MIDTERM / REPORT

Can Duyar - 171044075

Solving The Problem And Design Decisions:

Handling termination signal:
When user send a termination signal such as Ctrl+C then handle_signal function will handle the command. When it get signal I]it will unlink system POSIX named semaphores and send a termination message to all running processes. Doesn't require shared memory to be free cause system will free memory automatic allocated with mmap function.

Reading arguments:
Check argument identifiers and based on argument identifiers assign arguments. After that the program will check every required argument is given or not, if every required arguments not given then show message and finish program.

Shared memory:
Mapping shared memory for sharing data and communicating between processes. This memory will store all vaccines, buffer vaccines, citizens' pid and vaccine number, signal state termination, total vaccine 1 in buffer, total vaccine 2 in buffer and citizens' state. Those data will be accessed by all child processes.

Vaccines from file:
Opening the vaccines file by the given path. At first it will read a vaccine at a time and stores in shared memory and also count total vaccines added to shared memory from file. After that it checks total vaccines stores in shared memory from file and if there is not enough vaccines then it will print message and exit the program.

POSIX Semaphore:
Get shared named semaphores. Those semaphores will be used for synchronization. We will synchronize all nurses so, only one nurse can take a vaccine to buffer at a time. All vaccinators so, only one vaccinator can vaccinate a citizen at a time. We also need to synchronize nurses and vaccinators so they will not access buffer at the same time. And also we need to synchronize citizen process and vaccinator process to maintain vaccination output.

Creating child processes:

Here it calls three functions. First function will creates all citizen processes and defines their work. Second function will creates all nurse processes and defines their works. And the last one will creates all vaccinator processes.

Wait and exit:

Here program will wait for every child processes to be finish. And after all child processes has been finished the program will unlink all shared named semaphores. And then finish the program.

FUNCTIONS

Void handle_signal(int):

This function takes the signal and unlink all shared named semaphores and then exit the program. This function is used for handling termination signal from the user.

Void print_usage():

This function prints the right format for running the program. And then exit.

Void* mem_map(int):

This function takes size as argument and map the memory and returns void pointer of mapped memory.

Void nurses_work(int):

This function takes total nurses processes to be create. And it fork those processes and defines their works.

char get_vaccine():

This function will be used by nurse process to took a vaccine from shared memory that is stored from vaccine file. This function returns a char vaccine.

Int put_vaccine(char):

This function also used by nurse process. This function takes a vaccine and stores it in buffer. If success it returns 1 else 0.

Void vaccinators_work(int):

This function takes total vaccinator processes to be create. And it fork those processes and defines their works.

Int* get_citizens():
This function is used to get sorted citizens list. This is for bonus part. This function takes all citizens and sorted them by their pid. And returns the array. This function is used by vaccinator processes.

Char get_vaccine1():
This function is used by vaccinator processes. It tooks a vaccine1 from buffer and freturns it. If buffer is empty then its retuns null char.

Char get_vaccine2():
This function is used by vaccinator processes. It takes a vaccine2 from buffer and returns it. If buffer is empty then its returns null char.

Void citizens_work(int):
This function takes total citizen processes to be create. And it fork those processes and defines their works.

Int register_citizen(int):
This function is used by citizen processes. This function takes pid and stores them into shared memory so then vaccinator processes can access them and vaccinate them. It returns the index of shared memory where pid is putted.

I checked it with valgrind and there was no memory leak, and I also checked the number of zombie processes after the program running and number of zombie processes were 0.

Which requirements I achieved and which I have failed:

Every requirements are provided and it works properly.