Vamsi Kurkal and Armin Grossrieder

So this is how our program works:

The bank is represented as a hashtable of hashnodes which each point to an account struct

The bank has a mutex and so does each account struct.

On startup, the bank initializes everything and binds a socket to an open port on the machine its on

Begins bankserver print thread

It prints out domain name and port

Then, it starts listening.

After every connection, it’ll create a new thread to communicate with the connection.

In every connection handler thread, the program waits for a command

Balance, credit and debit will not open outside of an account session

Start <accountname> will try to start an account session with <accountname>.

The bank mutex is locked before and unlocked after the search.

Then it checks if the accountname is already being used, the program will send a waiting message, and retry every two seconds.

If <accountname> doesn’t exist, it’ll send error message.

If everything goes fine, the account mutex for <accountname> will be set.

Then, balance, credit and debit will work as planned.

Start and open will send error messages during an account session.

Finish will unlock the account mutex and leave the account session

Open <accountname> creates the account with name <accountname> in the server. It locks the bank mute before and unlocks after insertion attempt and fails if accountname already exists or if 20 accounts already exist. It does NOT start the session, the user must do that themselves.

Exit will have the client disconnect from the server. The server sees if they have an open account session, if they do, it unlocks the account mutex before ending the exit message to the client program.

Debit will fail if the operation causes a negative balance.

The print thread runs every 20 seconds. It locks the bank mutex during and unlocks after.

The print thread runs through the hash map and trys to lock every account mutex. If lock is achieved, it

Prints the balance, then unlocks mutex. If not acheived, it’ll print “In Session”

SIG-INT causes the server to send a server-wide exit message to all clients before exiting out of the program. Clients receive exit message and close down.

If the client disconnects unexpectedly, the server sees if they had an open account session, if they do it unlocks that mutex.

The client program takes two args, domain name and port number, both of which would be given at server start up. If connection fails, sleeps for two seconds and tries again.

The Client program has two threads, one for reading from the server and one for writing to. This allows asynchronous I/O so that the client can receive server messages at anytime.

The write to thread sleeps for two seconds, allows for an enter of a message, and sleeps for two seconds before sending the command out.