CSE344 Homework3 Report

Yeşim Yalçın-200104004094

1-) Problem Definition and My Solution Approach

The problem was to simulate a wholesaler-chefs relationship in a specific way. The wholesaler puts ingredients to a shared array and waits for a dessert to be made. Meanwhile the chef who requires the ingredients take the ingredients from the array and returns a dessert.

Command Line Argument Analysis
My first action was to check if the command line arguments were inputed properly. The unnamed version of the homework requires only the data file as an argument. The named version of the homework requires the data file and all the semaphore names as arguments. All these arguments are mentioned in the "Systems Requirements" Part of the homework.

Shared Memory Handling
I created shared memory to store the ingredient array. This memory is created before any child process is created. Therefore any child process that the main process (wholesaler) creates has access to the shared memory.
There are a few more data I store in the shared memory. One of them being an endedFlag. This endedFlag becomes 1 if all the ingredients have been delivered. This endedFlag helps all the other processes aside from the wholesaler know this and exit properly. The wholesaler makes this flag 1 when it needs to otherwise it is 0.
The last data I hold in shared memory is a forbiddenIngredients array. This is used only if a chef exits because of an error in the middle of processing. If that happens, it writes the ingredients it needs to this array to mean no chef needs these ingredients anymore. If a wholesaler sees that the ingredients it tries to deliver is not needed by any chef, it does not try to deliver them.

Wholesaler-Chef Communication
I use 9 semaphores to achieve this. 1 for each chef, 1 full semaphore, 1 empty semaphore and 1 critical region semaphore.
The critical region semaphore is waited on whenever a process needs to do stuff related to the shared memory.
Whenever the wholesaler inserts ingredients to the array, it posts the critical region semaphore and then posts the full semaphore indicating the array is full now. This wakes up a pusher process which was waiting on the full semaphore.
This pusher process checks the ingredients in the array and posts the related chef's semaphore directly. Meaning the wholesaler does not know which chef needs what. Only the pusher knows.
When a chef wakes up, it consumes the products in the array and puts XX in the array meaning it is empty now. It also increments the dessert count in the shared memory to deliver the dessert. (The array was initialized with XX at the start as well) Afterwards it posts the empty semaphore meaning the array is empty now. This wakes up the wholesaler who was waiting on the empty semaphore. This repeats until no ingredients are left to deliver.
End of Communication
When the wholesaler reaches the end of the data file, it makes the endFlag 1 and posts full semaphore to wake up the pusher. Pusher wakes up but this time because the endFlag is 1, it

wakes up all chefs. When all chefs wake up and see that the endFlag is 1, they return the dessert count they have made so far.
Meanwhile, the wholesaler waits with waitpid on all of it's children. After waitpid returns for a child, the wholesaler gathers the return value of the child and sums all of them to get the total made dessert count.

2-) System Requirements

Example of How to Run the Programs:

- For unnamed:

./hw3unnamed -i ./exampleInputs/data

- For named:

./hw3named -i ./exampleInputs/data -n crit1 -e empty1 -f full1 -a chef0 -b chef1 -c chef2 -d chef3 -g chef4 -h chef5

- -n is for critical region semaphore name
- -e is for empty semaphore name
- -f is for full semaphore name
- -a is for chef0 semaphore name
- -b is for chef1 semaphore name
- -c is for chef2 semaphore name
- -d is for chef3 semaphore name
- -g is for chef4 semaphore name
- -h is for chef5 semaphore name

Other Things to Mention

1-) The program should run both with absolute and relative paths
2-) The program assumes the data file includes at least 1 line and has valid input. Meaning it has 2 capital letters in each line. However, the program handles unnecessary spaces and newlines also possible invalid ingredient combinations. The example of this can be seen in the data files I included in exampleInputs folder.

3-) The Parts I Have Completed

I have completed all the requirements of the homework and tested in various ways.