April 29th, 2022

CSE344 – System Programming - Homework #4 - v1
Threads and System V semaphores

Objective

Let’s keep it simple and abstract. There is one supplier thread and multiple consumer threads. The supplier brings materials, one by one. And the consumers consume them, two by two. That’s it. Each actor will be modeled by its own thread.

```
Example: ./hw4 -C 10 -N 5 -F inputfilePath
```

Input File (ASCII)

It will contain in total NxC times the character ‘1’ and NxC times the character ‘2’ in an arbitrary order. This file will be provided externally as input (i.e. assume it’s there). Each character in the file will correspond to a type of material. ‘1’ will correspond to the first material, and ‘2’ will correspond to the second material.

Supplier x 1

There will be only one supplier thread. It will read the input file’s contents, one character at a time and output a message concerning its activity. For every character/material read from the file, it will augment/post a semaphore representing its amount. If it reads a ‘1’ it will post the semaphore representing the amount of ‘1’s read so far, and if it reads a ‘2’ it will post the semaphore representing the amount of ‘2’s read so far. It will terminate once it reaches the end of the file. The supplier will be a detached thread. You will have no other variables for keeping count of the delivered materials.

Output examples:

// Before delivery
```
Supplier: read from input a ‘1’. Current amounts: 0 x ‘1’, 0 x ‘2’.
```

// After delivery
```
Supplier: delivered a ‘1’. Post-delivery amounts: 1 x ‘1’, 0 x ‘2’.
```

// At EOF
```
The Supplier has left.
```

Consumers x C

Each consumer will have its own thread (not detached). Its task is to loop N times. At each iteration it will take/remove one ‘1’ and one ‘2’ by reducing the corresponding semaphores’ values. Careful, if either is not available (e.g. if there is no ‘1’ or ‘2’) then it will not take the other. In other words, it will either take two items (one ‘1’ and one ‘2’) or wait until two (one ‘1’ and one ‘2’) are available. This way each consumer will consume in total exactly N x ‘1’s and N x ‘2’s.

Output examples:

//Before consuming (**i denotes the integer id of the consumer 0 to C-1, j is the iteration number, 0 to N-1**)
```
Consumer-i at iteration j (waiting). Current amounts: 8 x ‘1’, 12 x ‘2’.
```

//After consuming
```
Consumer-i at iteration j (consumed). Post-consumption amounts: 7 x ‘1’, 11 x
‘2’.
```

// After consuming N times
```
Consumer-i has left.
```

Restrictions and requirements
- C > 4 (integer)
- N > 1 (integer)
- All threads must run concurrently.
- inputFilePath: relative or absolute
- There will be only one process.
- Don't use more than 2 semaphores.
- Don't create additional threads besides a supplier and C consumers.
- You must use only System V semaphores for synchronization.
- All output will be written to STDOUT without buffering (each line will begin with a timestamp).
- In case of SIGINT, all processes and threads will terminate gracefully, closing all open files and freeing all allocated resources.

Evaluation
Your submission will be evaluated with multiple (valid) input files and various (valid and invalid) input parameters. The assistants will try hard to cause a deadlock. Take the required precautions.

Grading (the rules have been updated!)
1) Compilation error: grade set to 1; if the error is resolved during the demo, then evaluation continues.
2) Compilation warning (with respect to the -Wall flag); -1 for every warning until 10. -20 points if there are more than 10 warnings; no chance of correction at demo.
3) No makefile, makefile without -Wall, makefile without "make clean": -30
4) No pdf report submitted (or submitted but insufficient, e.g. 3 lines of text with no design explanation, etc), file formats other than pdf: -20
5) The program crashes/locks or produces wrong output with valid input: -100
6) Poor synchronization (e.g., sleeps, busy waiting, timed waits, etc): -100
7) The program doesn't satisfy a requirement or violates a restriction: -100
8) Presence of memory leak (regardless of amount – checked with valgrind) -30
9) -15 for every day of late submission.
10) In case of an arbitrary and fatal error, exit by printing to STDERR a nicely formatted informative message. Otherwise: -10
11) If you don't cleanup after children processes and/or leave zombies: -50

Is my homework submission valid?
It is valid if given a correct input file, at least one consumer consumes a '1' and a '2'.

Submission rules:
- Your source files, your makefile and a report; place them all in a directory with your student number as its name, and zip the directory.
- Your report must contain: how you solved this problem, your design decisions, which requirements you achieved and which you have failed.
- The report must be in English.
- Your makefile must only compile the program, not run it!
- Do not submit any binary executable files. The TAs will compile them on their own boxes.
- Your code will be compared against online sources; you are free to be inspired, but you are also expected to write your own code.
- Homeworks are individual tasks. Proven cases of plagiarism will be punished to the full extent.

Hints:

- Plan/design carefully.
- Check for memory leaks with valgrind before submitting.
- Compile and run your program on more than one POSIX systems to ensure portability, if you can.
- Control whether you satisfy each and every one of the requirements prior to submission.
- Test your programs with arbitrary inputs and put it through a stress-test.

Good luck.