PROTOCOALE DE COMUNICATIE: Tema #3
Sistem de Partajare a Fisierelor

Termen de predare: 5 MAI 2015

Titulari curs: *Valentin CRISTEA, Gavril GODZA, Florin POP*

Responsabil Tema: **Elena APOSTOL**

Obiectivele Temei

Scopul temei este realizarea unui sistem de partajare a fisierelor in retea. Obiectivele temei sunt:

- Intelegerea mecanismelor de dezvoltare a aplicatiilor folosind *sockets*.
- Dezvoltarea unei aplicatii practice de tip client-server ce foloseste *sockets*.

Enuntul Temei

Se doreste implementarea unui sistem de partajare a fisierelor in retea. In cadrul sistemului se considera existenta a doua tipuri de entitati: clienti ce partajeaza fisiere si un server central ce ajuta la descoperirea clientilor si a fisierelor pe care acestia le partajeaza.

La pornire serverul va primi ca parametru un port pe care va asculta cereri de conexiune. Modul de apelare al serverului este:

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

Un client va primi ca parametru al executiei un nume cu ajutorul caruia se va identifica in sistem, numele unui director ce contine fisierele partajate, un port pe care va astepta conexiuni din partea altor clienti, ,si adresa si portul serverului central de descoperire (se considera existenta unui singur astfel de server in sistem). Numele clientului va contine doar caractere alphanumerice. Formatul de apelare a clientului este:

```
./client <nume_client> <nume_director> <port_client> <ip_server> <port_server>
```

Serverul va fi folosit pentru descoperirea clientilor conectati iar trimiterea de fisiere se va realiza direct intre clienti. Atat serverul cat si clientii vor porni un *server socket* si vor folosi apelul *select* pentru multiplexarea comunicatiei.

Functionalitate

La pornire, serverul va crea un socket si va astepta cereri de conexiune pe portul specificat. Un client se conecteaza la serverul central si trimite datele de identificare (numele primit la pornire, portul pe care asculta eventuale cereri de conexiune). Serverul poate raspunde cu accept sau reject, in functie de numele clientului (daca deja exista un client in sistem cu respectivul nume deoarece nu sunt admise duplicate de nume). Serverul va retine pentru fiecare client conectat adresa IP, numele si portul pe care acesta asculta.

Fiecare client isi va crea la pornire un fisier de log cu numele *<nume_client>.log*.

Coduri eroare

Se vor folosi pentru afisare si logare un set de coduri de eroare. Acestea sunt urmatoarele:

- -1 : Eroare la conectare
- -2 : Eroare la citire/scriere pe socket
- -3 : Client inexistent
- -4 : Client necunoscut
- -5 : Fisier inexistent
- -6 : Fisier partajat cu acelasi nume

Comenzi Client

Dupa conectarea la server un client poate primi un set de comenzi de la tastatura.

Orice comanda impreuna cu rezultatul ei se va scrie in fisierul de log. Pentru fiecare comanda, scrierea se va face dupa executarea acesteia si aflarea rezultatului. Daca comanda nu se executa cu succes se va intoarce un cod de eroare (prezentate in Sectiunea Coduri eroare). Rezultatele comenzilor vor fi afisate si la consola, cu scopul de a oferi un feedback utilizatorului.

Comenzile implementate la client sunt urmatoarele:

1. ***infoclients***
   Clientul trimite o cerere serverului care ii va intoarce lista tuturor clientilor conectati si informatii de conectare. Pentru fiecare client se va primi numele clientului, IP-ul cu care s-a conectat la server, portul pe care acesta asculta. Informatia va fi scrisa in fisierul de log, si va fi memorata pentru a fi folosita ulterior pentru conectare la un anume client (pentru transfer de fisiere). Clientii vor fi afisati in ordinea conectarii la server.
   La fiecare apel al comenzii *infoclients* se va updata local informatia referitoare la clientii activi pe server.
   Exemplu de functionare:

```
client1> infoclients
client1 127.0.0.1 10011
client2 127.0.0.1 10012
client3 127.0.0.1 10013
```

Listing 1: Exemplu 'infoclients' din fisierul de log.

2. ***getshare*** $<$ *nume_client* $>$
   Se trimite un mesaj serverului prin care se cere lista fisierelor partajate de catre un anumit client.
   Pentru fiecare fisier trebuie sa afisati numele si dimensiunea intr-un format user-friendly (nu afisati 1024B ci 1KiB). *Cand transformati dimensiunea din Bytes(pentru a fi afisata), afisati doar partea intreaga Exemplu pentru* ***304771B*** *afisati* ***297KiB***. Considerati ca un fisier poate avea dimensiunea maxim pana la ordinul de GiB.
   Serverul intoarce Cod de eroare -3 in cazul in care clientul nu exista.
   Exemplu de functionare:

```
client3> getshare client2
3
Readme.txt 12KiB
movie.avi 647MiB
tema.c 787B
```

Listing 2: Exemplu 'getshare' din fisierul de log.

3. ***share*** $<$ *nume_fisier* $>$
   Se trimite un mesaj serverului prin care anunta faptul ca se partajeaza un nou fisier.

La primirea acestei comenzi de la terminal, e necesar adaugarea (din program) la sfarsitul mesajului a dimensiunii fisierului. Recomand pentru aflarea dimensiunii instructiunea *stat*(vezi *man 2 stat*).
Serverul doar va inregistra numele si dimensiunea noului fisier in lista asociata clientului respectiv.
$NOTA_1$: Se pot partaja doar fisiere din directorul corespunzator clientului (cel dat ca parametru la executie). In caz contrar se intoarce codul de eroare -5 (iar comanda NU se va mai trimite serverului).
Exemplu de functionare:

```
client2> share picture.png
Succes
client2> share enunt.pdf
-5 : Fisier inexistent
```

Listing 3: Exemplu 'share' din fisierul de log.

$NOTA_2$: Catre server se va trimite de fapt mesajul: "share picture.png $< DIMENSIUNE >$".
Paramentru $< DIMENSIUNE >$ poate fi considerat ca fiind dimnesiunea fisierului in **Bytes**.
$NOTA_3$: Clientul poate partaja de oricate ori un fisier cu acelasi nume. Se memoreaza informatia de la ultimul apel de *share*.
$NOTA_4$: Este necesar sa se mentina lista fisierelor partajate si local fiecarui client.

4. ***unshare*** $< nume\_fisier >$
   Se trimite un mesaj serverului continand numele fisierului care trebuie sters din lista fisierelor partajate.
   Serverul poate intoarce *Succes* sau Cod de eroare -5.
   Exemplu de functionare:

```
client2> unshare lab.pdf
-5 : Fisier inexistent
client2> unshare picture.png
Succes
```

Listing 4: Exemplu 'unshare' din fisierul de log.

5. ***getfile*** $< nume\_client >$ $< nume\_fisier >$
   Se cere transferul unui fisier de la un alt client. Acest lucru se face printr-o conexiune directa intre cei doi clienti.
   Trimiterea unui fisier trebuie facuta in segmente de maxim **1024B**. Intre doua bucati transferate trebuie sa se verifice daca exista o noua comanda de executat din partea utilizatorului. Este nevoie de acest mecanism pentru ca un client sa nu fie blocat in timpul transferului si sa poata sa primeasca si alte comenzi. Un client e necesar sa functioneze cu pana la 3 trimiteri si 3 receptionari de fisiere in derulare in acelasi timp.

   Clientul va salva fisierul in directorul corespunzator lui. Fisierul va fi salvat sub acelasi nume. Daca clientul care a facut cererea are partajat un fisier cu acelasi nume, *getfile* va intoarce Codul de eroare -6. Daca fisierul exista, dar NU e partajat acesta se va suprascrie.

   La terminarea transferului se va afisa "Succes $nume\_fisier$" si se va loga in *logfile* comanda impreuna cu mesajul de succes.
   Exemplu de functionare:

```
client1> getfile client5 Readme1.txt
-4 : Client necunoscut
client1> getfile client2 Readme1.txt
-5 : Fisier inexistent
client1> getfile client2 Readme.txt
-6 : Fisier partajat cu acelasi nume
client1> unshare Readme.txt
Succes
client1> getfile client2 Readme.txt
Succes Readme.txt
```

Listing 5: Exemplu 'getfile' din fisierul de log.

$NOTA_1$: Cererea de transfer fisier se poate da doar unui client cunoscut, a carui informatie este retinuta in prealabil din rezultatul comenzii *infoclients*. Daca intre timp clientul de la care se cere fisierul s-a deconectat, fara ca local sa se re-apeleze comanda *infoclients*, *getfile* va intoarce Codul de eroare -1 (acelasi cod intors si de intructiunea "connect" (Vezi *man 2 connect*)). Exemplu:

```
client1> infoclients
client1 127.0.0.1 10011
client2 127.0.0.1 10012
client3 127.0.0.1 10013
client1> getfile client2 Readme.txt
-1 : Eroare la conectare
client1> infoclients
client1 127.0.0.1 10011
client3 127.0.0.1 10013
```

Listing 6: Exemplu 2 'getfile' din fisierul de log.

6. ***history***
   Aceasta comanda va afisa si loga lista fisierelor trimise catre alti clienti. O inregistrare va contine numele clientului care a facut cererea impreuna cu fisierul cerut. Aceasta se adauga in istorie doar dupa trimiterea cu succes a fisierului catre clientul care l-a cerut. Trebuie sa se permita memorarea a cel putin 20 de inregistari.
   Exemplu de functionare:

```
client2> history
3
client1 enunt.pdf
client1 pic1.jpg
client3 Readme.txt
```

Listing 7: Exemplu 'history' din fisierul de log.

7. ***quit***
   Clientul trimite un mesaj serverului prin care anunta ca va parasi sistemul, termina de trimis fisierele care incepuse sa le trimita, apoi inchide toate conexiunile si iese.

NOTA:

- Pastrati acelasi format de afisare cu cel din exemplele de *logfile* date in aceasta sectiune. In fata oricarei comenzi scrise in *logfile* se va adauga sirul *nume_client* >.
- De la terminal comenzile pot fi date in orice ordine.
- Comenzile si raspunsurilor vor fi scrise in fisierul de log fara linii libere intre ele.

Comenzi Server

Serverul poate primi de la tastatura doar comanda *quit*, care inchide serverul. Cand clientii detecteaza ca a fost inchisa conexiunea cu serverul vor iesi si ei.

Cerinte Privind Implementarea Temei

Tema (*client* si *server*) va fi realizata folosind sockets stream (peste TCP) in C sau C++.

Apelurile de sistem si mecanismele necesare pentru realizarea temei sunt descrise pe larg in suportul de curs si in cadrul laboratorului de socketi TCP.

Formatele de mesaje si protocolul de comunicatie folosit in implementarea aplicatiei trebuie sa fie descrise in fisierul *Readme* (cu justificare asupra alegerii). Pentru multiplexarea comunicatiei folositi apelul *select* (studiat in cadrul laboratorului). Nu aveti voie sa folositi crearea de procese sau fire de executie. Rezumati-va la folosirea apelului *select*.

Testare si Notare

Arhiva trebuie sa aiba numele conform regulamentului si trebuie sa contina pe langa sursele C:

- Makefile cu target-urile **build** si **clean** obligatoriu
- README in care sa se specifice modul de implementare a temei

Nerespectarea cerintei de mai sus conduce la necorectarea temei.

Tema se va puncta astfel:

- *Readme + Makefile* : 5p
- *client:infoclients* : 10p
- *client:getshare* : 10p
- *client:share* : 10p
- *client:unshare* : 10p
- *client:getfile* : 30p
- *client:history* : 15p
- *client:quit* : 5p
- *server:quit* : 5p

Comenzile de la clienti sau server sunt punctate daca sunt implementate in totalitate si functioneaza conform cu specificatiile.

Tema nu va fi testata pe *vmchecker*.