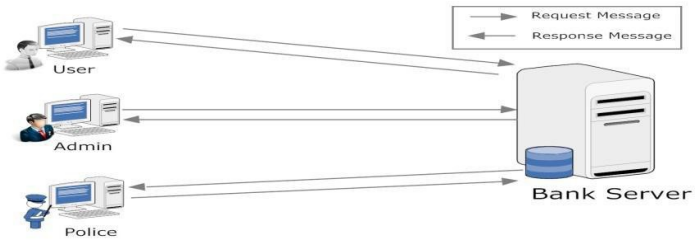
Application #1: *Banking System using Client-Server socket programming*

In this application, you require implementing two C programs, namely Client and Bank Server, and they communicate with each other based on TCP sockets. The goal is to implement a simple Banking System.



Initially, the client will connect to the bank server using the server's TCP port already known to the client. After successful connection, the client sends a Login Message (containing the Username and password) to the bank server. The client side, we can have three different types of user modes namely, Bank_Customer, Bank_Admin and Police. The bank server has the following files with him: Login_file (contains the login entries, assume limited number of static entries only), Customer_Account_files (Assume the bank has 10 Bank_Customers only and one file for each customer, which maintains the transaction history. Refer to Login_file and Customer_Account_files formats for more details). Once the Bank server receives the login request, it validates the information and performs the functionalities according to the user mode type. The system must provide the following functionalities to the following users:

- **Bank_Customer:** The customer should able to see AVAILABLE BALANCE in his/her account and MINI STATEMENT of his/her account.
- **Bank_Admin:** The admin should be able to CREDIT/DEBIT the certain amount of money from any Bank_Customer ACCOUNT (as we do it in a SBI single window counter.☺). The admin must update the respective "Customer_Account_file" by appending the new information. Handel the Customer account balance underflow cases carefully.
- **Police:** The police should only be able to see the available balance of all customers. He is allowed to view any Customers MINI STATEMENT by quoting the Customer _ID (i.e. User_ID with user_type as 'C').

Login_file entry format:

User_ID	Password	User_Type (C/A/P)

Customer _Account_files entry format:

Transaction_Date	Transaction Type (Credit/Debit)	Available Account_Balance

Implement the functionalities using proper REQUEST and RESPONSE Message formats. After each negotiation phase, the TCP connection on both sides should be closed gracefully releasing the socket resource. You should accept the IP Address and Port number from the command line (Don't use a hard-coded port number). Prototype for command line is as follows:

Prototypes for Client and Server

Client: <executable code><Server IP Address><Server Port number>
Server: <executable code><Server Port number>

NB: Please make necessary and valid assumptions whenever required.