CS39002 Operating Systems Laboratory
Spring 2024

Lab Assignment: 4
Date of submission: 31–Jan –2024

Shared Memory without Synchronization

One producer and n consumers share a memory $M[]$ capable of storing two int variables. The producer generates items (random integers) in $M[1]$ for a predetermined number t of times. For each item generated, the producer specifies in $M[0]$ the consumer (an integer in the range 1, 2, 3, . . . , n) for which the item written in $M[1]$ is meant. The designated consumer reads $M[1]$, and sets $M[0]$ to 0, indicating that the item is consumed. After all of the t items are generated and consumed, the producer writes –1 to $M[0]$. After reading this special value of $M[0]$, each consumer prints some aggregate information, and terminates. Finally, the producer terminates too after printing some aggregate information.

In order to implement this set of actions, write a C program *prodcons.c*. The parent process (call it P) plays the role of the producer. P reads n (the number of consumers) and t (the number of items to be produced) from the user or as command-line arguments. Then, P creates a shared-memory segment M capable of storing two int variables. P also initializes $M[0]$ to 0 (implying that no item is available for consumption at the beginning). P then forks n child processes C_1, C_2, . . . , C_n which play the roles of the n consumers. These child processes (or consumers) are numbered 1, 2, . . . , n. After this, P goes to a production loop, and each C_i goes to a consumption loop. The loops run until all of the t items are produced and consumed. These loops work as follows.

Production loop

For each i = 1, 2, . . . , t, the producer P (parent in our case) generates a random 3-digit int value *item* and a random consumer c in the range 1, 2, . . . , n. P waits (*busy wait*) until $M[0]$ becomes 0. When $M[0]$ becomes 0, P sets $M[0]$ to c and $M[1]$ to *item* (in that order). An optional delay (you can use *usleep*()) between setting $M[0]$ and setting $M[1]$ should be used if a compile-time macro `SLEEP` is set.

After producing t items, P waits (*busy wait*) until $M[0]$ becomes 0 (that is, the last item is consumed by the designated consumer child). P then writes –1 to $M[0]$, and waits until all of the n child processes terminate. P then prints, for each consumer c, the count of items produced for c, and the sum of these items.

P finally removes the shared-memory segment M, and exits.

Consumption loop

The c-th consumer waits until $M[0]$ becomes c or –1. If $M[0]$ becomes c, the consumer reads $M[1]$ as the next item meant for it. When $M[0]$ becomes –1, the consumption loop is broken. The number of items read by the consumer and the sum of these items are then printed, and the child process terminates.

Compile-time flags

The default behavior of your program should be to print only an initial message and the final statistics. If the compile-time flag `VERBOSE` is set, then the production and the consumption of each item should also be printed (see the Sample Output). Another compile-time flag `SLEEP` (already mentioned above) dictates whether there is no delay between the setting of $M[0]$ and the setting of $M[1]$ by the producer (this should be the default behavior if the flag is not set) or there is a small delay (of 1–10 microseconds) between these two assignments. This delay simulates preemption of P (which would otherwise be very difficult to reproduce), and highlights the necessity of synchronization for this producer-consumer problem.

Submit a single C source file *prodcons.c*.

Sample Output

```
$ gcc -Wall prodcons.c ; ./a.out
n = 5
t = 100
                                                Consumer 1 is alive
                                                Consumer 2 is alive
                                                Consumer 3 is alive
Producer is alive
                                                Consumer 5 is alive
                                                Consumer 4 is alive
                                                Consumer 5 has read 25 items: Checksum = 13488
                                                Consumer 2 has read 23 items: Checksum = 13657
                                                Consumer 1 has read 17 items: Checksum = 10204
                                                Consumer 4 has read 18 items: Checksum = 10798
                                                Consumer 3 has read 17 items: Checksum = 7715
Producer has produced 100 items
17 items for Consumer 1: Checksum = 10204
23 items for Consumer 2: Checksum = 13657
17 items for Consumer 3: Checksum = 7715
18 items for Consumer 4: Checksum = 10798
25 items for Consumer 5: Checksum = 13488

$ gcc -Wall -DVERBOSE prodcons.c ; ./a.out
n = 4
t = 10
                                                Consumer 1 is alive
                                                Consumer 2 is alive
Producer is alive
Producer produces 288 for Consumer 2
                                                Consumer 2 reads 288
Producer produces 281 for Consumer 3
                                                Consumer 3 is alive
                                                Consumer 3 reads 281
Producer produces 326 for Consumer 4
                                                Consumer 4 is alive
                                                Consumer 4 reads 326
Producer produces 535 for Consumer 2
                                                Consumer 2 reads 535
Producer produces 505 for Consumer 1
                                                Consumer 1 reads 505
Producer produces 848 for Consumer 2
                                                Consumer 2 reads 848
Producer produces 799 for Consumer 3
                                                Consumer 3 reads 799
Producer produces 828 for Consumer 4
                                                Consumer 4 reads 828
Producer produces 884 for Consumer 4
                                                Consumer 4 reads 884
Producer produces 688 for Consumer 4
                                                Consumer 4 reads 688
                                                Consumer 1 has read 1 items: Checksum = 505
                                                Consumer 2 has read 3 items: Checksum = 1671
                                                Consumer 3 has read 2 items: Checksum = 1080
                                                Consumer 4 has read 4 items: Checksum = 2726
Producer has produced 10 items
1 items for Consumer 1: Checksum = 505
3 items for Consumer 2: Checksum = 1671
2 items for Consumer 3: Checksum = 1080
4 items for Consumer 4: Checksum = 2726

$ gcc -Wall -DVERBOSE -DSLEEP prodcons.c ; ./a.out
n = 2
t = 10
                                                Consumer 1 is alive
Producer is alive
Producer produces 846 for Consumer 1
                                                Consumer 1 reads 0
                                                Consumer 2 is alive
Producer produces 648 for Consumer 1
                                                Consumer 1 reads 846
Producer produces 889 for Consumer 1
                                                Consumer 1 reads 648
Producer produces 861 for Consumer 2
                                                Consumer 2 reads 889
Producer produces 913 for Consumer 1
                                                Consumer 1 reads 861
Producer produces 924 for Consumer 2
                                                Consumer 2 reads 913
Producer produces 450 for Consumer 1
                                                Consumer 1 reads 924
Producer produces 671 for Consumer 1
                                                Consumer 1 reads 450
Producer produces 168 for Consumer 2
                                                Consumer 2 reads 671
Producer produces 364 for Consumer 2
                                                Consumer 2 reads 168
                                                Consumer 1 has read 6 items: Checksum = 3729
                                                Consumer 2 has read 4 items: Checksum = 2641
Producer has produced 10 items
6 items for Consumer 1: Checksum = 4417
4 items for Consumer 2: Checksum = 2317
```