming/HW1$ ./myFind -w /home/akif/Desktop/source -f 'test' -t d -b 4096 -l 3 -p 'rwxrwxr-x'
No file found
akif@ubuntu:~/Desktop/system-programming/HW1$
```

Figure 1