Chapter 6 Lab Work: Synchronization

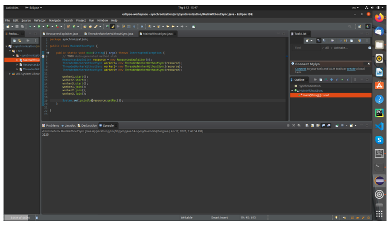
Nguyen Ngoc Lam - 20162316

Friday 5th June, 2020

Contents

1 Launch this program several times. What do you notice? Explain it!

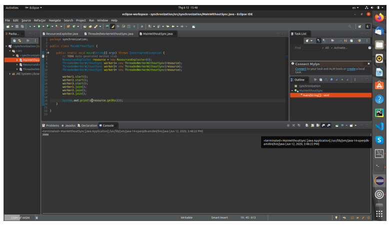
There are some times when the program print out the wrong answer (expected 3000 every time, got less than 3000 at some instances)



```
package synchronization;

public class MainWithoutSync {

    public static void main(String[] args) throws InterruptedException {
        // TODO Auto-generated method stub
        ResourcesExploiter resource = new ResourcesExploiter(0);
        ThreadedWorkerWithoutSync worker1= new ThreadedWorkerWithoutSync(resource);
        ThreadedWorkerWithoutSync worker2= new ThreadedWorkerWithoutSync(resource);
        ThreadedWorkerWithoutSync worker3= new ThreadedWorkerWithoutSync(resource);

        worker1.start();
        worker2.start();
        worker3.start();
        worker1.join();
        worker2.join();
        worker3.join();

        System.out.println(resource.getRsc());
    }

}
```

```
<terminated> MainWithoutSync [Java Application] /usr/lib/jvm/java-14-openjdk-amd64/bin/java (Jun 12, 2020, 3:46:54 PM)
2225
```

(a) Unexpected result



```
package synchronization;

public class MainWithoutSync {

    public static void main(String[] args) throws InterruptedException {
        // TODO Auto-generated method stub
        ResourcesExploiter resource = new ResourcesExploiter(0);
        ThreadedWorkerWithoutSync worker1= new ThreadedWorkerWithoutSync(resource);
        ThreadedWorkerWithoutSync worker2= new ThreadedWorkerWithoutSync(resource);
        ThreadedWorkerWithoutSync worker3= new ThreadedWorkerWithoutSync(resource);

        worker1.start();
        worker2.start();
        worker3.start();
        worker1.join();
        worker2.join();
        worker3.join();

        System.out.println(resource.getRsc());
    }

}
```

```
<terminated> MainWithoutSync [Java Application] /usr/lib/jvm/java-14-openjdk-amd64/bin/java (Jun 12, 2020, 3:48:22 PM)
3000
```

(b) The right result

Figure 1: Two types of result

Explanation: Since there is not synchronization between each worker, sometimes two workers will try to access the resource at the same time. Each one, get the old value and add 1 to that and return the resource. Because of that, there are ”less” addition happened and led to smaller result.

2 Differences after synchronization. Explain

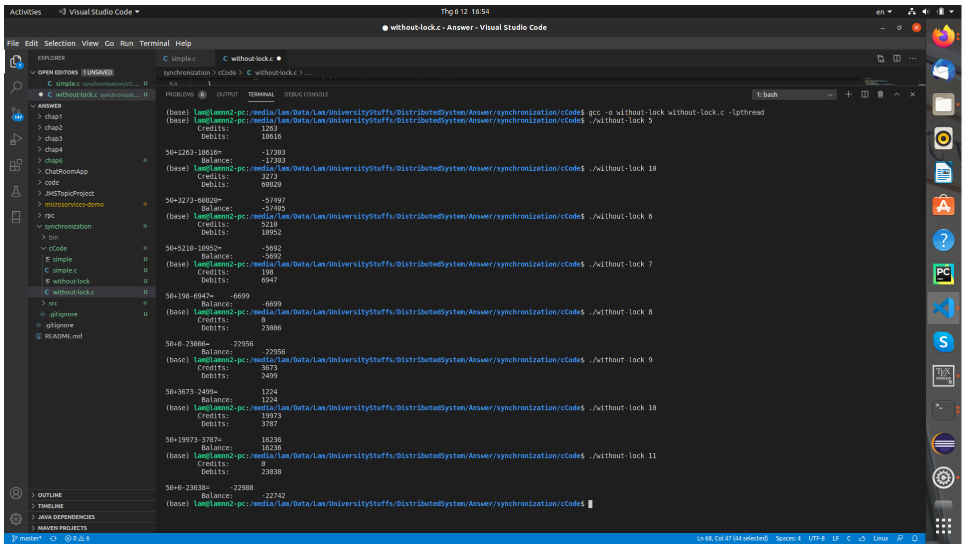
The result is now consistently 3000 due to applying synchronization that makes workers access the resource in an ”orderly” fashion.

3 Differences after using lock. Explain

The result is now consistently 3000 due to applying lock mechanism that makes only one worker can access the resource at one time.

4 Complete this file above (in the part YOUR-CODE-HERE) with a loop to increase the variable shared by 1 for 5 seconds.

5 Try to increase the value of threads and the value of the constant NUM_TRANS after each execution time until you obtain the different results between Balance and INIT_BALANCE + credits − debits. Explain why do you get this difference.

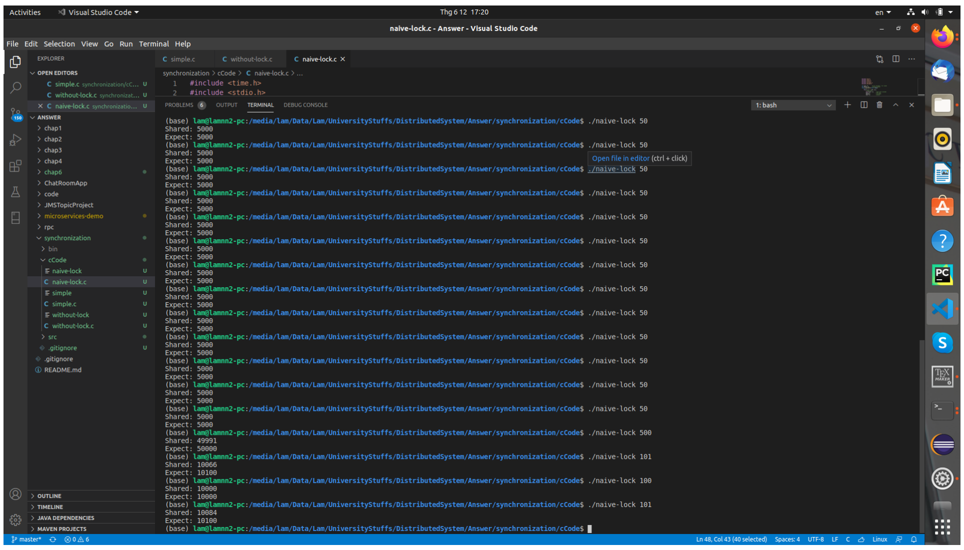


```
(base) lam@lamnn2-pc:/media/lam/Data/Lam/UniversityStuffs/DistributedSystem/Answer/synchronization/cCode$ gcc -o without-lock without-lock.c -lpthread
(base) lam@lamnn2-pc:/media/lam/Data/Lam/UniversityStuffs/DistributedSystem/Answer/synchronization/cCode$ ./without-lock 5
        Credits:        1263
         Debits:        18616

50+1263-18616=          -17303
         Balance:       -17303
(base) lam@lamnn2-pc:/media/lam/Data/Lam/UniversityStuffs/DistributedSystem/Answer/synchronization/cCode$ ./without-lock 10
        Credits:        3273
         Debits:        60820

50+3273-60820=          -57497
         Balance:       -57405
(base) lam@lamnn2-pc:/media/lam/Data/Lam/UniversityStuffs/DistributedSystem/Answer/synchronization/cCode$ ./without-lock 6
        Credits:        5210
         Debits:        10952

50+5210-10952=          -5692
         Balance:       -5692
(base) lam@lamnn2-pc:/media/lam/Data/Lam/UniversityStuffs/DistributedSystem/Answer/synchronization/cCode$ ./without-lock 7
        Credits:        198
         Debits:        6947

50+198-6947=    -6699
         Balance:       -6699
(base) lam@lamnn2-pc:/media/lam/Data/Lam/UniversityStuffs/DistributedSystem/Answer/synchronization/cCode$ ./without-lock 8
        Credits:        0
         Debits:        23006

50+0-23006=     -22956
         Balance:       -22956
(base) lam@lamnn2-pc:/media/lam/Data/Lam/UniversityStuffs/DistributedSystem/Answer/synchronization/cCode$ ./without-lock 9
        Credits:        3673
         Debits:        2499

50+3673-2499=           1224
         Balance:       1224
(base) lam@lamnn2-pc:/media/lam/Data/Lam/UniversityStuffs/DistributedSystem/Answer/synchronization/cCode$ ./without-lock 10
        Credits:        19973
         Debits:        3787

50+19973-3787=          16236
         Balance:       16236
(base) lam@lamnn2-pc:/media/lam/Data/Lam/UniversityStuffs/DistributedSystem/Answer/synchronization/cCode$ ./without-lock 11
        Credits:        0
         Debits:        23038

50+0-23038=     -22988
         Balance:       -22742
(base) lam@lamnn2-pc:/media/lam/Data/Lam/UniversityStuffs/DistributedSystem/Answer/synchronization/cCode$
```

Figure 2: Result when there is not any lock

Since every threads can access freely to the resource, when calculate the balance and there are more than one thread try to calculate that, it will not return the desired result (the whole operations of N thread involved) instead it updates the value according to the last to complete it operation. The higher the number of threads, the bigger chance of this to happened. The save option would be to make variable credits and debits exclusively used of each other and each thread only performs a credit or debit but not both.

6 Try to build and run this program. Launch it repeatedly until you see the difference between Shared and Expect values. Analyze the source code to understand the problem that leads to this difference.



```
#include <time.h>
#include <stdio.h>
```

```
(base) lam@lamnn2-pc:/media/lam/Data/Lam/UniversityStuffs/DistributedSystem/Answer/synchronization/cCode$ ./naive-lock 50
Shared: 5000
Expect: 5000
(base) lam@lamnn2-pc:/media/lam/Data/Lam/UniversityStuffs/DistributedSystem/Answer/synchronization/cCode$ ./naive-lock 50
Shared: 5000
Expect: 5000
(base) lam@lamnn2-pc:/media/lam/Data/Lam/UniversityStuffs/DistributedSystem/Answer/synchronization/cCode$ ./naive-lock 50
Shared: 5000
Expect: 5000
(base) lam@lamnn2-pc:/media/lam/Data/Lam/UniversityStuffs/DistributedSystem/Answer/synchronization/cCode$ ./naive-lock 50
Shared: 5000
Expect: 5000
(base) lam@lamnn2-pc:/media/lam/Data/Lam/UniversityStuffs/DistributedSystem/Answer/synchronization/cCode$ ./naive-lock 50
Shared: 5000
Expect: 5000
(base) lam@lamnn2-pc:/media/lam/Data/Lam/UniversityStuffs/DistributedSystem/Answer/synchronization/cCode$ ./naive-lock 50
Shared: 5000
Expect: 5000
(base) lam@lamnn2-pc:/media/lam/Data/Lam/UniversityStuffs/DistributedSystem/Answer/synchronization/cCode$ ./naive-lock 50
Shared: 5000
Expect: 5000
(base) lam@lamnn2-pc:/media/lam/Data/Lam/UniversityStuffs/DistributedSystem/Answer/synchronization/cCode$ ./naive-lock 50
Shared: 5000
Expect: 5000
(base) lam@lamnn2-pc:/media/lam/Data/Lam/UniversityStuffs/DistributedSystem/Answer/synchronization/cCode$ ./naive-lock 50
Shared: 5000
Expect: 5000
(base) lam@lamnn2-pc:/media/lam/Data/Lam/UniversityStuffs/DistributedSystem/Answer/synchronization/cCode$ ./naive-lock 50
Shared: 5000
Expect: 5000
(base) lam@lamnn2-pc:/media/lam/Data/Lam/UniversityStuffs/DistributedSystem/Answer/synchronization/cCode$ ./naive-lock 50
Shared: 5000
Expect: 5000
(base) lam@lamnn2-pc:/media/lam/Data/Lam/UniversityStuffs/DistributedSystem/Answer/synchronization/cCode$ ./naive-lock 50
Shared: 5000
Expect: 5000
(base) lam@lamnn2-pc:/media/lam/Data/Lam/UniversityStuffs/DistributedSystem/Answer/synchronization/cCode$ ./naive-lock 50
Shared: 5000
Expect: 5000
(base) lam@lamnn2-pc:/media/lam/Data/Lam/UniversityStuffs/DistributedSystem/Answer/synchronization/cCode$ ./naive-lock 500
Shared: 49991
Expect: 50000
(base) lam@lamnn2-pc:/media/lam/Data/Lam/UniversityStuffs/DistributedSystem/Answer/synchronization/cCode$ ./naive-lock 101
Shared: 10066
Expect: 10100
(base) lam@lamnn2-pc:/media/lam/Data/Lam/UniversityStuffs/DistributedSystem/Answer/synchronization/cCode$ ./naive-lock 100
Shared: 10000
Expect: 10000
(base) lam@lamnn2-pc:/media/lam/Data/Lam/UniversityStuffs/DistributedSystem/Answer/synchronization/cCode$ ./naive-lock 101
Shared: 10084
Expect: 10100
(base) lam@lamnn2-pc:/media/lam/Data/Lam/UniversityStuffs/DistributedSystem/Answer/synchronization/cCode$
```

Figure 3: Result when using naive lock

Since naive lock can only work for when the number of thread is smaller than 100 (as declared in the increment function), if the number of threads is larger than 100, it will immediately have a problem.

7 What is the improvement after using mutex lock

When use mutex lock, I saw that the number of error reduced dramatically. The error rate is nearly zero (out of all the time I run the program it return the different answer once).

8 Compare the run times of the two strategies to prove that Fine Locking is faster and much faster on larger load sets

On 10000 threads, Coarse Locking runs in about 3 seconds. Meanwhile, Fine Locking runs in about 1.5 seconds.

9 Run this program and what do you get as output? Explain what the deadlock is

The program will forever runs and it never give the output.
Deadlock: deadlock is a state in which each member of a group is waiting for another member, including itself, to take action, such as sending a message or more commonly releasing a lock. In this particular problem, fun_1 gets the lock of resource a first while fun_2 gets the lock of resource b. 2 processes need both lock to work, but since neither can acquire the pair of locks so they will forever wait each other.