CSE 344 – System Programming

HW4 Report Muhammet Fikret ATAR

1801042693

Objective:

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.

Problem Definition:

It is a thread model that one supplier multi consumer. There are N types consumer which has been taken by from command line. There are main thread, detach thread and consumer threads. Consumer threads takes the 1 and 2 from supplier that produces one by one.

Problem Solution:

System V semaphores

The fact that semaphore creation and initialization must be performed by separate system calls, instead of in a single atomic step, leads to possible race conditions when initializing a semaphore (another process might attempt to use before it's initialized).

1) avoidance: ensure that a single process is in charge of creating & initializing the semaphore

2) A trick with the sem_otime field in the semid_ds data structure based on the historical (and now standard) fact that the field is set to 0 upon creation and only changes with a semop. The process that does not create the semaphore can wait until the first process has both initialized the semaphore and executed a no-op semop() call that updates the sem_otime field, but does not modify the semaphore's value

In main thread detach thread and Consumer threads have been created. Also two semaphores have been created with system 5 semaphore. Semaphores are used for controlling the 1 and 2. They are providing to a protection for operation of producer and consumers.

Design Plan Explanation

->N>4 and C>1 conditions were checked and if not appropriate, the necessary message was given and the program was terminated.

-> I created detach thread with attribute of detach flag. It does not have to wait a join inside of main thread.

-> Threads are default needs to be joined. I did not declare any flag and it needs to be joined

-> It waits 1 and 2 at the same time from detach thread. It gets the value from semctl GETVALUE and prints out after that it operates -1 and again prints out the after values of semaphores.

-> It controls all the possible signal and close the semaphores

->In the supplier thread function, the input file file is opened and in a loop until the file becomes EOF, 1 character is read from the file each time, and if the currently read buffer is not NULL, first the char_1 semaphore values are retrieved with semctl GETVAL. It is then printed on the screen as required. If the read character is 1, char_1 semaphore 1 is posted with semop, and if the read character is 2, char_1 other semaphore 1 is posted with semop. Then the semaphore value is taken with semctl and GETVAL and it is printed on the screen correctly. This continues until EOF becomes file and the last supplier leaves

-> In the consumer thread function, the i value entered as a parameter in the create function is taken as args with the id variable. For each consumer, the cycle takes N times. The values of the Char_1 semaphores are taken with semctl and GETVAL and printed to the screen as necessary

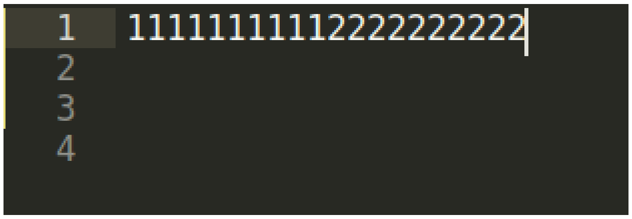
-> Whether the file is valid or not is checked while reading each character from the file. If anything other than 1 or 2 comes, the program is closed instantly.

Which requirements I achieved?

I achieved all requirement from assignment file.

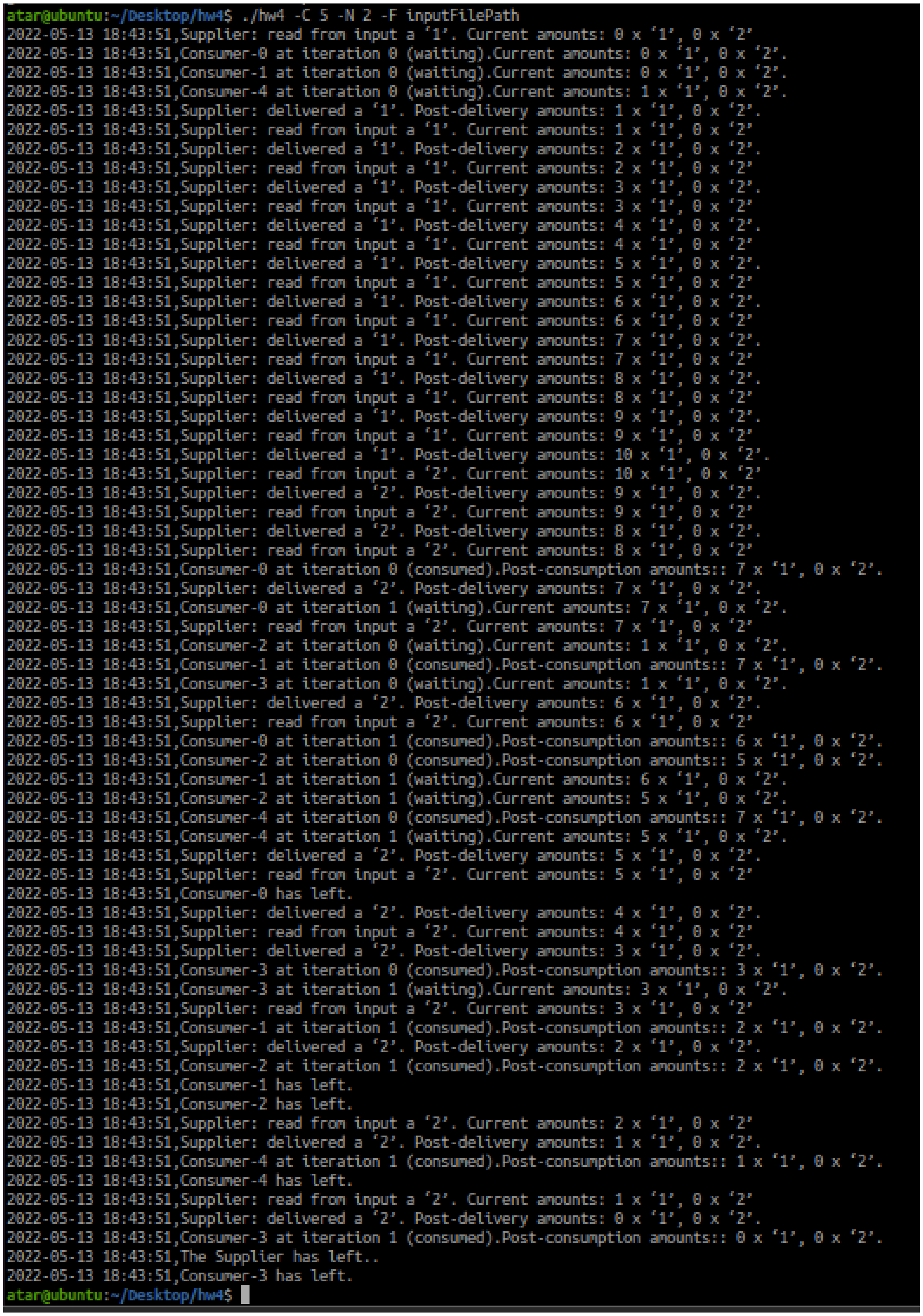
TEST AND RESULT:

Input file:



```
11111111112222222222
```

Output:



```
atar@ubuntu:~/Desktop/hw4$ ./hw4 -C 5 -N 2 -F inputFilePath
2022-05-13 18:43:51,Supplier: read from input a ‘1’. Current amounts: 0 x ‘1’, 0 x ‘2’
2022-05-13 18:43:51,Consumer-0 at iteration 0 (waiting).Current amounts: 0 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Consumer-1 at iteration 0 (waiting).Current amounts: 0 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Consumer-4 at iteration 0 (waiting).Current amounts: 1 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Supplier: delivered a ‘1’. Post-delivery amounts: 1 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Supplier: read from input a ‘1’. Current amounts: 1 x ‘1’, 0 x ‘2’
2022-05-13 18:43:51,Supplier: delivered a ‘1’. Post-delivery amounts: 2 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Supplier: read from input a ‘1’. Current amounts: 2 x ‘1’, 0 x ‘2’
2022-05-13 18:43:51,Supplier: delivered a ‘1’. Post-delivery amounts: 3 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Supplier: read from input a ‘1’. Current amounts: 3 x ‘1’, 0 x ‘2’
2022-05-13 18:43:51,Supplier: delivered a ‘1’. Post-delivery amounts: 4 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Supplier: read from input a ‘1’. Current amounts: 4 x ‘1’, 0 x ‘2’
2022-05-13 18:43:51,Supplier: delivered a ‘1’. Post-delivery amounts: 5 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Supplier: read from input a ‘1’. Current amounts: 5 x ‘1’, 0 x ‘2’
2022-05-13 18:43:51,Supplier: delivered a ‘1’. Post-delivery amounts: 6 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Supplier: read from input a ‘1’. Current amounts: 6 x ‘1’, 0 x ‘2’
2022-05-13 18:43:51,Supplier: delivered a ‘1’. Post-delivery amounts: 7 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Supplier: read from input a ‘1’. Current amounts: 7 x ‘1’, 0 x ‘2’
2022-05-13 18:43:51,Supplier: delivered a ‘1’. Post-delivery amounts: 8 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Supplier: read from input a ‘1’. Current amounts: 8 x ‘1’, 0 x ‘2’
2022-05-13 18:43:51,Supplier: delivered a ‘1’. Post-delivery amounts: 9 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Supplier: read from input a ‘1’. Current amounts: 9 x ‘1’, 0 x ‘2’
2022-05-13 18:43:51,Supplier: delivered a ‘1’. Post-delivery amounts: 10 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Supplier: read from input a ‘2’. Current amounts: 10 x ‘1’, 0 x ‘2’
2022-05-13 18:43:51,Supplier: delivered a ‘2’. Post-delivery amounts: 9 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Supplier: read from input a ‘2’. Current amounts: 9 x ‘1’, 0 x ‘2’
2022-05-13 18:43:51,Supplier: delivered a ‘2’. Post-delivery amounts: 8 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Supplier: read from input a ‘2’. Current amounts: 8 x ‘1’, 0 x ‘2’
2022-05-13 18:43:51,Consumer-0 at iteration 0 (consumed).Post-consumption amounts:: 7 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Supplier: delivered a ‘2’. Post-delivery amounts: 7 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Consumer-0 at iteration 1 (waiting).Current amounts: 7 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Supplier: read from input a ‘2’. Current amounts: 7 x ‘1’, 0 x ‘2’
2022-05-13 18:43:51,Consumer-2 at iteration 0 (waiting).Current amounts: 1 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Consumer-1 at iteration 0 (consumed).Post-consumption amounts:: 7 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Consumer-3 at iteration 0 (waiting).Current amounts: 1 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Supplier: delivered a ‘2’. Post-delivery amounts: 6 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Supplier: read from input a ‘2’. Current amounts: 6 x ‘1’, 0 x ‘2’
2022-05-13 18:43:51,Consumer-0 at iteration 1 (consumed).Post-consumption amounts:: 6 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Consumer-2 at iteration 0 (consumed).Post-consumption amounts:: 5 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Consumer-1 at iteration 1 (waiting).Current amounts: 6 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Consumer-2 at iteration 1 (waiting).Current amounts: 5 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Consumer-4 at iteration 0 (consumed).Post-consumption amounts:: 7 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Consumer-4 at iteration 1 (waiting).Current amounts: 5 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Supplier: delivered a ‘2’. Post-delivery amounts: 5 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Supplier: read from input a ‘2’. Current amounts: 5 x ‘1’, 0 x ‘2’
2022-05-13 18:43:51,Consumer-0 has left.
2022-05-13 18:43:51,Supplier: delivered a ‘2’. Post-delivery amounts: 4 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Supplier: read from input a ‘2’. Current amounts: 4 x ‘1’, 0 x ‘2’
2022-05-13 18:43:51,Supplier: delivered a ‘2’. Post-delivery amounts: 3 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Consumer-3 at iteration 0 (consumed).Post-consumption amounts:: 3 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Consumer-3 at iteration 1 (waiting).Current amounts: 3 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Supplier: read from input a ‘2’. Current amounts: 3 x ‘1’, 0 x ‘2’
2022-05-13 18:43:51,Consumer-1 at iteration 1 (consumed).Post-consumption amounts:: 2 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Supplier: delivered a ‘2’. Post-delivery amounts: 2 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Consumer-2 at iteration 1 (consumed).Post-consumption amounts:: 2 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Consumer-1 has left.
2022-05-13 18:43:51,Consumer-2 has left.
2022-05-13 18:43:51,Supplier: read from input a ‘2’. Current amounts: 2 x ‘1’, 0 x ‘2’
2022-05-13 18:43:51,Supplier: delivered a ‘2’. Post-delivery amounts: 1 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Consumer-4 at iteration 1 (consumed).Post-consumption amounts:: 1 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Consumer-4 has left.
2022-05-13 18:43:51,Supplier: read from input a ‘2’. Current amounts: 1 x ‘1’, 0 x ‘2’
2022-05-13 18:43:51,Supplier: delivered a ‘2’. Post-delivery amounts: 0 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,Consumer-3 at iteration 1 (consumed).Post-consumption amounts:: 0 x ‘1’, 0 x ‘2’.
2022-05-13 18:43:51,The Supplier has left..
2022-05-13 18:43:51,Consumer-3 has left.
atar@ubuntu:~/Desktop/hw4$ 
```