PDP - Lab 1

Statement Bank Account

At a bank, we have to keep track of the balance of some accounts. Also, each account has an associated log (the list of records of operations performed on that account). Each operation record shall have a unique serial number, that is incremented for each operation performed in the bank.
We have concurrently run transfer operations, to be executed on multiple threads. Each operation transfers a given amount of money from one account to some other account, and also appends the information about the transfer to the logs of both accounts.
From time to time, as well as at the end of the program, a consistency check shall be executed. It shall verify that the amount of money in each account corresponds with the operations records associated to that account, and also that all operations on each account appear also in the logs of the source or destination of the transfer.
Two transaction involving distinct accounts must be able to proceed independently (without having to wait for the same mutex).

Solution

Operation:
- serial number
- amount
- source id (the source account)
- destination id (the destination account)

Account:
- account id
- initial balance (read from the file)
- current balance (after performing each operation)
- mutex (for controlling concurrent access to the account)
- list of operations

readAccountsFromFIle():
- read the account information from file

generateRandomNumberInRange(min, max):
- generate a random integer number from the interval [min, max]

getRandomAccount():
- get a random account id from the accounts read from the file

getRandomAmount():
- get a random amount of money from the interval [10, 30]

generateRandomOperations(size):
- generate size random operations

performConsistencyCheck():
- go through every account
- lock the account
- from the initial balance, perform all the operations associated with the account and check if it is equal to the actual balance
- unlock the account
- if the condition is not met, set checkPassed to false

threadHandler(threadId):
- go through the associated operations and run them
- lock the source account and check if the transfer can be performed, if yes then update the source account information, otherwise go to the next operation
- if the operation can be performed, lock the second account and update its information
- check if it is time to perform a consistency check

main:
- check if the number of threads and the number of operations were provided
- read the accounts information from file
- generate the operations
- start the threads
- after the threads finished, perform a consistency check

Accounts are stored in a text file, having the following form: ID BALANCE
```
EX:  1 2000
     2 1500
     3 11000
```

Operations are randomly generated in the main thread and then are distributed evenly between the other threads.

Tests:
- 100 000 operations
  - 1 thread: 8.28836 seconds
  - 2 threads: 4.68282 seconds
  - 3 threads: 3.54427 seconds
  - 4 threads: 2.71864 seconds

- 5 threads: 2.41536 seconds
- 6 threads: 2.22078 seconds
- 7 threads: 2.09055 seconds
- 8 threads: 1.98733 seconds