Jon Cobi Delfin and Jeffrey Pham
Systems Programming
Project 3: MultiThreaded Bank System

Design, Assumptions, and Difficulties

Starting the project was not as difficult as the previous projects since it was similar to the multithreaded CSV sorter with the addition of sockets. We began by trying to setup a simple server-client socket where we could send messages between the two. We found that there were already two structs within C that were established, addrinfo and sockaddr. We found the addrinfo struct first so that was what we ended up using, but the same thing could easily be accomplished with the sockaddr struct as well. With the addrinfo struct, we established its flags, family, and socktype so that they could connect to eachother. Then, we used the getaddrinfo function and passed in the user-inputted host name and port number to start the connection between the client and server.

Once we had that done, the client side of the program was fairly easy. There are two threads in the client side, one that sends information to the server and one that receives messages from the server. The error checking of the command inputs is done within the thread sent to the server.

As for the server, it creates as many threads as there are client connections. The threads main function is the server_client_interaction. It continuously reads input being sent from the client and according to what the input is, it will send back the appropriate message to the client, update the global account table accordingly, and store the socket file descriptor in the global linked list. Within the server, we also keep track of any accounts that are currently being serviced so that no other client may service that same account at the same time. Furthermore, an itimer set to 15 second intervals was used in tandem with a semaphore to lock the table in order to print out all the account data.

Upon receiving CTRL+C, the sockets are closed(if they weren't already closed with the 'quit' command) and nodes are freed. A message is also sent to all clients notifying them that the server is shutting down and subsequently shutting down the clients as well.

Using Our Program

To use our program, untar our file and call make on it. Then, have two or more terminals open(can be on same machine or different machines). One of the terminals must be the server, so call:

```
./bankingServer <port number>
```

On the other terminals, you must establish a client that connects to the server. To do this, you call:

```
./bankingClient <host name> <port number>
```

The host name will be one of the supported ilab machines, such as ls.cs.rutgers.edu and the port number must be a number above 8192 or it will give an error and not initialize the server. The port number must be the same as the ones called in setting up the server.

If the terminal keeps printing "Unable to find server", please wait around 10 seconds and re run the executables.