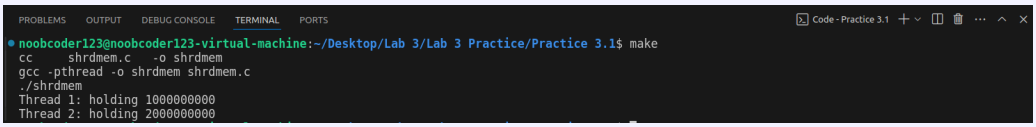
REPORT LAB 3: Synchronization

LÔ HOÀNG BẢO - 2252066 - CN01

Ngày 09 tháng 05 năm 2024

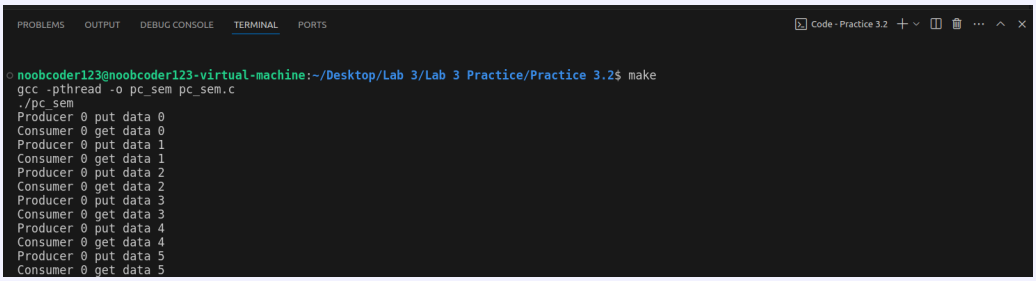
Practice 1: Shared buffer problem

After recognizing the wrong issue and propose a fix mechanism using the provided synchronization tool. I got the answer as follow for this Shared buffer problem:



Practice 2: Bounded buffer problem problem

After recognizing the wrong issue and propose a fix mechanism using the provided synchronization tool. I got the answer as follow for this Bounded buffer problem problem:



Practice 3: Dining-Philosopher problem problem

After proposing a solution to make it work. I got the answer as follow:

```
noobcoder123@noobcoder123-virtual-machine:~/Desktop/Lab 3/Lab 3 Practice/Practice 3.3$ make
gcc -pthread -o din dinPhl.c
./din
Philosopher 0 has entered room
Philosopher 3 has entered room
Philosopher 1 has entered room
Philosopher 2 has entered room
Philosopher 4 has entered room
Philosopher 2 takes fork 2 and 3
Philosopher 2 is eating
Philosopher 2 puts fork 3 and 2 down
Philosopher 2 is thinking
Philosopher 0 takes fork 0 and 1
Philosopher 0 is eating
Philosopher 0 puts fork 1 and 0 down
Philosopher 0 is thinking
Philosopher 1 takes fork 1 and 2
Philosopher 1 is eating
Philosopher 1 puts fork 2 and 1 down
Philosopher 1 is thinking
Philosopher 3 takes fork 3 and 4
Philosopher 3 is eating
Philosopher 3 puts fork 4 and 3 down
Philosopher 3 is thinking
Philosopher 4 takes fork 4 and 0
Philosopher 4 is eating
Philosopher 4 puts fork 0 and 4 down
Philosopher 4 is thinking
Philosopher 2 takes fork 2 and 3
Philosopher 2 is eating
Philosopher 2 puts fork 3 and 2 down
Philosopher 2 is thinking
Philosopher 4 takes fork 4 and 0
Philosopher 4 is eating
Philosopher 4 puts fork 0 and 4 down
Philosopher 4 is thinking
Philosopher 1 takes fork 1 and 2
Philosopher 1 is eating
Philosopher 1 puts fork 2 and 1 down
Philosopher 1 is thinking
Philosopher 0 takes fork 0 and 1
Philosopher 0 is eating
                                                    Ln 12, Col 1  Spaces: 4  UTF-8  LF  {} C  Linux
```

Exercise 1: Sequence Lock API

In my implemtation of sequence lock, I implement base on the policy as follow:

- When no one is in the critical section, one writer can enter the critical section and takes the lock, increasing the sequence number by one to an odd value. When the sequence number is an odd value, the writing is happening. When the writing has been done, the sequence is back to even value. Only one writer is allow into critical section.
- When the reader wants to read data, it checks the sequence number which is an odd value, then it has to wait until the writer finish.
- When the value is even, many readers can enter the critical section to read the value.
- When there are only a reader and no writer in the critical section, if a writer want to enter the critical section it can take the lock without blocking.

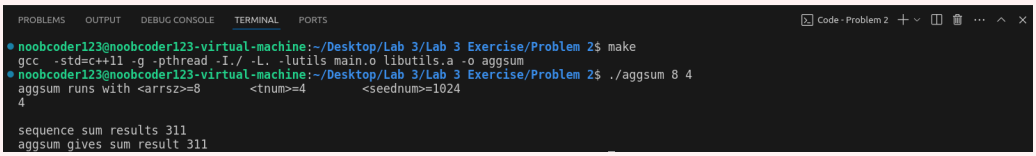
After designing and implementing sequence lock API. I got the result as follow after running my code.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                    Code - Problem 1

noobcoder123@noobcoder123-virtual-machine:~/Desktop/Lab 3/Lab 3 Exercise/Problem 1$ make
gcc -pthread seqlock.c -o seqlock
./seqlock
val = 1
```

Exercise 2: Aggregated Sum

In this code, I implement the thread-safe program to calculate the sum of a given integer array using $< tnum >$ number of threads.
After implementing thread routine. I got the result as follow after running my code.



Exercise 3: Interruptable System Logger

In this code, I design and implement a logger support the two operations *wrlog*() and *flushlog*() to manipulate the log data buffer "logbuf"
After implementing thread routine. I got the result as follow after running my code.

```
noobcoder123@noobcoder123-virtual-machine:~/Desktop/Lab 3/Lab 3 Exercise/Problem 3$ make
gcc -pthread -o logbuf logbuf.c
./logbuf
wrlog(): 0
wrlog(): 7
wrlog(): 12
wrlog(): 26
wrlog(): 4
wrlog(): 5
flushlog()
Slot  0: 0
Slot  1: 7
Slot  2: 12
Slot  3: 26
Slot  4: 4
Slot  5: 5
wrlog(): 6
wrlog(): 1
wrlog(): 8
wrlog(): 9
wrlog(): 10
wrlog(): 11
flushlog()
Slot  0: 6
Slot  1: 1
Slot  2: 8
Slot  3: 9
Slot  4: 10
Slot  5: 11
wrlog(): 13
wrlog(): 14
wrlog(): 15
wrlog(): 17
wrlog(): 18
wrlog(): 19
flushlog()
Slot  0: 13
Slot  1: 14
Slot  2: 15
Slot  3: 17
Slot  4: 18
Slot  5: 19
wrlog(): 21
wrlog(): 22
wrlog(): 23
wrlog(): 24
wrlog(): 25
wrlog(): 27
flushlog()
Slot  0: 21
Slot  1: 22
Slot  2: 23
Slot  3: 24
Slot  4: 25
Slot  5: 27
wrlog(): 28
wrlog(): 29
wrlog(): 16
wrlog(): 2
wrlog(): 20
wrlog(): 3
flushlog()
Slot  0: 28
Slot  1: 29
Slot  2: 16
Slot  3: 2
Slot  4: 20
Slot  5: 3
```