MyFileTransferProtocol
Proiect (B) - Rețele de Calculatoare

Pichiu Cristina-Cătălina, grupa 2B4

Universitatea Alexandru Ioan Cuza, Facultatea de Informatică Iasi
cristina.pichiu@info.uaic.ro
https://www.info.uaic.ro/

Abstract. În aceast raport tehnic voi descrie modul în care m-am gândit să implementez proiectul **MyFileTransferProtocol** și voi prezenta diferite tehnologii și concepte pe care le-am folosit.
Acest proiect are la bază diferite operații cu fișiere și directoare pe care doar un utilizator conectat la aplicație le poate face.

1 Introducere

Proiectul My File Transfer Protocol urmărește implementarea unei aplicații de tip client-server ce permite lucrul cu fișiere și directoare, având la bază o serie de operații ce pot fi aplicate asupra acestora, precum: copiere, ștergere, mutare, redenumire etc. Proiectul are, așadar, în vedere transferul de fișiere între client și server, asupra cărora pot fi efectuate anumite modificări. Implementarea se bazează pe o serie logică de pași. Pentru a efectua operațiile enumerate mai sus pe fișiere, clientul trebuie, prima dată, să se autentifice în aplicație. Dacă a reușit acest lucru, adică dacă username-ul și parola se află în baza de date atașată, acesta este liber să lucreze cu fișierele și directoarele dorite până se deconectează sau iese din program. Pe lângă aceste aspecte, va exista și un mecanism de autorizare pentru conturile utilizatorilor, de tip whitelist/blacklist, astfel încât programul să aprobe doar conectarea utilizatorilor din whitelist, considerați a fi siguri pentru program și să îi respingă pe cei din blacklist, a căror prezență ar putea fi considerată o breșă de securitate. Tot din punct de vedere al securității, se dorește și implementarea unui mecanism de criptare a parolei la autentificare, astfel încât simbolurile acesteia să nu mai fie vizibile atunci când sunt tastate.

2 Tehnologii utilizate

Pentru că, în acest proiect, transmiterea în mod corect a informațiilor din fișiere este cea cu adevărat importantă, deoarece conținutul acestora ne interesează în final, pentru comunicarea dintre client și server am folosit un protocol TCP (Transmission Control Protocol), cunoscut pentru transferul exact și sigur al datelor. Datele trimise de acesta sunt grupate sub forma unor pachete(segmente) și reasamblate atunci când ajung la destinație.

Spre deosebire de acesta, protocolul UDP (User Datagram Protocol) nu ar fi foarte util, deoarece acesta nu garantează o transmitere sigură a informațiilor, întrucât pot exista pierderi de date. Deși este mult mai rapid și poate mai eficient decat protocolul precedent, deoarece datele sunt transmise în mod direct de la sursă la destinație, acest lucru nu este benefic aplicației descrise, deoarece importantă este transmiterea în mod corect a acestora, nu și timpul de livrare, în mod special.
Pentru a reține informațiile despre utilizatori am folosit o bază de date SQL. Cu ajutorul acesteia, datele, precum numele, parola și status-ul (whitelist/blacklist) sunt mai ușor de accesat. Tabelul "utilizatori" în care am introdus aceste câmpuri este următorul:

```
mysql> select * from utilizatori;
+----+-----------------+----------+-----------+
| id | name            | password | status    |
+----+-----------------+----------+-----------+
|  1 | cristina.pichiu | al5gtuv* | whitelist |
|  2 | catalina.tanase | bg*ab5fd | whitelist |
|  3 | cristian.pichiu | defawxi7 | blacklist |
|  4 | andreea.popescu | fex*w67a | blacklist |
|  5 | catalin.morariu | dfgm*rva | whitelist |
+----+-----------------+----------+-----------+
```

3 Arhitectura aplicatiei

3.1 Concepte implicate

Aplicația se bazează pe un model clasic de comunicare între client și server, unde clientul trimite anumite comenzi serverului, acesta le recepționează, le execută și îi trimite răspunsul înapoi clientului. Partea corespunzătoare serverului este mult mai complexă, deoarece aici vor avea loc toate operațiile pe fișiere și directoare, acesta va interacționa cu baze de date propriu-zisă, ca mai apoi să poată să trimită răspunsul corespunzător solicitat de client.
Comunicarea dintre client și server va fi asigurata prin intermediul primitivei socket(), prezentă în ambele fișiere.
De asemenea, pentru a asigura conexiunea între cele 2 programe se vor folosi și următoarele primitive: read() și write() care au rolul de a citi si scrie date, astfel între cele 2 programe are loc un schimb de mesaje, bind(), cu ajutorul căreia legăm socketul de un port, listen(), prin care serverul "ascultă" dacă există noi conexiuni (clienți care vor să se conecteze), accept(), în care se acceptă o cerere de conectare (programul se blochează pana la primirea unei altei cereri de la un alt user), connect(), care stabilește legătura cu un server care a făcut accept() și close().
În implementarea proiectului se va folosi un server concurent ce permite comunicarea cu mai mulți utilizatori în același timp. Spre deosebire de serverul iterativ, acesta este mult mai eficient deoarece nu se așteaptă terminarea execuției clientului curent pentru a trece la următorul, ci toate cererile și comenzile sunt tratate

simultan. Pentru implementarea acestui aspect voi folosi thread-uri.

3.2 Diagramele aplicatiei detaliate:

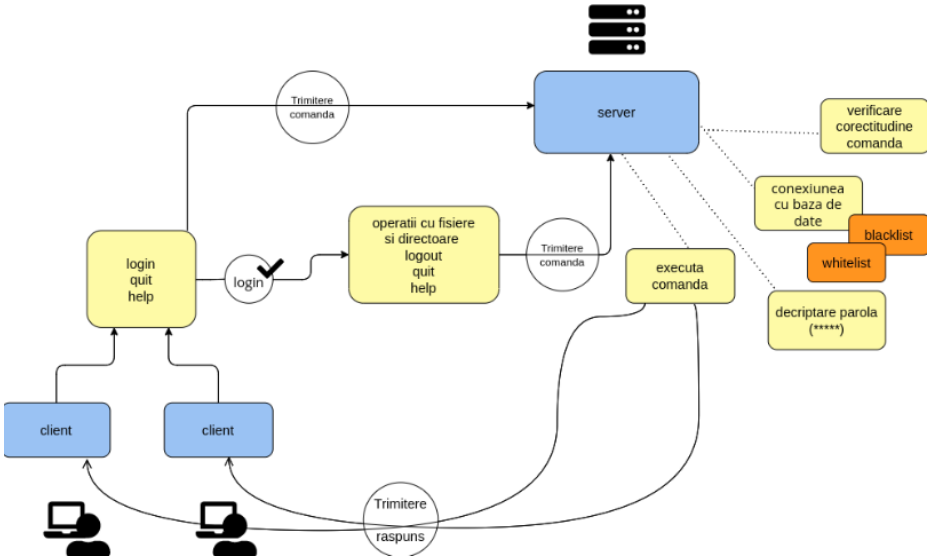

Fig. 1. Diagrama generala a aplicatiei ce ilustreaza functionalitatile acesteia

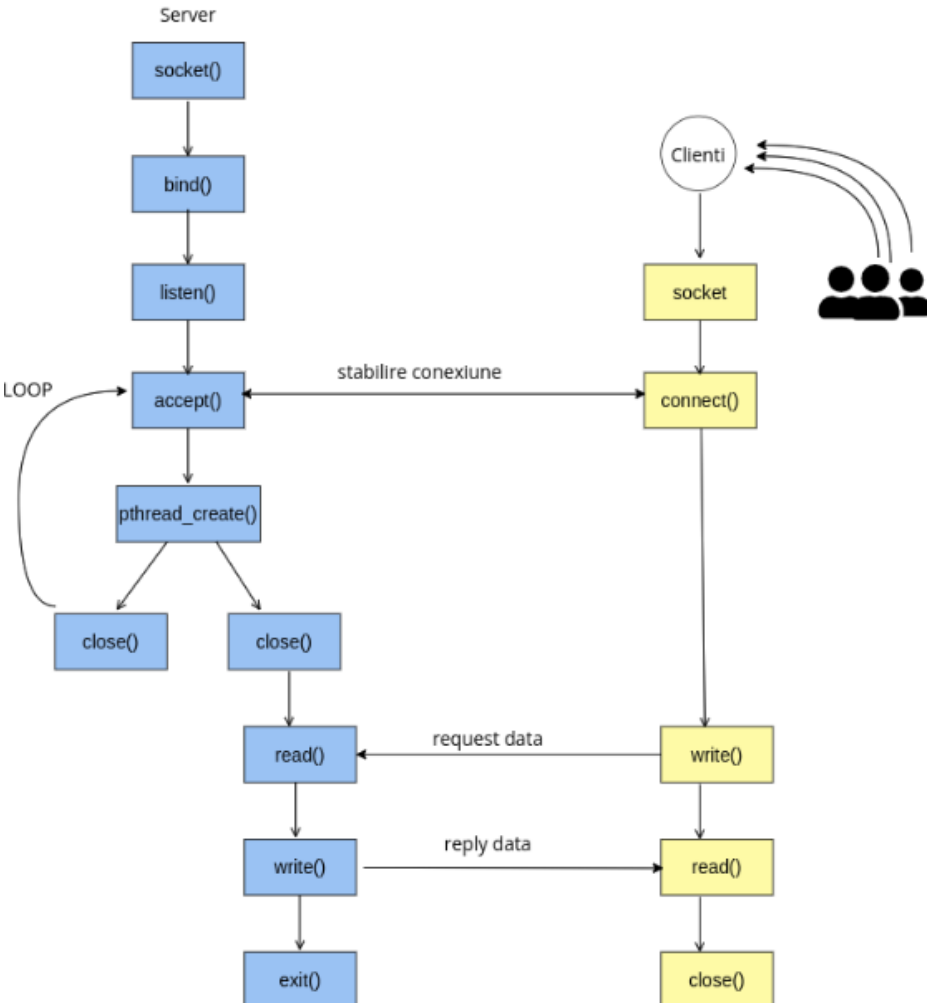

Fig. 2. Diagrama structurala a aplicatiei

4 Detalii de implementare

4.1 Cod relevant particular proiectului

Un aspect foarte important este crearea legăturii dintre client și server. Aceasta se realizează prin intermediul primitivei socket:

```
38
39
40
41  //crearea socket-ului:
42
43  if ((sd = socket (AF_INET, SOCK_STREAM, 0)) == -1)
44    {
45      perror ("Eroare la socket().\n");
46      return errno;
47    }
48
49
50  //realizarea conexiunii client-server:
51
52  server.sin_family = AF_INET;
53  server.sin_addr.s_addr = inet_addr(argv[1]);
54  server.sin_port = htons (port);
55
56
57  if (connect (sd, (struct sockaddr *) &server,sizeof (struct sockaddr)) == -1)
58    {
59      perror ("[client]Eroare la connect().\n");
60      return errno;
61    }
62
63
64
```

În server este atașat socket-ul și prin intermediul primitivei listen() acesta asculta daca există noi cereri de la clienți:

```
50
51
52
53    if (bind (sd, (struct sockaddr *) &server, sizeof (struct sockaddr)) == -1)
54    {
55        perror ("[server]Eroare la bind().\n");
56        return errno;
57    }
58
59    //serverul asculta daca vin alti clienti care vor sa se conecteze:
60
61    if (listen (sd, 1) == -1)
62    {
63        perror ("[server]Eroare la listen().\n");
64        return errno;
65    }
66
67
68
```

Validarea datelor utilizatorilor este făcută cu ajutorul unei baze de date SQL. Următoarea funcție realizeaza conexiunea cu baza de date și caută anumite valori în tabelul asociat acesteia cu ajutorul unei interogari pe care o are ca și parametru.

```
48 int search_bd(char *sir, char * interogare)
49 {
50
51    MYSQL *apel_bd;
52    MYSQL *rezultat_bd;
53    MYSQL_ROW inregistrare;
54
55    char * server = "localhost";
56    char * user = "utilizator";
57    char* password = "baze10";
58    char * database = "db";
59
60
61    apel_bd=mysql_init(NULL);
62
63    //conectarea propriu-zisa la baza de date
64
65    if(!mysql_real_connect(apel_bd, server, user, password, database, 0, NULL, 0))
66    {
67       fprintf(stderr, "%s\n", mysql_error(apel_bd));
68       return 1;
69    }
70
71    //trimitere interogare
72
73    if( mysql_query(apel_bd, interogare))
74    {
75      fprintf(stderr, "%s\n", mysql_error(apel_bd));
76      return 1;
77    }
78
79    rezultat_bd=mysql_use_result(apel_bd);
80
81    while((inregistrare=mysql_fetch_row(rezultat_bd))!=NULL)
82    {
83        if(strcmp(inregistrare[0], sir)==0)
84          {
85
86     mysql_free_result(rezultat_bd);
87     mysql_close(apel_bd);
88      return 1;}
89    }
90
91
92     mysql_free_result(rezultat_bd);
93     mysql_close(apel_bd);
94     return 0;
95
96
97 }
98
```

Implementarea unor comenzi cu ajutorul cărora facem anumite operații pe fișiere și directoare:

```
545    if(strstr(buf, "rename_c:")&&logat[w]==1&&status[w]==1)
546   {
547   int i, poz=0;
548
549   sir[0]='\0';
550   strncat(sir, buf+9, strlen(buf)-9);
551
552   for(i=0;i<strlen(sir);i++)
553    if(sir[i]=='/')
554       {poz=i; break;}
555
556     numeFisierVechi[0]='\0';
557     numeFisierNou[0]='\0';
558     strcpy(numeFisierVechi, "./Client/");
559     strcpy(numeFisierNou, "./Client/");
560
561      sir[strlen(sir)-1]='\0';
562
563     strncat(numeFisierVechi+9, sir, poz);
564     strncat(numeFisierNou+9, sir+poz+1, strlen(sir)-poz-1);
565
566      int result = rename(numeFisierVechi, numeFisierNou);
567       if (result == 0)
568      {
569       strcpy(raspuns, "Fisierul introdus a fost redenumit cu succes!!!");
570      }
571    else
572     {
573       strcpy(raspuns, "Eroare de sintaxa. Fisierul introdus nu a putut fi redenumit! ");
574     }
575
576   }
577    else
578       if(strstr(buf, "rename_c:")&&logat[w]==0)
579         strcpy(raspuns, "Nu sunteti conectat la sistem ca sa puteti efectua operatii cu fisiere!");
580    else
581       if(strstr(buf, "rename_c")&&status[w]==0)
582        strcpy(raspuns, "Nu aveti acces la aceste fisiere: status-->blacklist!");
583
```

```
660
661     //remove pentru server
662
663     if(strstr(buf, "remove_s:")&&logat[w]==1&&status[w]==1)
664    {
665      sir[0]='\0';
666      strcpy(sir, "./Server/");
667      strncat(sir+9, buf+9, strlen(buf)-9);
668           sir[strlen(sir)-1]='\0';
669           printf("%s\n", sir);
670           printf("%d\n",strlen(sir));
671
672     int  result = remove(sir);
673
674    if (result == 0)
675      {
676       strcpy(raspuns, "Fisierul introdus a fost eliminat cu succes!!!");
677      }
678    else
679     {
680       strcpy(raspuns, "Eroare de sintaxa. Fisierul introdus nu a putut fi eliminat! ");
681     }
682
683
684    }
685     else
686       if(strstr(buf, "remove_s:")&&logat[w]==0)
687         strcpy(raspuns, "Nu sunteti conectat la sistem ca sa puteti efectua operatii cu fisiere!");
688
689          else
690        if(strstr(buf, "remove_s")&&status[w]==0)
691         strcpy(raspuns, "Nu aveti acces la aceste fisiere: status-->blacklist!");
692
693
694
695
```

```
891  if(strstr(buf, "creareDir_c:")&&logat[w]==1&&status[w]==1)
892 {
893
894
895
896     sir[0]='\0';
897     strcpy(sir, "./Client/");
898     strncat(sir+9, buf+12, strlen(buf)-12);
899          sir[strlen(sir)-1]='\0';
900          printf("%s\n", sir);
901          printf("%d\n",strlen(sir));
902
903
904     int directory = mkdir(sir, 0777);
905
906  if (directory!= 0) {
907    strcat(raspuns, "Nu s-a putut crea folderul introdus in folderul Client");
908  }
909
910 strcat(raspuns, "Folderul introdus a fost creat cu succes in folderul Client");
911
912
913
914 }
915  else
916      if(strstr(buf, "creareDir_s:")&&logat[w]==0)
917        strcpy(raspuns, "Nu sunteti conectat la sistem ca sa puteti efectua operatii cu fisiere!");
918   else
919       if(strstr(buf, "creareDir_c")&&status[w]==0)
920        strcpy(raspuns, "Nu aveti acces la aceste fisiere: status-->blacklist!");
921
```

```
1192
1193      if(strstr(buf, "mutare_sc")&&logat[w]==1&&status[w]==1)
1194      {
1195
1196          char sursa[256], destinatie[256];
1197
1198          sursa[0]='\0';
1199          destinatie[0]='\0';
1200
1201          strcpy(sursa, "./Server/");
1202          strcpy(destinatie, "./Client/");
1203          strncat(sursa+9, buf+10, strlen(buf)-10);
1204          strncat(destinatie+9, buf+10, strlen(buf)-10);
1205          sursa[strlen(sursa)-1]='\0';
1206          destinatie[strlen(destinatie)-1]='\0';
1207
1208
1209
1210          FILE *fisier1 = fopen(sursa, "r");
1211          FILE *fisier2 = fopen(destinatie, "w");
1212
1213         char *linie = malloc(dimensiune);
1214
1215         size_t size;
1216         while ((size = fread(linie, 1, dimensiune, fisier1)) > 0) {
1217         fwrite(linie, 1, size, fisier2);
1218
1219
1220         }
1221
1222
1223    int  result = remove(sursa);
1224    if (result == 0)
1225     {
1226      strcpy(raspuns, "Fisierul introdus a fost mutat cu succes!!!");
1227     }
1228   else
1229    {
1230      strcpy(raspuns, "Eroare de sintaxa. Fisierul introdus nu a putut fi mutat! ");
1231    }
1232
1233  free(linie);
1234  fclose(fisier1);
1235  fclose(fisier2);
1236      }
1237        else
1238      if(strstr(buf, "mutare_sc")&&logat[w]==0)
1239        strcpy(raspuns, "Nu sunteti conectat la sistem ca sa puteti efectua operatii cu fisiere!");
1240   else
1241       if(strstr(buf, "mutare_sc")&&status[w]==0)
1242        strcpy(raspuns, "Nu aveti acces la aceste fisiere: status-->blacklist!");
1243
```

4.2 Scenarii de utilizare

Așa cum am menționat mai sus, proiectul funcționează după modelul clasic de client-server, în care clientul este cel care trimite o comandă, iar serverul o execută. Inițial, clientul nu este autentificat în aplicație. Așadar, acesta poate accesa doar cateva dintre comenzile implementate, și anume: login, quit și help. Am considerat implementarea comenzii help foarte utilă, deoarece odată apelată, aceasta afișează sintaxa corectă a fiecărei comenzi, venind în ajutorul userului. Prin intermediul comenzii quit, acesta are posibilitatea de a ieși din aplicație, operațiile făcute pe fișiere rămânând totuși salvate. Prin intermediul comenzii login, clientul încearcă să se conecteze la program. Acest lucru este făcut prin validarea de către server a username-ului și a parolei introduse, căutand aceste informații în baza de date. Dacă sintaxa comenzii este corectă si utilizatorul este găsit în fișier, următorul pas este verificarea cărei entități (whitelist/blacklist) aparține userul, accesul acestuia fiind permis doar dacă are atributul whitelist, așadar doar dacă este sigur. Un aspect important la partea de login este faptul că parola va fi criptată, aceasta nu este vizibilă așa cum este username-ul atunci când este introdus. Odată autentificat, utilizatorul poate efectua diferite operații pe fișiere și directoare, precum ștergere, copiere, redenumire, mutare etc., denumite sugestiv și în program, dar se poate și deconecta prin intermediul comenzii logout, progresul său rămânând salvat.

4.3 Concluzii

MyFileTransferProtocol este un proiect cu ajutorul cu ajutorul căruia se pot realiza diferite operații cu fișiere. Am încercat să îl fac cat mai ușor de folosit și să denumesc comenzile în mod sugestiv astfel încât utilizatorii să nu întâmpine probleme.

References

1. https://profs.info.uaic.ro/~computernetworks/cursullaboratorul.php
2. https://profs.info.uaic.ro/~andreis/index.php/computernetworks/
3. https://www.geeksforgeeks.org/differences-between-tcp-and-udp/
4. https://learn.microsoft.com/en-us/host-integration-server/core/iterative-vs-concurrent-tcp-ip-models1
5. https://profs.info.uaic.ro/~andreis/wp-content/uploads/2021/11/Laboratorul-7.pdf
6. https://www.youtube.com/watch?v=d9s_d28yJq0&t=316s
7. https://www.youtube.com/watch?v=cA9ZJdqzOoU
8. https://www.geeksforgeeks.org/c-program-delete-file/?ref=gcse
9. https://www.geeksforgeeks.org/c-program-copy-contents-one-file-another-file/?ref=lbp
10. https://www.geeksforgeeks.org/c-program-print-contents-file/?ref=lbp