Console Shopper

Anechitoaiei Codrin-Mihai

Facultatea de Informatica, Universitatea Alexandru Ioan Cuza Iasi

Abstract. In acest raport este prezentat modul in care am facut proiectul "Console Shopper", la materia "Retele de calculatoare". Aici se vor gasi tehnologiile utilizate, arhitectura aplicatiei si detalii de implementare.

1 Introducere

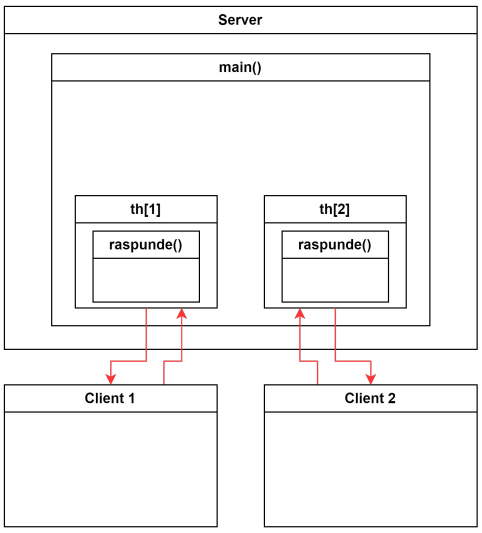
Scopul aplicatiei este de a crea un fel de site de online shopping, care va functiona doar prin comenzi scise din consola si va afisa tot la consola rezultate in funcie de acele comenzi.

2 Tehnologiile utilizate

Aceasta aplicatie va utiliza un server de tip TCP, deorece viteza de transimisie nu este esentiala, singurul lucru care este mai important pentru aplicatie ar fi transmiterea exacta de informatii, fara packet lossuri. Pentru aceasta aplicatie este esential serverul, iar alte tehologii mai importante ar fi threadurile, care permit conectarea a mai multor clienti la server(in acest caz).

3 Arhitectura aplicatiei

In aceasta aplicatie, se pot executa oricare dintre comenzile puse la dispozitie, cum ar fi login(inregistreaza userul cu ajutorul userului si a parolei, care este salvata in baza de date prin "incriptare"), logout, show_all(afiseaza lista de produse). Dupa ce un user s-a inregistrat, se pot executa comenzile pentru selectarea unui produs precum si numarul de produse pentru a fi puse in cos, plasarea produsului in cos, salvarea cosului si stergerea cosului.



Llsta de comenzi
help
login <user> <password>
logout
show_all
show_categories
show_category <category_name>
show_prod_info_by_id <product_id>
show_prod_info_by_name <product_name>
select <product_id>
set <product_count>
add_to_cart
show_cart
delete_cart
exit

4 Detalii de implementare

In aceasta sectiune vor fi prezentate detalii si informatii despre cod, si vor fi explicate, in detaliu, sectiunile imprtante de cod.

Cod din server:

```
prod=fopen("produse.txt", "r");
fscanf(prod, "%d", &prod_count);
for(int i=0; i<prod_count; i++)
{
    fscanf(prod, "%s %s %d", produse[i].nume, produse[i].categ, &produse[i].stoc);
    for(int j=0; j<strlen(produse[i].nume); j++)
        if(produse[i].nume[j]=='_')
            produse[i].nume[j]=' ';
    if(categ_count==0 || strcmp(categorii[categ_count-1], produse[i].categ)!=0)
    {
        strcpy(categorii[categ_count], produse[i].categ);
        categ_count++;
    }
}
fclose(prod);
```

Sectiunea de cod de mai sus salveaza intr-un array de structuri(structul, de tip produs, fiind denumit produse) informatii despre produse, cum ar fi numele si categoria produsului, precum si numarul de produse in stoc. Informatiile sunt luate dintr-un file de tip txt.

```c
s_sd = socket(AF_INET, SOCK_STREAM, 0);
if (s_sd < 0)
{
    perror("Eroare la socket");
    exit(1);
}
printf("Server TCP creat\n");

memset(&s_addr, '\0', sizeof(s_addr));
s_addr.sin_family=AF_INET;
s_addr.sin_addr.s_addr=htonl(INADDR_ANY);
s_addr.sin_port=htons(PORT);

n = bind(s_sd, (struct sockaddr*)&s_addr, sizeof(s_addr));
if (n < 0)
{
    perror("Eroare la bind");
    exit(1);
}
printf("Conectat la portul: %d\n", PORT);
```

Sectiunea de mai sus creaza un socket, cu familia "AF_INET", tipul "SOCK_STREAM" si protocolul "0"(protocolul nu este specificat de apelator). Acest socket este creat cu scopul de a crea serverul TCP. Dupa ce serverul este creat, i se vor atribui serverului prin structul "s_addr" domainul, ip-ul si portul. Dupa, se va verifica daca portul a fost legat sau nu.

```c
listen(s_sd, 5);
while(1)
{
    printf("Astept clientul...\n");

    addr_size = sizeof(c_addr);
    c_sd = accept(s_sd, (struct sockaddr*)&c_addr, &addr_size);
    cl_count++;
    printf("[client %d] - Client conectat\n", cl_count);
    pthread_create(&th[cl_count], NULL, &thread, (void *)(intptr_t)c_sd);
}
```

Aceasta sectiune de cod cauta clientii care sa ca conecteze la ip-ul si portul specificat. Acestia raman conectati la server prin threadul th, clientii fiind numerotati cu ajutorul valorii cl_count.

```
static void *thread(void * arg)
{
    printf ("[client %d] - Astept mesajul...\n", cl_count);
    fflush (stdout);
    pthread_detach(pthread_self());
    raspunde((intptr_t)arg);
    close ((intptr_t)arg);
    return(NULL);
};
```

Aceasta functie are scopul de a face threadul clientului sa numai poata fi apelat si sa stearga toate resursele dupa ce a fost inchis. In functie se mai apeleaza si functaia raspunde, functie care face posibila primirea de la client, procesarea si trimiterea inapoi la client de informatii.

Cod din client:

```
sd = socket(AF_INET, SOCK_STREAM, 0);
if (sd < 0){
    perror("Eroare la socket");
    exit(-1);
}
printf("Socketul pentru serverul TCP a fost creat\n");

memset(&addr, '\0', sizeof(addr));
addr.sin_family=AF_INET;
addr.sin_addr.s_addr=inet_addr(argv[1]);
addr.sin_port=htons(PORT);
```

Acesta parte creaza socketul de conectare la server, dupa care verifica da nu au fost erori la socket. A doua parte atribuie familia de socketuri ”AF_INET”, ip-ul specificat de utilizator si portul definit de ”PORT”.

5 Concluzii

In concluzie, aplicatia "Console Shopper" va utiliza un server de tip TCP deoarece este similar structurat cu un site de online shopping, unde este esentiala trimiterea de informatii cu exactitate, iar partea de server a aplicatiei va contine mare parte din tot codul priectului. O metoda prin care acest proiect poate fi retusat sete imbunatatirea functionalitatilor, de exemplu fiecare utilizator ar putea avea mai multe cosuri, care sa le dea un nume la legere.

References

1. Diferenta dintre TCP si UDP `https://www.spiceworks.com/tech/networking/articles/tcp-vs-udp/`
2. Utilizarea threadurilor in C `https://www.geeksforgeeks.org/multithreading-in-c/`
3. Pagina de curs si laborator a materiei `https://profs.info.uaic.ro/~computernetworks/cursullaboratorul.php`