April 30th, 2020

CSE344 – System Programming – Midterm project - v8

Your task is to create a program that will simulate the student mess hall of a university. There are 3 actors involved: the supplier, the cooks and the students, each of them will be a separate process. There are also 3 locations involved: the kitchen, the counter and the tables.

The kitchen is the place from which the cooks take the plates of food and where the supplier delivers them to. The tables are where every student individually eats.

There will be N cooks, M students, T tables, a counter of size S, and a kitchen of size K. Every plate of food can be of three types: soup (P), main course (C) or desert (D). A student is considered serviced when s/he gets one of each plate from the counter. The students are very hungry. Every student after eating goes to get more food from the counter, a total of L times.

Example: `$./program -N 3 -M 12 -T 5 -S 4 -L 13 -F filePath`

Constraints (all are integers)
M > N > 2
S > 3
M > T >= 1
L >= 3
K = 2LM+1

Supplier
The task of the supplier is to deliver already cooked plates of food to the kitchen. The supplier will deliver in total exactly LxM plates of each type (P, C and D, so 3LM in total); which is equal to the exact amount to be eaten in total by all students. If there is no room in the kitchen the supplier will wait for empty room, and once the supplier completes delivery, he'll leave the premises. At each visit to the kitchen the supplier will deliver **ONE plate. The supplier will read the file represented by the parameter filepath, from start to finish, which must contain exactly 3LM characters, each being either 'P','C' or 'D' in an arbitrary order. The character read will denote the plate to deliver to the kitchen. The supplier will terminate once it reaches the end of the file.**

The supplier process will print 3 types of messages:

// Entering the kitchen
```
The supplier is going to the kitchen to deliver soup:    kitchen items P:3,C:4,D:3=10
```

// Afer delivery
```
The supplier delivered soup – after delivery: kitchen items P:4,C:4,D:3=11
```

// Done delivering.
```
The supplier finished supplying – GOODBYE!
```

Cook
Each cook will get a plate (P, C or D) from the kitchen, if there is one, otherwise the cook will wait for plates to be delivered by the supplier. Then the cook will place the plate on the counter if there is any

room left on the counter. If there is no room, the cook will wait for room to open up. Then the cook will return to the kitchen and repeat. However, the cooks must be careful because they can cause deadlocks. Imagine them bringing always soups and deserts to the counter. The counter will fill up, but since there are no main courses, no student will be able to eat, and no cook will be able to place additional plates since the counter is full. So the cooks must choose intelligently the next plate to bring, so as to avoid this kind of a situation. **Your solution must not hinder the maximal use of the counter.** Solve it in such a way, that if the counter is full, containing exactly S items, then at least 3 of them will be distinct (one soup, one main course and one desert), and thus at least one student will be able to take them and eat. In other words, don't let there be a deadlock. The cooks are not allowed to put back to the kitchen any food taken from it.

Each cook process will print 4 types of messages:

// waiting for/getting deliveries
```
Cook 0 is going to the kitchen to wait for/get a plate - kitchen items P:4,C:4,D:3=11
```

// Going to the counter
```
Cook 0 is going to the counter to deliver soup – counter items P:2,C:1,D:1=4
```

// After delivery to counter
```
Cook 1 placed desert on the counter - counter items P:2,C:1,D:2=5
```

// After finishing placing all plates
```
Cook 0 finished serving - items at kitchen: 0 – going home – GOODBYE!!!
```

Student

The students wait in front of the counter. There is no queue or line of any form at the counter. Any student might be serviced at any time. A student will wait until at least one soup, one main course and one desert are available on the counter, in which case she/he will take all 3 of them at once and leave the counter.

e.g. what if one soup is available, should the student take it? NO! The student will wait until all three are available to take. She/he will either take all three plates together (P, C and D) or none.

Then the student will find an empty table, sit down and eat, if there are no empty tables, the student will wait for one. After eating, the student will once again go to the counter and repeat this behavior L times.

Each student process will print 5 types of messages: `(round x)` where `1 <= x <= L`

// arriving at the counter/waiting for food
```
Student 1 is going to the counter (round 1) - # of students at counter: 1 and counter items P:2,C:1,D:2=5
```

// waiting for/getting a table
```
Student 0 got food and is going to get a table (round 1) - # of empty tables: 4
```

// sitting to eat
```
Student 0 sat at table 1 to eat (round 4) - empty tables:1
```

// done eating, going again to the counter; times x, where x is increased to x+1
`Student 0 left table 1 to eat again (round 2) - empty tables:2`

// after finishing eating L times
`Student 2 is done eating L=5 times - going home – GOODBYE!!!`

Implementation
Implement every actor as a separate process. So you will have in total at least N+M+1 processes. The objective is to simulate this mess hall and print on screen what every actor is doing from the very start when the supplier arrives, until the last student finishes eating L times. Your main process will create and prepare all the necessary tools (semaphores, shared memory, etc) and then fork all actors involved and finally wait for them to finish.

Report: you will explain which synchronization problems you encountered, and how you solved them. Additionally, you will provide graphical plots. The data of the plots will be available to you from the output messages of your program. Each printed message will represent one temporal instance

Study the effect of each input parameter (M, N, S, T, L) on your program's output. Try various values for each parameter while keeping the rest of them fixed and for each of those combinations prepare a plot containing the # of students waiting at the counter, the # of plates at the counter and the # of plates at the kitchen across time.

----------BONUS (20 points)----------
Assume the students are either undergraduates (U) or graduates (G)

M=U+G > 3, U>G>=1

Example: `$./program -N 3 -T 5 -S 4 -L 13 -U 8 -G 2 -F filePath`

And that graduate students have priority at the counter over undergraduates. As long as there are graduates in front of the counter waiting for their food, no undergraduate is allowed to be serviced.

Rules:

- use the `getopt()` library method for parsing commandline arguments.
- Use system calls for all file I/O purposes
- Your programs are not allowed to crash due to any foreseeable reason. In case of an error, exit by printing to stderr a nicely formatted informative errno based message.
- If the command line arguments are missing/invalid your program must print usage information and exit.
- **In case of CTRL-C exit gracefully**
- **Free all allocated resources explicitly.**
- **No zombie processes.**
- **Don't use sleep, or goto, non blocking semaphore operations or busy waiting**
- **Don't use POSIX named semaphores or System V semaphores. You are expected to solve it using using unnamed POSIX semaphores and shared memory. This means you can still use signal handlers, extra variables, and extra processes.**

- **Compilation must be error and warning-free.**
- You might be requested to provide a demonstration of your program to the course assistants.
- **No late submissions will be allowed. The deadline is strict. Plan accordingly.**

Submission:

- source files, makefile and your report.

Grading:

- Does not compile: FAIL
- Deadlock: -50
- Crashes for n reasons despite normal input: -10 * n
- Violating the homework rules: FAIL
- No report: -30

Good luck.