CSE344 – System Programming
Midterm Project – Report

Berk Pekgöz
171044041

Solving The Problem & Design:

To explain solving the problem and design, there are 9 subjects:

- Output difference
- Shared Memory and Semaphores
- Nurse
- Vaccinator
- Citizen
- Parent Process
- Error Handling
- Ctrl + C Handling
- Bonus Part

Output difference:

In PDF file Vaccinators does not print the current amount of vaccines but citizen prints. In my program, the exact opposite is the output (Vaccinator prints amount of vaccines but citizen does not) because amount of vaccines will be updated while inviting Citizen. And I sent an e-mail about this to Mr.Aptoula and he said *"I think it would be better. Specify this in the report."* .

Shared Memory and Semaphores:

In this project, I have used two shared memory and seven semaphores. First shared memory keeps counters, unnamed semaphores and some useful values. Second shared memory keeps an citizen_info (struct type) array with size number of citizen. This struct type keeps pid of the citizen and information of "is citizen busy or not" .

Semaphores; *buffer_empty, vacc1_full* and *vacc2_full* are semaphores for ensuring the producer consumer problem between nurses and vaccinators. *sm_mutex* and *ct_mutex* are semaphores for protecting the shared memories for critical regions. *available_ct* is a semaphore which provides waiting for a avaliable citizen to vaccinate. *end_barrier* is a semaphore which provides end barrier for vaccinators to print how many vaccines they have applied.

Nurse:

Nurse processes have a while loop which keeps executing until all needed vaccines placed to the buffer. Buffer mechanism is achieved with 3 semaphores and 2 shared memory values. If there is no space in buffer, nurse processe wait with *buffer_empty* semaphore until a vaccinator posts that semaphore. Then read the file and gets the character. After that uses sem_post() with associated semaphore (*vacc1_full* or *vacc2_full*) and increases the associated shared memory value (vacc_1 or vacc_2). If the character is not 1 or 2 then error occurs.

For each vaccine received, a counter in the shared memory (total_vacc) increased by one. If total_vacc reaches to the total vaccine number, then it means job of nurses is done and that process sets n_done value in the shared memory to TRUE and terminates. After that other nurses checks that value and terminates themselves.

Vaccinator:

Vaccinator processes have a while loop which keeps executing until no vaccines to be applied left. In while loop, process first checks the amount of vaccine to be applied left (not in buffer, in general). If there is no vaccination left then exits the loop.

After the first check, if loop continues then wait for a available citizen to vaccinate with *available_ct* semaphore. If any citizen is available to be vaccinated (value of semaphore is bigger than zero) process can pass through this point. Then search for that available (busy is FALSE) citizen (in my program oldest available citizen for *bonus part*) in the shared memory which keeps informations (pid/busy) of the citizens. After finding the citizen, sets that citizen's busy status as TRUE to preventing other vaccinators to choose that citizen.

To check if there are any vaccine1 and vaccine 2, *vacc1_full* and *vacc2_full* semaphores are used. If there are no vaccine1 or vaccine2 then process waits for the nurses to place both of them to the buffer. After that point, process decrease the count of the vaccine1 and vaccine2 by one. Then prints the invite message and updated amount of vaccine1 and vaccine2 to the screen. After that sends SIGUSR1 signal to the citizen (who is waiting for a signal) and posts *buffer_empty* semaphore two times because two vaccines (vaccine1 and vaccine2) is used so we need to make place for two vaccine.

When process gets out of the loop, wait for the end_barrier semaphore to print amount of vaccination at the end of program.

Citizen:

Citizen processes wait for a signal with sigsupend at the beginnig of a while loop. If SIGUSR1 signal received, it means a vaccinator invited that process to be vaccinated. If it is the first time inviting, citizen finds its own index number in the citizen info shared memory. Then decreases dose left, posts *available_ct* semaphore and prints the "Vaccinated" message to the screen.

After that check for dose left. If there more then 0 dose left, updates its busy status as FALSE and continues to the loop. But if there is no does left, then decreases citizen left value in the shared memory and exits.

If the process is the last citizen to left, then sends SIGUSR1 signal to the parent process to finish program.

Parent Process:

Parent process have some initializations like shared memories, semaphores, signal handlers and signal masking. If an error occurs during these initializations, then program terminates without forking.

After the initializations, parent first forks nurses (they can start to place vaccines to the buffer). After that forks the citizens but stores the pid of the citizens to the citizen info shared memory. When all citizens are created then parent sorts the citizen info shared memory by age (for bonus part). Finally forks vaccinators and then waits for a signal.

If parent receives a signal, checks for the context. If signal is SIGUSR1, it means program succesfully finished and posts the end_barrier semaphore then cleans up to terminate(explained in Vaccinator section). If signal is SIGUSR2, it means an error occured in one of the child process. Then sends signal to all children to terminate and cleans up to terminate itself.

Error Handling:

If one of the child processes receives and error, that process sends SIGUSR2 (which means "sometihng bad happened" in my program) to the parent process and terminates. When parent receives SIGUSR2 signal, sends signal to all children to terminate and cleans up to terminate itself gracefully.

Ctrl + C Handling:

In the beginning of the program, parent sets a signal handler for SIGINT, but after forking all childeren switches back to the default action. Because children does not allocates any heap space so there is no need to free anything in child process. Default action for SIGINT terminates process and this is a gracefully termination for child processes (I askes about this to Mr.Aptoula and he approved.)

But in the parent process, when SIGINT comes, sources are freed and waited for all children to terminate (no zombie).

Bonus Part:

Bonus part is achieved in this program. As Mr.Aptoula said, important thing is the first time inviting of a citizen. In my program if a elder process (less pid) available to be vaccinated, then vaccinator definetly invites that citizen.

Notices

- All requirements metioned in PDF have been achivied.
- Input file must be in exactly same structure with given example file.
- One program must be executing at a time. Because shared memory is used.
- Not used more than 7 posix semaphores.
- All kinds of synchronization between processes achieved with shared memory, semaphores and signals.
- Parent process clean up after all its children.
- In case of an arbitrary error, exit by printing to stderr a nicely formatted informative message.
- Some function calls does not have error check because they have no valid error in our processes.
- In case of CTRL-C the program (that means all processes) stop execution, return all resources to the system and exit with an information message.
- If the required command line arguments are missing/invalid, program print usage information and exit.

- No memory leak with valgrind.
- Report prepared via latex