#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <unistd.h>

*// Declaration of a function to handle GET requests*

void handle\_get(int client\_socket, char \*request);

*// Declaration of a function to handle POST requests*

void handle\_post(int client\_socket, char \*request);

int main(int argc, char \*argv[]) {

int server\_socket, client\_socket;

struct sockaddr\_in server\_address, client\_address;

socklen\_t client\_address\_len;

*// Create socket*

server\_socket = socket(AF\_INET, SOCK\_STREAM, 0);

if (server\_socket == -1) {

perror("Unable to create socket");

exit(EXIT\_FAILURE);

}

*// Bind socket to port 5000*

memset(&server\_address, 0, sizeof(struct sockaddr\_in));

server\_address.sin\_family = AF\_INET;

server\_address.sin\_addr.s\_addr = INADDR\_ANY;

server\_address.sin\_port = htons(5000);

if (bind(server\_socket, (struct sockaddr \*) &server\_address, sizeof(server\_address)) == -1) {

perror("Unable to bind socket to port");

exit(EXIT\_FAILURE);

}

*// Listen for incoming connections*

if (listen(server\_socket, 10) == -1) {

perror("Unable to listen for incoming connections");

exit(EXIT\_FAILURE);

}

printf("Server is listening on port 5000...\n");

while (1) {

*// Accept incoming connection*

client\_address\_len = sizeof(client\_address);

client\_socket = accept(server\_socket, (struct sockaddr \*) &client\_address, &client\_address\_len);

if (client\_socket == -1) {

perror("Failed to accept incoming connection");

continue;

}

printf("Accepted new connection from %s:%d\n", inet\_ntoa(client\_address.sin\_addr), ntohs(client\_address.sin\_port));

*// Read HTTP request from client*

char request[1024];

int length = read(client\_socket, request, sizeof(request));

if (length < 0) {

perror("Failed to read from client");

close(client\_socket);

continue;

}

*// Determine the HTTP request method*

char http\_method[10];

int i = 0;

while (request[i] != ' ') {

http\_method[i] = request[i];

i++;

}

http\_method[i] = '\0';

*// Determine the URI from the HTTP request*

char uri[1024];

int j = 0;

i++;

while (request[i] != ' ') {

uri[j] = request[i];

i++;

j++;

}

uri[j] = '\0';

printf("Request received: method=%s, uri=%s\n", http\_method, uri);

*// Route the HTTP request*

if (strcmp(http\_method, "GET") == 0) {

handle\_get(client\_socket, uri);

} else if (strcmp(http\_method, "POST") == 0) {

handle\_post(client\_socket, uri);

} else {

char error\_message[] = "HTTP/1.1 400 Bad Request\r\nContent-Type: text/plain\r\nContent-Length: 0\r\n\r\n";

write(client\_socket, error\_message, strlen(error\_message));

}

*// Close client socket*

close(client\_socket);

printf("Connection closed\n");

}

*// Close server socket*

close(server\_socket);

return 0;

}

void handle\_get(int client\_socket, char \*request) {

if (strcmp(request, "/") == 0) {

char message[] = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\nContent-Length: 77\r\n\r\n<html><body><h1>Welcome to my website</h1></body></html>";

write(client\_socket, message, strlen(message));

} else if (strcmp(request, "/about") == 0) {

char message[] = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\nContent-Length: 72\r\n\r\n<html><body><h1>About me</h1><p>I'm a programmer</p></body></html>";

write(client\_socket, message, strlen(message));

} else {

char error\_message[] = "HTTP/1.1 404 Not Found\r\nContent-Type: text/plain\r\nContent-Length: 0\r\n\r\n";

write(client\_socket, error\_message, strlen(error\_message));

}

}

void handle\_post(int client\_socket, char \*request) {

char success\_message[] = "HTTP/1.1 200 OK\r\nContent-Type: text/plain\r\nContent-Length: 15\r\n\r\nPost received\n";

write(client\_socket, success\_message, strlen(success\_message));

}

דוגמה זו מאזינה ביציאה 5000 לחיבורים נכנסים, קוראת בקשות HTTP מלקוחות, מנתחת את הבקשה לקביעת שיטת HTTP ו-URI ומנתבת את הבקשה לפונקציית המטפל המתאימה. במקרה זה, **`handle\_get`** מטפל בבקשות GET ו-**`handle\_post`** מטפל בבקשות POST. אם ה-URI אינו מזוהה על ידי השרת, שגיאת 404 Not Found מוחזרת ללקוח. אם שיטת ה-HTTP אינה מזוהה, מוחזרת שגיאת 400 Bad Request. זוהי רק דוגמה פשוטה וניתן ליישם היגיון ניתוב מורכב יותר בהתבסס על הצרכים של הפרויקט הספציפי.

מ