Banking Account System

Using bankingServer

A program that will be used as a bank server for clients. The server will receive commands from the client and manage the accounts. Accounts store information on the account name, balance, and a flag of whether the client is in session. The server must handle account operations with synchronous locks to keep data consistent.

To use bankingServer:

./bankingServer <port>

- Port: the port that the server will be used on.

Using bankingClient

A program that will be used together with bankingServer. This program tries to establish a connection with the server. If the program cannot establish a connection, it will sleep for 3 seconds and then try again. When the client establishes a connect, the user can have 7 commands to use to give to the server.

Commands:

create <accountname (char) >: creates an account name

serve <accountname (char) > : allows account to be used for balance commands
deposit <amount (double) > : Adds amount to balance

withdraw <amount (double) > : Takes amount away from balance

Query : Prints the balance of the account in service

End: Removes in session flag from account

Quit: Disconnects client from server

To use bankingClient:

./bankingClient <name of machine> <port>

- Name of machine: the ip address of the machine
- Port: the port that the server will be used on.

Design

A global variable is used to hold all the account details in an account linked list, all sockets created in a linked list, and all threads created in a linked list. There signal handlers that catch SIGINT and SIGALRM, When the server receives a SIGINT, the server will shutdown, closing all clients, lock all accounts, join all threads, and stop the diagnostic output. When the server receives a SIGALRM (every 15 seconds), the server will lock the account linked list so no accounts can be created, and print a list of all the accounts on the server.

The server's socket is binded, and listen is used in a while loop to receive inputs. When a client connection is spotted by listen, the server will use accept on the connection and spawn a separate service thread to handle all commands from the client.

Assumptions

The user will not use ctrl+c on the client side. The user will not ctrl+q immediately after connecting to the server.

Difficulties

This project used a lot of string manipulation from user input and output, so often times there would be segmentation faults when memory outside of the reserved memory was accessed. Messing with IP addresses and connections was difficult at first since this is my first project on socket programming.