**Task No 4**

**C++ Programming**

**Name: Hamna Jahangir**

**Date: July 27,2024**

**Code:**

**#include <iostream>**

**#include <thread>**

**#include <vector>**

**#include <cstring>**

**#include <fstream>**

**#include <sstream>**

**#include <netinet/in.h>**

**#include <unistd.h>**

**const int PORT = 8080;**

**const int BUFFER\_SIZE = 1024;**

**void handle\_client(int client\_socket) {**

**char buffer[BUFFER\_SIZE];**

**int bytes\_received = recv(client\_socket, buffer, BUFFER\_SIZE - 1, 0);**

**if (bytes\_received < 0) {**

**std::cerr << "Error receiving data." << std::endl;**

**close(client\_socket);**

**return;**

**}**

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

**std::string request(buffer);**

**std::string response;**

**// Parse the HTTP request to extract the requested file**

**std::istringstream request\_stream(request);**

**std::string method, path;**

**request\_stream >> method >> path;**

**if (method == "GET") {**

**// Remove the leading '/' from the path**

**if (path == "/") path = "/index.html";**

**std::string file\_path = "www" + path;**

**std::ifstream file(file\_path);**

**if (file.is\_open()) {**

**std::stringstream content;**

**content << file.rdbuf();**

**file.close();**

**response = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n";**

**response += content.str();**

**} else {**

**response = "HTTP/1.1 404 Not Found\r\nContent-Type: text/html\r\n\r\n";**

**response += "<html><body><h1>404 Not Found</h1></body></html>";**

**}**

**} else {**

**response = "HTTP/1.1 405 Method Not Allowed\r\nContent-Type: text/html\r\n\r\n";**

**response += "<html><body><h1>405 Method Not Allowed</h1></body></html>";**

**}**

**send(client\_socket, response.c\_str(), response.size(), 0);**

**close(client\_socket);**

**}**

**int main() {**

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

**if (server\_socket == 0) {**

**std::cerr << "Socket creation failed." << std::endl;**

**return 1;**

**}**

**sockaddr\_in address;**

**address.sin\_family = AF\_INET;**

**address.sin\_addr.s\_addr = INADDR\_ANY;**

**address.sin\_port = htons(PORT);**

**if (bind(server\_socket, (sockaddr\*)&address, sizeof(address)) < 0) {**

**std::cerr << "Bind failed." << std::endl;**

**return 1;**

**}**

**if (listen(server\_socket, 3) < 0) {**

**std::cerr << "Listen failed." << std::endl;**

**return 1;**

**}**

**std::cout << "Server listening on port " << PORT << std::endl;**

**std::vector<std::thread> threads;**

**while (true) {**

**int client\_socket = accept(server\_socket, nullptr, nullptr);**

**if (client\_socket < 0) {**

**std::cerr << "Accept failed." << std::endl;**

**continue;**

**}**

**threads.emplace\_back(handle\_client, client\_socket);**

**}**

**// Join all threads (not typically reached, since server runs indefinitely)**

**for (auto& t : threads) {**

**if (t.joinable()) {**

**t.join();**

**}**

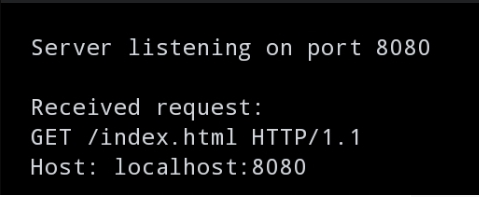
**}**

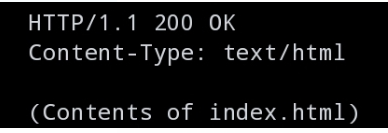
**close(server\_socket);**

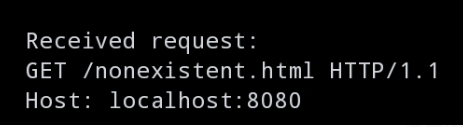
**return 0;**

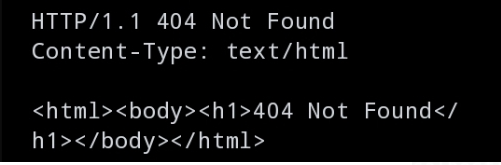
**}**

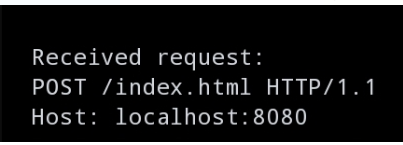
**Output**:

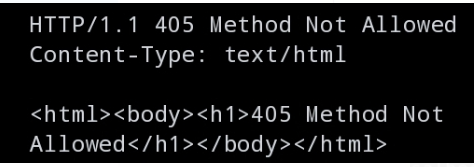


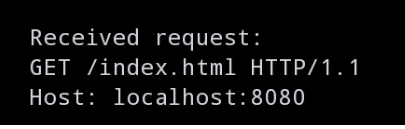


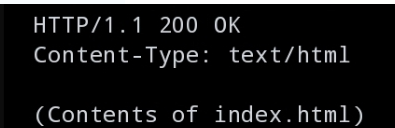












**Github link:**

**<https://github.com/Hamna-123739/DEP-C-Programming>**

**Linkedln link:**

**<https://www.linkedin.com/in/hamna-jahangir-51b3a9311?utm_source=share&utm_campaign=share_via&utm_content=profile&utm_medium=android_app>**