МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

**ИНТЕЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ**

ОТЧЁТ

По лабораторной работе № 2

Выполнил:

Студент группы ИИ-22

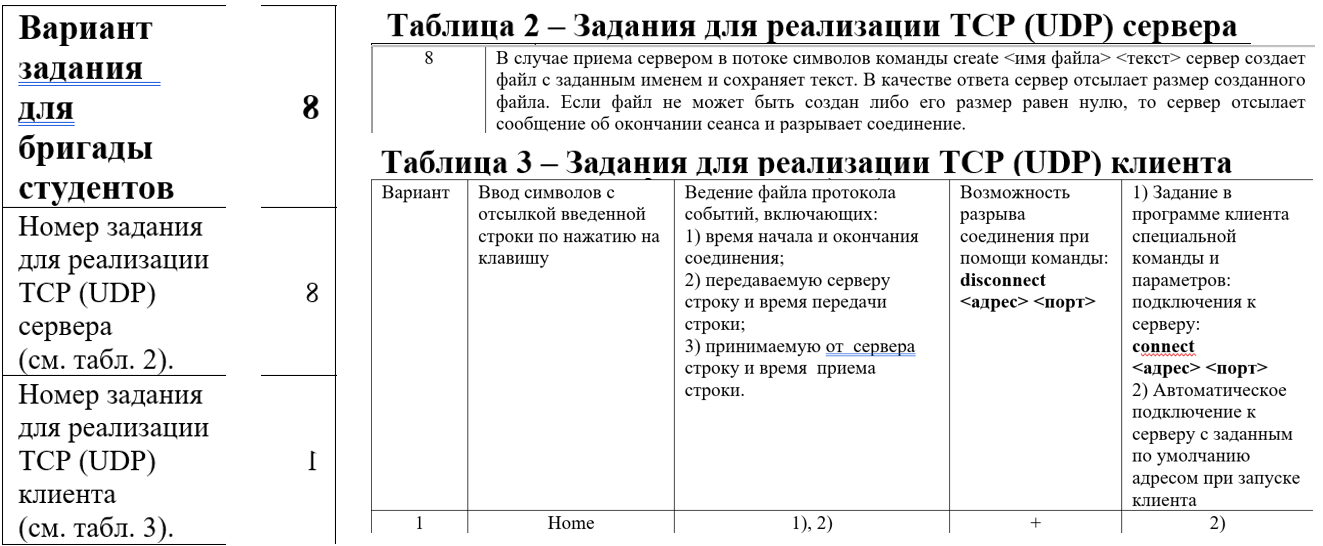
Копанчук Евгений Романович

Проверил:

Касьяник В.В.

Брест – 2023

**Ход работы**



*Код программы UDP client:*

#pragma comment(lib, "ws2\_32.lib")

#include <winsock2.h>

#include <iostream>

#include <fstream>

#include <chrono>

#include <string>

#include <thread>

#include <mutex>

#include <stdio.h>

#include <string.h>

#pragma warning(disable: 4996)

#define ServerPORT 3000

#define ServerIP "127.0.0.1"

#define ClientPORT 3001

#define ClientIP "127.0.0.1"

SOCKET ClientSocket;

SOCKADDR\_IN serverAddr;

int serverAddrSize;

std::mutex g\_lock;

std::string filename;

void ClientRecvHandler() {

char msg[256];

while (true) {

int bytesReceived = recvfrom(ClientSocket, msg, sizeof(msg), 0, (SOCKADDR\*)&serverAddr, &serverAddrSize);

if (bytesReceived != SOCKET\_ERROR) {

g\_lock.lock();

long long currentTimeMillis = std::chrono::duration\_cast<std::chrono::milliseconds>(

std::chrono::system\_clock::now().time\_since\_epoch()

).count();

std::ofstream file(filename, std::ios::app);

if (file.is\_open()) {

file << "Server: " << msg << " at: " << std::to\_string(currentTimeMillis) << std::endl;

file.close();

}

else { std::cerr << "Ошибка при создании или открытии файла." << std::endl; }

g\_lock.unlock();

if (strstr("code1", msg)) {

std::cerr << "Server interrupted connection." << std::endl;

exit(0);

}

else {

std::cout << "Server: " << msg << std::endl;

}

}

}

}

void ClientSendHandler() {

char msg[256];

while (true) {

bool HomeKeyState = !GetKeyState(VK\_HOME);

bool isMessageEntered = false;

bool isCommandDisconect = false;

char msg[256];

while (true) {

if (!isMessageEntered) {

std::cin.getline(msg, sizeof(msg));

std::cout << "Press HOME to sent it." << std::endl;

isMessageEntered = true;

HomeKeyState = !GetKeyState(VK\_HOME);

char\* ptr = strstr(msg, "disconnect");

if (ptr) {

isCommandDisconect = true;

}

}

if (isCommandDisconect) {

std::string command(msg);

std::string test = "disconnect " + std::to\_string(ServerPORT) + ' ' + ServerIP;

if (command == test) {

std::cout << "Connection interrupted.\n";

g\_lock.lock();

long long currentTimeMillis = std::chrono::duration\_cast<std::chrono::milliseconds>(

std::chrono::system\_clock::now().time\_since\_epoch()

).count();

std::ofstream file(filename, std::ios::app);

if (file.is\_open()) {

file << "Connection interrupted at: " << std::to\_string(currentTimeMillis) << std::endl;

file.close();

}

else { std::cerr << "Ошибка при создании или открытии файла." << std::endl; }

g\_lock.unlock();

exit(0);

}

isCommandDisconect = false;

isMessageEntered = false;

}

if (isMessageEntered && HomeKeyState == GetKeyState(VK\_HOME)) {

std::cout << "Massage was sended." << std::endl;

sendto(ClientSocket, msg, sizeof(msg), 0, (SOCKADDR\*)&serverAddr, sizeof(serverAddr));

g\_lock.lock();

long long currentTimeMillis = std::chrono::duration\_cast<std::chrono::milliseconds>(

std::chrono::system\_clock::now().time\_since\_epoch()

).count();

std::ofstream file(filename, std::ios::app);

if (file.is\_open()) {

file << "Client: " << msg << " at: " << std::to\_string(currentTimeMillis) << std::endl;

file.close();

}

else { std::cerr << "Ошибка при создании или открытии файла." << std::endl; }

g\_lock.unlock();

isMessageEntered = false;

}

}

}

}

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

WSAData wsaData;

WORD DLLVersion = MAKEWORD(2, 1);

if (WSAStartup(DLLVersion, &wsaData)) {

std::cout << "Error\n";

exit(1);

}

SOCKADDR\_IN clientAddr;

int sizeofClientAddr = sizeof(clientAddr);

clientAddr.sin\_addr.s\_addr = inet\_addr(ClientIP);

clientAddr.sin\_port = htons(ClientPORT);

clientAddr.sin\_family = AF\_INET;

ClientSocket = socket(AF\_INET, SOCK\_DGRAM, 0);

bind(ClientSocket, (SOCKADDR\*)&clientAddr, sizeofClientAddr);

serverAddrSize = sizeof(serverAddr);

serverAddr.sin\_addr.s\_addr = inet\_addr(ServerIP);

serverAddr.sin\_port = htons(ServerPORT);

serverAddr.sin\_family = AF\_INET;

long long currentTimeMillis = std::chrono::duration\_cast<std::chrono::milliseconds>(

std::chrono::system\_clock::now().time\_since\_epoch()

).count();

filename = "conection\_id" + std::to\_string(currentTimeMillis) + ".txt";

std::ofstream file(filename);

if (file.is\_open()) {

file << "Client connected at: " << std::to\_string(currentTimeMillis) << std::endl;

std::cout << "Client connected at: " << std::to\_string(currentTimeMillis) << std::endl;

file.close();

}

else { std::cerr << "Ошибка при создании или открытии файла." << std::endl; }

std::thread thr1(ClientRecvHandler);

std::thread thr2(ClientSendHandler);

thr1.join();

thr2.join();

closesocket(ClientSocket);

WSACleanup();

return 0;

}

*Код программы UDP server:*

#pragma comment(lib, "ws2\_32.lib")

#include <winsock2.h>

#include <iostream>

#include <fstream>

#include <string>

#include <algorithm>

#include <vector>

#include <sstream>

#include <thread>

#pragma warning(disable: 4996)

SOCKET ServerSocket;

SOCKADDR\_IN clientAddr;

int clientAddrSize;

#define ServerPORT 3000

#define ServerIP "127.0.0.1"

#define ClientPORT 3001

#define ClientIP "127.0.0.1"

void ServerRecvHandler() {

char msg[256];

while (true) {

sockaddr\_in clientAddr;

int clientAddrSize = sizeof(clientAddr);

int bytesReceived = recvfrom(ServerSocket, msg, sizeof(msg), 0, (SOCKADDR\*)&clientAddr, &clientAddrSize);

if (bytesReceived != SOCKET\_ERROR) {

char\* ptr = strstr(msg, "create");

if (ptr) {

std::cout << "Received: " << msg << std::endl;

std::istringstream iss(msg);

std::vector<std::string> tokens;

std::string token;

while (iss >> token) {

tokens.push\_back(token);

if (tokens.size() > 1) {

break;

}

}

std::string rest;

std::getline(iss, rest);

if (tokens.size() < 2 || tokens[1].find(".txt") == std::string::npos) {

const char\* err = "code1";

sendto(ServerSocket, err, sizeof(err), 0, (SOCKADDR\*)&clientAddr, sizeof(clientAddr));

}

else {

std::ofstream file(tokens[1]);

if (file.is\_open()) {

file << rest << std::endl;

file.close();

}

else {

std::cerr << "Ошибка при создании или открытии файла." << std::endl;

}

std::ifstream filebin(tokens[1], std::ios::binary);

filebin.seekg(0, std::ios::end);

std::streamsize fileSize = filebin.tellg();

filebin.close();

std::string fileSizeString = std::to\_string(static\_cast<long long>(fileSize)).c\_str();

sendto(ServerSocket, fileSizeString.c\_str(), sizeof(fileSizeString), 0, (SOCKADDR\*)&clientAddr, sizeof(clientAddr));

}

}

else {

std::cout << "Received from client: " << msg << std::endl;

}

}

}

}

void ServerSendHandler() {

bool HomeKeyState = !GetKeyState(VK\_HOME);

bool isMessageEntered = false;

char msg[256];

while (true) {

if (!isMessageEntered) {

std::cin.getline(msg, sizeof(msg));

std::cout << "Press HOME to sent it." << std::endl;

isMessageEntered = true;

HomeKeyState = !GetKeyState(VK\_HOME);

}

if (isMessageEntered && HomeKeyState == GetKeyState(VK\_HOME)) {

std::cout << "Massage was sended." << std::endl;

sendto(ServerSocket, msg, sizeof(msg), 0, (SOCKADDR\*)&clientAddr, sizeof(clientAddr));

isMessageEntered = false;

}

}

}

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

WSAData wsaData;

WORD DLLVersion = MAKEWORD(2, 1);

if (WSAStartup(DLLVersion, &wsaData)) {

std::cout << "Error\n";

exit(1);

}

SOCKADDR\_IN serverAddr;

int sizeofServerAddr = sizeof(serverAddr);

serverAddr.sin\_addr.s\_addr = inet\_addr(ServerIP);

serverAddr.sin\_port = htons(ServerPORT);

serverAddr.sin\_family = AF\_INET;

ServerSocket = socket(AF\_INET, SOCK\_DGRAM, 0);

bind(ServerSocket, (SOCKADDR\*)&serverAddr, sizeofServerAddr);

clientAddr.sin\_addr.s\_addr = inet\_addr(ClientIP);

clientAddr.sin\_port = htons(ClientPORT);

clientAddr.sin\_family = AF\_INET;

int clientAddrSize = sizeof(clientAddr);

std::cout << "UDP Server is running...\n";

std::thread thr1(ServerRecvHandler);

std::thread thr2(ServerSendHandler);

thr1.join();

thr2.join();

closesocket(ServerSocket);

WSACleanup();

return 0;

}

