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

Termen de predare: 28 APRILIE 2014

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 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). 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*.

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.

Codurile de eroare folosite sunt urmatoarele:

- -1 : Client inexistent
- -2 : Fisier inexistent
- -3 : Exista fisier local cu acelasi nume
- -4 : Eroare la conectare
- -5 : Eroare la citire de pe socket

Rezultatele comenzilor vor fi afisate si la consola, cu scopul de a oferi un feedback utilizatorului.

Comenzile implementate la client sunt urmatoarele:

1. *listclients*
   Clientul trimite o cerere serverului care ii va intoarce lista tuturor clientilor conectati.
   Exemplu de functionare:

```
> listclients
client1
client2
client3
```

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

2. *infoclient* < *nume_client* >
   Aceasta comanda cere serverului informatii suplimentare despre un client. Serverul va intoarce numele clientului, IP-ul cu care s-a conectat la server, portul pe care acesta asculta si timpul cand s-a conectat la server / SAU Cod de eroare −1, in cazul in care clientul nu exista.
   Informatia va fi scrisa in fisierul de log, si va fi memorata pentru a fi folosita ulterior pentru conectare la acel client (pentru transfer de fisiere).
   Exemplu de functionare:

```
> infoclient client2
client2 127.0.0.1 10012 09:51:57 AM
```

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

3. *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 1KB). *Cand transformati dimensiunea din Bytes(pentru a fi afisata), afisati doar partea intreaga Exemplu pentru **304771B** afisati **297KB**.*
   Serverul intoarce Cod de eroare −1 in cazul in care clientul nu exista.
   Exemplu de functionare:

```
> getshare client2
Readme.txt 12KB
movie.avi 647MB
tema.c 787B
```

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

4. *infofile* < *nume_fisier* >
   Clientul va interoga serverul central asupra identitatii unui client/clientilor care partajeaza un fisier avand respectivul nume.

Serverul intoarce cod de eroare -2 in cazul in care fisierul nu e partajat de nimeni.
Exemplu de functionare:

```
> infofile Readme.txt
client2 Readme.txt 12KB
client4 Readme.txt 12KB
client5 Readme.txt 17KB
```

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

5. *sharefile* $<$ *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 -2 (iar comanda NU se va mai trimite serverului).
Exemplu de functionare:

```
> sharefile picture.png
Succes

> sharefile enunt.pdf
-2 : Fisier inexistent
```

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

$NOTA_2$: Catre server se va trimite de fapt mesajul: "sharefile picture.png $< DIMENSIUNE >$".
Paramentru $< DIMENSIUNE >$ poate fi considerat ca fiind dimnesiunea fisierului in **Bytes**.

6. *unsharefile* $<$ *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 -2.
Exemplu de functionare:

```
> unsharefile lab.pdf
-2 : Fisier inexistent

> unsharefile picture.png
Succes
```

Listing 6: Exemplu 'unsharefile' din fisierul de log.

7. *getfile* $<$ *nume_client* $>$ $<$ *nume_fisier* $>$
Se transfera un fisier de la un alt client. Acest lucru se face printr-o conexiune directa la clientul sursa.
Trimiterea unui fisier trebuie facuta in segmente de maxim **1024B** si intre doua bucati transferate sa verificati daca aveti cumva 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.
Clientul va salva fisierul in directorul corespunzator lui. Fisierul va fi salvat sub acelasi nume.
Daca exista deja local un fisier avand respectivul nume este interogat utilizatorul daca doreste suprascrierea vechiului fisier sau renuntarea la operatie.
Exemplu de functionare:

```
> getfile client8 Readme.txt
-1 : Client inexistent

> getfile client2 Readme.txt
-3 : Exista fisier local cu acelasi nume
Suprascris

> getfile client4 Readme.txt
-3 : Exista fisier local cu acelasi nume
Comanda anulata

> getfile client2 movie.avi
Succes
```

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

$NOTA_1$: Transferul de fisier se poate face doar cu un client cunoscut: asupra caruia s-a dat in prealabil comanda *infoclient*.
$NOTA_2$: In cazul unui fisier care exista deja local, urmatorii pasi trebuie executati la consola clientului:

```
      getfile client2 Readme.txt
      -3 : Exista fisier local cu acelasi nume.
      Suprascriere? y/n?
      y

```

8. *quit*
   Clientul trimite un mesaj serverului prin care anunta ca va parasi sistemul, inchide toate conexiunile si iese.

NOTA: Pastrati acelasi format de afisare cu cel din exemplele de *logfile* date in aceasta sectiune.

Comenzi Server

Serverul poate primi de la tastatura doar urmatoarele comenzi:

1. *status*
   Afiseaza in consola lista clientilor conectati, adresele lor ip si porturile pe care acestia asculta.
2. *quit*
   Se inchide serverul. Cand clientii detecteaza ca a fost inchisa conexiunea cu serverul vor iesi si ei.

Serverul si clientii pot apela comenzile in orice ordine. Serverul central trebuie sa mentina intern un socket pe care asculta cereri de noi conexiuni din partea clientilor. Un client odata conectat si inregistrat poate mentine conexiunea respectiva si o poate refolosi pentru schimbul de mesaje. Pentru primirea de comenzi de la tastatura in paralel cu mesajele se poate adauga file descriptor-ul pentru *stdin* in *select*.

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 de catre asistent.

Tema se va puncta astfel:

- *Readme + Makefile* : 10p
- *client:listclients* : 5p
- *client:infoclient* : 10p
- *client:getshare* : 10p
- *client:infofile* : 10p
- *client:sharefile* : 5p
- *client:unsharefile* : 5p
- *client:getfile* : 30p
- *client:quit* : 5p
- *server:status* : 5p
- *server:quit* : 5p

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