ReadsProfiler

Duca Alina

Decembrie 2022

Rezumat

Acest proiect a fost creat cu scopul de a avea un server concurent ușor de utilizat, rapid și interactiv, care va oferi acces la o librărie online. serverul va avea acces la un fișier XML care ne va oferi informații despre cărți, precum autorul, anul apariției, genuri și subgenuri, ISBN, rating etc, și la un fișier care va ține evidența căutărilor și a descărcărilor fiecărui utilizator, astfel încât sistemul de recomandări să poată fi capabil de propuneri cât mai adecvate utilizatorilor.

1 Introducere

Acest raport este conceput pentru a demonstra implementarea unui client și a unui server concurent pentru proiectul „ReadsProfiler".

Vom implementa un client și un server concurent care va prelua cererile clienților și le va oferi posibilitatea de a descărca, evalua și de a primi recomandări de cărți. Pe măsură ce activitatea clienților crește, sistemul de recomandări se va putea îmbunătăți în acuratețe, luând în considerare factori precum gusturile unui utilizator, preferințele altor utilizatori cu gusturi similare sau rating-ul pe care l-au primit cărțile de la clienții cărora le-au fost recomandate.

2 Tehnologii utilizate

2.1 TCP – Protocol de control al transmisiei

Protocolul TCP a fost folosit în acest proiect pentru a asigura fiabilitate în transmiterea datelor între client și server.

TCP este un protocol de transport orientat conexiune, fără pierdere de informații, care vizează oferirea calității maxime a serviciilor, integrează mecanisme de stabilire și de eliberare a conexiunii și controlează fluxul de date. O conexiune TCP reprezintă un flux de octeți, iar stabilirea conexiunii se realizează prin mecanismul „Three-way handshake" care are la bază setarea unor flag-uri de control din antetul TCP.

Acest mecanism creează conexiunea dintre client și server înainte ca datele să fie trimise și se ocupă de închiderea corectă a conexiunii la finalul sesiunii de lucru. De asemenea, este capabil să detecteze eventuale erori (cu *checksum* – câmp folosit pentru verificarea sumei de control al pachetului TCP), să secvențieze datele (TCP împarte fluxul de octeți în segmente TCP), și să le retransmită în caz că se pierd (destinatarul verifică numărul de secvență pentru fiecare segment și trimite înapoi câte un număr de confirmare specificând numărul de secvență al următorului octet care se așteaptă a fi recepționat, detectarea segmentelor pierdute realizându-se cu un timer de retransmisie a datelor).

2.2 Parserul XML

Pentru extragerea datelor din fișierul XML, voi folosi un parser XML dintr-o bibliotecă numită pugixml. Aceasta este o bibliotecă de procesare XML C++, care constă dintr-o interfață asemănătoare DOM cu capacități bogate de traversare/modificare, un parser XML extrem de rapid care construiește arborele DOM dintr-un fișier XML și o implementare XPath 1.0 pentru date complexe. Codul este distribuit sub licența MIT.

Pentru utilizarea acestui parser, am inclus biblioteca `#include "files/pugixml.hpp"`, a cărei implementare se regăsește în folderul `"files"`.

Mai precis, în implementarea aplicației voi folosi două fișiere XML: unul în care se vor regăsi informațiile despre cărțile din librărie și unul în care voi ține evidența căutărilor și descărcărilor fiecărui utilizator. În funcție de informațiile regăsite în acest ultim fișier, sistemul va fi capabil să facă clienților recomandări cât mai bune.

3 Arhitectura aplicației

3.1 Arhitectura clientului

- **Clientul** apelează funcția `socket()`, salvează descriptorul returnat într-o variabilă și se conectează la server prin apelul primitivei `connect()`.
- După conectarea la server, se citește comanda care va fi dată în terminal de către utilizator și se trimite serverului.
- După ce serverul va trimite un răspuns adecvat, îl voi citi și îl voi afișa în terminal.
- Dacă se va trimite serverului o comandă care presupune descărcarea unei cărți, atunci serverul va trimite înapoi clientului URL-ul unde poate fi găsită cartea, acestuia revenindu-i sarcina de a descărca pdf-ul la care ne conduce link-ul, apelând funcții din biblioteca `#include <curl/curl.h>`.

3.2 Arhitectura serverului

- **Serverul** apelează funcția `socket()`, salvează descriptorul returnat într-o variabilă, populează structura de date folosită de server și atașează socket-ului informațiile respective, apelând funcția `bind()`.
- Ascultă dacă se conectează clienți, prin apelul primitivei `listen()`.
- Intră într-o buclă infinită, în cadrul căreia acceptă conexiunea cu un client prin apelul funcției `accept()`, salvând totodată descriptorul returnat.
- Creează un thread cu descriptorul întors de `accept()` printr-un apel al funcției `pthread_create()`.
- În handler-ul thread-ului, `treat()`, va intra într-o buclă infinită, în cadrul căreia va citi comanda trimisă de către client și o va procesa.
- În cadrul procesării comenzii, serverul va apela la informațiile conținute în două baze de date, **books** și **users**, prima dintre ele conținând toate informațiile despre cărțile disponibile în librărie (precum titlul, autorul, anul apariției, ISBN-ul, rating-ul, genurile și subgenurile în care se încadrează și link-ul de unde poate fi descărcată cartea), iar a doua conținând informații despre utilizatori (precum numele de utilizator, cărțile căutate, cărțile descărcate, nota oferită fiecărei cărți în cazul în care utilizatorul a dorit să evalueze cartea, genul și subgenul preferat al utilizatorului), pentru a le putea oferi acestora recomandări cât mai bune.
- În cadrul programului, vom lucra cu utilizatorii și cu cărțile disponibile prin intermediul a două clase: **Books** și **Users**.
- În urma procesării comenzii, serverul va trimite clientului un răspuns conform cu comanda trimisă.
- În cazul în care clientul părăsește aplicația (de exemplu, prin trimiterea unui semnal precum SIGINT, SIGTSTP sau SIGQUIT) sau trimite serverului comanda „quit”, acesta va ieși din bucla infinită a handler-ului de thread. Altfel, serverul va relua execuția buclei.

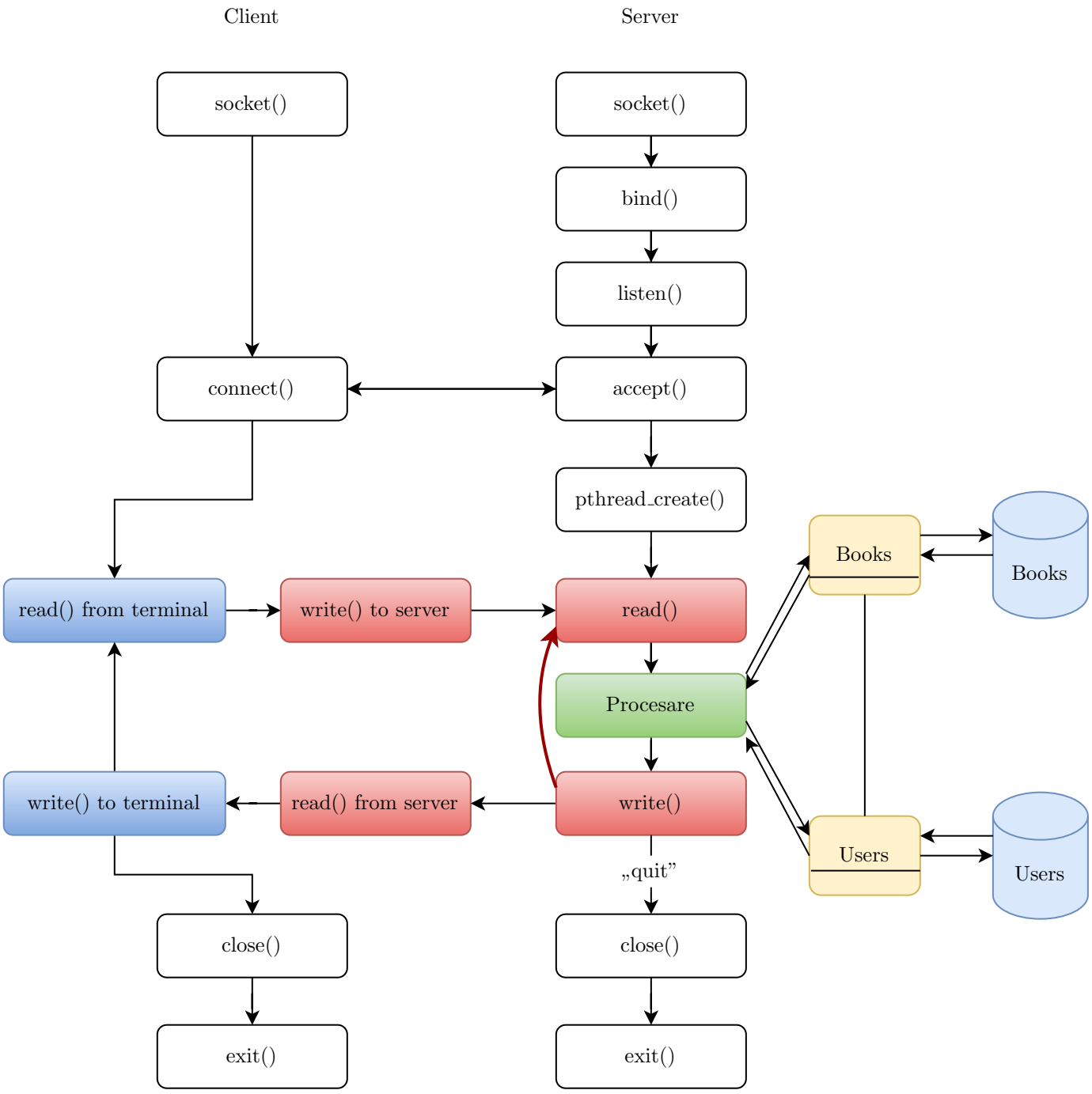

Figura 1: Structura programului

4 Detalii de implementare

În cadrul programului, toate mesajele reprezintă șiruri de caractere care permit o lungime maximă de 1000 de caractere.

Comunicarea efectivă de la client către server se realizează cu ajutorul unor comenzi specifice:

- „login : <username>" - va căuta în baza de date existența utilizatorului <username>, iar în cazul în care acesta nu există, va fi adăugat în baza de date. În cazul în care această comandă nu va fi dată, sau va fi dată, dar cu un nume de utilizator care să conțină mai puțin de 3 caractere, nu se va putea afla nicio informație despre cărțile disponibile în librărie.
- „search-title : <titlu>" - va căuta în baza de date cartea cu titlul menționat. Dacă sunt mai multe astfel de cărți (de exemplu, „Poezii" de Mihai Eminescu și „Poezii" de George Coșbuc), se vor afișa primele 5 cărți încă nedescărcate de utilizatorul curent, în ordinea care ne va fi oferită de către sistemul de recomandare.
- „search-author : <autor>" - va căuta în baza de date cărțile scrise de autorul menționat. Dacă sunt mai multe astfel de cărți, se vor afișa primele 5 cărți încă nedescărcate de utilizatorul curent, în ordinea care ne va fi oferită de către sistemul de recomandare.
- „search-year : <an>" - va căuta în baza de date cărțile publicate în anul menționat. Dacă sunt mai multe astfel de cărți, se vor afișa primele 5 cărți încă nedescărcate de utilizatorul curent, în ordinea care ne va fi oferită de către sistemul de recomandare.
- „search-genre : <gen>" - va căuta în baza de date cărțile care se încadrează în genul menționat. Dacă sunt mai multe astfel de cărți, se vor afișa primele 5 cărți încă nedescărcate de utilizatorul curent, în ordinea care ne va fi oferită de către sistemul de recomandare.
- „search-subgenre : <subgen>" - va căuta în baza de date cărțile care se încadrează în subgenul menționat. Dacă sunt mai multe astfel de cărți, se vor afișa primele 5 cărți încă nedescărcate de utilizatorul curent, în ordinea care ne va fi oferită de către sistemul de recomandare.
- „search-isbn : <isbn>" - va căuta în baza de date cartea cu ISBN-ul menționat.
- „search-rating : <rating>" - va căuta în baza de date cărțile care au rating-ul cuprins între [rating] și [rating]+1, acesta constând într-un număr între 1 și 10 cu o singură zecimală. Dacă sunt mai multe astfel de cărți, se vor afișa primele 5 cărți încă nedescărcate de utilizatorul curent, în ordinea care ne va fi oferită de către sistemul de recomandare.
- „eval-book : <titlu>_<autor>_<nota>" - va permite utilizatorului să evalueze o carte și, pe baza notei, va modifica rating-ul cărții în baza de date corespunzătoare.
- „download-book : <titlu>_<autor>" - va căuta în baza de date cartea menționată, după care i se va prelua adresa URL și i se va trimite clientului de către server, pentru ca acesta să o descarce pentru utilizator.
- „logout" - deloghează utilizatorul.
- „quit" - permite utilizatorului să părăsească serverul.
- „man" - va afișa comenzile pe care utilizatorul le poate da.

4.1 Client

În cadrul implementării clientului se va utiliza primitiva `socket()`, după cum urmează:

```
if((sd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
    perror("Eroare la socket().\n");
    return errno;
}
```

După apelul primitivei `socket()` urmează popularea structurii de date `server` cu IP-ul și portul oferite la lansarea în execuție a clientului și conectarea la server cu ajutorul unui apel al funcției `connect()`:

```
server.sin_family = AF_INET;
server.sin_addr.s_addr = inet_addr(argv[1]);
server.sin_port = htons(port);
if(connect(sd, (struct sockaddr *) &server, sizeof(struct sockaddr)) == -1) {
    perror("Eroare la connect().\n");
    return errno;
}
```

În continuare, clientul va intra într-o buclă infinită în cadrul căreia va citi comenzile date de utilizator, i le va trimite serverului, va aștepta un răspuns din partea acestuia și le va afișa, până când acesta va trimite o comandă sau un semnal de părăsire a serverului.

Tot aici, clientul va verifica dacă nu cumva mesajul trimis de server conține un link la o carte, iar în caz afirmativ, va descărca această carte în felul următor:

```
int i = url.size();
for(; i >= 0; i--)
    if(url[i] == '/')
        break;
file_path = url.substr(i + 1, url.size() - 1);
CURL* curl;
FILE* fp;
CURLcode res;
curl = curl_easy_init();
if(curl) {
    fp = fopen(file_path.c_str(), "wb");
    curl_easy_setopt(curl, CURLOPT_URL, url.c_str());
    curl_easy_setopt(curl, CURLOPT_WRITEDATA, fp);
    res = curl_easy_perform(curl);
    curl_easy_cleanup(curl);
    fclose(fp);
}
```

4.2 Server

În cazul serverului, apelul primitivei `socket()` și popularea structurii de date `server` se realizează într-o manieră asemănătoare. Acesta, în schimb, apelează încă două primitive înainte de a realiza conexiunea cu clientul - `bind()` și `listen()`:

```
if(bind(sd, (struct sockaddr *) &server, sizeof(struct sockaddr)) == -1) {
    perror("Eroare la bind().\n");
    return errno;
}
if(listen(sd, 5) == -1) {
    perror("Eroare la listen().\n");
    return errno;
}
```

Acceptarea efectivă a clienților va avea loc într-o buclă infinită, va fi realizată prin apelul primitivei `accept()` și va fi urmată de crearea thread-ului corespunzător clientului curent, astfel încât serverul își va servi clienții în mod concurent:

```
if((client = accept(sd, (struct sockaddr *) &from, &length)) < 0) {
    perror("Eroare la accept().\n");
    continue;
}
```

```
td = (struct thData *)malloc(sizeof(struct thData));
td->idThread = thNumber++;
td->cl = client;
pthread_create(&th[thNumber], NULL, &treat, td);
```

În cadrul handler-ului thread-ului, `treat`, voi avea o buclă infinită în care voi interacționa cu clientul, până când acesta va părăsi serverul:

```
 while(1)
    if(raspuns((struct thData*) arg, user) == -1) {
        close((intptr_t)arg);
        printf("Clientul cu thread-ul %d a parasit serverul.\n", tdL.idThread);
        return(NULL);
    }
```

Funcția `raspuns()`, apelată în cadrul handler-ului mai sus menționat, va realiza comunicarea cu clientul: va citi comanda transmisă de acesta, va apela funcția `procesare()` și va trimite răspunsul așteptat. În cazul în care clientul părăsește serverul, funcția va returna valoarea `-1` și va determina închiderea conexiunii cu clientul în cadrul funcției `treat()`.

```
tdL = *((struct thData*) arg);
if(read(tdL.cl, msg, sizeof(msg)) <= 0) {
    perror("Eroare la read().\n");
    return -1;
}
msg[strlen(msg) - 1] = '\0';
if(!strcmp(msg, "quit"))
    return -1;
procesare(msg, user, rasp);
if(write(tdL.cl, rasp, sizeof(rasp)) <= 0) {
    perror("Eroare la write().\n");
    return -1;
}
```

4.3 Scenarii de utilizare

1. Clientul părăsește brusc serverul – serverul nu va aștepta la infinit o comandă din partea clientului, ci va închide conexiunea cu acesta, returnându-se `-1` în cadrul funcției `raspuns()`.
2. Utilizatorul introduce o comandă inexistentă – serverul va trimite mesajul „comandă inexistentă" și va continua interacțiunea cu clientul.
3. Clientul va da o comandă de căutare a unei cărți, de evaluare, de descărcare sau de logout fără să se fi logat – serverul va trimite clientului mesajul „Userul nu este logat!".
4. Clientul va da comanda login, dar cu mai puțin de 3 caractere pentru numele de utilizator – serverul va trimite mesajul „Numele de utilizator trebuie sa contina minim 3 caractere."

5 Concluzii

Scopul proiectului este acela de a oferi acces la o librărie online într-un mod ușor și rapid, fiind capabil să facă recomandări clienților în funcție de gusturile acestora.

Aplicația poate fi îmbunătățită prin implementarea unei interfețe grafice care să permită clienților o interacțiune mai facilă cu cărțile, eventual afișarea unui preview pentru cărțile căutate, și bhjkl.gbhyuj,kghyui,îmbunătățirea sistemului de recomandare în funcție nu numai de căutările altor clienți, ci și de vârsta, sexul, regiunea sau religia celorlalți clienți.

Bibliografie

[1] Lenuța Alboaie, *Nivelul transport* (Curs *Rețele de calculatoare*), 2022.

[2] https://github.com/zeux/pugixml.

[3] https://curl.se/libcurl/c/CURLOPT_XFERINFOFUNCTION.html

[4] https://leimao.github.io/blog/Download-Using-LibCurl/

[5] https://app.diagrams.net/

[6] https://profs.info.uaic.ro/~computernetworks/