How to Run Test & Our Test Results

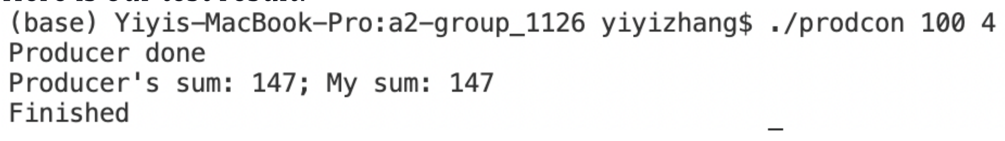
1. Run make

2. Run the test file prodcon.

   Prodcon created a producer and a consumer thread. The producer first sends a message indicating how many numbers will be sent later, and then sends the numbers while both the producer and the consumer are calculating the sum of those numbers. Prodcon compares the sum calculated by the producer with the sum calculated by the consumer to see if any synchronization problem occurred.

   Prodcon takes two arguments; the first one for the number of bytes of queue capacity; the second for the number of digits we will transfer.

   Here is our test result:



```
(base) Yiyis-MacBook-Pro:a2-group_1126 yiyizhang$ ./prodcon 100 4
Producer done
Producer's sum: 147; My sum: 147
Finished
```

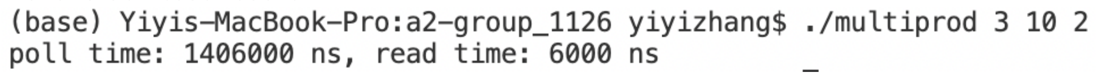
3. Run the test file multiprod.

   Multiprod creates one consumer thread and multiple producer threads. Each producer is assigned to one queue and will send the same number of messages. The consumer thread read from all the queues in non-blocking style (meaning it will not block in read(); it uses poll() to wait until a result is ready, and then read()).

   When we implemented the poll function correctly, the poll time should be much larger than the read time.

   Multiprod takes three arguments; the first one for the number of messages that can store in a queue; the second one for the number of producers; the third one for the number of messages each producer will send.

   Here is our test result:



```
(base) Yiyis-MacBook-Pro:a2-group_1126 yiyizhang$ ./multiprod 3 10 2
poll time: 1406000 ns, read time: 6000 ns
```