Operating System
10610 CS342302
Prof. Pai H. Chou
TA:

Hsu PO-CHUN
Student ID: s105062803
Submitted: Dec. 04, 2017
Due Date: Dec. 05, 2017

Assignment 7

1 Problem 1

[20 points] 7.4 In Section 7.4.4, we describe a situation in which we prevent deadlock by ensuring that all locks are acquired in a certain order. However, we also point out that deadlock is possible in this situation if two threads simultaneously invoke the transaction() function. Fix the transaction() function to prevent deadlocks.

```
void transaction(Account from, Account to, double amount) {
   mutex lock1, lock2;
   lock1 = get_lock(from);
   lock2 = get_lock(to);
   acquire(lock1);
      acquire(lock2);
         withdraw(from, amount);
         deposit(to, amount);
      release(lock2);
   release(lock1);
}
```

1.1 Answer

Add a new lock to this function. This third lock must be acquired before the two locks associated with the accounts are acquired.
The transaction() function now appears as follows:

```
void transaction(Account from, Account to, double amount)
{
Semaphore lock1, lock2, lock3;
wait(lock3);//acquire lock3
        lock1 = getLock(from);//request to get lock for lock1
        lock2 = getLock(to);//request to get lock for lock2
        wait(lock1);//acquire the lock1 to perform the transaction
                wait(lock2);//acquire the lock2 to perform the transaction
                        withdraw(from, amount);
                        deposit(to, amount);
                signal(lock3);        //first release the lock3
                signal(lock2);
        signal(lock1);
}
```

2 Problem 2

[20 points] 7.7 Consider a system consisting of four resources of the same type that are shared by three processes, each of which needs at most two resources. Show that the system is deadlock free.

2.1 Answer

This system is deadlock-free.

To show it by contradiction:

Suppose the system is deadlocked. This implies that each process is holding one resource and is waiting for one more. Since there are three processes and four resources, one process must be able to obtain two resources. This process requires no more resources and, therefore it will release its resources when it is finished. Then the available resources are always larger or equal to the each needs. Hence, the system is deadlock-free.

By Banker's algorithm:

$$
\begin{aligned}
Max_i &= 2 \\
\sum_i^n Need_i &= \sum_i^n Max_i - \sum_i^n Allocation_i \\
&= 2 * 3 - 4 \\
&= 2 < N = 3
\end{aligned}
$$

This imply that at least one process can release Max = 2 resources. Hence, the system is deadlock-free.