МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра информационных технологий**

**Отчёт по лабораторной работе №5**

**«Сетевое клиент-серверное приложение на основе протокола HTTP»**

Работу выполнил

студент 22 группы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н.Е. Лучин

Направление подготовки 01.03.02 Прикладная математика и информатика

Проверил

канд. техн. наук, доц.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.А. Полупанов

Краснодар 2022

**Цель работы:** изучить основные положения протокола HTTP, реализовать сетевое клиент-серверное приложение на основе протокола HTTP.

**Задача:**

**a:** Написать на C++ клиент-серверное приложение, осуществляющее взаимодействие по протоколу HTTP на основе TCP-сокетов

**б:** Используя HTTP-клиент, написанный на C++, обратиться методом GET к одному из стандартных Web-серверов (например, json.org)

**в:** Используя стандартный браузер, обратиться к HTTP-серверу, написанному на C++

**г:** С помощью программы TELNET отправить HTTP-запрос к веб-серверу.

**Практическая часть:**

Листинг клиента:

#include <winsock2.h>

#include <windows.h>

#include <WS2tcpip.h>

#include <iostream>

#include <string>

#pragma comment(lib, "Ws2\_32.lib")

#define MAX\_PACKET\_SIZE 4096

using namespace std;

int main() {

    int port = 80;

    string path, website;

    cout << "Enter path: ";

    cin >> path;

    if (path == "0")

    {

        path = "";

        port = 8000;

    }

    cout << "Enter website: ";

    cin >> website;

    string request = "GET /" + path + " HTTP/1.0 \r\nHost: " + website + " \r\n\r\n";

    WSADATA wsData;

    int erStat = WSAStartup(MAKEWORD(2, 2), &wsData);

    if (erStat != 0) {

        cout << "Error WinSock initializaion # ";

        cout << WSAGetLastError();

        return 1;

    }

    else

        cout << "WinSock initialization is OK" << endl;

    SOCKET ClntSock = socket(AF\_INET, SOCK\_STREAM, 0);

    if (ClntSock == INVALID\_SOCKET) {

        cout << "Error initialization socket # "

<< WSAGetLastError() << endl;

        closesocket(ClntSock);

        WSACleanup();

        return 1;

    }

    else

        cout << "Client socket initialization is OK"

<< endl;

    struct addrinfo hints, \*res;

    struct in\_addr addr;

    memset(&hints, 0, sizeof(hints));

    hints.ai\_socktype = SOCK\_STREAM;

    hints.ai\_family = AF\_INET;

    // новый аналог gethostbyname()

    erStat = getaddrinfo(website.data(), NULL, &hints, &res);

    if (erStat != 0) {

        cout << "Error getting website information # ";

        cout << WSAGetLastError();

        return 1;

    }

    else

        cout << "WinSock initialization is OK" << endl;

    sockaddr\_in\* serv\_info = (sockaddr\_in\*)res -> ai\_addr;

    char ipstr[INET6\_ADDRSTRLEN];

    inet\_ntop(AF\_INET, &(serv\_info->sin\_addr), (PSTR)ipstr, sizeof(ipstr));

    cout << "Commencing connection via address: "

<< ipstr << endl;

    serv\_info->sin\_port = htons(port);

    erStat = connect(ClntSock, (sockaddr\*)serv\_info, sizeof(\*serv\_info));

    if (erStat != 0) {

        cout << "Error connecting to the website # ";

        cout << WSAGetLastError();

        return 1;

    }

    else

        cout << "Website connection is OK" << endl;

    erStat = send(ClntSock, request.data(), sizeof(char) \* request.size(), 0);

    int len = 0;

    char buff[MAX\_PACKET\_SIZE];

    do

    {

        len = recv(ClntSock, (char\*)&buff, MAX\_PACKET\_SIZE, 0);

        buff[len] = 0;

        cout << buff;

    } while (len != 0);

    if (SOCKET\_ERROR == closesocket(ClntSock))

        return -1;

    return 1;

}

Листинг сервера:

#include <winsock2.h>

#include <string>

#include <sstream>

#include <windows.h>

#include <WS2tcpip.h>

#include <iostream>

#pragma comment(lib, "Ws2\_32.lib")

using namespace std;

int main()

{

    WSADATA wsData;

    int erStat = WSAStartup(MAKEWORD(2, 2), &wsData);

    if (erStat != 0) {

        cout << "Error WinSock initializaion # ";

        cout << WSAGetLastError();

        return 1;

    }

    else

        cout << "WinSock initialization is OK" << endl;

    addrinfo\* addr = NULL;

    addrinfo hints;

    ZeroMemory(&hints, sizeof(hints));

    hints.ai\_family = AF\_INET;

    hints.ai\_socktype = SOCK\_STREAM;

    hints.ai\_protocol = IPPROTO\_TCP;

    hints.ai\_flags = AI\_PASSIVE;

    erStat = getaddrinfo("127.0.0.1", "8000", &hints, &addr);

    if (erStat != 0) {

        cout << "Gettint address has failed # ";

        cout << WSAGetLastError();

        return 1;

    }

    SOCKET ServSock = socket(addr->ai\_family, addr->ai\_socktype, addr->ai\_protocol);

    if (ServSock == INVALID\_SOCKET) {

        cout << "Error initialization socket # " << WSAGetLastError() << endl;

        closesocket(ServSock);

        WSACleanup();

        return 1;

    }

    else

        cout << "Server socket initialization is OK"

<< endl;

    erStat = bind(ServSock, addr->ai\_addr, (int)addr->ai\_addrlen);

    if (erStat != 0) {

        cout << "Error Socket binding. Error # "

<< WSAGetLastError() << endl;

        closesocket(ServSock);

        WSACleanup();

        return 1;

    }

    else

        cout << "Binding socket to Server info is OK"

<< endl;

    erStat = listen(ServSock, SOMAXCONN);

    if (erStat != 0) {

        cout << "Can't start to listen to. Error # "

<< WSAGetLastError() << endl;

        closesocket(ServSock);

        WSACleanup();

        return 1;

    }

    else {

        cout << "Listening..." << endl;

    }

    const int max\_client\_buffer\_size = 1024;

    char buf[max\_client\_buffer\_size];

    long long unsigned int ClntSock = INVALID\_SOCKET;

    while (true) {

        ClntSock = accept(ServSock, NULL, NULL);

        if (ClntSock == INVALID\_SOCKET) {

            cout << "Can't accept connection. Error # " << WSAGetLastError() << endl;

            closesocket(ServSock);

            WSACleanup();

            return 1;

        }

        erStat = recv(ClntSock, buf, max\_client\_buffer\_size, 0);

        stringstream response;

        stringstream response\_body;

        if (erStat == SOCKET\_ERROR) {

            cout << "Recieving data failed. Error # "

<< WSAGetLastError() << endl;

            closesocket(ServSock);

            WSACleanup();

            return 1;

        }

        else if (erStat == 0) {

            cout << "Connection was closed " << endl;

        }

        else if (erStat > 0) {

            buf[erStat] = '\0';

            response\_body << "<title>Test C++ HTTP Server</title>\n"

                << "<h1>Test page</h1>\n"

                << "<p>This is body of the test page...</p>\n"

                << "<h2>Request headers</h2>\n"

                << "<pre>" << buf << "</pre>\n"

                << "<em><small>Test C++ Http Server</small></em>\n";

            response << "HTTP/1.1 200 OK\r\n"

                << "Version: HTTP/1.1\r\n"

                << "Content-Type: text/html; charset=utf-8\r\n"

                << "Content-Length: "

<< response\_body.str().length()

                << "\r\n\r\n"

                << response\_body.str();

            erStat = send(ClntSock, response.str().c\_str(),

                response.str().length(), 0);

            if (erStat == SOCKET\_ERROR) {

                cout << "Sending message failed. Error#" << WSAGetLastError() << endl;

                closesocket(ServSock);

                WSACleanup();

                return 1;

            }

            closesocket(ClntSock);

        }

    }

    closesocket(ServSock);

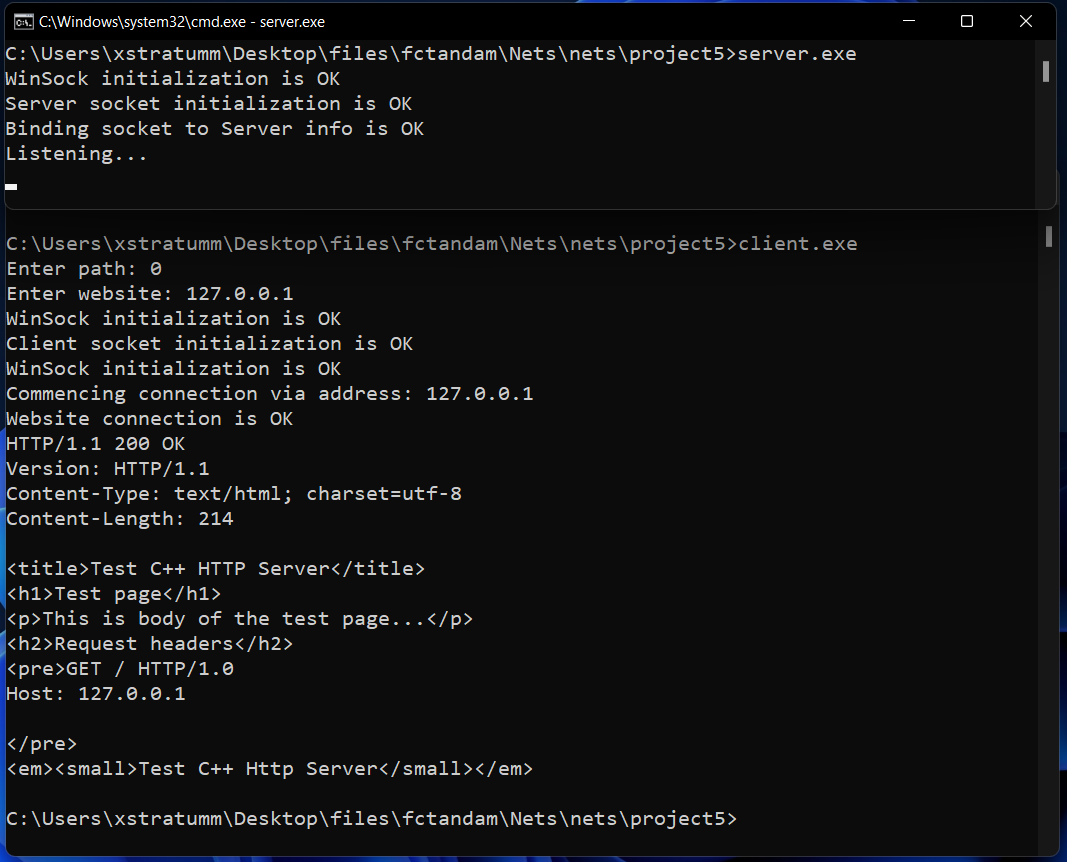
    freeaddrinfo(addr);

    WSACleanup();

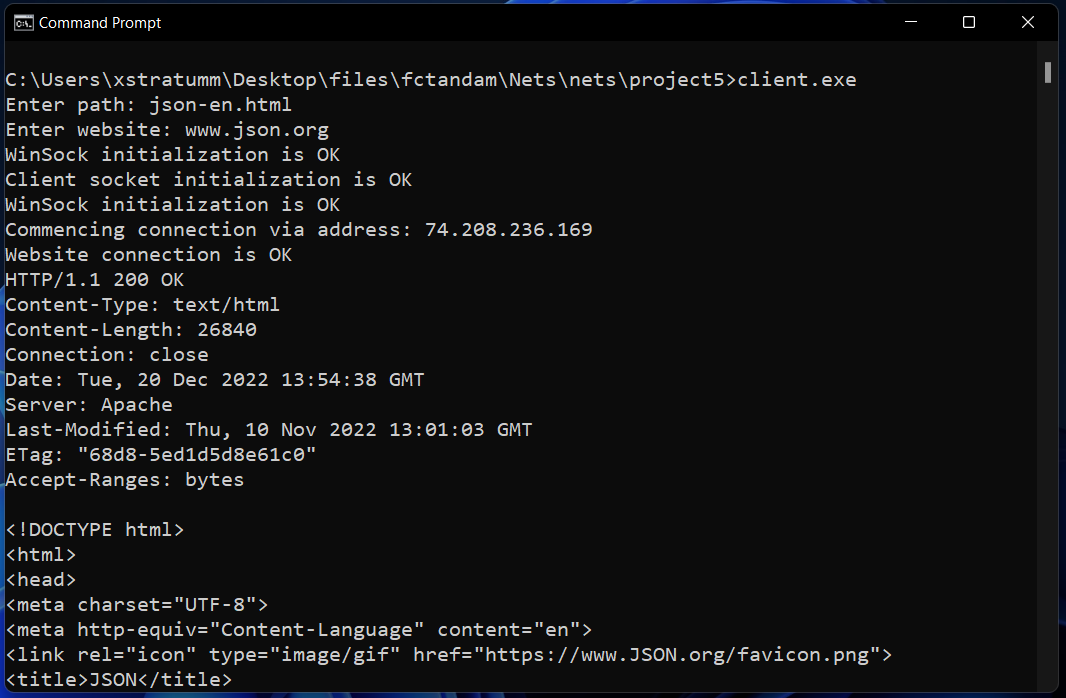
    return 0;

}

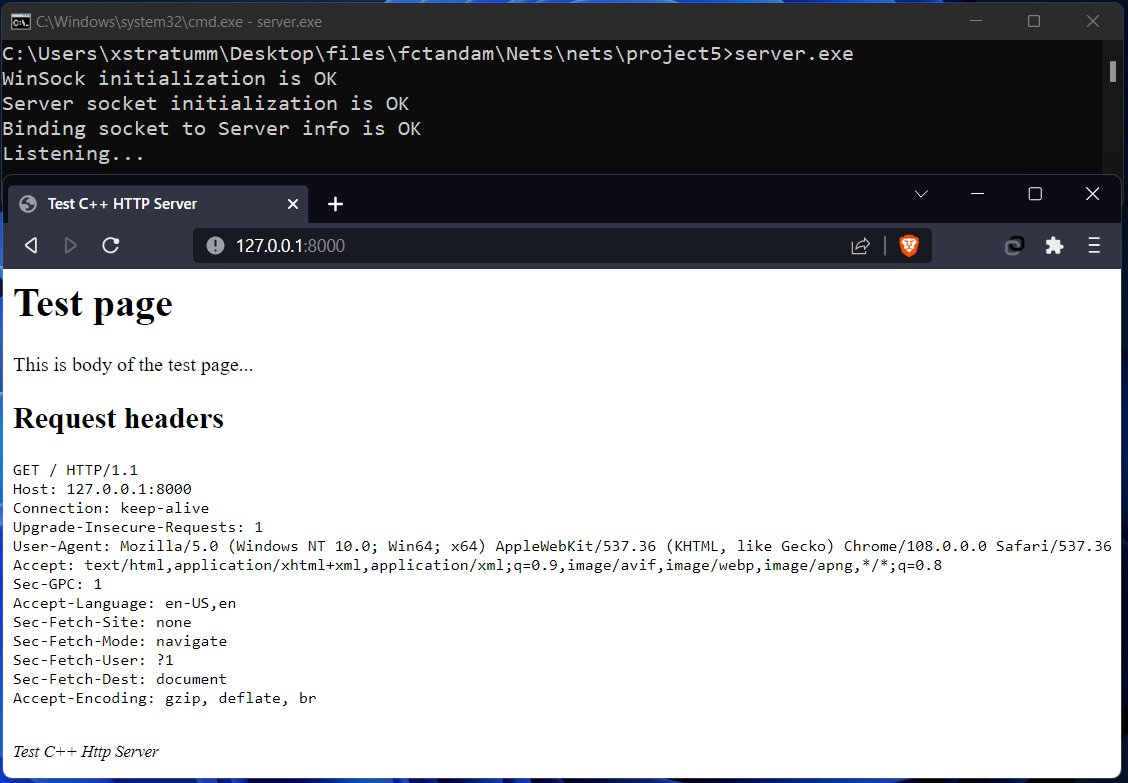
Пример работы программ:



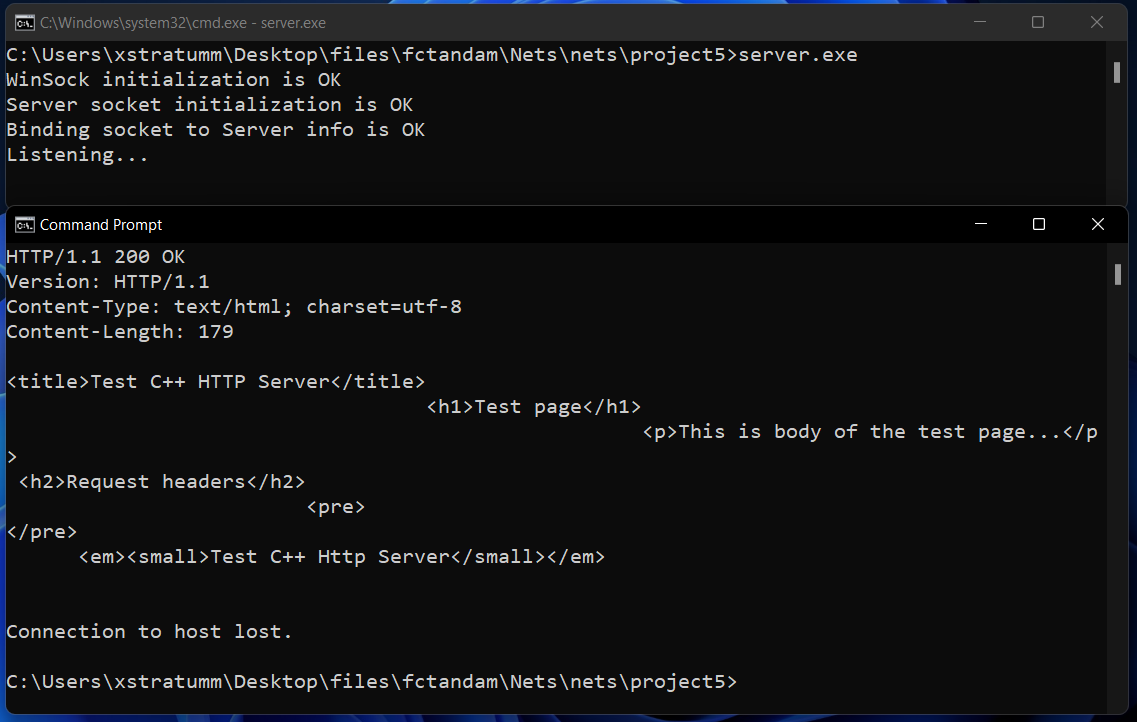
*Рисунок 1 – обращение клиента к серверу*

**

*Рисунок 2 – обращение клиента к некоторому веб-серверу*

**

*Рисунок 3 – обращение к серверу через веб-браузер*

**

*Рисунок 4 – обращение к серверу клиента telnet*

**Вывод:** были изучены основные положения протокола HTTP, было разработано сетевое клиент-серверное приложение на основе протокола HTTP.