Stephen Quickel: svq3 and Masood Karami: mkk102 – Assignment 3

Server/Client banking connection

Server:

Main account struct has an integer inUse to determine if a session is active, a double balance to store a given accounts current balance, and an account Name which has a maximum of 255 characters and cannot be changed after creation.

int main(void)

Sets up initial socket for connecting to the client, and uses port number that is inputted

Then begins listening on this socked and once a client is accepted will create a new thread for handling this new client. All of the commands passed to the server from the client are dealt with in handle_connection which is the function that each new thread is created off of. This will initially create the proper signals as well as the timer to print out the accounts information at the proper intervals this also create a thread for the signal handlers both quitting out of the server as well as printing data sigalrm.

Void* handle_connection(void* arg)

This is the master method for dealing with all possible combinations of incoming messages. The argument is a socket connection to a client. Then given the client input of either serve create or quit it will go into the respective functions to handle the input or return an error to the client if a wrong input was entered. If serving an account the client can either deposit, withdraw, query, or quit from the account they are currently servicing, no two accounts are able to be serviced simultaneously.

int serve(char * accountName)

The serve message will taking in the accountName as requested by the client. If this name is a part of the account array the client will be able to service the account, and if not an error message will be displayed to inform the client that an account by that name does not exist or if the account is already in use they will be told it is currently being serviced and deny them access.

double deposit(double amount, int accountNum)

While currently servicing an account, the client can deposit funds into the accounts balance. When this is called it locks the mutex before adding the amount then unlocks the mutex before returning the balance to the client.

double withdraw(double amount, int accountNum)

While currently servicing an account, the client can withdraw funds from the accounts balance by locking the mutex then checking if the funds are available to withdraw. If there is enough the amount will be subtracted, then the new balance will be returned and if not an error will be returned to the user that there is not enough of a balance to withdraw the amount requested.

double query(int accountNum)

Querying simply returns the currently balance of account provided and does a simple mutex lock and unlock before the read.

void * signal_thread(void * arg)

This will be a switch statement to either deal with the sigint signal or the sigalrm which will use a semaphore to block new account creating while printing out values. The SIGINT signal will send a message to the clients to shutdown and join all threads back in addition to disconnecting socket and freeing memory allocation.

Client:

Opens a socket to connect to the server and attempts to connect based on the two input arguments from the command line which are the machine name and the port number. Then connects and goes into a while loop which will only break when the user either types quit or the server sends the message to shut down. The main method will create another thread for each of the client threads to deal with writing to the server in addition to reading.

void * handleInput (void * arg)

This is the new thread created to handle writing to the server so both writing and reading can be simultaneous

Assumptions: We assumed that there will not be more than 10000 accounts at any given time and that accounts could not be deleted once created since exiting from the server essentially erases all accounts.