Proiect Retele de Telecomunicatii

Autori: Szasz Alexandru Profesori coordinatori:

Livezeanu Tudor Ing. Gabriel LAZAR

Echipa rtc1s3e1 Sl.dr.ing. Iustin-Alexandru IVANCIU

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

// DECLARAREA FUNCTIILOR

void display\_menu();

int connect\_to\_server(int \*sock, struct sockaddr\_in \*server\_addr);

void disconnect\_from\_server(int \*sock);

void send\_http\_get(int sock);

void save\_to\_file(const char \*data);

// FUNCTIA PRINCIPALA

int main() {

// Variabila pentru optiunea aleasa din meniu

int option;

// Variabila pentru descriptorul socketului

// (-1 inseamna ca nu exista conexiune)

int sock = -1;

// Structura care stocheaza adresa si portul serverului

struct sockaddr\_in server\_addr;

// Detaliile serverului (IP si port)

// IP-ul icio.us

char server\_ip[] = "107.181.87.5"; // IP-ul icio.us

// Conform cerintei, se va folosi portul 80, adesea folosit in comunicarea HTTP

int server\_port = 80;

// Folosim memset pentru a initializa obiectul server\_addr cu 0-uri, eliminand

// datele garbage;

memset(&server\_addr, 0, sizeof(server\_addr));

// Clarificare de ce AF\_INET si nu PF\_INET?

server\_addr.sin\_family = AF\_INET; // Address Family for IPv4

// Setam portul 80. Folosim htons pentru a converti numarul portului de la host byte order la network byte order

server\_addr.sin\_port = htons(server\_port);

// Convertim reprezentarea IP-ului sub forma de string in binar si il stocam in server\_addr.sin\_addr

inet\_pton(AF\_INET, server\_ip, &server\_addr.sin\_addr);

// Bucla de afisare a meniului principal

do {

// Apelam functia de afisare a meniului principal

display\_menu();

printf("Introduceti optiunea dorita: ");

scanf("%d", &option);

// Implementam fiecare optiune aleasa

switch(option) {

case 1:

// Conectare la server: conexiunea a avut succes daca functia returneaza 0

if (connect\_to\_server(&sock, &server\_addr) == 0) {

printf("Conectat la server.\n");

}

break;

case 2:

// Deconectare de la server

disconnect\_from\_server(&sock);

break;

case 3:

// Rulare comanda "GET / HTTP/1.0\r\n\r\n"

// Daca suntem conectati la server (sock != -1), atunci apelam functia de GET HTTP

if (sock != -1) {

send\_http\_get(sock);

} else {

// Daca nu suntem conectati la server (sock == -1)

printf("Este necesar sa fii conectat la server mai intai!\n");

}

break;

case 4:

// Iesire din aplicatie

// Ne deconectam de la server si anuntam iesirea

disconnect\_from\_server(&sock);

printf("Aplicatia se inchide...\n");

break;

default:

// Pentru numerele diferite de 1, 2, 3 sau 4

printf("Optiune invalida. Incercati din nou.\n");

}

} while (option != 4);

return 0;

}

// DEFINIREA FUNCTIILOR

// Functia de afisare a meniului

void display\_menu() {

printf("\n--- Meniu ---\n");

printf("1. Conectare la server\n");

printf("2. Deconectare de la server\n");

printf("3. Rulare comanda \"GET / HTTP/1.0\\r\\n\\r\\n\"\n");

printf("4. Iesire din aplicatie\n");

}

/\* Functia de conectare la server:

\* Stabilim o conexiune la server folosind un socket TCP.

\*

\* parametri:

\* - sock: Pointer spre descriptorul socketului.

\* Acesta ne indica statusul conexiunii (daca e -1, nu suntem conectati)

\* - server\_addr: Pointer spre structura adresei serverului.

\* Contine adresa IP si portul serverului.

\* Pasat prin referinta (\*) pentru a evita copierea datelor inutil.

\* Returnam 0 daca conexiunea a avut succes, -1 daca a esuat.

\*/

int connect\_to\_server(int \*sock, struct sockaddr\_in \*server\_addr) {

// Verificam sa nu fim conectati deja.

if (\*sock != -1) {

printf("Sunteti deja conectat la server.\n");

return -1;

}

// Cream un socket nou folosind functia socket()

// AF\_INET -> IPv4

// SOCK\_STREAM -> TCP

\*sock = socket(AF\_INET, SOCK\_STREAM, 0);

if (\*sock < 0) {

// Daca crearea socketului esueaza, afisam o eroare

perror("EROARE: Crearea socketului a esuat!");

return -1;

}

// Incercam sa ne conectam folosind functia connect()

// Daca esueaza (functia connect() returneaza < 0), atunci:

// inchidem socketul, resetam sock la -1 si afisam eroare

if (connect(\*sock, (struct sockaddr \*)server\_addr, sizeof(\*server\_addr)) < 0) {

perror("EROARE: Conexiunea a esuat.\n");

close(\*sock);

\*sock = -1;

return -1;

}

// Daca functia returneaza 0, inseamna ca a reusit conectarea la server

return 0;

}

/\* Functia de deconectare de la server:

\* Inchide conexiunea activa cu serverul, daca exista

\*

\* parametri:

\* - sock: Pointer spre descriptorul socketului

\* Acesta ne indica statusul conexiunii. (daca e -1, nu suntem conectati)

\*/

void disconnect\_from\_server(int \*sock) {

// Daca exista o conexiune activa

if (\*sock != -1) {

// Inchidem conexiunea, resetam socketul la -1 si afisam mesaj

close(\*sock);

\*sock = -1;

printf("Deconectat de la server\n");

} else {

// Daca eram deja deconectati, afisam un mesaj sugestiv

printf("Nu exista conexiuni active de la care sa ne deconectam.\n");

}

}

/\* Functie de rulare a comenzii GET HTTP:

\* Trimites un request HTTP GET spre serverul la care suntem conectati si

\* procesam raspunsul

\*

\* parametri:

\* -sock - Descriptorul socketului pentru conexiunea activa

\* FOlosit pentru a trimite si primi date de la server

\*/

void send\_http\_get(int sock) {

// Stringul requestului din cerinta

char request[] = "GET / HTTP/1.0\r\n\r\n";

char buffer[1024];

int bytes\_received;

// Folosim functia send() pentru a trimite requestul

send(sock, request, strlen(request), 0);

// Primim raspunsul si il stocam

bytes\_received = recv(sock, buffer, sizeof(buffer) - 1, 0);

if (bytes\_received > 0) {

// Ne asiguram ca sirul de date este null-terminated ('\0')

buffer[bytes\_received] = '\0';

// Afisam raspunsul si il salvam in index.html folosind functia save\_to\_file

printf("Raspuns de la server:\n%s\n", buffer);

save\_to\_file(buffer);

} else {

// Daca nu avem raspuns, afisam mesaj sugestiv

printf("Niciun raspuns de la server.\n");

}

}

/\* Functie de salvare a raspunsului de la server in fisierul index.html:

\* Salvam raspunsul de la server conform cerintei.

\*

\* parametri:

\* - data: Pointer spre sirul de bytes care formeaza raspunsul

\* Raspunsul este stocat ca un sir de bytes (char) si este null-terminated

\*/

void save\_to\_file(const char \*data) {

// Deschidem fisierul in write mode ("w")

FILE \*file = fopen("index.html", "w");

if (file) {

// Scriem raspunsul si inchidem fisierul dupa

fprintf(file, "%s", data);

fclose(file);

printf("Raspunsul a fost salvat in index.html\n");

} else {

perror("EROARE: Fisierul nu a putut fi creat");

}

}