PROTOCOALE DE COMUNICAȚIE : Tema #2
Sistem monetar de tip Internet Banking

Termen de predare: 05 MAI 2018

Responsabili Tema:
Elena APOSTOL, Razvan POPA
Giorgiana VLASCEANU, Andru MOCANU

Obiectivele Temei

Scopul temei este realizarea unui sistem pentru efectuarea de operaţiuni bancare. Obiectivele temei sunt:

- Înțelegerea mecanismelor de dezvoltare a aplicațiilor folosind *socketi* UDP si TCP.
- Dezvoltarea unei aplicații practice de tip client-server ce folosește *socketi*.

Enuntul Temei

Se dorește implementarea unui sistem pentru efectuarea de operaţiuni bancare online. În cadrul sistemului se consideră existența a două tipuri de entități: un *server bancar* care ofera doua servicii: banking (pentru interogare sold, tranfer numeral,...) si serviciul deblocare; și *clienți* care vor permite utilizatorilor accesarea facilităților oferite de server.

La pornire serverul primește două argumente: un numar de port și numele unui fișiere ce conține datele clienților, a cărei structură va fi explicată ulterior (Sectiunea Fisierul de date folosit de server). Modul de pornire al serverului este:

```
./server <port_server> <users_data_file>
```

Un client va primi ca argumente la pornire IP-ul și portul serverului. Modul de pornire al clientului este:

```
./client <IP_server> <port_server>
```

Codul pentru implementarea serverului și al clientului va fi secvențial (nu se vor folosi mai multe thread-uri) și se va folosi apelul *select* pentru multiplexarea comunicației (cu entitățile din sistem și/sau interfața utilizator).

Functionalitate

Serverul va oferi cele două servicii/functionalitati: internet banking (iBANK) și serviciul de deblocare. Pentru serviciul de tip iBANK, serverul va crea un socket *TCP* și va aștepta cereri de conexiune pe portul specificat. De asemenea, serverul va creea un socket *UDP* care va folosi acelasi numar de port (dat ca prim argument). Pe acest socket serverul va primi cereri de deblocare cont.

Un client se conectează (TCP) la serverul indicat de argumentele primite la pornire și așteaptă primirea de comenzi de la utilizator (prin *stdin*) pentru a-și exercita rolul de interfață/terminal în sistem. Întrucât ne dorim ca accesul la sistem să se poată face numai prin autentificare, serverul va ține o evidență a sesiunilor deschise de către clienți.

Între server și fiecare client conectat va fi prezentă o conexiune TCP care va servi drept flux de comandă în comunicarea dintre aceste două entități.

Fiecare client își va crea un fișier de log cu numele de forma *client-<ID>.log*, unde *ID* este ID-ul procesului prin care acesta a fost lansat.

NOTA$_1$: Pentru determinarea ID-ului procesului curent se poate folosi apelul funcției *getpid()* din *<unistd.h>*.

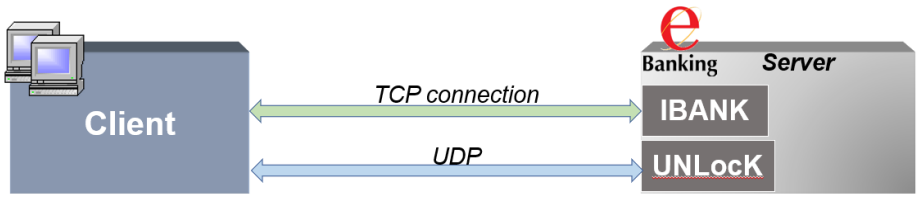

Figure 1: Funcționarea sistemului Internet Banking

Coduri eroare

Se vor folosi pentru afișare și logare un set de coduri de eroare. Acestea sunt următoarele:

- -1 : Clientul nu este autentificat *Explicație: Pentru a folosi o comandă diferită de login trebuie să fii autentificat.*
- -2 : Sesiune deja deschisa *Explicație: Fiecărui proces client îi este asociat o sesiune unică./ Pe server poate exista o singură sesiune deschisă pentru un anume număr de card.*
- -3 : Pin gresit
- -4 : Numar card inexistent
- -5 : Card blocat
- -6 : Operatie esuata *Explicație: Operația cerută de către client nu a putut fi efectuată cu succes*
- -7 : Deblocare esuata *Explicație: Deblocarea contului nu a putut fi efectuată cu succes.*
- -8 : Fonduri insuficiente *Explicație: Retragerea nu a reușit pentru că nu există suficienți bani în cont.*
- -9 : Operatie anulata
- -10 : Eroare la apel *nume-functie* *Explicație: Pentru erori întampinate la anumite apeluri (ex: bind, accept, connect, etc.). Acest tip de eroare se folosește la discreția voastră, cu scopul de a ușura procesul de debugging.*

Fisierul de date folosit de server

La pornirea serverul va citi dintr-un fișiere de date (*users_data_file*), dat ca argument în linia de comandă.

users_data_file va conține pe prima linie un număr natural N, iar pe următoare N linii seturi de informații ale clienților din sistem, în următorul format:

<nume> <prenume> <numar_card> <pin> <parola_secretă> <sold>

```
3
Popovici George 456123 8799 qwertyp4 9000.00
Ionescu Georgeta 111789 8997 123geo 11820.30
Iordache Mimi 678990 2356 m1m1klh 56000.50
```

Listing 1: Exemplu de fișier users_data

RESTRICTII:

- <nume> - șir de caractere alfabetice cu o lungime maximă de 12 caractere.

- <prenume> - șir de caractere alfabetice cu o lungime maximă de 12 caractere.
- <numar_card> - număr format din 6 cifre, acesta fiind unic pentru fiecare client.
- <pin> - număr format din 4 cifre.
- <parola_secretă> - șir de caractere alfa-numerice (fără spații), având lungimea maximă de 8 caractere.
- <sold> - număr double cu o precizie de două zecimale.

Comenzi Client

După conectarea la server (serviciul IBANK), un client poate primi un set de comenzi de la tastatură.

Orice comandă, împreună cu rezultatul ei se va scrie în fișierul de log al clientului. Pentru fiecare comandă scrierea se va face după executarea acesteia și aflarea rezultatului. Dacă operațiunea nu se execută cu succes se va întoarce un cod de eroare (dintre cele prezentate în Secțiunea Coduri eroare). Rezultatele comenzilor vor fi afișate și la consolă, cu scopul de a oferi un feedback utilizatorului.

$NOTA_1$: Orice raspuns primit de un client pe socketul de TCP va fi prefixat de sirul *IBANK>*.

$NOTA_2$: Orice raspuns primit de un client pe socketul de UDP va fi prefixat de sirul *UNLOCK>*.

Comenzile implementate de un client sunt următoarele:

1. ***login*** $< numar\_card > < pin >$
   Clientul trimite o cerere serverului (IBANK) pentru a realiza deschiderea unei sesiuni de lucru.
   Verificarea corectitudinii credidențialelor oferite se va face la server.
   În cazul în care în cadrul procesului client curent există deja o sesiune deschisă - cu ajutorul comenzii *login* -, clientul va returna codul de eroare -2 și **nu** va mai trimite comanda către server.

   În cazul în care într-o sesiune se oferă un set de credențiale incorect de 3 ori consecutiv, serverul va întoarce codul de eroare -5, si va bloca clientul cu acel numar de card. Orice operatie de logare ulterioara pentru acel card va intoarce codul de eroare -5. În acest caz, procesul client va trebui sa discute cu serviciul de deblocare (pe socket-ul UDP). (Vezi comanda *unlock*)

   $NOTA_1$: Pe server va exista la un moment dat doar o singură sesiune deschisă pentru un anumit număr de card. Dacă se incearcă logarea din alt proces client cu acelasi număr de card, **serverul** va intoarce codul de eroare **-2** (IBANK> -2 : Sesiune deja deschisa)
   Exemple functionare:

```
1  login 456127 8795
2  IBANK> -4 : Numar card inexistent
3
4  login 456123 87992
5  IBANK> -3 : Pin gresit
6
7  login 456123 8799
8  IBANK> Welcome Popovici George
9
10 login 123456 1234
11 IBANK> -2 : Sesiune deja deschisa
```

Listing 2: Exemplu 1 'login' din fisierul de log.

```
1  login 456123 8795
2  IBANK> -3 : Pin gresit
3
4  login 456123 8795
5  IBANK> -3 : Pin gresit
6
7  login 456123 8795
8  IBANK> -5 : Card blocat
9
10 login 456123 8799
11 IBANK> -5 : Card blocat
```

Listing 3: Exemplu 2 'login' din fisierul de log.

2. ***logout***
   Clientul va reseta sesiunea curentă și va anunța serverul (IBANK) de încheierea sesiunii. Serverul va raspunde cu un mesaj de *deconectare de la bancomat*. În cazul în care clientul nu este logat, se va întoarce codul de eroare -1 și nu se va trimite nicio solicitare către server.
   Exemplu de funcționare:

```
logout
IBANK> Clientul a fost deconectat

logout
-1 : Clientul nu este autentificat
```

Listing 4: Exemplu 'logout' din fișierul de log.

3. ***listsold***
   Permite unui client autentificat să vizualizeze soldul curent. Serverul va intoarce întotdeauna soldul în formatul *double cu doua zecimale*.
   Exemplu de funcționare:

```
listsold
IBANK> 7345.85
```

Listing 5: Exemplu 'listsold' din fișierul de log.

4. ***transfer*** $<$ *numar_card* $>$ $<$ *suma* $>$
   Clientul va solicita serverului (IBANK) transferul unei sume de bani catre numarul de cont specificat in comanda. Se considera ca suma poate fi data ca un numar intreg sau ca un double cu doua zecimale. Serverul verifica prima data daca numarul de cont al destinatarului este corect, in caz contrar va intoarce codul -4 . In pasul 2 serverul va verifica suma ce trebuie transferata. Aceasta trebuie să nu depășească suma deținută de client, altfel va intoace codul -8 . Daca cei doi parametri sunt corecti, i se va cere clientului sa confirme. Serverul nu trebuie sa se blocheze in asteptarea raspunsului de la client. Daca raspunsul clientului este *'y'*, serverul va scadea suma ceruta din contul clientului si o va adauga la contul destinatarului; dupa care anunta clientul ca transferul a fost realizat cu succes.
   $NOTA_1$: Se presupune ca de la tastatură se va citi mereu un număr cu cel mult doua zecimale.
   $NOTA_2$: 'y' reprezinta confirmare, orice altceva se considera anulare. Serverul va lua in considerare doar prima litera (caracter) din raspunsul clientului.
   Exemplu de funcționare:

```
listsold
IBANK> 8000.40

transfer 111788 9000
IBANK> -4 : Numar card inexistent

transfer 111789 9000
IBANK> -8 : Fonduri insuficiente

transfer 111789 2000
IBANK> Transfer 2000 catre Ionescu Georgeta? [y/n]
z
IBANK> -9 : Operatie anulata

transfer 111789 2000.10
IBANK> Transfer 2000.10 catre Ionescu Georgeta? [y/n]
y
IBANK> Transfer realizat cu succes

listsold
IBANK> 4000.30
```

Listing 6: Exemplu 'transfer' din fișierul de log.

5. ***unlock***
   Aceasta comanda va fi trimisă către server (*serviciul de deblocare*), pe socket-ul UDP. Înainte de a fi trimisă la server, se va adauga la mesaj (din program) numărul de card.

$NOTA_1$: Către server (serviciul de deblocare) se va trimite de fapt mesajul:
"unlock <*numar_card*>".
$NOTA_2$: Numarul de card ce trebuie adaugat in client este cel corespunzator ultimei comenzi de *login* apelate !
$NOTA_3$: **In testare se garanteaza ca in prealabil a fost apelata cel putin o data comanda *login*!**

În funcție de caz, serverul (*serviciul de deblocare*) va raspunde cu:

- $CAZ1$ UNLOCK> Cod eroare -4 în cazul în care nu există client cu acel număr de card.
- $CAZ2$ UNLOCK> Trimite parola secretă Se cere parola secretă a utilizatorului. Ca răspuns clientul va tasta parola sa secretă.
  $NOTA_4$: Către server (serviciul de deblocare) se va trimite de fapt mesajul:
  "<*numar_card*> <*parola_secreta*>".
  Dacă parola este corectă, serverul va debloca clientul (cu respectivul numar de card) și va raspunde cu mesajul "UNLOCK> Card deblocat", altfel va trimite codul de eroare -7: "UNLOCK> -7 : Deblocare eșuata"
- $CAZ3$ Daca acel numar de card **nu este blocat**, serverul va trimite codul de eroare -6

La încheierea cu succes a operației de deblocare, clientul se va putea conecta din nou la server (IBANK), folosind comanda *login*.
Exemplu de funcționare:

```
1  login 456123 8799
2  IBANK> -5 : Card blocat
3
4  unlock
5  UNLOCK> Trimite parola secreta
6  dgjkdf
7  UNLOCK> -7 : Deblocare esuata
8
9  unlock
10 UNLOCK> Trimite parola secreta
11 qwerty
12 UNLOCK> Card deblocat
13
14 login 456123 8799
15 IBANK> Welcome Popovici George
16
17 unlock
18 UNLOCK> -6 : Operatie esuata
```

Listing 7: Exemplu 'unlock' din fișierul de log.

6. ***quit***
   Clientul trimite serverului (IBANK) mesajul 'quit', prin care anunță că se va deconecta, apoi închide conexiunea cu serverul.

NOTA:

- Păstrați același format de afișare cu cel din exemplele de *logfile* date în această secțiune.
- După autentificarea clientului, comenzile pot fi date în orice ordine.
- Comenzile și răspunsurile vor fi scrise în fișierul de log fără linii libere între ele.

Comenzi Server

Serverul poate primi de la tastatură doar comanda *quit*, comandă ce va închide serverul. Înainte de închiderea execuției, serverul anunță clienții conectați de intenția sa de închidere.

Cerinte privind implementarea temei

Tema (*client* și *server*) va fi realizată folosind sockets stream/datagram (peste TCP și UDP) în C sau C++.

Apelurile de sistem și mecanismele necesare pentru realizarea temei sunt descrise pe larg în suportul de curs și în cadrul laboratoarelor de socketi UDP și TCP.

Formatele de mesaje și protocolul de comunicație folosit în implementarea aplicației trebuie să fie descrise în fișierul *Readme* (cu justificare asupra alegerii). Pentru multiplexarea comunicației folosiți apelul *select* (studiat în cadrul laboratorului 8). Nu aveți voie să folosiți crearea de procese sau fire de execuție. Rezumați-vă la folosirea apelului *select*.

Testare si notare

Arhiva trebuie să aibă numele conform regulamentului și trebuie să conțină pe lângă sursele C/C++:

- Makefile cu target-urile **build** și **clean**
- README în care să se specifice modul de implementare al temei

Este obligatoriu ca numele celor 2 executabile să fie *server* și *client* și să primească argumentele în ordinea specificată în enunț.

Nerespectarea cerințelor de mai sus conduce la necorectarea temei!

Tema se va puncta astfel:

- *client:login* : 20p
- *client:logout* : 15p
- *client:listsold* : 15p
- *client:transfer* : 20p
- *client:unlock* : 20p
- *client:quit* : 5p
- *server:quit* : 5p

Comenzile de la clienți sau server sunt punctate dacă sunt implementate în totalitate și funcționeaza conform cu specificațiile.

Tema nu va fi testata pe *vmchecker*.