Raport al aplicatiei Netls

Balaita Anca-Georgiana, IIA4

Facultatea de Informatica, Universitatea Alexandru Ioan Cuza
Iasi, Iasi, Romania
anca.balaita@info.uaic.ro

Abstract. Netls este o aplicatie client/server care listeaza de la distanta continutul unui director, oferind informatii despre acesta si despre subdirectoarele sale. Folosirea unui server TCP concurent este datorata modului in care acesta transmite informatiile, si anume in intregime si in ordine, astfel oferindu-se utilizatorului un mediu stabil.

Keywords: recurenta, TCP, fork.

1.Introducere

Netls este o aplicatie client/server care listeaza de la distanta continutul unui director, precizindu-se tipul, dimensiunea, data ultimei schimbari de status, data ultimei modificari, data ultimei accesari, permisiunile si proprietarul acelui director. De asemenea, aplicatia furnizeaza si numarul total de octeti ocupati de toate fisierele din acel director.

Aplicatia presupune implementarea unui server TCP concurent si a unei metode recursive de listare a continutului subdirectoarelor directorului in cauza.

2. Tehnologii utilizate

Transmission Control Protocol (TCP) este un protocol folosit de obicei de aplicatii care au nevoie de confirmare de primire a datelor. Acesta efectueaza o conectare virtuala full duplex intre doua puncte terminale, fiecare punct fiind definit de catre o adresa IP si de catre un port TCP.

Aplicatia Netls foloseste acest protocol pentru implementarea serverului tocmai datorita carateristicilor sale de a oferi servicii orietate-conexiune, full-duplex si sigure pentru transportul fluxului de octeti.

Serverul va fi unul concurent, bazandu-se pe fork-uri. Serverul va crea cate un proces copil pentru fiecare client care se va conecta, astfel incat acesta sa poata servi toti clientii conectati in acelasi timp.

Comunicarea dintre server si client se va face pe baza de socket-uri.

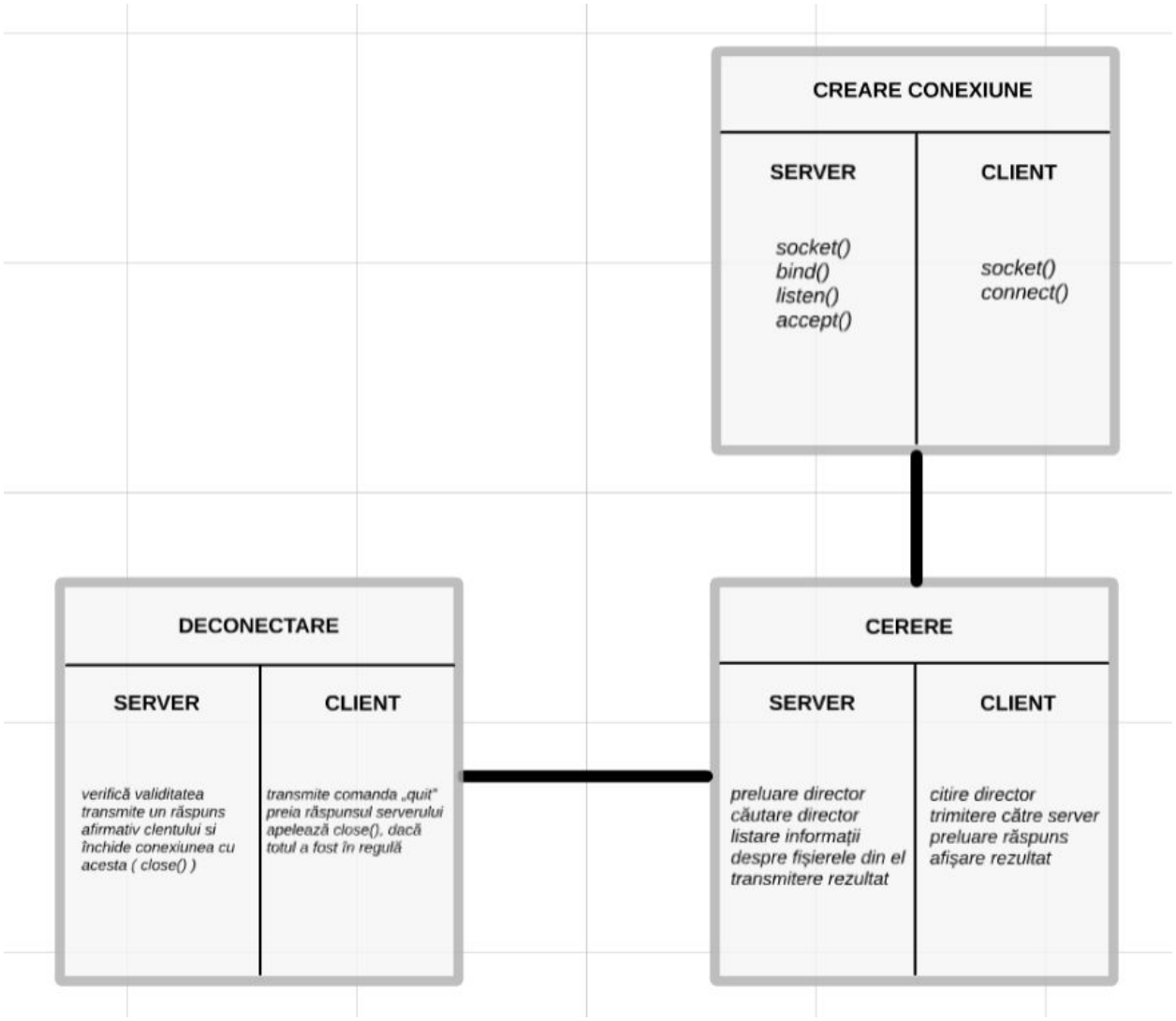
3. Arhitectura aplicatiei

3.1. Concepte implicate

Initial, serverul isi va deschide serviciul pe o adresa locala si pe un port predefinit. Cand un client doreste sa se conecteze la server, se initiază o conexiune TCP cu serverul, iar acesta va crea un nou proces copil prin intermediul primitivei fork().

Comunicarea dintre server si client se va face folosind primitivele de sistem, *read()* si *write()*. In implementarea serverului nu sunt utilizate mijlioace de comunicare intre procese, acestea nefiind necesare in contextul actual. Singurul canal de comunicatie folosit va fi socketul prin care se realizeaza comunicarea dintre server si client.

3.2. Diagrama aplicatiei



4. Detalii de implementare

4.1. Cod relevant

```
void procesare_intrare(char cale[MAX_SIZE],char raspuns[MAX_SIZE],long long *marime_director)
{
  struct stat st;
  struct passwd * pwd;
  char perm[10] = "---------";
  int cod, err = 0;
  char sir_final[MAX_SIZE], sir_aux[100];

  bzero(sir_final,MAX_SIZE);
  bzero(sir_aux,100);
```

```
strcat(sir_final,"Fisierul/folder-ul: ");
strcat(sir_final,cale);
strcat(sir_final,"\n");
if (stat(cale, &st) != 0)
{
  printf("Eroare la stat pentru %s .\t", cale);//%s-->string
  perror("Cauza este");
  strcpy(raspuns,"Eroare la stat!");
  return;
}

strcat(sir_final, "\tTipul fisierului: ");
switch (st.st_mode &S_IFMT)
{
case S_IFSOCK:
  strcat(sir_final, "Socket\n");
  break;
case S_IFLNK:
  strcat(sir_final, "Link\n");
  break;
case S_IFREG:
  strcat(sir_final, "Fisier obisnuit\n");
  break;
case S_IFBLK:
  strcat(sir_final, "Block device\n");
  break;
case S_IFCHR:
  strcat(sir_final, "Character device\n");
  break;
case S_IFIFO:
  strcat(sir_final, "FIFO\n");
  break;
case S_IFDIR:
  strcat(sir_final, "Director\n");
  break;
default:
  strcat(sir_final, "Unknown file type\n");
}

*marime_director+=(long long)st.st_size;
sprintf(sir_aux,"\tDimensiunea: %lld octeti\n", (long long)st.st_size);

strcat(sir_final, sir_aux);

if (S_IRUSR & st.st_mode) perm[0] = 'r';
if (S_IWUSR & st.st_mode) perm[1] = 'w';
if (S_IXUSR & st.st_mode) perm[2] = 'x';
if (S_IRGRP & st.st_mode) perm[3] = 'r';
if (S_IWGRP & st.st_mode) perm[4] = 'w';
if (S_IXGRP & st.st_mode) perm[5] = 'x';
if (S_IROTH & st.st_mode) perm[6] = 'r';
if (S_IWOTH & st.st_mode) perm[7] = 'w';
if (S_IXOTH & st.st_mode) perm[8] = 'x';

char s1[64], s2[64], s3[64];
struct tm t1, t2, t3;
time_t status_time = st.st_ctime, modify_time = st.st_mtime, access_time = st.st_atime;
```

```
  localtime_r( & status_time, & t1);//convertesc timpul din formatul din structura st, intr-un
format de timp efectiv(data, ora)
  localtime_r( & modify_time, & t2);
  localtime_r( & access_time, & t3);

  strftime(s1, sizeof(s1), "%c", & t1);
  strftime(s2, sizeof(s2), "%c", & t2);
  strftime(s3, sizeof(s3), "%c", & t3);

  strcat(sir_final, "\tData ultimei schimbari de status: ");
  strcat(sir_final, s1);
  strcat(sir_final, "\n");
  strcat(sir_final, "\tData ultimei modificari: ");
  strcat(sir_final, s2);
  strcat(sir_final, "\n");
  strcat(sir_final, "\tData ultimei accesari: ");
  strcat(sir_final, s3);
  strcat(sir_final, "\n");

  strcat(sir_final, "\tPermisiunile: ");
  strcat(sir_final, perm);
  strcat(sir_final, "\n");
  if ((pwd = getpwuid(st.st_uid)) != NULL) //in pwd va fi owner-ul fisierului si id-ul
acestuia
  {
    strcat(sir_final, "\tProprietarul: ");
    strcat(sir_final, pwd -> pw_name);
    strcat(sir_final, " cu UID-ul: ");
    sprintf(sir_aux, "%ld", (long)st.st_uid);

    strcat(sir_final, sir_aux);
    strcat(sir_final, "\n");
  }
  else //daca proprietarul nu are un nume efectiv sau este ascuns afisez doar id-ul acestuia
  {
    sprintf(sir_aux, "\tProprietarul are UID-ul: %ld\n", (long)st.st_uid);
    strcat(sir_final, sir_aux);
  }
  strcat(sir_final, "\n");
  //am terminat de construit sirul cu toate informatiile despre fisier
  //si il pun in "raspuns", sir care va fi trimis catre client la sfarsit
  strcat(raspuns,sir_final);
}


void cautare_director(char cale[MAX_SIZE], char raspuns[MAX_SIZE],char director[MAX_SIZE],long
long *marime_director)
{
       struct stat st;
       struct dirent * rdir;
       DIR* dir;
       char nume[MAX_SIZE];
       bzero(nume,MAX_SIZE);
       //aplic stat pe cale la momentul curent pentru a vedea daca fisierul este folder
       if (0 != stat(cale, & st))
       {
```

```
                printf("Eroare la aplicarea stat pentru: %s", cale);
                perror("Cauza: ");
                return;
        }
        if (S_ISDIR(st.st_mode))//daca este folder
        {
                if ((dir = opendir(cale)) == NULL)
                {
                  printf("Eroare la deschiderea folder-ului %s", cale);
                  perror("Cauza: ");
                  return;
                }
            while (NULL != (rdir = readdir(dir))) //parcurg fisierele din director, cat timp
mai exista
            {
               if (strcmp(rdir -> d_name, ".") && strcmp(rdir -> d_name, ".."))// ".” este
directorul curent, iar "..” este directorul parinte; accesarea acestora duce la o recurenta
infinita
               {
                       char temp[MAX_SIZE];
                       bzero(temp,MAX_SIZE);
                       if(director[0]!='/')
                                       strcat(temp,"/");
                               strcat(temp,director);
                               strcat(temp,"/");
                       sprintf(nume, "%s/%s", cale, rdir -> d_name);
                       if(strstr(nume,temp)!=0)
                               procesare_intrare(nume,raspuns,marime_director);
                       cautare_director(nume,raspuns,director,marime_director);
               }
            }
            closedir(dir);
        }
}
```

4.2. Scenarii de utilizare

Initial, serverul isi va deschide serviciul pe o adresa locala si pe un port predefinit. Cand un client doreste sa se conecteze la server, se initiază o conexiune TCP cu serverul, iar acesta va crea un nou proces copil prin intermediul primitivei fork().

Clientul va trebui sa introduca de la tastatura directorul pe care il doreste a fi listat. Acest director este trimis catre server.

Serverul va prelua numele directorului si il va cauta recursiv pornind din radacina "/home", astfel parcurgandu-se toate fisierele existente. Pe masura ce se parcurge directorul respectiv, se calculeaza dimeniunea fiecarui fisier si se salveaza informatii despre el pentru ca mai tarziu sa fie trimise ca raspuns clientului. In cazul in care dimensiunea directorului este zero, in consola se va afisa un mesaj corespunzator faptului ca acel director nu exista sau este gol.

Dupa ce clientul primeste mesajul, acesta poate alege intre a introduce un nou director pentru listare sau se poate deconecta prin "quit".

5. Concluzii

Avand in vedere ca protocolul de comunicatie ales este TCP/IP, aplicatia va oferi utilizatorului un mediu stabil, in care toate serviciile vor fi transmise corect si in ordinea în care au fost transmise.

Pentru imbunatatirea proiectului, se poate inlocui folosirea primitivei fork() cu crearea thread-urilor. Astfel, se va eficientiza executia programului.

6. Bibliografie

https://profs.info.uaic.ro/~computernetworks/cursullaboratorul.php
https://profs.info.uaic.ro/~eonica/rc/lab06.html
http://pubs.opengroup.org/onlinepubs/009695399/functions/stat.html
http://pubs.opengroup.org/onlinepubs/7908799/xsh/dirent.h.html