Documentatie Proiect Nfiles

– Retele de calculatoare –

Adam Cristian

09 January 2019

1 Introducere

Proiectul Nfiles consta in crearea unui server care este capabil sa preia toate cererile de la clienti in mod concurent. Aceste cereri constau in cautarea unor informatii despre fisierele aflate la distanta (nume,lungime,permisiuni,propietar) deja stocate pe diferite masini. Acestea sunt stocate identic pe toate masinile, astfel incat daca masina este inchsa, exista posibilitatea de a gasi fisierul intr-o alta masina. Numai serverului principal ii este permisa comunicarea cu aceste masini, el tinand un fisier cu toate adresele acestora la care se poate face conexiunea. Daca un client solicita cautarea unui fisier intr-o masina cu un IP necunoscut de server, atunci IP-ul nou este adaugat la lista din server si completate listele cu fisiere din fiecare masina. Acest concept poate fi privit ca un drive, doar ca singura diferenta este ca clientul cere doar informatii despre acel fisier si nu continutul acestuia.

2 Tehnologii Utilizate

Dupa cum se poate observa din introducere, serverul este cel care face legatura dintre clienti si masini, mediul in care sunt stocate fisierele si informatiile despre acestea. Din acest motiv, cred ca nu ar fi in concordanta cu cele deja mentionate pentru nici o componenta a programului sa se piarda date cand se realizeaza conexunile intre clienti si serverul principal si apoi intre server si masini. De aceea am ales ca si protocol de transmitere a datelor TCP si in plus pentru ca este un serviciu fiabil si optimizat de livrare a fluxului, care garanteaza ca toti octetii primiti vor fi identici cu octetii expediati in ordinea corecta. TCP-ul are de asemenea implementat o metoda prin care receptorul raspunde cu un mesaj de confirmare pe masura ce primeste datele. Expeditorul pastreaza o inregistrare a fiecarui pachet pe care il trimite si mentine un cronometru de la trimiterea datelor. Cronometrul este necesar in cazul in care un pachet se pierde sau este corupt. Aceasta tehnica se numeste 3-Way Handshaking.

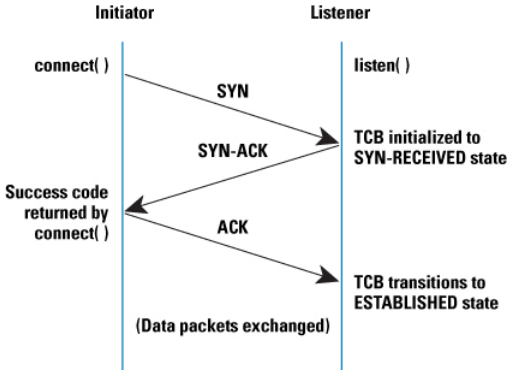

Figure 1: 3 Way Handshaking Exemplificat

Am ales TCP-ul in defavoarea UDP-ului, deoarece TCP-ul se bazeaza pe acuratete, siguranta, iar UDP-ul, in schimb, se bazeaza mai mult pe viteza decat pe primirea cererii in intregime. In plus, TCP-ul poate detecta si recupera erori.

3 Arhitectura Aplicatiei

Pentru a face ca programul sa ruleze cat mai rapid posibil, atat serverul cat si masinile unde sunt stocate fisierele, vor trebui sa fie concurente. Astfel, concurenta se va realiza cu ajutorul thread-urilor, deoarece sunt mult mai rapide in practica decat procesele copil create de fork.

Un thread este un flux secvential din interiorul unui proces. Deoarece acestea nu sunt independente de alte procese, thread-urile impart cu alte thread-uri sectiunea de cod, sectiunea de date, resurserle sistemului de operare, precum si fisierele deschise si semnalele. Dar, ca si un proces, un thread are propriul program de calcul, registri si spatiu pe stiva.

Thread-urile sunt o modalitate populara de a imbunatati aplicatia prin paralelism. Acestea functioneaza mai repede decat procesele copil generate de fork datorita faptului ca crearea firelor de executie, schimbarea de date si comunicarea intre thread-uri sunt mult mai rapide si ca se pot termina cu usurinta.

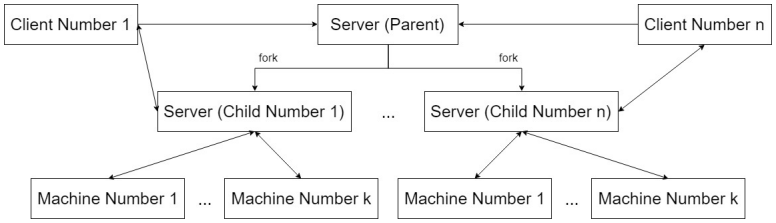

Figure 2: Schema Proiect Simplificata

In schema de mai jos am ilustrat exact cum se face conexiunea intre server si un client si intre server si o masina, folosind de data aceasta procese create cu ajutorul fork-ului. Fiecare proces este initiat cu un singur thread, numit adesea thread-ul principal, oferind resursele necesare pentru a executa un program. Diferenta principala fata de thread-uri este ca procesele ruleaza la adrese de memorie diferita.

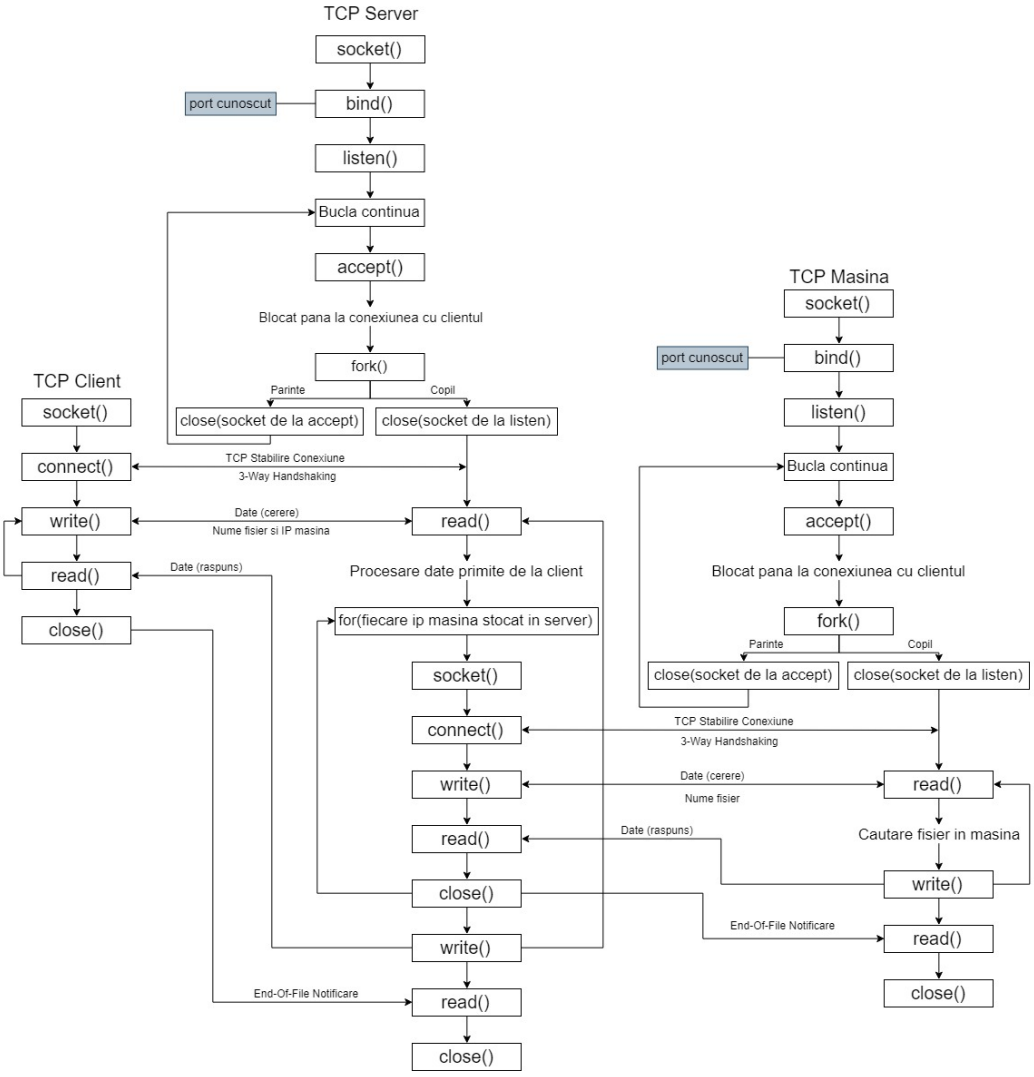

Figure 3: Schema Detaliata

4 Detalii de Implementare

Un aspect important din implementare este prelucarea datelor primite de la client in server si stabilirea daca clientul, cel care a trimis cererea, vrea ca fisierul dat sa fie cautat numai in masina cu IP-ul mentionat sau in toate masinile la care serverul se poate conecta.

```
char file[100]={0};
bzero (file, 100);

char machine_ip[11];
bzero(machine_ip,11);

if('0' <= string_received[0] && string_received[0] <= '9')
{
    char sep[]=" ";
    char *p=strtok(string_received,sep);
    strcpy(machine_ip,p);
    p=strtok(NULL,sep);
    strcpy(file,p);
}
else strcpy(file,string_received);
```

In cazul in care IP-ul dat de catre client se afla in lista de IP-uri a serverului principal si acesta se poate conecta la masina respectiva, se va afisa ori raspunsul la cererea primita ori faptul ca fisierul nu este stocat in fisierele masinii respective.

```
if(masina_gasita)
    if(masina_conectata)
    {
        char text[]="IP Masina gasit si conectat!\n";
        if(strcmp(raspuns,"File not found!")!=0) sprintf(string_send,"%s%s",text,raspuns);
        else
        {
            char aux[]="Fisierul cautat nu se gaseste in masina cu ip-ul dat!\n";
            sprintf(string_send,"%s%s",text,aux);
        }
    }
```

In cazul in care IP-ul dat de catre client se afla in lista de IP-uri a serverului principal, dar masina este deconectata, nemaiputand fi realizata conexiunea, i se specifica clientului ca masina de la IP-ul respectiv nu este activa fiind stearsa din lista si in plus se mai transmite o lista cu serverele deschise unde poate fi gasit fisierul dorit.

```
char text[]="IP Masina gasita dar neconectat!\nMasini deconectate sterse!\nAdrese ip a masinilor conectate unde
poate fi gasit fisierul:";
strcpy(rezultat,text);

char connected_machines_with_file[1000];
bzero(connected_machines_with_file,1000);

strcpy(file,file+1);
gasireIPCuFisierDat(file,connected_machines_with_file);

sprintf(rezultat,"%s%s",rezultat,connected_machines_with_file);
```

In cazul in care IP-ul dat de catre client nu se afla in lista, atunci acesta este adaugat la lista serverului principal daca masina este activa.

```
FILE* fisier_ip;
fisier_ip=fopen("MachinesIPs.txt","a");
if(fisier_ip==NULL)
{
    perror("[server]Eroare la deschidere fisier detalii!\n");
    return errno;
}

machine_ip[strlen(machine_ip)]='\n';
fprintf(fisier_ip,"%s",machine_ip);

if(fclose(fisier_ip) == -1)
{
    perror("[server]Eroare la inchidere fisier cu ip-urile stocate!\n");
    return errno;
}

actualizareFisiereInMasini();

char text[]="IP Masina dat nu a fost gasit, dar a fost adaugat la lista daca masina este activa!\n";
strcpy(rezultat,text);
```

De asemenea, le este permis clientilor sa adauge un fisier nou in masinile conectate pe langa fisierele deja stocate, fie fara nici o informatie despre acesta sau fie prn calea absoluta a acestuia, lasand serverul sa gaseasca toate informatiile necesare despre fisier.

```
FILE* fisier;
fisier=fopen("MachinesIPs.txt","r");
```

```
char ip_masina[11];
bzero(ip_masina,11);
for(int i=strlen(file)-1;i>=0;--i) file[i+1]=file[i];
file[0]='1';

while( fgets(ip_masina,11,fisier) != NULL )
{
    int sd_masina;
    struct sockaddr_in masina;
    int port_masina = atoi (ip_masina);
    if ((sd_masina = socket (AF_INET, SOCK_STREAM, 0)) == -1)
    {
        perror ("[server]Eroare la socket().\n");
        return errno;
    }
    char adresa[]="127.0.0.1";
    masina.sin_family = AF_INET;
    masina.sin_addr.s_addr = inet_addr(adresa);
    masina.sin_port = htons (port_masina);
    if (connect (sd_masina, (struct sockaddr *) &masina,sizeof (struct sockaddr)) == -1) continue;
    if (write (sd_masina, file, strlen(file)) == -1)
    {
        perror ("[server]Eroare la write() spre masina.\n");
        return errno;
    }
    char raspuns_masina[100];
    bzero(raspuns_masina, 100);
    if (read (sd_masina, raspuns_masina, 100) == -1)
    {
        perror ("[server]Eroare la read() de la masina.\n");
        return errno;
    }
close(sd_masina);
```

Functia de cautare fisier printre toate directoarele.

```
void cautareFisierPrintreDirectoare(char path[],char file[], char rezultat[])
{
    struct stat statut;
    stat(path,&statut);
    if( S_ISDIR(statut.st_mode) )
    {
        DIR *director;
        director=opendir(path);
        if(director==NULL)
        {
```

```
            perror("Eroare la deschidere director!\n");
            return errno;
        }
        else
        {
            struct dirent *fisier_actual;
            while( ( fisier_actual=readdir(director) ) != NULL )
            {
                stat(fisier_actual->d_name,&statut);
                if( S_ISDIR(statut.st_mode) && strcmp(fisier_actual->d_name,".")!=0 && strcmp(fisier_actual->d_na
                {
                    char noul_director[256];
                    sprintf(noul_director, "%s/%s", path, fisier_actual->d_name);
                    cautareFisierPrintreDirectoare(noul_director,file,rezultat);
                }
                if(strstr(fisier_actual->d_name,file)!=NULL)
                {
                    char path_complet[256];
                    bzero(path_complet,256);
                    sprintf(rezultat,"%s/%s",path,fisier_actual->d_name);
                }
            }
        }
        if(closedir(director) == -1)
        {
            perror("Eroare la inchidere director!\n");
            return errno;
        }
    }
}
```

Functia de determinare a tuturor informatiilor necesare despre fisierul dat.

```
void determinareStatutFisier(char path[], char rezultat[])
{
    char comanda[255];
    bzero(comanda,255);
    sprintf(comanda,"%s %s","stat",path);
    FILE *fd;
    fd=popen(comanda,"r");
    if(fd==NULL)
    {
        perror("[server]Eroare la rulare comanda!\n");
        return errno;
    }
    char sir[255];
    bzero(sir,255);
```

```
    while(fgets(sir,255,fd) != NULL)
    {
        char sep[]=" \t\n";
        char *p=strtok(sir,sep);
        while(p)
        {
            if(strcmp(p,"File:")==0) break;
            else
            {
                if('a'<=*(p+0) && *(p+0)<='z') sprintf(rezultat,"%s %s",rezultat,"Type:");
                sprintf(rezultat,"%s %s",rezultat,p);
                p=strtok(NULL,sep);
            }
        }
    }
}
```

Functia de actualizare fisiere din masini deja active dupa ce s-a adaugat o noua masna activa si invers.

```
FILE* fisier;
fisier=fopen("MachinesIPs.txt","r");

char ip_masina[11];
bzero(ip_masina,11);
char sir[]="2";

while( fgets(ip_masina,11,fisier) != NULL )
{
    int sd_masina;
    struct sockaddr_in masina;

    int port_masina = atoi (ip_masina);

    if ((sd_masina = socket (AF_INET, SOCK_STREAM, 0)) == -1)
    {
        perror ("[server]Eroare la socket().\n");
        return errno;
    }

    char adresa[]="127.0.0.1";
    masina.sin_family = AF_INET;
    masina.sin_addr.s_addr = inet_addr(adresa);
    masina.sin_port = htons (port_masina);

    if (connect (sd_masina, (struct sockaddr *) &masina,sizeof (struct sockaddr)) == -1) continue;
```

```
    if (write (sd_masina, sir, strlen(sir)) == -1)
    {
        perror ("[server]Eroare la write() spre masina.\n");
        return errno;
    }

    char raspuns_masina[10000];
    bzero(raspuns_masina, 10000);

    if (read (sd_masina, raspuns_masina, 10000) == -1)
    {
        perror ("[server]Eroare la read() de la masina.\n");
        return errno;
    }

    raspuns_masina[strlen(raspuns_masina)]='\n';
    char sep[]="\n";
    char *p=strtok(raspuns_masina,sep);

    while(p)
    {
        char aux[1000];
        bzero(aux,1000);

        strcpy(aux,p);

        adaugareFisierInMasini(aux);
        p=strtok(NULL,sep);
    }
    close(sd_masina);
}
```

O alta parte importanta din implementare este cautarea fisierului transmis de serverul principal si adaugarea acestiua la lista de fisiere de catre masina respectiva.

```
for(int i=1;i<=strlen(string_received);++i) string_received[i-1]=string_received[i];
char aux[1000],sep[]=" ";
bzero(aux,1000);
strcpy(aux,string_received);
char *file=strtok(aux,sep);
char rezultat[255];
bzero(rezultat,255);

cautareFisier(file,rezultat);
if(strcmp(rezultat,"File not found!")==0)
{
```

```
    adaugareFisier(string_received);
    strcpy(string_send,"Adaugat!");
}
else strcpy(string_send,"Este adaugat deja!");
```

Functia de cautare fisier dat de server in lista masinii respective.

```
FILE* fisier;
fisier=fopen("FilesDetails.txt","r");
if(fisier==NULL)
{
    perror("[masina]Eroare la deschidere fisier detalii!\n");
    return errno;
}
char sir[1000],sep[]=" ";
bzero(sir,1000);
while(fgets(sir,1000,fisier) != NULL)
{
    char *p=strtok(sir,sep);
    if(strcmp(p,file)==0)
    {
        p=strtok(NULL,sep);
        while(p!=NULL)
        {
            char s[100];
            bzero(s,100);
            strcpy(s,p);
            sprintf(rezultat,"%s %s",rezultat,s);
            p=strtok(NULL,sep);
        }
        break;
    }
}
if(fclose(fisier) == -1)
{
    perror("[masina]Eroare la inchidere fisier cu datele stocate!\n");
    return errno;
}
if(rezultat[0]==NULL)
{
    char text[]="File not found!";
    strcpy(rezultat,text);
}
```

5 Concluzii

Cu toate ca Nfiles este un proiect destul de greu de realizat, in acelasi timp este si unul foarte interesant, cu ajutorul caruia am mereu ceva nou de invatat, lucru ce ma ajuta sa-mi dezvolt cunostintele.

Din punctul meu de vedere, sunt foarte multumit de mine, deoarece am reusit sa imbunatatesc proiectul propus prin urmatoarele adaugari foarte utile si esentiale pentru acest tip de implementare:

1. Le este permis clientilor sa adauge un fisier nou in masinile conectate pe langa fisierele deja stocate, fie fara nici o informatie suplimentara despre acesta or fie prin calea absoluta a acestuia pentru a lasa serverul sa gaseasca toate informatiile necesare despre fisier.

2. Daca o noua masina este adaugata in lista serverului principal, atunci toate masinile deschise care se pot conecta la serverul principal isi vor actualiza fisierele adaugandu-le numai pe cele noi de la masina recent adaugata si invers, adica masina noua isi va actualiza si ea lista cu acele fisiere care se gasesc numai in masinile vechi.

6 Bibliografie

https://en.wikipedia.org/wiki/Transmission_Control_Protocol

https://www.geeksforgeeks.org/tcp-server-client-implementation-in-c/

https://www.geeksforgeeks.org/multithreading-c-2/

https://stackoverflow.com/questions/200469/what-is-the-difference-between-a-process-and-a-thread