Gebze Technical University

Computer Engineering Faculty

Midterm Report

# Processes and Semaphores

Student: Muhammed Okumuş

No: 151044017

## Contents

# 1 Program Design

Under this section, I will share function prototypes, their parts in the work flow of the whole program and their algorithms. I'll also mention the shared memory contents and semaphores purposes.

## 1.1 Functions

```
// Valdiation and printing.
void print_usage(void);
void debug_printf(const char *format, ...);
int validate_inputs(void);
void print_inputs(void);
void sig_handler(int sig_no);

// Actors
void nurse(char *input_file, struct ClinicData *data, int id);
void vaccinator(struct ClinicData *data, int id);
void citizen(struct ClinicData *data, int id);

// Wrappers
int s_wait(sem_t *sem);
int s_post(sem_t *sem);
int s_init(sem_t *sem, int val);
```

Functions are gathered in three different catagories. Input validation and printing functions are mainly used for debugging and checking the input validity. Wrappers are small functions and implemented to improve code readablity and prevent code repeating. These functions wont be explained in detail as they are mostly self explanatory. Actors are the worker functions that represent different actors in the assignment.

## 1.2 About Shared Memory and Semaphores

I used unnamed posix semaphores and shared them accross the process by putting them in shared memory structure. There is a single ClinicData structure initlized in the program and it's sent to all nurses, vaccinators and citizens procesesses. This way they can share information easily and syncronize on unnamed semaphores.

```
// Shared data
struct ClinicData
{
    sem_t sem_shm_access;        // For accesing this struct
    sem_t sem_full;              // producer/consumer sem
    sem_t sem_empty;             // producer/consumer sem
    sem_t sem_vacc_available;    // producer/consumer sem

    int pfizer;                  // vacc 1
    int sputnik;                 // vacc 2

    int file_index;              // file offset for pread

    int nurses_done;
    int total_carried;

    int citizens_to_vaccinate;
    int vaccinators_done;
    int vacc_grabbed;

    char results[1024][60];      // Result of vaccinators
};
```

## 1.3 Nurse Actor Function

This function takes three parameters: input file path, shared memory structure and an id. It doesn't return anything.

It's main purpose is the open and read the given file using pread and carry the file contents into the vaccine buffers. Since there will be multiple nurse processes, sequential reading of the file is done by keeping the file offset in the shared memory of the program. Nurse process locks the shared memory and in the critical region it performs a read of a single char from the file and increments the file offset.

In it's core it acts as a producer process. It implements the simple producer pattern with a small difference. A regular producer would increment "full/stored" semaphore for each resource carried to the buffer. Since we need pair of each vaccine, nurse only increments this semaphore only if a new pair is available. Here is a lite pseudo code of the function that outlines the semaphore usage.

```
void nurse(char *input_file, struct ClinicData *data, int id){
    // Open file ...
    while (1){
        wait(empty);
        wait(shared_memory);

        if (resource carried >= _T * _C * 2){
            post(full); //Required for multi producer/consumer
            post(empty);
            post(shared_memory);
            break;
        }

        pread(i_fd, &c, 1, data->file_offset++);
        if (c == '1')   data->pfizer++;
        if (c == '2')data->sputnik++;

        // Smart increment for new pair of vaccines
        if ((c == '1' && data->pfizer <= data->sputnik) ||
            (c == '2' && data->sputnik <= data->pfizer))
            post(full);
        post(shared_memory);
    }
    return;
}
```

## 1.4 Vaccinator Actor Function

This function takes two parameteres, it takes the same parameters as the nurse except the file path since it doesn't need it.

It acts as a mid-consumer process which can be described as a middle buffer between the nurses and citizens. It's purpose is the mark available vaccines and notify citizens of them. It doesn't take from buffer(vaccines are decremented when citizens takes the shot) it only posts "vaccine available" to citezens and makes note in the shared memories result buffer before exitting about the citizens vaccinated. Below is the pseudo code of the vaccinator function.

```
void vaccinator(struct ClinicData *data, int id){
    int doses = 0;
    while (1){
        wait(full);
        wait(shared_memory);

        if (vacc_grabbed >= _T * _C * 2) {
            post(full);
            post(shared_memory);
            // Note the result buffer ...
            break;
        }

        doses++;
        vacc_grabbed += 2;
        post(vacc_available);
        post(shared_memory);
    }
    return;
}
```

## 1.5 Citizen Actor Function

This function takes two parameteres too, it takes the same parameters as the vaccinator function.

It waits for vaccinator to signal available vaccines and decrements vaccines from the buffer. In the pseudo code below botice that function posts "empty" semaphore twice because 2 shots are done at one go.

```
void citizen(struct ClinicData *data, int id){
    while (1)
    {
        wait(vacc_available);
        wait(shared_memory);

        data->pfizer--;
        data->sputnik--;

        post(empty);
        post(empty);
        post(shared_memory);

        dose_taken++;
        if (dose_taken == _T){
            citizens_to_vaccinate--;
            break;
        }
    }
    return;
}
```

### 1.6 Forking N Processes and Assigning Tasks

Actor processes are created by calling fork and each processes pid is recorded in a dynamicly allocated pid array. This array must be initlized before the fork and must be free'ed at parent and all childeren process after fork to prevent valgrind warnings. Tasks are assigned to process under a simple for loop by checking their pid to make sure each childeren assigned a single task. Parent performs wait on each pid and frees resources before exit. I won't put the code here since those parts are pretty self explanatory in the main.c file.

## 2 Evaluation

### 2.1 Requirements Met

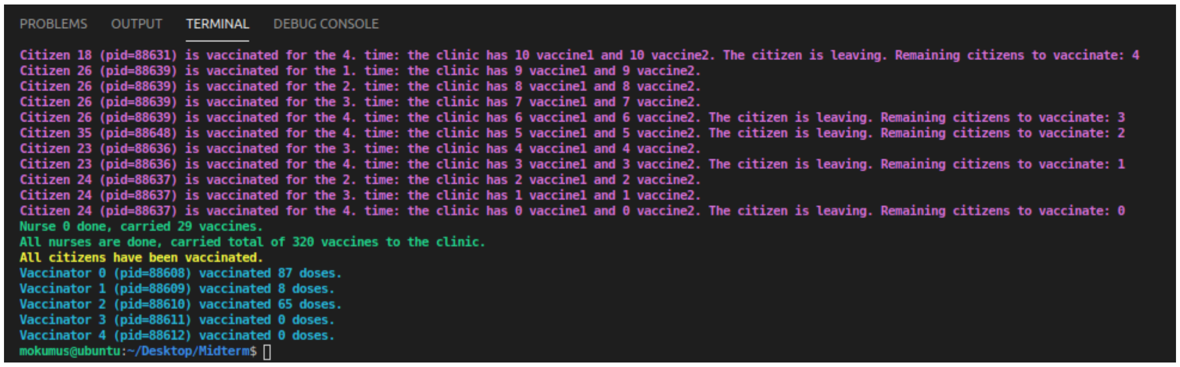
- Input validity checking and usage printing.
- Signal handling and suitable design to outside termination requests.
- No compilation warning or errors with ggc, -g -c -Wall flags on Ubuntu 18.04
- No memory leaks detected by valgrind.
- Only 4 semaphores used.
- Input file is opened with read only flag. No modification on the input file.
- No System V semaphores used.
- All citizens are always vaccinated.
- No dead locks detected tried with different inputs and parameters.

### 2.2 Requirements Failed

- Vaccinators calls random citizens that require a vaccine.

## 3 Extras

Input coloring added via macros, this will work on Linux variants and macOS. If the program input is redirected to a file macros will printed to the file and look weird. It is designed to work on console only. Sample coloring below:



```
Citizen 18 (pid=88631) is vaccinated for the 4. time: the clinic has 10 vaccine1 and 10 vaccine2. The citizen is leaving. Remaining citizens to vaccinate: 4
Citizen 26 (pid=88639) is vaccinated for the 1. time: the clinic has 9 vaccine1 and 9 vaccine2.
Citizen 26 (pid=88639) is vaccinated for the 2. time: the clinic has 8 vaccine1 and 8 vaccine2.
Citizen 26 (pid=88639) is vaccinated for the 3. time: the clinic has 7 vaccine1 and 7 vaccine2.
Citizen 26 (pid=88639) is vaccinated for the 4. time: the clinic has 6 vaccine1 and 6 vaccine2. The citizen is leaving. Remaining citizens to vaccinate: 3
Citizen 35 (pid=88648) is vaccinated for the 4. time: the clinic has 5 vaccine1 and 5 vaccine2. The citizen is leaving. Remaining citizens to vaccinate: 2
Citizen 23 (pid=88636) is vaccinated for the 3. time: the clinic has 4 vaccine1 and 4 vaccine2.
Citizen 23 (pid=88636) is vaccinated for the 4. time: the clinic has 3 vaccine1 and 3 vaccine2. The citizen is leaving. Remaining citizens to vaccinate: 1
Citizen 24 (pid=88637) is vaccinated for the 2. time: the clinic has 2 vaccine1 and 2 vaccine2.
Citizen 24 (pid=88637) is vaccinated for the 3. time: the clinic has 1 vaccine1 and 1 vaccine2.
Citizen 24 (pid=88637) is vaccinated for the 4. time: the clinic has 0 vaccine1 and 0 vaccine2. The citizen is leaving. Remaining citizens to vaccinate: 0
Nurse 0 done, carried 29 vaccines.
All nurses are done, carried total of 320 vaccines to the clinic.
All citizens have been vaccinated.
Vaccinator 0 (pid=88608) vaccinated 87 doses.
Vaccinator 1 (pid=88609) vaccinated 8 doses.
Vaccinator 2 (pid=88610) vaccinated 65 doses.
Vaccinator 3 (pid=88611) vaccinated 0 doses.
Vaccinator 4 (pid=88612) vaccinated 0 doses.
mokumus@ubuntu:~/Desktop/Midterm$ 
```

Pink is used for citizens, green for nurses, blue for vaccinators and yellow for parent process.