May 13th, 2020

CSE344 – System Programming - Homework #4 - v2
Fun with threads and semaphores

Let’s assume for the sake of simplicity that one needs exactly 4 ingredients for preparing güllaç:
- milk (M)
- flour (F)
- walnuts (W)
- sugar (S).

There are 6 chefs in a street, and each has an endless supply of two distinct ingredients and lacks the remaining two:

e.g.
chef1 has an endless supply of milk and flour but lacks walnuts and sugar,
chef2 has an endless supply of milk and sugar but lacks flour and walnuts,
chef3 has an endless supply of milk and walnuts but lacks sugar and flour,
etc.

Every chef sits in front of her/his store and waits for the remaining two ingredients to arrive in order to go and prepare güllaç.

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 and then waits for one of the chefs to prepare the dessert. We are assuming there is exactly one chef among the 6 with the right missing ingredients that can prepare the dessert. Once the güllaç is ready, the wholesaler takes it for sale and the chefs wait again for the next round.

Your objective is to implement the chefs and the wholesaler as 6+1 threads that print on screen their activities. If your program is succesful it could for instance print something like the following:

```
...
chef2 is waiting for flour and walnuts
chef3 is waiting for sugar and flour
chef1 is waiting for walnuts and sugar
...
the wholesaler delivers sugar and flour
chef3 has taken the sugar
the wholesaler is waiting for the dessert
chef3 has taken the flour
chef3 is preparing the dessert
chef3 has delivered the dessert to the wholesaler
the wholesaler has obtained the dessert and left to sell it
chef3 is waiting for sugar and flour
...
```

The wholesaler will read the ingredients to deliver from a file containing two letters at each line, representing the ingredients to deliver; e.g.
MS
FM

WS
etc

Assume the file has valid content and at least 10 rows. The wholesaler will be represented by the main thread, and will join all the threads of the chefs before terminating.

```
./program -i filePath
```

The ingredients will be delivered to a data structure of your choosing (e.g. array, stack, etc), located at the heap and thus shared among all involved threads. Be careful while delivering and retrieving ingredients. Once the wholesaler is done, she/he'll need a way to notify the chefs that there won't be any more ingredients.

Rules:

- use the `getopt()` library method for parsing commandline arguments.
- **All chef threads will execute the same function. Don't implement separate functions for every chef. Moreover every chef must use the same semaphores for synchronization. Assume that the wholesaler is unaware of how many threads/chefs are waiting.**
- **The chef can simulate dessert preparation by sleeping for a random number of seconds (1 to 5 inclusive).**
- **Solve any and all synchronization issues with semaphores (named, unnamed, sys V or POSIX, that's up to you).**
- You are free to create additional threads if you need them.
- In case of an error, exit by printing to stderr a nicely formatted informative message.
- If the command line arguments are missing/invalid your program must print usage information and exit.
- Each thread should remove allocated resources explicitly.
- Never allow zombie processes!
- Compilation must be warning-free.
- You will provide a demonstration of your program to the course assistants. They will provide the details.
- No late submissions will be allowed.

Submission:

- your source files, your makefile and a report on how you solved the issues in this homework.

Grading:

- Does not compile: fail
- Compiles but crashes despite normal input: fail
- Deadlock: fail
- Crashes for n reasons: -10 * n
- Violating the homework rules: fail
- No report: -30

Good luck.