Sabancı University

Faculty of Engineering and Natural Sciences
CS204 Advanced Programming
Spring 2017

Homework 7 – Simulation of Bank Account Transactions using Threads

Due: 12/05/2017, Friday, 21:00

PLEASE NOTE:

Your program should be a robust one such that you have to consider all relevant programmer mistakes and extreme cases; you are expected to take actions accordingly!

You can NOT collaborate with your friends and discuss solutions. You have to write down the code on your own. Plagiarism will not be tolerated!

Introduction

In this homework, you are asked to write a **multithreaded** C++ program that simulates transactions of multiple people on the same bank account. There are two possible types of transactions: *withdrawing* and *depositing* money. It is possible that multiple people try to access the account at the same time performing different types of transactions. For example, there could be four people and three of them want to withdraw money, while the other person wants to deposit money. Naturally, only one person can access the account at a time. We can achieve this by representing each person as a *different thread* in the program.

The time required for any type of transaction is probabilistic. Thus, depositing and withdrawing transactions take a *random* amount of time. The parameters of these random values will be input (see Section "Details of Simulation" for details).

The simulation starts after taking some inputs from the keyboard, and continues until a certain number of transactions, which is a user input, have been performed. The time interval between transactions is probabilistic and they are also represented by some user inputs. During the simulation, you are going to display some verbose output about transactions. Please see “Details of Simulation” section for these details.

Using Threads

There will be **four threads** (other than main thread) in your program. **Three threads** are dedicated to the **withdrawal** transactions (to simulate three different people withdrawing money), while the other thread is used for a **deposit** transaction (there is only one person to deposit money on the account).

The account balance is stored in a file as a single integer value. In that file, there should be no other data, but a single integer for balance. Initially the balance is zero. In each thread, similar operations are performed: first reading the account balance from file and then overwriting the balance with the updated value accordingly. Please note that withdrawing money is only possible if the account balance is positive.

Since simultaneous access to the bank account is impossible, you must make sure that at any given time exactly one transaction (deposit or withdrawal) is performed, meaning that only one of the four threads can read and update the balance. To guarantee such a synchronization, you must take some precautions in the threads. Please refer to lecture and lab materials which include methods for dealing with these cases using *mutex*.

If you cannot resolve the abovementioned synchronization issues properly, you might have some inconsistencies in the transactions, such as negative balance, wrong balance, withdrawing the same money multiple times (e.g. suppose the balance is 2 TL and two withdrawal transactions successfully withdraw 1 TL each. You expect zero balance, but due to synchronization problems the balance could be 1 TL), etc. You have to check the outputs for the transactions carefully to see such inconsistencies in order to debug and fix the problem in your code.

At the end, your threads must be joined properly and your program must terminate without any complications.

Details of Simulation

Before the simulation begins, seven inputs are entered via keyboard. First, the file name used to store the account balance is entered. Second, the minimum and maximum waiting time between withdrawal transactions of a thread are entered in seconds (two inputs). Then, the same information is entered for deposit transactions (two inputs). Afterwards, you enter the total number of withdrawal transactions to be performed per withdrawal thread (say *countW*). Since there are three threads of this type, each one has to perform this total number of withdrawal transactions (all successful and failed withdrawal transactions count, see "Sample runs" for details). Finally, the total number of deposit transactions to be performed (say *countD*) is entered. **Your program must stop once each of the four threads has performed the user-specified total number of transactions.**

You may assume that all inputs are correct, so no input checks are necessary.

After the inputs are entered, the file for storing the account balance is created and you initialize the balance with value 0. Please note that we use this file as both output file (to initialize and update the balances after each transaction) and as input file to read the balance before the transaction. The first use is initialization of the balance to zero (output file); thus you must not check existence of the file name.

In the next step, you can create and start the four threads: **first start the thread for depositing money** and **afterwards start the three threads for withdrawing money.**

At the beginning of simulation, display a message that says that the simulation has started and the time of start (see "Sample Runs" for examples).

First, let us explain the general strategy for all threads. Then we will continue with the specific operations of withdrawal and deposit threads.

For all threads, before each transaction (including the first transaction), the threads must wait a random amount of time (in seconds). Then, the transaction is performed. These operations are repeated for *countW* or *countD* times for each thread, depending on whether it is withdrawal or deposit thread. After a thread finishes all of its transactions, its job is done.

The random waiting times before the transactions are determined via a function that returns a random integer between (and including) a minimum and maximum value passed as parameters. **Do not use the `RandGen` class from** CS201 for random integer generation in multithreaded applications because it is

not thread-safe. Instead, **you must use the following thread-safe function, `random_range`, for generating random waiting times inside the threads**. This function returns a random number between `min` and `max` parameters. In the threads, you have call it by passing minimum and maximum waiting times of the corresponding thread as arguments.

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

int random_range(const int & min, const int & max) {
    static mt19937 generator(time(0));
    uniform_int_distribution<int> distribution(min, max);
    return distribution(generator);
}
```

To pause a thread for a certain amount of time, you should use the `this_thread::sleep_for(chrono::seconds(`*time_in_seconds*`))` command; you need to include the `thread` and `chrono` libraries in your program for this to work.

In each withdrawal transaction, first read the balance from the file. If the balance is greater than 0, decrement it by 1 TL, overwrite the new balance in the file and display a message for this successful withdrawal transaction. Otherwise, do not update the balance and display a message about this failed transaction.

In each deposit transaction, first read the balance from the file. Then, increment it by 1 TL, overwrite the new balance in the file and display a message for this transaction.

The messages that you display after each transaction are verbose ones that include thread details, success/failure (for withdrawal), current balance after transaction and current time. See "Sample Runs" for the message format. Here please remark that a single line of output from a particular thread must be displayed without being interleaved by the output of another thread.

At the end of your program, do not forget to join on four threads properly. Moreover, display a message to specify that the simulation has ended and the ending time.

Use of global variables

Finally, some good news; you may use global variables in this homework. Actually, it would be miserable not to use them in a program that has several threads. However, we kindly request you not to exaggerate the global usage since after a certain point you may lose control over your program (as the famous Turkish proverb says "azı karar, coğu zarar").

Sample Runs

Some sample runs are given below, but these are not comprehensive, therefore you have to consider all cases, to get full mark.

Due to the probabilistic nature of the homework and due to the scheduling of threads, same inputs may yield different outputs for your code. However, the order of the events must be consistent with the homework requirements and the given inputs.

The inputs from the keyboard are written in ***`boldface and italic`***.

Sample Run 1:

```
Please, enter file name in which balance will be stored:
a.txt
Please, enter min and max waiting times between withdrawal transactions per thread
(in seconds):
3 3
Please, enter min and max waiting times between deposit transactions (in seconds):
1 1
Please, enter the total number of withdrawal transactions per thread:
4
Please, enter the total number of deposit transactions:
10

Simulation starts at 00:21:34

1 TL has been deposited: balance is 1, 00:21:35

1 TL has been deposited: balance is 2, 00:21:36

Withdrawal thread 2 successfully withdrawn 1 TL; balance is 1, 00:21:37

Withdrawal thread 0 successfully withdrawn 1 TL; balance is 0, 00:21:37

Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:21:37

1 TL has been deposited: balance is 1, 00:21:37

1 TL has been deposited: balance is 2, 00:21:38

1 TL has been deposited: balance is 3, 00:21:39

Withdrawal thread 2 successfully withdrawn 1 TL; balance is 2, 00:21:40

Withdrawal thread 0 successfully withdrawn 1 TL; balance is 1, 00:21:40

Withdrawal thread 1 successfully withdrawn 1 TL; balance is 0, 00:21:40

1 TL has been deposited: balance is 1, 00:21:40

1 TL has been deposited: balance is 2, 00:21:41

1 TL has been deposited: balance is 3, 00:21:42

Withdrawal thread 2 successfully withdrawn 1 TL; balance is 2, 00:21:43

Withdrawal thread 0 successfully withdrawn 1 TL; balance is 1, 00:21:43

Withdrawal thread 1 successfully withdrawn 1 TL; balance is 0, 00:21:43

1 TL has been deposited: balance is 1, 00:21:43

1 TL has been deposited: balance is 2, 00:21:44

Withdrawal thread 2 successfully withdrawn 1 TL; balance is 1, 00:21:46

Withdrawal thread 0 successfully withdrawn 1 TL; balance is 0, 00:21:46

Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:21:46

Simulation ends at 00:21:46
```

Sample Run 2:

```
Please, enter file name in which balance will be stored:
a.txt
Please, enter min and max waiting times between withdrawal transactions per thread
(in seconds):
1 2
Please, enter min and max waiting times between deposit transactions (in seconds):
3 4
Please, enter the total number of withdrawal transactions per thread:
4
Please, enter the total number of deposit transactions:
10

Simulation starts at 00:20:09

Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:20:10

Withdrawal thread 0 failed to withdraw 1 TL; balance is 0, 00:20:10

Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:20:10

Withdrawal thread 0 failed to withdraw 1 TL; balance is 0, 00:20:11

Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:20:12

Withdrawal thread 0 failed to withdraw 1 TL; balance is 0, 00:20:12

Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:20:12

1 TL has been deposited: balance is 1, 00:20:13

Withdrawal thread 1 successfully withdrawn 1 TL; balance is 0, 00:20:13

Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:20:14

Withdrawal thread 0 failed to withdraw 1 TL; balance is 0, 00:20:14

Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:20:15

1 TL has been deposited: balance is 1, 00:20:16

Withdrawal thread 2 successfully withdrawn 1 TL; balance is 0, 00:20:16

1 TL has been deposited: balance is 1, 00:20:20

1 TL has been deposited: balance is 2, 00:20:24

1 TL has been deposited: balance is 3, 00:20:30

1 TL has been deposited: balance is 4, 00:20:34

1 TL has been deposited: balance is 5, 00:20:38

1 TL has been deposited: balance is 6, 00:20:41

1 TL has been deposited: balance is 7, 00:20:45

1 TL has been deposited: balance is 8, 00:20:48

Simulation ends at 00:20:48
Press any key to continue . . .
Press any key to continue . . .
```

Sample Run 3:

```
Please, enter file name in which balance will be stored:
a.txt
Please, enter min and max waiting times between withdrawal transactions per thread
(in seconds):
2 5
Please, enter min and max waiting times between deposit transactions (in seconds):
1 2
Please, enter the total number of withdrawal transactions per thread:
4
Please, enter the total number of deposit transactions:
10

Simulation starts at 00:18:44

1 TL has been deposited: balance is 1, 00:18:46

Withdrawal thread 1 successfully withdrawn 1 TL; balance is 0, 00:18:46

Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:18:46

Withdrawal thread 0 failed to withdraw 1 TL; balance is 0, 00:18:47

1 TL has been deposited: balance is 1, 00:18:47

1 TL has been deposited: balance is 2, 00:18:49

Withdrawal thread 0 successfully withdrawn 1 TL; balance is 1, 00:18:50

Withdrawal thread 1 successfully withdrawn 1 TL; balance is 0, 00:18:51

Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:18:51

1 TL has been deposited: balance is 1, 00:18:51

1 TL has been deposited: balance is 2, 00:18:52

Withdrawal thread 1 successfully withdrawn 1 TL; balance is 1, 00:18:53

1 TL has been deposited: balance is 2, 00:18:53

Withdrawal thread 0 successfully withdrawn 1 TL; balance is 1, 00:18:55

1 TL has been deposited: balance is 2, 00:18:55

Withdrawal thread 1 successfully withdrawn 1 TL; balance is 1, 00:18:56

Withdrawal thread 2 successfully withdrawn 1 TL; balance is 0, 00:18:56

1 TL has been deposited: balance is 1, 00:18:56

1 TL has been deposited: balance is 2, 00:18:57

1 TL has been deposited: balance is 3, 00:18:58

Withdrawal thread 0 successfully withdrawn 1 TL; balance is 2, 00:19:00

Withdrawal thread 2 successfully withdrawn 1 TL; balance is 1, 00:19:00

Simulation ends at 00:19:00
Press any key to continue . . .
```

Sample Run 4:

```
Please, enter file name in which balance will be stored:
a.txt
Please, enter min and max waiting times between withdrawal transactions per thread
(in seconds):
1 2
Please, enter min and max waiting times between deposit transactions (in seconds):
3 4
Please, enter the total number of withdrawal transactions per thread:
10
Please, enter the total number of deposit transactions:
4

Simulation starts at 00:17:04

Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:17:05

Withdrawal thread 0 failed to withdraw 1 TL; balance is 0, 00:17:05

Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:17:05

Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:17:06

Withdrawal thread 0 failed to withdraw 1 TL; balance is 0, 00:17:07

Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:17:07

1 TL has been deposited: balance is 1, 00:17:08

Withdrawal thread 2 successfully withdrawn 1 TL; balance is 0, 00:17:08

Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:17:08

Withdrawal thread 0 failed to withdraw 1 TL; balance is 0, 00:17:09

Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:17:09

Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:17:10

Withdrawal thread 0 failed to withdraw 1 TL; balance is 0, 00:17:11

Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:17:11

1 TL has been deposited: balance is 1, 00:17:12

Withdrawal thread 0 successfully withdrawn 1 TL; balance is 0, 00:17:12

Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:17:12

Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:17:12

Withdrawal thread 0 failed to withdraw 1 TL; balance is 0, 00:17:13

Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:17:13

Withdrawal thread 0 failed to withdraw 1 TL; balance is 0, 00:17:14

Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:17:14

Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:17:15

1 TL has been deposited: balance is 1, 00:17:16

Withdrawal thread 0 successfully withdrawn 1 TL; balance is 0, 00:17:16
```

```
Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:17:16

Withdrawal thread 0 failed to withdraw 1 TL; balance is 0, 00:17:17

Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:17:17

Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:17:17

Withdrawal thread 0 failed to withdraw 1 TL; balance is 0, 00:17:19

Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:17:19

Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:17:19

1 TL has been deposited: balance is 1, 00:17:20

Withdrawal thread 1 successfully withdrawn 1 TL; balance is 0, 00:17:21

Simulation ends at 00:17:21
Press any key to continue . . .
```

Sample Run 5:

```
Please, enter file name in which balance will be stored:
a.txt
Please, enter min and max waiting times between withdrawal transactions per thread
(in seconds):
2 4
Please, enter min and max waiting times between deposit transactions (in seconds):
1 2
Please, enter the total number of withdrawal transactions per thread:
10
Please, enter the total number of deposit transactions:
4

Simulation starts at 00:15:35

1 TL has been deposited: balance is 1, 00:15:37

Withdrawal thread 0 successfully withdrawn 1 TL; balance is 0, 00:15:37

Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:15:37

Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:15:38

1 TL has been deposited: balance is 1, 00:15:38

Withdrawal thread 1 successfully withdrawn 1 TL; balance is 0, 00:15:40

1 TL has been deposited: balance is 1, 00:15:40

Withdrawal thread 0 successfully withdrawn 1 TL; balance is 0, 00:15:40

Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:15:41

Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:15:42

1 TL has been deposited: balance is 1, 00:15:42

Withdrawal thread 0 successfully withdrawn 1 TL; balance is 0, 00:15:43

Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:15:44
```

```
Withdrawal thread 0 failed to withdraw 1 TL; balance is 0, 00:15:45

Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:15:46

Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:15:47

Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:15:48

Withdrawal thread 0 failed to withdraw 1 TL; balance is 0, 00:15:49

Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:15:50

Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:15:51

Withdrawal thread 0 failed to withdraw 1 TL; balance is 0, 00:15:51

Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:15:52

Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:15:53

Withdrawal thread 0 failed to withdraw 1 TL; balance is 0, 00:15:53

Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:15:55

Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:15:56

Withdrawal thread 0 failed to withdraw 1 TL; balance is 0, 00:15:56

Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:15:57

Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:15:59

Withdrawal thread 0 failed to withdraw 1 TL; balance is 0, 00:16:00

Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:16:00

Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:16:02

Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:16:02

Withdrawal thread 0 failed to withdraw 1 TL; balance is 0, 00:16:03

Simulation ends at 00:16:03
Press any key to continue . . .
```

Sample Run 6:

```
Please, enter file name in which balance will be stored:
a.txt
Please, enter min and max waiting times between withdrawal transactions per thread
(in seconds):
3 3
Please, enter min and max waiting times between deposit transactions (in seconds):
1 1
Please, enter the total number of withdrawal transactions per thread:
10
Please, enter the total number of deposit transactions:
4

Simulation starts at 00:13:19

1 TL has been deposited: balance is 1, 00:13:20

1 TL has been deposited: balance is 2, 00:13:21
```

```
Withdrawal thread 0 successfully withdrawn 1 TL; balance is 1, 00:13:22
Withdrawal thread 2 successfully withdrawn 1 TL; balance is 0, 00:13:22
Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:13:22
1 TL has been deposited: balance is 1, 00:13:22
1 TL has been deposited: balance is 2, 00:13:23
Withdrawal thread 0 successfully withdrawn 1 TL; balance is 1, 00:13:25
Withdrawal thread 2 successfully withdrawn 1 TL; balance is 0, 00:13:25
Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:13:25
Withdrawal thread 0 failed to withdraw 1 TL; balance is 0, 00:13:28
Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:13:28
Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:13:28
Withdrawal thread 0 failed to withdraw 1 TL; balance is 0, 00:13:31
Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:13:32
Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:13:32
Withdrawal thread 0 failed to withdraw 1 TL; balance is 0, 00:13:35
Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:13:35
Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:13:35
Withdrawal thread 0 failed to withdraw 1 TL; balance is 0, 00:13:38
Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:13:38
Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:13:38
Withdrawal thread 0 failed to withdraw 1 TL; balance is 0, 00:13:41
Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:13:41
Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:13:41
Withdrawal thread 0 failed to withdraw 1 TL; balance is 0, 00:13:44
Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:13:44
Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:13:44
Withdrawal thread 0 failed to withdraw 1 TL; balance is 0, 00:13:47
Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:13:47
Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:13:47
Withdrawal thread 0 failed to withdraw 1 TL; balance is 0, 00:13:50
Withdrawal thread 2 failed to withdraw 1 TL; balance is 0, 00:13:50
Withdrawal thread 1 failed to withdraw 1 TL; balance is 0, 00:13:50
```

```
Simulation ends at 00:13:50
Press any key to continue . . .
```

Please see the previous homework specifications for the other important rules and the submission guidelines

Good Luck!
Albert Levi, Stefan Räbiger