April 25th, 2021

CSE344 – System Programming - Midterm – v5
Processes and semaphores

Objective

This is basically a (compound) producer/consumer paradigm. You will emulate a covid19 vaccination flow. We'll keep this simple: nurses bring vaccine products to a clinic/buffer, vaccinators vaccinate the citizens by taking two of them (both the first and second shots) and the citizens leave the clinic. Each nurse, vaccinator and citizen will be represented by a distinct process. Each citizen needs to get the two shots t times.

How

```
./program –n 3 –v 2 –c 3 –b 11 –t 3 –i inputfilepath
```

`n >= 2`: the number of nurses (integer)
`v >= 2`: the number of vaccinators (integer)
`c >= 3`: the number of citizens (integer)
`b >= tc+1`: size of the buffer (integer),
`t >= 1`: how many times each citizen must receive the 2 shots (integer),
`i`: pathname of the input file

The input ASCII file (provided externally) will contain the numbers 1 and 2 in **<u>an arbitrary order</u>** and $t \times c$ of each (*her birinden* $t \times c$ *adet*). The '1' and '2' characters in the file, represent respectively the first and second vaccine shots that every citizen must receive. In other words, there is **exactly enough** vaccines for all c citizens.

Nurse

The nurse processes have a simple job. They read the input file, one character at a time (yes, I know it's inefficient) (in the usual order, from left to right, top to bottom, nothing fancy), and place the vaccine they read (either the character '1' or '2') into the clinic/buffer (of which there will be one!). Imagine this as bringing one vaccine from cold storage (the file) to the clinic (the buffer). The nurse processes terminate after having carried all the vaccines into the clinic/buffer. If there is not enough room in the buffer for vaccines, the nurses are supposed to wait.

Vaccinator

The vaccinators are medical personel, and if there is at least one '1', and one '2' at the buffer/clinic, then a vaccinator invites a citizen into the clinic, applies both vaccines/shots, by removing the two characters from the buffer, and sends the citizen away. If there are not sufficient shots in the buffer (or only '1's or only '2's), they wait until there is. The vaccinators terminate after having vaccinated all citizens t times. The vaccinators must not prevent the maximal use of the buffer by the nurses.

Citizen

Each citizen process waits for a vaccinator to call her inside the clinic. If that happens she gets vaccinated, and waits to be called again for a total of t times. Once the citizen receives both shots t times, she leaves and lives happily ever after.

Optional bonus for +40 points: However, the citizen processes must be vaccinated according to their age, oldest first. Their pid will be related to their age; so that the citizen process with the smallest pid will be considered the oldest. Don't let vaccinators sit idle. While a citizen x receives her n-th dose, a younger citizen y should still be able to receive her n-1 dose.

Issues

- Protect the buffer against underflow and overflow

- Avoid race conditions to the buffer
- Arrange for multiple processes to read the input file sequentially
- Make sure both shots are present at the buffer before inviting a citizen into the clinic.
- Optional: Make sure the citizens are vaccinated according to their age.
- and more...

Output

```
Welcome to the GTU344 clinic. Number of citizens to vaccinate c=8 with t=2 doses.
Nurse 1 (pid=3451) has brought vaccine 1: the clinic has 1 vaccine1 and 0 vaccine2.
Nurse 3 (pid=3452) has brought vaccine 1: the clinic has 2 vaccine1 and 0 vaccine2.
...
Nurse 2 has brought vaccine 2: the clinic has 5 vaccine1 and 1 vaccine2.
Vaccinator 1 (pid=3454) is inviting citizen pid=3456 to the clinic
...
Nurse 2 has brought vaccine 2: the clinic has 5 vaccine1 and 2 vaccine2.
...
Citizen 1 (pid=3456) is vaccinated for the 1st time: the clinic has 6 vaccine1 and 1 vaccine2
...
Citizen 1 (pid=3456) is vaccinated for the 5th time: the clinic has 5 vaccine1 and 3 vaccine2. The
citizen is leaving. Remaining citizens to vaccinate: 7
...
Nurses have carried all vaccines to the buffer, terminating.
...
All citizens have been vaccinated .
Vaccinator 1 (pid=3454) vaccinated 6 doses. Vaccinator 2 (pid=3455) vaccinated 10 doses. The clinic
is now closed. Stay healthy.
```

Evaluation

You program will be evaluated with an arbitrary input file and arbitrary parameters.

Requirements:

- In case of CTRL-C, all processes must terminate gracefully.
- You are not allowed to use System V semaphores.
- You are not allowed to use more than 7 posix semaphores.
- Don’t modify the input file in any way

Rules:

1) Compilation error: grade set to 1; if the error is resolved during the demo, then evaluation continues.
2) Compilation warning (with respect to the **-Wall** flag); -1 for every warning until 10. -20 points if there are more than 10 warnings; no chance of correction at demo.
3) No makefile: -20
4) No pdf (other formats are inadmissible) report submitted (or submitted but insufficient, e.g. 3 lines of text or no design explanation, etc): -20.
5) A report prepared via latex (pdf and tex source submitted together): +5
6) If the required command line arguments are missing/invalid, your program must print usage information and exit. Otherwise: -10
8) The program crashes/freezes and/or doesn’t produce expected output with normal input: -80
9) Presence of memory leak (regardless of amount – checked with valgrind) -30
10) Zombie process (regardless of number) -30
11) Deadlock of any kind due to poor synchronization -50
12) Use of poor synchronization: busy waiting/trylock/trywait/timedwait/sleep: -80
13) Late submissions will not be accepted
14) In case of an arbitrary error, exit by printing to stderr a nicely formatted informative message. Otherwise: -10

Is my homework submission valid?

If all processes get created, and at least one citizen is vaccinated at least once, yes.

Submission rules:

- Your source files, your makefile and a report; place them all in a directory with your student number as its name, and zip the directory.
- Your report must contain: how you solved this problem, your design decisions, which requirements you achieved and which you have failed.
- The report **must be in English**.
- Your makefile must only compile the program, not run it!
- Do not submit any binary executable files. The TAs will compile them on their own boxes.
- Any deviation from the submission rules will results in automatic (without content evaluation) failure of the homework.

Good luck.