CS214 Project 3: Multithreaded Bank System

Hongju Shin: hs757, Jeeho ahn: ja709

December 12, 2018

1 contents of submission

1.1 bankingServer.c

bankingServer.c process spawn a single session-acceptor thread. A session-acceptor thread accepts client connections from multiple client processes and it is designed to handle different connections safely. For each new connection, the session-acceptor thread create client-service threads that handles a single client connection with synchoronized behavior. The bank server process maintains a simple bank with multiple accounts. Initially your bank will have no accounts. When clients create accounts bank server stores unlimited number of accounts. When the bank stores an account, it holds information of account name, current balance and In-session flag. The server creates each client in a separate client-service thread.

1.2 bankingClient.c

bankingClient.c seeks for a server running using socket(), gethostbyname(), and connect(). These helps to find the name of the machine running the server process and port as a command-line argument. The client process waits for attempts to connect to the server every three seconds when the matching server is not found. Once connected, the client process prompts for commands. The allowed commands are create ¡accountname (char) ¿, serve ¡accountname (char) ¿, deposit ¡amount (double) ¿, withdraw ¡amount (double) ¿, query, end, and quit. Command entry must be throttled every two seconds to give space between the commands inputs. This intentionally delay client interaction with the server and simulates many thousands of clients using the bank server. Your client implementation includes a commandinput thread to read commands from the user and send them to the server. Every response-output thread to read messages from the server and send them to the user.

2 Design of the code

The bank server diagonize a list of all accounts every 15 seconds using SIGALARM. The information is protected by synchronization with semaphore. New accounts can not be created while the bank is printing out the account information. The diagnoisis output uses timers, signal handlers and semaphores to achieve safe output while multiple clients interact with the server.
Synchronization behaviors are implemented in creating, and serving. While the account is being created, the name specified uniquely identifies the account and the new account is created only once at a time. This part requires mutex to lock the behavior since it forbid other clients from accessing other accounts against their will. for serving command, it has used the mutex lock as well for the equivalent issue.