CSE 344 HOMEWORK 3 REPORT

EMRE SEZER
1901042640

I created 6 chef processes and 4 pusher processes with fork(). Wholesaler process reads the input file line by line and stores it in a string array. This string array is stored on shared memory. Wholesaler process calls sem_post() function for each pusher semaphore. Normally pusher processes are waiting on sem_wait of that pusher semaphore. When wholesaler process posts the pusher semaphores. Each pusher process runs one time and continue waiting. Depending on the string at the share memory pusher processes sem_post the required chef semaphores.

Chef processes are normally waiting on sem_wait of the chef semaphore.When pusher processes post the chef semaphore. That particular chef process works one time and creates a gullac and sem_post the wholesaler semaphore. After that continues waiting. Those actions repeat for each line of the input file.

After a chef creates a gullac. It makes the content of the string at the shared memory “XX” which means there is no ingredients left on the string. At the each chef output in the end it prints contents of the string at the shared memory. If it prints “XX” chef already used the delivered ingredients and there is no ingredients left to be used.

At the pdf it writes “There is also a wholesaler that every once in a while delivers two distinct ingredients out of the four
to the street of the chefs”. So, I put a 1 second sleep for each input. Those sleeps are for just providing this content like written at the pdf file.

After there is no inputs left. Wholesaller sends a SIGUSR1 signal to all pusher and chef processes. Pusher processes, just exit do nothing more. Chef processes, returns the amount of gullac he/she made so far and makes the necessary prints.
In the end wholesaler increases the total gullac counter with return values of the chefs and prints the total gullac counter.

At each operation I used the mutex semaphore in order to provide synchronization.

There is a mutex semaphore
There is a semaphore for wholesaler
There are 4 semaphores for pushers

There are 6 semaphores for chefs
Running the homework:

For hw3_unnamed you need to type:

./hw3_unnamed -i inputFilePath

For hw3_named you need to type: (4 arguments)

./hw3_named -i inputFilePath -n chefSemaphorePath
-w wholesalerSemaphorePath – p pusherSemaphorePath

inputFilePath: path of the input file (argv[2])
chefSemaphorePath: path of the chef semaphore (argv[4])
wholesalerSemaphorePath: name of the wholesaler semaphore (argv[6])
pusherSemaphorePath: name of the pusher semaphore (argv[8])

I added an example input file at the folder.
Input file needs to end after the last ingredient. There shouldn't be a next line after the last input line.

I also added a lib folder. This folder contains code from course book. For using errExit just like in course book. My work is hw3_named.c and hw3_unnamed.c files.

Update:
I updated my homework according to the messages on the courses Teams page. So, my new homework file is 1901042640 (1).