Operating System - Mini Project

IMT2017003 - Amogh Johri

May 2020

1 Introduction

An online banking-management system has been developed and the functionalities provided are as follows:

1.1 Stored Information

The database contains the following information :
1. For each account we store :
a) ID's connected to that account (The assumption is that multiple people can be dealing with the same account)
b) Account balance
c) Account password
Other book-keeping details which are stored are :
a.1) *this_id* (This is to manage the ID which is accessing the account at the current time-instance)
2. For the admin account we store :
a) ID's connected to that account (but this is always 1 and this account ID is also fixed to 1)
b) Account password
Other book-keeping details which are stored are :
a.1) *account_balance* (This variable stores the number of account structs present in the database file. This is used to manage the space efficiently, adding-deleting account combinations to not lead to a lot of holes in the database, etc)
a.2) *this_ids* (This is to manage the ID which is accessing the account, as of now this is also fixed to 1 for the case of administrator)

1.2 Types of User

There are two types of users involved :
1. Account Handlers (normal clients)
2. Administrator

2 Functionalities

2.1 User Functionalities

The following functionalities are provided to the user:
1. Login
2. Deposit Money
3. Withdraw Money
4. Change Password
5. View Account Balance
6. View Account Details
7. Logout

2.1.1 Login

The user can login to an existing account using the ID and password. The ID can be any of the ID's attached to the password but the password is unique to an account. The user can login from one ID from a maximum of 1 place at a time, i.e. two logins cannot be made simultaneously from the same ID (however, two logins into the same account can be made from different IDs).

2.1.2 Deposit Money

Having logged in, the user can deposit the money in their account. This will get reflected immediately for all the IDs attached to the account. Any amount of money can be deposited.
Potential Bug: The deposit can also consist of a negative sum which will be dealt as a withdraw. In case the total account balance ¡ 0, the action (whatever it might be, a negative deposit or a positive withdraw) shall not be carried out. However, the user shall not get informed of the failure of the action

2.1.3 Withdraw Money

Having logged in, the user can withdraw the money in their account. This will get reflected immediately for all the IDs attached to the account. The withdrawal should only consider of an amount such that the account balance = 0, otherwise an error message gets written on the user screen.
Potential Bug: Same as above.

2.1.4 Change Password

Having logged in, the user can change the password of the account that their ID is attached to. This will result in a change in the passwords for all the IDs attached to the account.

2.1.5 View Account Balance

Having logged in, the user can view the account balance. This shows the remaining balance of the account

2.1.6 View Account Details

Having logged in, the user can view the details of the account. This shows the following:
1. ID of the user calling the function.
2. Remaining balance in the account.
3. Whether the account is a single account or a joint account
4. Password for the account

2.1.7 Logout

Having logged in, the user can choose to logout. This closes the connection on the client side as well, however, the server continues to run.

2.2 Administrator Functionalities

The following functionalities are provided to the administrator:
1. Login (same as user)
2. Add Account
3. Delete Account
4. Modify Account
6. View Account Details
7. Logout (same as user)
8. Shut the Server Down

2.2.1 Add Account

Having logged in, the administrator can add accounts to the database. The accounts can be single-user or joint-accounts. The new IDs should not contain already existing ones (within the same account as well across accounts). Once the account has been created, the users associated with the account IDs can access the account.

2.2.2 Delete Account

Having logged in, the administrator can choose to delete an existing account. If the account which is to be deleted is being accessed (by one or more of the attached IDs) the server waits for the associated IDs to logout. Once they have logged out, the server deletes the account.

2.2.3 Modify Account

Having logged in, the administrator can choose to modify the account details of existing accounts. As of now, this only involves modifying the account's password (this is the way in which admin can change it's own password as well).

2.2.4 View Account Details

Having logged in, the administrator can choose to view the account details of existing accounts. This, for other IDs is the same as "View Account Details" feature for the user. For the administrator, instead of the balance amount it shows the number of accounts that are stored in the database.

2.2.5 Shut the Server Down

Having logged in, the administrator can indicate the server to shut down. The client process exists from the administrator's end. The server has been signalled to shut down and it should continue running as long as all the other clients exit. Once this happens, the server process exits as well.
Potential Bugs: Once the server is in the state where it has been signalled to shutdown but it is still catering the logged in clients, if there are any more clients which attempt to join, they get stuck in an infinite wait at the "connect" call.

3 Miscellaneous

1. The program **DOES NOT** provide for a stringent error checking mechanism, the goal was to implement the above mentioned functionalities. The program suffers from bugs such as - 'deleting the admin account', 'depositing/withdrawing from the account by providing a non-numeric input', etc. If the program exits through a SIGKILL, etc the behavior is undefined.
2. To provide for database efficiency, while deleting a record which exists somewhere in between the file, the memory is simply set as 0 for the entire range of the record, and while adding a new account, if such a 'hole' is encountered in the database, the new record gets added there.
3. Since we are providing for a system of authorization, the database has been saved as a binary file since these are safer than normal files.
4. If an account has verified its ID but not the password and it gets deleted mid-way, the client exits, but the exit message is - Invalid Password instead of Account Deleted.

4 How To Run

1. Compile and run the setup file. This will create a database and initializes it with 20 dummy records, 10 of which are single-user accounts and 10 are joint-user accounts (there is also an admin account present in the first position, so there are a total of 21 records).
2. All the records are given the IDs as single digit positive integers greater than 1, and their password is set as the smallest ID corresponding to the account. All the accounts have a default balance of 0.
3. Compile and run the server file. The server is meant to run in the background and it only outputs on STDERR or to the client.
4. Compile and run the client file. A total of 1024 connections can be made to the server between each server start and shutdown. Connections can be made simultaneously but two connections cannot be made through the same ID.
5. To shut the server down, login as the admin (ID = 1, Password = password) and select the appropriate option.

5 Testing

These are the following tests that have been done:
1. Add a single-user account and exit. (Admin)
2. View a single-user account and exit. (Admin)
3. Add a joint-user account and exit. (Admin)
4. View a joint-user account and exit. (Admin)
5. Logout and log-back in and exit. (Admin)
6. Delete a single-user account and exit. (Admin)
7. Delete a joint-user account and exit. (Admin)
8. View a deleted account (single and joint) - shows invalid ID. (Admin)
9. Remove a single account and add a joint account containing ID of the deleted single account (Admin).
10. Perform all 5 functions (Deposit, Withdraw, Balance Inquiry, View Account details, Password Change) and exit. (User)
11. Perform all 5 functions on an added account and exit. (User)
12. Accessing a deleted account - shows invalid ID. (User)
13. Deleting an account being accessed. (Concurrent)
14. Concurrently accessing the same account from two IDs. (Concurrent-User) (Tried all combinations of withdraw-deposit)
15. Trying to delete an account being accessed from more than one ID. (Concurrent)
16. Shutting the server down while users are still online. (Concurrent)
17. Shutting the server down while users are in between the login process. (Concurrent)
18. Delete all the accounts (except the admin). (Admin)
19. Create accounts having deleted all the accounts. (Admin)
20. Access accounts having deleted all the accounts. (Concurrent)