

PROTOCOALE DE COMUNICATIE: Tema #2

Sistem de Backup și Partajare a Fișierelor

Termen de predare: 30 APRILIE 2016

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

Responsabili Tema: **Elena APOSTOL, Dorinel FILIP, Loredana GROZA**

Obiectivele Temei

Scopul temei este realizarea unui sistem de stocare și partajare a fișierelor în rețea. Obiectivele temei sunt:

- Înțelegerea mecanismelor de dezvoltare a aplicațiilor folosind *socket*ii.
- Dezvoltarea unei aplicații practice de tip client-server ce folosește *socket*ii.

Enunțul Temei

Se dorește implementarea unui sistem de stocare și partajare a fișierelor în rețea. În cadrul sistemului se consideră existența a două tipuri de entități: un server care va face managementul și stocarea fișierelor partajate în rețea și clienți care vor permite utilizatorilor accesarea facilităților oferite de server.

La pornire serverul primește ca parametri 3 valori: un port pe care va asculta cereri de conexiune și numele a 2 fișiere de configurare a căror utilitate va fi explicată ulterior (Secțiunea Fișiere folosite în configurarea inițială a serverului). Modul de apelare al serverului este:

```
./server <port_server> <users_config_file> <static_shares_config_file>
```

Un client va primi ca parametru al execuției IP-ul serverului și portul pe care acesta ascultă. Formatul de apelare a clientului este:

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

Serverul va juca rolul de storage pentru fișiere. (În rețea pot exista mai multe servere, însă acestea vor lucra independent, iar un client va fi conectat la un singur server la un moment dat.)

Ambii algoritmi (rulați de către server și clienți) vor fi implementați secvențial (fără utilizarea de mai multe thread-uri) și vor folosi apelul select pentru multiplexarea comunicației (cu membrii sistemului și/sau interfața utilizator).

Funcționalitate

La pornire, serverul va crea un socket și va aștepta cereri de conexiune pe portul specificat. Un client se conectează la serverul indicat de parametrii primiți la execuție și așteaptă primirea de comenzi de la utilizator (prin stdin) pentru a își exercita rolul de interfață/terminal în sistem. Întrucât ne dorim ca accesul la sistem să se poată face numai cu 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 2 entități.

~~Pentru fiecare fișier transferat se va crea, adițional, o conexiune TCP la server.~~

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

NOTA₁: Pentru determinarea ID-ului procesului curent se poate folosi apelul funcției *getpid()* din *<unistd.h>*.

Coduri eroare

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

- **-1** : Clientul nu e autentificat *Explicatie: Pentru a folosi această comandă trebuie să fii autentificat*
- **-2** : Sesiune deja deschisa *Explicatie: Nu poți deschide 2 sesiuni în cadrul aceluiași client*
- **-3** : User/parola gresita
- **-4** : Fisier inexistent
- **-5** : Descarcare interzisa *Explicatie: Nu ai dreptul de a descărca acest fișier*
- **-6** : Fisier deja partajat
- **-7** : Fisier deja privat
- **-8** : Brute-force detectat
- **-9** : Fisier existent pe server *Explicatie: Fișierul există deja pe server*
- **-10** : Fisier in transfer *Explicatie: Fișier aflat momentan in transfer*
- **-11** : Utilizator inexistent

Fișiere folosite în configurarea inițială a serverului

La lansarea în execuție serverul va citi 2 fișiere de configurare, indicate prin parametri în linie de comandă. Acestea au următoarea semnificație și folosire:

- *users_config* va conține pe prima linie un număr natural *N*, iar pe următoarele *N* linii seturile de credențiale (username și parolă) ale utilizatorilor din sistem

```
1 3
2 Doru protocoale
3 Elena 123456
4 Loredana qwerty
```

Listing 1: Exemplu de fișier *users_config*

- *shared_files* va conține pe prima linie un număr natural *M*, iar pe următoarele *M* linii câte un fișier partajat de un utilizator, în următorul format: *<nume_utilizator>:<Nume_fișier>*

```
1 3
2 Doru:Tema3_PC.pdf
3 Elena:Catalog.xlsx
4 Elena:Fisiere_laborator.zip
```

Listing 2: Exemplu de fișier *shared_files*

RESTRICTII:

- Numele de utilizator și parolele sunt șiruri de caractere alfa-numerice (fără spații), având lungimea maxim 24 de caractere.
- În sistem nu pot exista 2 utilizatori cu același nume, dar același utilizator poate fi conectat, în paralel, la același server, prin mai multe instanțe ale programului client.

- La parcurgerea fișierului `shared_files` se va verifica corectitudinea directivelor găsite. Dacă numele de utilizator sau fișierul nu există, atunci se va afișa un mesaj relevant în consola serverului și se va ignora acea linie.

NOTA₁: În afară de fișierele partajate, în folderele utilizatorilor pot exista și alte fișiere, iar serverul va trebui să fie capabil să le listeze, atunci când se cere lista de fișiere a unui utilizator.

NOTA₂: Întrucât în cadrul directoarelor utilizatorilor organizarea nu va fi una ierarhică (nu vor exista subdirectoare), pentru parcurgerea tuturor fișierelor din director puteți folosi funcția standard C `readdir` (vezi man 3 `readdir`).

Comenzi Client

După conectarea la server un client poate primi un set de comenzi de la tastatură.

Orice comandă, împreună cu rezultatul ei se va scrie în fișierul de log. 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.

Comenzile implementate la client sunt următoarele:

1. **login** < nume_utilizator > < parola >

Clientul trimite o cerere serverului pentru a realiza deschiderea unei sesiuni de lucru. Acesta va confirma deschiderea sesiunii cerând clientului deblocarea tuturor celorlalte comenzi sau va întoarce un cod de eroare. Pe parcursul unei sesiuni de lucru, toate comenzile introduse în log-ul clientului vor fi precedate de un prompt de tipul < nume_utilizator >

Verificarea corectitudinii credențialelor oferite se va face la server.

În cazul în care în cadrul clientului curent există deja o sesiune deschisă cu ajutorul comenzii `login`, va returna codul de eroare **-2** și nu va mai trimite comanda către server.

În cazul în care într-o sesiune se oferă, de 3 ori consecutiv un set de credențiale incorect, serverul va întoarce codul de eroare **-8** și va închide imediat conexiunea respectivului client. În acest caz, procesul client ar trebui să se termine imediat.

Exemple functionare:

```
1 $ login root 123456
2 -3 User/parola gresita
3
4 $ login Doru protocoale
5
6 Doru> login Elena 123456
7 -2 Sesiune deja deschisa
```

Listing 3: Exemplu 1 'login' din fisierul de log.

```
1 $ login root 123456
2 -3 User/parola gresita
3
4 $ login root admin
5 -3 User/parola gresita
6
7 $ login root qwery
8 -8 Brute-force detectat
```

Listing 4: Exemplu 2 'login' din fisierul de log.

2. **logout**

Clientul va reseta sesiunea curentă și va anunța serverul de încheierea sesiunii. În cazul în care utilizatorul nu este logat, va întoarce codul de eroare **-1** și nu va trimite nicio solicitare către server.

Exemple de functionare:

```
1 Doru> logout
2
3 $ logout
4 -1 Clientul nu e autentificat
5
```

```

6 $ login Elena 123456
7 Elena>

```

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

3. *getuserlist*

Permite unui client autentificat să vizualizeze lista de utilizatori înregistrați în sistem. În acest sens, utilizatorul va trimite o solicitare către server, iar acesta îi va oferi ca răspuns lista solicitată.

Exemplu de funcționare:

```

1 Doru> getuserlist
2 Elena
3 Doru
4 Loredana

```

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

Ordinea afișării utilizatorilor va fi cea a apariției lor în fișierul de configurare.

4. *getfilelist* < nume_utilizator >

Cientul va solicita serverului lista de fișiere ale utilizatorului specificat.

În cazul în care utilizatorul specificat nu există, serverul va întoarce codul de eroare **-11**, iar execuția comenzii va înceta.

Serverul va trimite clientului solicitant, lista de fișiere cerută, alături de dimensiunea fișierelor și starea lor (shared/private).

Exemplu de funcționare:

```

1 Elena> getuserlist
2 Elena
3 Doru
4 Loredana
5
6 Elena> getfilelist Elena
7 Catalog.xlsx      94.523 bytes    SHARED
8 Film.avi         13.316.249.020 bytes PRIVATE
9
10 Elena> getfilelist Jimmy
11 -11 Utilizator inexistent
12
13 Elena> getfilelist Doru
14 Falstad.zip      666.053 bytes    SHARED
15 Big_Secret.zip   16.249.020 bytes PRIVATE

```

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

5. *upload* < nume_fisier >

Cientul va trimite către server fișierul indicat din folderul curent, iar acesta îl va stoca în folderul corespunzător utilizatorului logat. În cazul în care fișierul nu există, atunci clientul va întoarce codul de eroare **-4** și nu va mai trimite solicitarea către server. În cazul în care același fișier sau un alt fișier cu același nume există deja pe server în folderul dedicat utilizatorului logat, serverul va preveni suprascrierea lui, returnând codul de eroare **-9**.

6. *download* < nume_utilizator > < nume_fisier >

Cientul va solicita serverului fișierul < nume_fisier > din directorul utilizatorului < nume_utilizator > pe care îl va salva în folderul său de lucru numele < nume_fisier > prefixat cu prefixul <ID>_, unde ID este PID-ul procesului client.

Fiecare utilizator va avea acces toate fișierele ce sunt plasate în folderul ce îi este destinat pe server, respectiv la toate fișierele partajate global de ceilalți utilizatori.

În cazul în care fișierul nu există, serverul va întoarce codul de eroare **-4**, iar execuția comenzii va fi întreruptă. Pentru situația în care utilizatorul nu are permisiunea de a accesa un fișier, se va întoarce codul de eroare **-5**.

Pentru a face referire la un fișier din folderul personal, un utilizator va putea înlocui numele de utilizator din comandă cu caracterul special

Ambele comenzi de *upload* / *download* sunt asincrone și vor afișa un mesaj de call-back la finalizarea operațiunii de transfer.

Atenție! Având în vedere aplicarea politicii de round-robin la prelucrarea segmentelor din fișiere transferate, este foarte probabil ca execuția comenzilor să nu se încheie în ordinea în care acestea au fost introduse, motiv pentru care în mesajul de încheiere al comenzilor upload și download va trebui să specificați toți parametri aferenți comenzii.

Exemplu de functionare:

```
1 Doru> getfilelist Doru
2 Falstad.zip      666.053 bytes    SHARED
3 Big_Secret.zip   16.249.020 bytes PRIVATE
4
5 Doru> upload Test1.zip
6
7 Doru> upload test_inexistent.zip
8 -4 Fisier inexistent
9
10 Doru> upload Test2.zip
11
12 Doru> getfilelist Doru
13 Falstad.zip      666.053 bytes    SHARED
14 Big_Secret.zip   16.249.020 bytes PRIVATE
15
16 Doru> Upload finished: Test2.zip - 62.312 bytes
17
18 Doru> getfilelist Doru
19 Falstad.zip      666.053 bytes    SHARED
20 Big_Secret.zip   16.249.020 bytes PRIVATE
21 Test2.zip        31.312 bytes PRIVATE
22
23 Doru> Upload finished: test1.zip - 1.100.233.522 bytes
24
25 Doru> getfilelist Doru
26 Falstad.zip      666.053 bytes    SHARED
27 Big_Secret.zip   16.249.020 bytes PRIVATE
28 Test2.zip        31.312 bytes PRIVATE
29 test1.zip        1.100.233.522 bytes PRIVATE
30
31 Doru> download @ VS.zip
32 -4 Fisier inexistent
33
34 Doru> download @ Falstad.zip
35 Doru> Download finished: Falstad.zip - 666.053 bytes
36
37 Doru> getfilelist Elena
38 Catalog.xlsx     94.523 bytes    SHARED
39 Film.avi         13.316.249.020 bytes PRIVATE
40
41 Doru> download Elena Film.avi
42 -5 Descarcare interzisa
43
44 Doru> download Elena Ceva.txt
45 -4 Fiser inexistent
46
47 Doru> download Elena Catalog.xlsx
48 Doru> Download finished: Catalog.xlsx - 94.523 bytes
```

Listing 8: Exemplu 'download' din fisierul de log.

Principala funcționalitate a sistemului va fi aceea de a încărca / descărca fișiere de la server, la care se adaugă posibilitatea de a partaja fișierele personale cu ceilalți utilizatori. Întreaga experiență a utilizatorului trebuie să fie non-blocantă (o nouă comandă poate apărea oricând și trebuie pusă în execuție imediat, chiar dacă în paralel rulează o altă comandă).

Întrucât fișierele pot fi foarte mari, va fi necesar și ne dorim limitarea utilizării memoriei la ambele părți ale transmisiei, va fi necesar să faceți implementarea unei politici de segmentare a transferurilor, care să respecte următoarele specificații:

- Dimensiunea maximă a unui fragment trimis pe rețea va fi 4096 de octeți.
- Toate transferurile vor avea acces în mod egal la resursele sistemului, iar procesarea transferurilor concurente se va face procesând – pe rând – câte un fragment din fiecare fișier recepționat / trimis.
- Va folosi apelul select pentru a evita blocarea serverului / unui client în cazul în care, din transferul luat în calcul de politica round-rubin, nu este disponibil pentru recepție niciun fragment.

Fișierele fiecărui utilizator vor fi stocate într-un director din folderul de lucru al serverului, având aceeași denumire cu hadle-ul folosit de acesta pentru autentificare. Exemplu: Dacă serverul este lansat din folderul `/home/protocoale/NTFTP`, atunci fișierele utilizatorului Foo vor fi stocate în `/home/protocoale/NTFTP/Foo`.

7. **share** < nume_fisier >

Se trimite un mesaj serverului prin care anunța faptul că fișierul < nume_fisier > va fi accesibil tuturor utilizatorilor.

Se presupune că fișierul există deja pe server și se va face doar schimbarea atributului de partajare.

NOTA₁: Fișierele al căror atribut se va schimba din PRIVATE în SHARED vor fi fișiere deja existente pe server. În caz contrar, se va returna codul de eroare **-4**.

NOTA₂: Clienții nu trebuie să mențină local o evidență a fișierelor de pe server, motiv pentru care detectarea situațiilor de eroare pentru această comandă trebuie făcută la server. Această observație se aplică și comenzii *unshare*.

Exemplu de functionare:

```
1 Elena> getfilelist Elena
2 Catalog.xlsx      94.523 bytes
   SHARED
3 Film.avi         13.316.249.020 bytes
   PRIVATE
4
5 Elena> share Catalog.xlsx
6 -6 Fisierul este deja partajat
7
8 Elena> share Cheatsheet.pdf
9 -4 Fisier inexistent
10
11 Elena> share Film.avi
12 200 Fisierul Film.avi a fost
   partajat.
13
14 Elena> getfilelist Elena
15 Catalog.xlsx      94.523 bytes
   SHARED
16 Film.avi         13.316.249.020 bytes
   SHARED
```

```
1 Doru> download Elena Film.avi
2 -5 Descarcare interzisa
3
4
5
6
7
8
9
10
11
12
13
14
15 Doru> download Elena Film.avi
16 Download finished: Film.avi -
   13.316.249.020 bytes
```

Listing 9: Exemplu 'share' din fisierul de log în paralel cu răspunsurile pe care le-ar primi un alt utilizator care dorește să acceseze fișierele.

8. *unshare* < nume_fisier >

Se trimite un mesaj serverului continand numele fisierului care trebuie sters din lista fisierelor partajate. Serverul va confirma executarea comenzii sau va întoarce un cod de eroare conform scenariilor din exemplu.

Exemplu de functionare:

```
1 Elena> getfilelist Elena
2 Catalog.xlsx      94.523 bytes    SHARED
3 Film.avi         13.316.249.020 bytes PRIVATE
4
5 Elena> unshare Film.avi
6 -7 Fisier deja privat
7
8 Elena> unshare Cheatsheet.pdf
9 -4 Fisier inexistent
10
11 Elena> unshare Catalog.avi
12 200 Fisierul a fost setat ca PRIVATE
13
14 Elena> getfilelist Elena
15 Catalog.xlsx      94.523 bytes    PRIVATE
16 Film.avi         13.316.249.020 bytes PRIVATE
```

Listing 10: Exemplu 'unshare' din fisierul de log.

9. *delete* < nume_fisier >

Clientul va solicita ștergerea fișierului nume_fisier din registrul serverului și din directorul core-spunzător de pe disc.

Pentru ștergerea fișierului recomandăm folosirea apelului *unlink(const char * filename)* din biblioteca standard C (vezi man 1 unlink).

În cazul în care pentru fișierul de șters există un transfer în curs de desfășurare, serverul va respinge cererea cu codul de eroare -10.

Exemplu de functionare:

```
1 Elena> getfilelist Elena
2 Catalog.xlsx      94.523 bytes
   SHARED
3 Film.avi         13.316.249.020 bytes
   PRIVATE
4
5 Elena> delete Catalog.xlsx
6 -10 File is busy
7
8 Elena> delete Film.avi
9 200 Fisier sters
10
11 Elena> getfilelist Elena
12 Catalog.xlsx      94.523 bytes
   SHARED
```

```
1
2
3
4
5 Doru> download Elena Catalog.xlsx
6
7
8 Download finished: Catalog.xlsx -
   94.523 bytes
```

Listing 11: Exemplu 'delete' din fisierul de log.

10. *quit*

Clientul trimite un mesaj serverului prin care anunta ca va parasii 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. Anunță clienții de intenția de închidere a serverului, blochează recepția de noi comenzi, termină transferurile în curs de desfășurare, apoi se închide.

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* : 10p
- *client:login* : 10p
- *client:logout* : 5p
- *client:upload* : 15p
- *client:download* : 20p
- *client:share* : 5p
- *client:unshare* : 5p
- *client:getuserlist* : 5p
- *client:getfilelist* : 5p
- *client:delete* : 10p
- *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*.