CSE344 Midterm Report

161044070

Kaan Can Bozdoğan

I used shared memory and semaphores to solve this producer-consumer problem.

Semaphores:

critSec: Process which takes the semaphore can process data in the critical section, which is making changes on the shared memory space, in other words buffer.

empty: Determines the count of empty spaces in the buffer. If it gets down to zero, nurses can not add vaccines to the buffer. When the vaccinator vaccinates a citizen, it increments the empty semaphore so nurses can now add vaccines to the buffer again.

vac1 and **vac2:** Count of the vaccines in the buffer. In order for the vaccinators to consume, buffer needs to have at least one vaccine1 and one vaccine2. So they wait for both vac1 and vac2 semaphores to start consuming. When nurses read vaccines from the input file, they control the type of vaccine, and increment the corresponding semaphore after adding that vaccine to the buffer. So the vaccinator can understand when the vaccines in the buffer are ready for vaccination.

getVac: After the vaccinator got vaccine1 and vaccine2 from the buffer they are ready to vaccinate the citizen. They invite the citizen by incrementing the getVac semaphore so citizens waiting for that semaphore can come to the clinic and get vaccinated.

citLeaving: After citizens get vaccinated they post that semaphore so the vaccinator waiting for that semaphore (the vaccinator who vaccinates that citizen) can understand that citizen's vaccination process has ended so they can continue vaccinating others who are waiting.

nursesDone: Parent process waits for that semaphore. When the buffer work of the nurses are done this semaphore is incremented so the parent process can message the user of the program that their work is done.

Shared Memory Spaces:

buf: buffer/clinic. Has the given size as a parameter. In every index of it, there is a char which represents the corresponding vaccines. With taking the **critSec** semaphore, nurses can put vaccines into it or vaccinators can take vaccines from it.

inf: Array with size 3. Has the information of size, vaccine1 and vaccine2 counts of the buffer/clinic.

Processes:

Nurse:

Nurses start by waiting for **empty** semaphore. After it is available (there is enough space on the clinic for new vaccines) with taking the **critSec** semaphore, they put vaccines they read from the input file into the first empty index of the buffer (buffer size variable in the **inf** array). Updates the buffer size and corresponding vaccine counts in the **inf** array. Posts the **vac1** and **vac2** semaphores. Repeats it until end of file of the input file is reached. Posts the **nursesDone** semaphore and exits successfully.

Vaccinator:

Waits for semaphores **vac1** and **vac2**. Takes the **critSec** semaphore. Finds one vaccine1 and one vaccine2 at the **buf** array. Removes them from it and shifts the remaining vaccines to fill the gaps of the removed vaccines. Updates the **inf** variables according to the changes it has made. Posts the **getVac** semaphore to invite an available citizen. Waits for the **citLeave** semaphore and then leaves the critical section by posting the **critSec** semaphore. Posts the **empty** semaphore to let nurses know there is available space in the buffer for them to put new vaccines into it. Repeats it until all the citizens got vaccinated.

Citizen:

Waits for the **getVac** semaphore. Gets vaccinated. Posts **citLeave** to let the vaccinator know. Loops t times (amount of shots a citizen is gonna get). And lives happily ever after.