Министерство образования Республики Беларусь

Учреждение образования

Белорусский государственный университет информатики

и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Операционные среды и системное программирование

**ОТЧЕТ**

к лабораторной работе №7

на тему

«Интерфейс сокетов и основы сетевого программирования (Windows). Программирование взаимодействия через сеть с использованием интерфейса сокетов. Реализация сетевых протоколов: собственных или стандартных.»

Выполнил:

студент гр. 153504

Сивый А.А.

Проверил:

Гриценко Н.Ю.

Минск 2023

**СОДЕРЖАНИЕ**

[1 Цели работы 3](#_gjdgxs)

[2 Краткие теоретические сведения 4](#_30j0zll)

[3 Полученные результаты 5](#_1fob9te)

[Вывод 6](#_4q3p755gzys6)

[Приложение А (обязательное) Листинг кода 7](#_4q3p755gzys6)

**1 ЦЕЛИ РАБОТЫ**

1 Изучить основные принципы работы сокетов в операционной системе Windows.

2 Ознакомиться с принципами сетевого программирования и взаимодействия через сеть.

3 Изучить принципы передачи данных через сеть с помощью различных сетевых протоколов.

4 Реализовать приложение для передачи файлов между клиентом и сервером с использованием сокетов TCP протокола.

**2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

TCP/IP – это семейство протоколов, которые используются для связи в сетях. Протокол TCP (Transmission Control Protocol) обеспечивает надежную передачу данных между узлами в сети. Он управляет установлением соединения, обменом данными и разрывом соединения между отправителем и получателем. Протокол IP (Internet Protocol) отвечает за маршрутизацию и доставку данных в сети.

В операционной системе Windows для работы с сетью и реализации сетевого программирования используется интерфейс сокетов, который предоставляет программисту возможность создания сетевых приложений. Сокеты – это программный интерфейс, который позволяет приложениям взаимодействовать с сетью. С их помощью приложения могут отправлять и принимать данные через сеть, а также устанавливать соединения с удаленными узлами.

Существует два основных типа сокетов: TCP и UDP. TCP-сокеты обеспечивают надежную передачу данных, гарантируя доставку и порядок пакетов, в то время как UDP-сокеты предоставляют более быструю, но менее надежную передачу данных.

Для работы с TCP в операционной системе Windows используется библиотека Winsock (Windows Sockets), которая предоставляет набор функций для создания сетевых приложений. Некоторые из основных функций Winsock для работы с TCP включают:

1 socket()– создание нового сокета для обмена данными.

2 connect() – установление соединения с удаленным сервером.

3 send() – отправка данных через установленное соединение.

4 recv() – прием данных от удаленного сервера.

5 close() – закрытие соединения.

Эти функции позволяют программисту создавать клиент-серверные приложения, обмениваться данными через сеть и обрабатывать сетевые события.

Клиент-серверное приложение в – это приложение, которое состоит из двух основных компонентов: клиента и сервера. Клиент отправляет запросы серверу, а сервер отвечает на эти запросы, обрабатывая их и отправляя обратно клиенту необходимую информацию.

В таком приложении клиент и сервер обмениваются данными через сеть с использованием протокола TCP. Клиент инициирует соединение с сервером, отправляет запросы и получает ответы, а сервер принимает запросы, обрабатывает их и отправляет ответы.

Такие приложения широко используются для организации распределенных систем, где клиенты обращаются к серверу для получения различных услуг или данных.

**3 ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ**

При запуске программы на экране появляется приглашение для пользователя. Пользователю необходимо ввести путь к рабочей директории. Если пользователь введет некорректный путь, программа предложит ввести его заново. Начало программы продемонстрировано на рисунке 3.1.

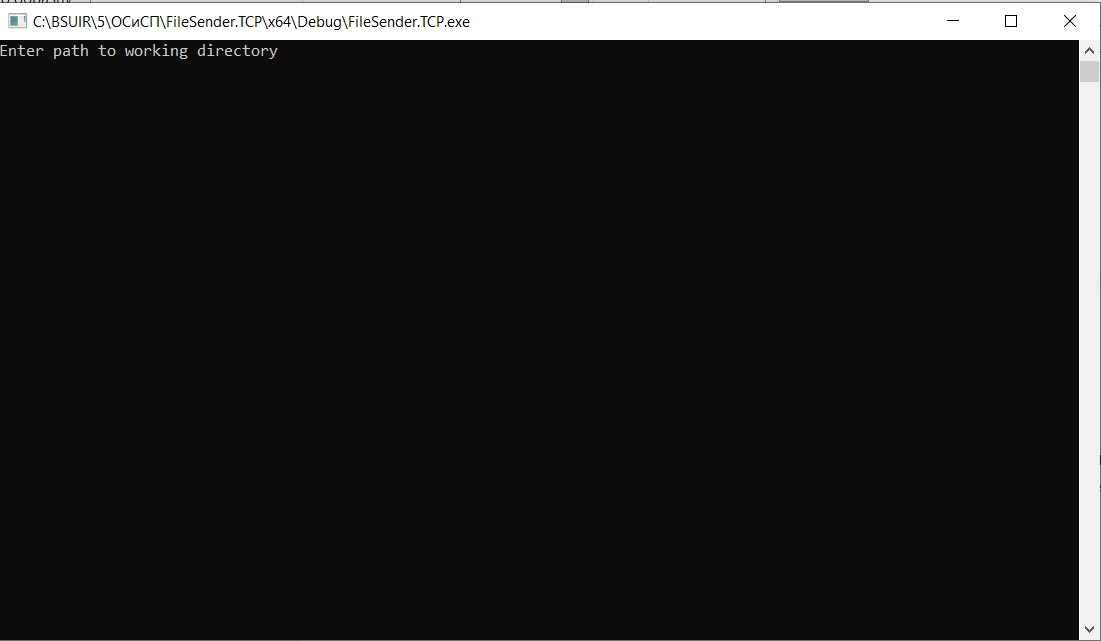


Рисунок 3.1 – Начало работы программы.

После ввода пути к рабочей директории, программа позволяет пользователю выбрать действие, которое он хочет выполнить. Пользователь может получать файл, отправлять файл а также получать список файлов, находящихся на данный момент в рабочей директории. Это можно наблюдать на рисунке 3.2.

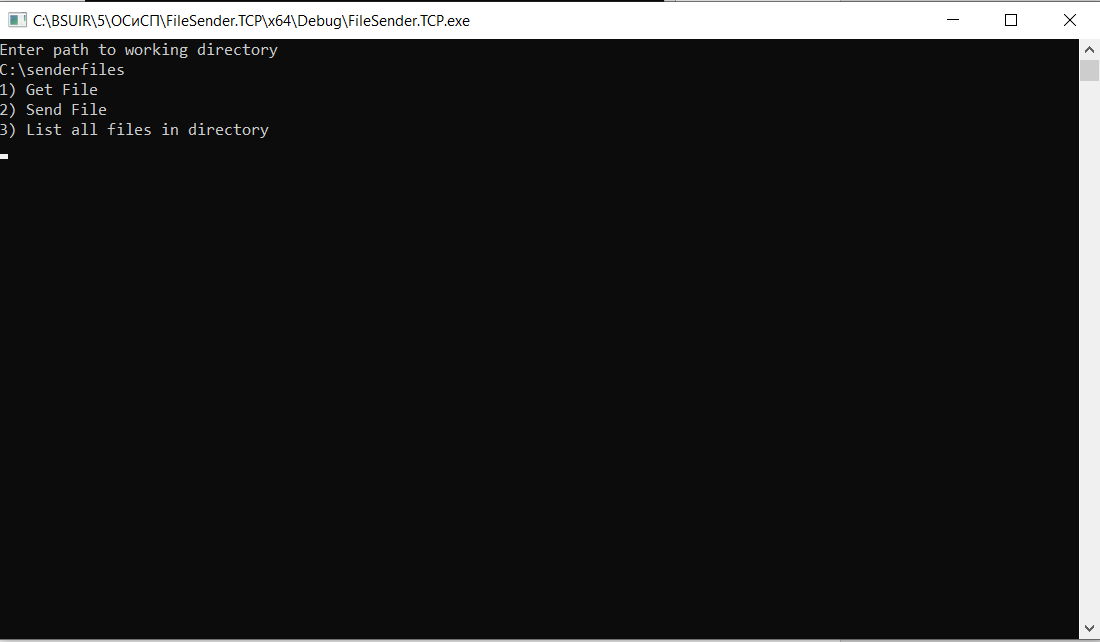


Рисунок 3.2 – Главное меню приложения.

Для получения файла пользователю необходимо нажать на кнопку «Get File», после чего программа будет ожидать подключения отправителя файла. Н а рисунке 3.3 можно наблюдать программу, которая ожидает файл.

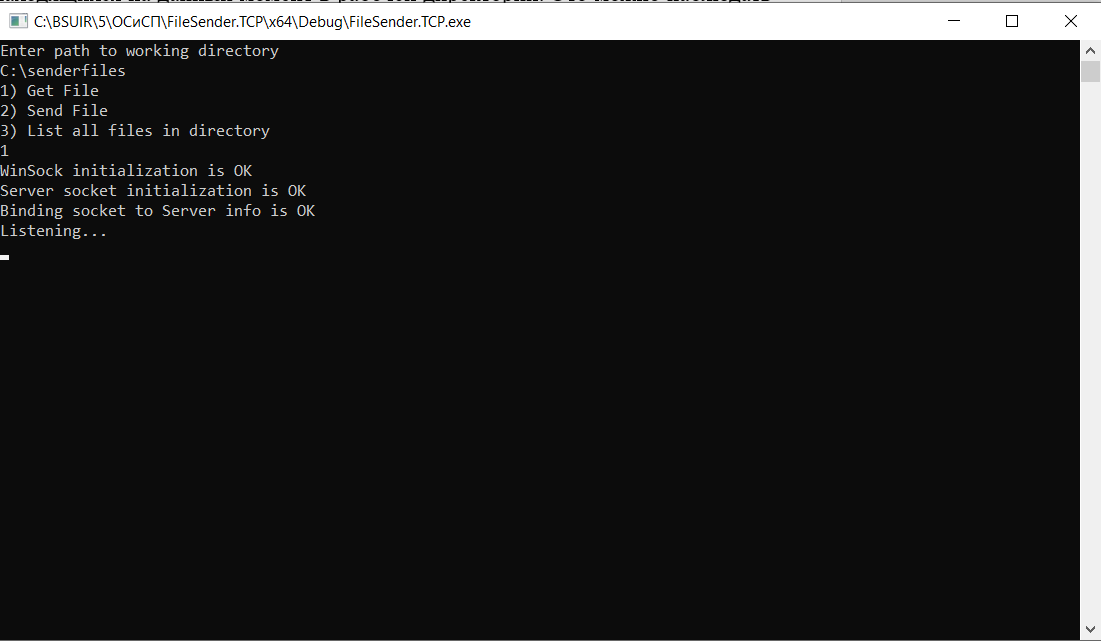


Рисунок 3.3 – Ожидание подключения пользователя.

После подключения, пользователю, который отправляет файл, будет предложено ввести имя файла. На рисунке 3.4 представлено состояние программы у обоих пользователей.

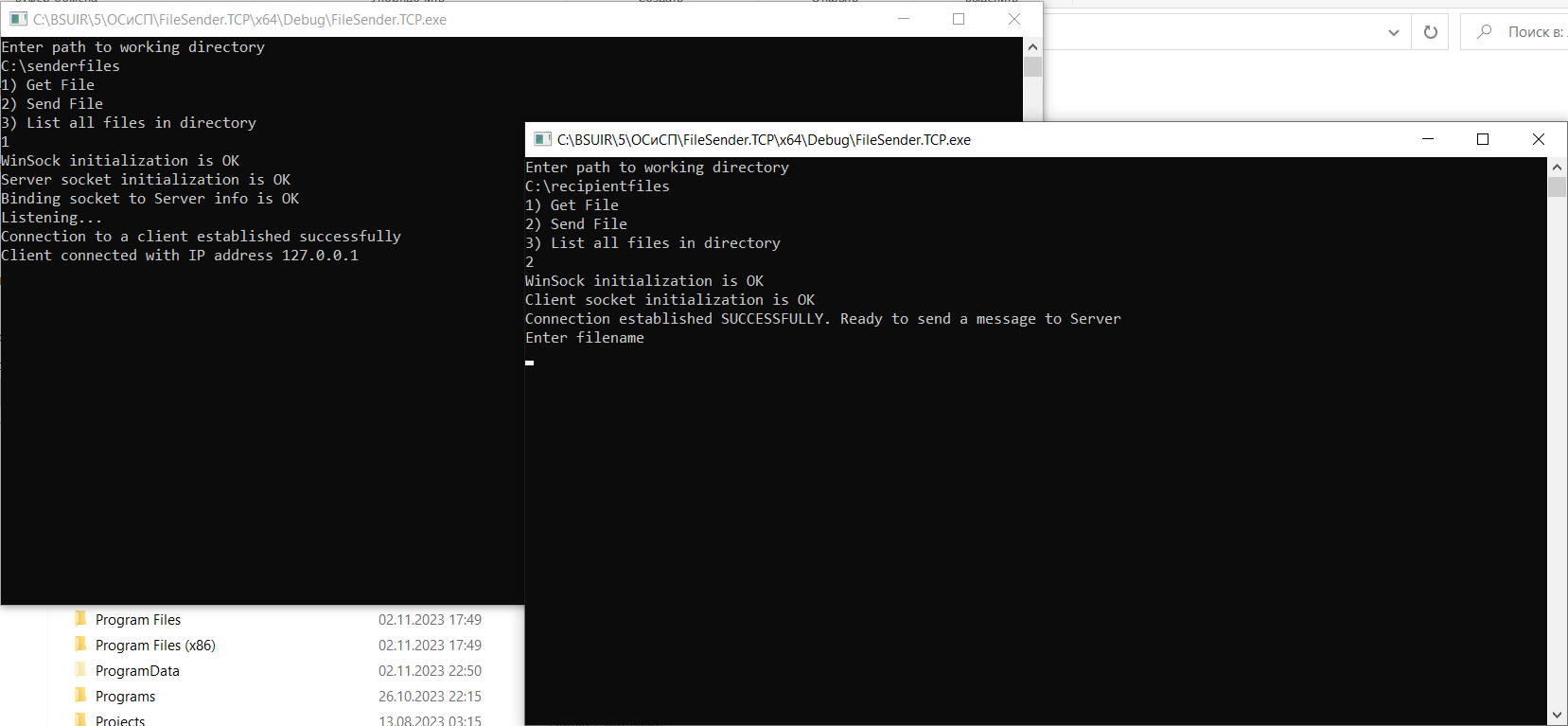


Рисунок 3.4 – Состояние программы после подключения.

После ввода имени файла, он отсылается и в результате можно видеть уведомление об успешном отправлении файла.

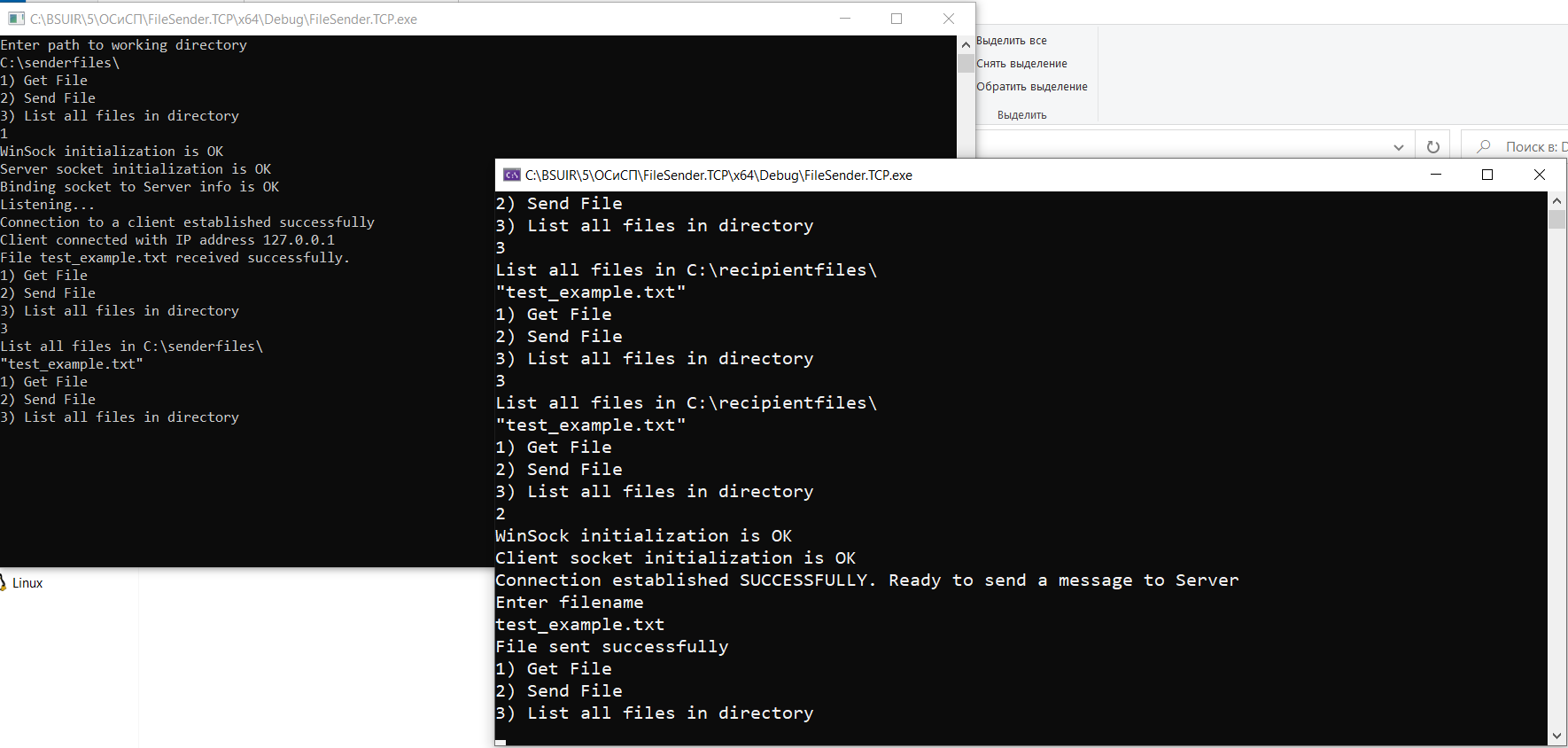


Рисунок 3.5 – Уведомление об успешной передаче файла.

В результате файл успешно отправлен, что продемонстрировано на рисунке 3.5.

**ВЫВОД**

В ходе лабораторной работы были изучены основы сетевого программирования, включая работу с сокетами TCP. Было реализовано приложение для передачи файлов между клиентом и сервером с использованием сокетов TCP.

В процессе работы были изучены основные принципы работы сокетов, установления соединения, отправки и приема данных. Также были рассмотрены основные аспекты безопасности при работе с сокетами.

**ПРИЛОЖЕНИЕ А**

**(обязательное)**

**Листинг кода**

#include <iostream>

#include <string>

#include <fstream>

#include <WinSock2.h>

#include <WS2tcpip.h>

#include <filesystem>

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

using namespace std;

char ip[10] = "127.0.0.1";

int port = 8888;

const short BUFFER\_SIZE = 1024;

std::string recipient\_folder = "C:\\recipientfiles\\";

std::string sender\_folder = "C:\\senderfiles\\";

std::string working\_directory;

int send\_file() {

// Key variables for all program

int erStat;

in\_addr ip\_to\_num;

inet\_pton(AF\_INET, ip, &ip\_to\_num);

WSADATA wsData;

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

if (erStat != 0) {

cout << "Error WinSock version initializaion #";

cout << WSAGetLastError();

return 1;

}

else

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

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

if (сlientSocket == INVALID\_SOCKET) {

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

closesocket(сlientSocket);

WSACleanup();

}

else

cout << "Client socket initialization is OK" << endl;

sockaddr\_in servInfo;

ZeroMemory(&servInfo, sizeof(servInfo));

servInfo.sin\_family = AF\_INET;

servInfo.sin\_addr = ip\_to\_num;

servInfo.sin\_port = htons(port);

erStat = connect(сlientSocket, (sockaddr\*)&servInfo, sizeof(servInfo));

if (erStat != 0) {

cout << "Connection to Server is FAILED. Error # " << WSAGetLastError() << endl;

closesocket(сlientSocket);

WSACleanup();

return 1;

}

else

cout << "Connection established SUCCESSFULLY. Ready to send a message to Server" << endl;

std::string filename;

std::string path;

do {

cout << "Enter filename" << endl;

std::cin >> filename;

path = sender\_folder + filename;

} while (!filesystem::exists(path) && !filesystem::is\_regular\_file(path));

send(сlientSocket, filename.c\_str(), filename.size(), 0);

std::ifstream inputFile(path, std::ios::binary);

if (!inputFile.is\_open()) {

std::cerr << "Error opening file for reading." << std::endl;

closesocket(сlientSocket);

WSACleanup();

return 1;

}

char buffer[BUFFER\_SIZE];

int bytesRead;

do {

inputFile.read(buffer, BUFFER\_SIZE);

bytesRead = inputFile.gcount();

if (bytesRead > 0) {

send(сlientSocket, buffer, bytesRead, 0);

}

} while (inputFile.good());

cout << "File sent successfully" << endl;

closesocket(сlientSocket);

WSACleanup();

return 0;

}

int get\_file() {

int erStat; // Keeps socket errors status

//IP in string format to numeric format for socket functions. Data is in "ip\_to\_num"

in\_addr ip\_to\_num;

erStat = inet\_pton(AF\_INET, ip, &ip\_to\_num);

if (erStat <= 0) {

cout << "Error in IP translation to special numeric format" << endl;

return 1;

}

// WinSock initialization

WSADATA wsData;

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

if (erStat != 0) {

cout << "Error WinSock version initializaion #";

cout << WSAGetLastError();

return 1;

}

else

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

// Server socket initialization

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

if (serverSocket == INVALID\_SOCKET) {

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

closesocket(serverSocket);

WSACleanup();

return 1;

}

else

cout << "Server socket initialization is OK" << endl;

// Server socket binding

sockaddr\_in servInfo;

ZeroMemory(&servInfo, sizeof(servInfo)); // Initializing servInfo structure

servInfo.sin\_family = AF\_INET;

servInfo.sin\_addr = ip\_to\_num;

servInfo.sin\_port = htons(port);

erStat = bind(serverSocket, (sockaddr\*)&servInfo, sizeof(servInfo));

if (erStat != 0) {

cout << "Error Socket binding to server info. Error # " << WSAGetLastError() << endl;

closesocket(serverSocket);

WSACleanup();

return 1;

}

else

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

//Starting to listen to any Clients

erStat = listen(serverSocket, SOMAXCONN);

if (erStat != 0) {

cout << "Can't start to listen to. Error # " << WSAGetLastError() << endl;

closesocket(serverSocket);

WSACleanup();

return 1;

}

else {

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

}

sockaddr\_in clientInfo;

ZeroMemory(&clientInfo, sizeof(clientInfo)); // Initializing clientInfo structure

int clientInfo\_size = sizeof(clientInfo);

SOCKET clientSocket = accept(serverSocket, (sockaddr\*)&clientInfo, &clientInfo\_size);

if (clientSocket == INVALID\_SOCKET) {

cout << "Client detected, but can't connect to a client. Error # " << WSAGetLastError() << endl;

closesocket(serverSocket);

closesocket(clientSocket);

WSACleanup();

return 1;

}

else {

cout << "Connection to a client established successfully" << endl;

char clientIP[22];

inet\_ntop(AF\_INET, &clientInfo.sin\_addr, clientIP, INET\_ADDRSTRLEN); // Convert connected client's IP to standard string format

cout << "Client connected with IP address " << clientIP << endl;

}

char filenameBuffer[BUFFER\_SIZE];

int filenameBytesReceived = recv(clientSocket, filenameBuffer, 1024, 0);

if (filenameBytesReceived <= 0) {

std::cerr << "Error receiving file name." << std::endl;

closesocket(clientSocket);

closesocket(serverSocket);

WSACleanup();

return 1;

}

std::string filename(filenameBuffer, filenameBytesReceived);

std::string path = recipient\_folder + filename;

char buffer[BUFFER\_SIZE];

int bytesReceived;

std::ofstream outputFile(path, std::ios::binary);

if (!outputFile.is\_open()) {

std::cerr << "Error opening file for writing." << std::endl;

closesocket(clientSocket);

closesocket(serverSocket);

WSACleanup();

return 1;

}

do {

bytesReceived = recv(clientSocket, buffer, BUFFER\_SIZE, 0);

if (bytesReceived > 0) {

outputFile.write(buffer, bytesReceived);

}

} while (bytesReceived > 0);

outputFile.close();

closesocket(clientSocket);

closesocket(serverSocket);

WSACleanup();

std::cout << "File " << filename << " received successfully." << std::endl;

return 0;

}

void list\_files() {

cout << "List all files in " << working\_directory << endl;

for (const auto& entry : std::filesystem::directory\_iterator(working\_directory)) {

if (filesystem::is\_regular\_file(entry.path())) {

cout << entry.path().filename() << endl;

}

}

}

int main() {

setlocale(LC\_ALL, "ru");

do {

cout << "Enter path to working directory" << endl;

cin >> working\_directory;

} while (!filesystem::exists(working\_directory) && !filesystem::is\_directory(working\_directory));

while (true) {

int choice;

cout << "1) Get File" << endl;

cout << "2) Send File" << endl;

cout << "3) List all files in directory" << endl;

cin >> choice;

switch (choice) {

case 1:

get\_file();

break;

case 2:

send\_file();

break;

case 3:

list\_files();

break;

}

}

}