Brandon Dunlap, Vikram Krishna

PLEASE NOTE: Samani Gikandi gave us extention until Monday April 20th , on Thursday April
16th during his office hours.
Systems Programming
Project Assignment 4 readme document

Usage: To invoke the server/client, run make and then in either order, invoke ./server and ./client computername. Another important thing to take note of is when client A is serving account B, if client C attempts this, they will be blocked but the command prompt will come up, DO NOT enter any commands till a message stays you can. Also negative numbers can be entered in serve, for some reason the atof does not make them negative but positive, but entering www will trigger an error. Similarly for the command prompt, it will sleep every 2 seconds in order to throttle but any commands entered will be sent to the server

Weird Things: When I was coding serve commands, for some reason when I tried to use a method for serve instead of writing the code in the client session thread, it completely broke. Granted I locked the account before invoking the method but it did not work until I transfer the code into the client session thread, another weird thing is it takes a while for the message to pop up indicating the server has accepted the client, about 5-10 seconds. A message will pop up but will take time.

I.
bank.h: header file that will be passing two structs and several methods to bank.c.
A.
struct Account: typedef as “account”; account represents each of the bank accounts that
will be stored in struct Bank. Members include a “name” for the account owner, represented by a
character array with a 100 character capacity. The account “balance” represented by float.
“service”, represented by an integer that will be either 1 or 0, indicating the server is serving the
account or not respectively. And finally a mutex lock is included so every account can be locked
when money is withdrawn or deposited.
B.
struct Bank: typedef as “bank”; represents a bank holding an array of a maximum 20 accounts, and integer counting the number of accounts created, and a mutex lock so the bank can
be locked when printing all account information.
C.
The following methods will be passed to bank.c
1.
account mallocAccount()
2.
bank mallocBank()
3.
int createAccount(char* name, int socket, bank x)
4.
int PrintoutBankInfo(bank x)
5.
int PromptSelector(char* prompt , char* promptarg, int socket,bank x)

6.
int serveAccount(char* name, int socket, float amount, bank x)
7.
int Quit()
8.
account findAccounttoServe(char* name, int socket, bank x)
II.
bank.c:
A.
account mallocAccount(): allocates an account to dynamic memory and returns it.
B.
bank mallocBank(): allocates a bank to dynamic memory and returns it.

C.
int createAccount(char* name, int socket, bank x): this method creates an account with
‘name’ as the ‘accountname’, and the balance and service state initialized to 0. During the
process, the method will write to the client using the ‘socket’ argument. If successful the method
will return 1 after; locking the bank, incrementing the number of accounts, adding the new
account to the bank, unlocking the bank, and writing a success prompt to the client. If
unsuccessful the method will return -1 and write an appropriate prompt to the client. The first
thing the method will do is check to see if the bank is printing information and therefore locked, a
prompt will be sent in this case and the user will have to wait. In the case of failure createAccount
will write a prompt to the client if the bank already has 20 accounts, or if the ‘accountname’ is
already in the bank.
D.
int PrintoutBankInfo(bank x) : returning a 1 upon success, the method will check if the
bank is already locked and wait if so. When the bank is not locked PrintOutBankInfo will lock the
mutex of the bank and proceed to print the name, balance, and service status of each account
using a for loop. When printing is complete it will unlock the mutex of the bank. NOTE: this
method was not utilized, there were issues calling it as method, so the code was transferred to
server.c.
E.
account findAccounttoServe(char* name, int socket, bank x): findAccounttoServe does
as the name suggests by locking the mutex of the bank, and searching the array of accounts, via a
for loop, for the account with an accountname that matches the argument ‘name’. Upon success it
unlocks the back and returns the matching account. Upon failure the bank is unlocked, a prompt
is written to the client, and the method returns NULL.
F.
int ServeSelector(char* prompt, char* accname, int socket, account serveaccount, float
amount): This method deals with the various serve commands, it is called directly when serve is
used. The method starts by setting the account’s service status to 1. NOTE: this method was not
utilized, there were issues calling it as method, so the code was transferred to server.c.
III.
client.c:
A.
void ServePrompt() and void Prompt(): print out the commands and approiate prompts.
B.

Void* ServerResponse(void *c): reads and prints out error conformation messages, and
shuts down the client if the server goes down.
C.
int isValid(char* prompt): checks to see if the user command is valid.
D.
Main: modified Professor Russel’s code to pass the command line prompts to the server.
It also can shut down the client if the quit command is entered.
IV.
server.c:
A.
void* periodic_action_cycle_thread(void * arg): This method prints out the bank information every 20 seconds, using a semaphore and timer(similar to russel's code). The code
enters a while loop which is blocked by sem wait, every 20 seconds a signal is sent which invokes

sem post() which unblocks the while loop and allows it to print out the information and the
process repeats

B.
void * client_session_thread(void* bank): This thread communicates with client, and
reads the arguments sent by the client to invoke the correct action on the bank. Depending on
which argument is sent, the thread either invokes createAccount() from bank.c or preforms the
serve actions in the main while loop. Also if the client enters quit on their side, the while loop
exits and this thread shuts down as there is no need for it