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

ОТЧЁТ

к лабораторной работе №7

на тему

СРЕДСТВА ОБМЕНА ДАННЫМИ (*WINDOWS*). ИЗУЧЕНИЕ И ИСПОЛЬЗОВАНИЕМ СРЕДСТВ ОБМЕНА ДАННЫМИ И СОВМЕСТНОГО ДОСТУПА.

Выполнил студент гр.153502 Богомолов М.А.

Проверил ассистент кафедры информатики Гриценко Н.Ю.

Минск 2023

**СОДЕРЖАНИЕ**

[1 Формулировка задачи 3](#_Toc148046360)

[2 Теоретические сведения 4](#_Toc148046361)

[3 Описание функций программы 5](#_Toc148046362)

[Список использованных источников 7](#_Toc148046363)

[Приложение А (обязательное) Листинг кода 8](#_Toc148046364)

# **1 ФОРМУЛИРОВКА ЗАДАЧИ**

Целью выполнения лабораторной работы является построение системы передачи файлов между клиентами через сеть с возможностью выбора и отправки файлов.

# **2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Сетевой порт – то число, которое записывается в заголовках протоколов транспортного уровня сетевой модели *OSI*. Оно используется для определения программы или процесса-получателя пакета в пределах одного IP-адреса. Порты позволяют определить сетевые приложения, работающие на компьютере, множество которых может быть запущено одновременно [1].

*Winsock* – это сокращение от *Windows Socket*, что в переводе означает “сокеты *Windows*”.

*Winsock* представляет собой интерфейс программирования приложений (*API*), который позволяет разработчикам создавать сетевые приложения в операционной системе *Windows*. Он обеспечивает возможность взаимодействия сетевых программ с сетевым стеком операционной системы, что позволяет приложениям отправлять и получать данные через сеть.

В основе работы *Winsock* лежит понятие сокета, который является основным строительным блоком сетевого взаимодействия. Сокет представляет собой комбинацию *IP*-адреса и порта, которые идентифицируют конкретное сетевое соединение. С помощью функций *Winsock* приложения могут создавать сокеты, устанавливать связи с другими узлами в сети и передавать данные по этим соединениям.

Основные функции *Winsock* включают в себя создание сокета, установку соединения, передачу данных, прослушивание порта, обработку событий сокета и разрыв соединения. Эти функции позволяют разработчикам создавать сетевые приложения различной сложности, включая веб-сервера, чаты, клиенты *FTP* и другие.

Работа с *Winsock* обычно включает несколько этапов, таких как создание сокета, установка параметров соединения, обмен данными и разрыв соединения. Для удобства программирования существуют различные библиотеки и фреймворки, которые предоставляют обертки над функциями *Winsock* и упрощают процесс разработки сетевых приложений [2].

# **3 ОПИСАНИЕ ФУНКЦИЙ ПРОГРАММЫ**

Для полноценного использования программы необходимо запустить серверную составляющую программы и программу-клиент. При запуске клиент автоматически совершает подключение к серверу и пользователю предлагается выбрать директорию сохранения получаемых от других клиентов файлов. Для вызова окна выбора директории необходимо нажать кнопку “Выбрать директорию”, после чего откроется окно выбора директории, в которой пользователь может создать папку или выбрать существующую (Рисунок 1).

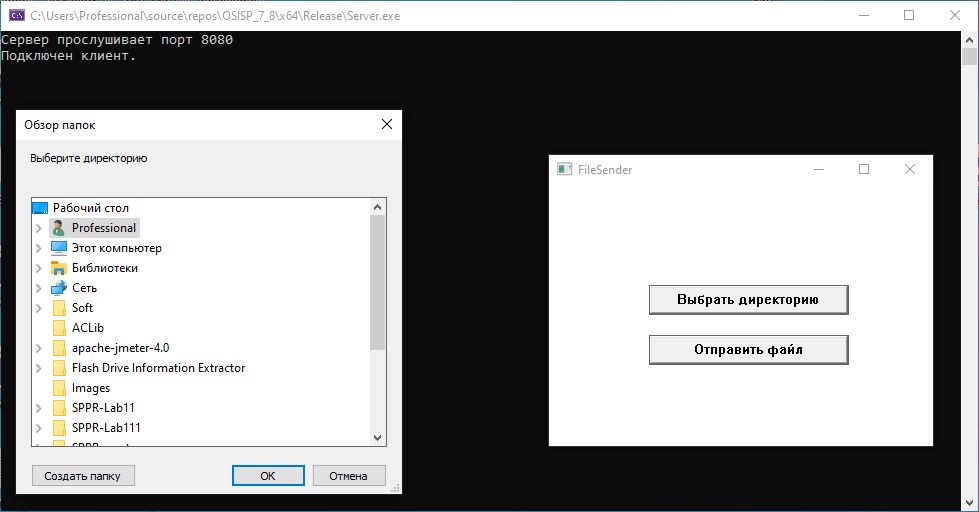


Рисунок 1 – Результат нажатия кнопки выбора директории

При нажатии на кнопку “Отправить файл” для пользователя открывается окно выбора файлов, в котором он может произвести выбор файла в любой существующей директории (Рисунок 2).

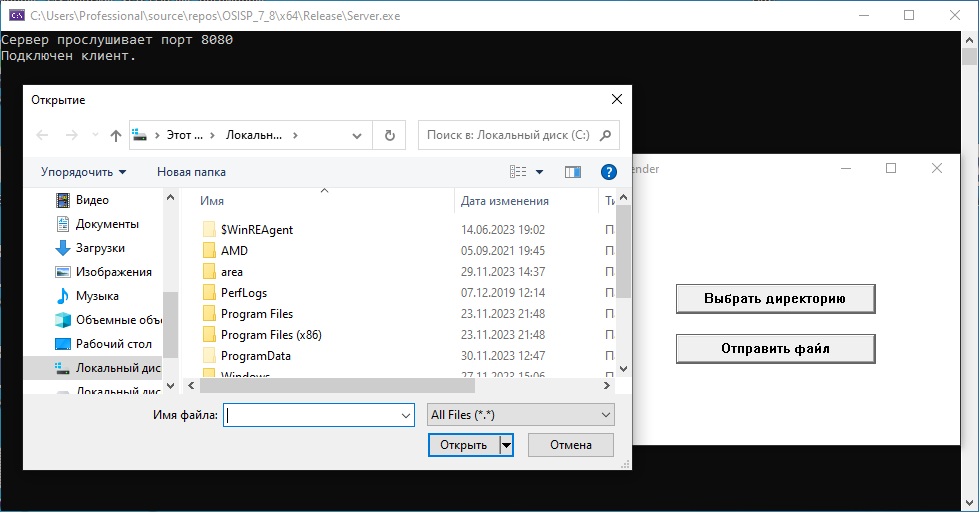


Рисунок 2 – Результат нажатия кнопки отправки файла

После выбора файла и нажатия кнопки “Открыть” в окне “Открытие” файл передаётся на сервер и пользователю выводится сообщения о статусе отправки выбранного файла (Рисунок 3).

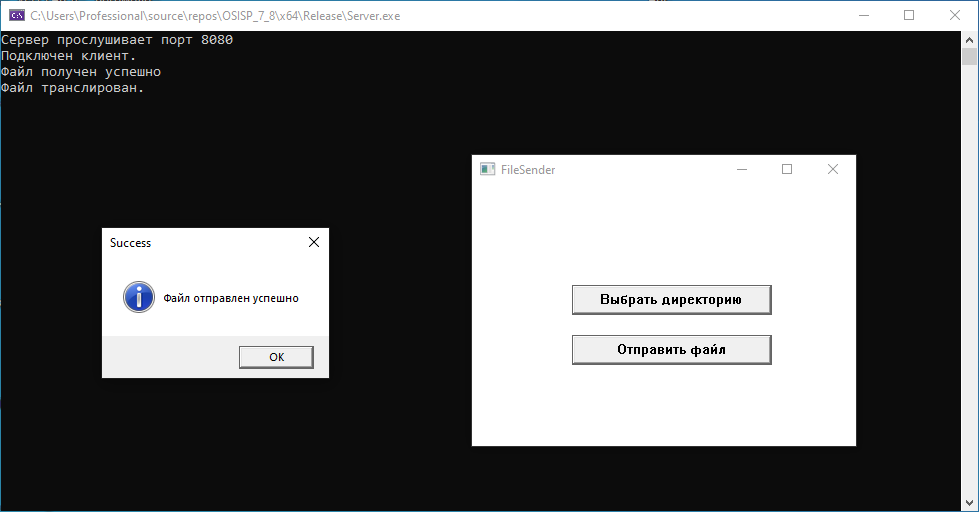


Рисунок 3 – Вывод сообщение об отправке файла

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Что такое порты, какие бывают и зачем они нужны? [Электронный ресурс]. – Режим доступа: <https://vc.ru/u/1193668-fenixhost/441676-chto-takoe-porty-kakie-byvayut-i-zachem-oni-nuzhny> – Дата доступа: 06.12.2023.

[2] Что такое Winsock? Определение и особенности работы [Электронный ресурс]. – Режим доступа: <https://uchet-jkh.ru/i/cto-takoe-winsock-opredelenie-i-osobennosti-raboty/> – Дата доступа: 06.12.2023.

# **ПРИЛОЖЕНИЕ А (обязательное) Листинг кода**

Листинг 1 – Файл OSISP\_7\_8.cpp

#include <winsock2.h>

#include <windows.h>

#include <ws2tcpip.h>

#include <iostream>

#include <commdlg.h>

#include <fstream>

#include <shlobj.h>

#include <thread>

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

HINSTANCE hInst;

HWND hwndButtonSelectDir, hwndButtonSendFile, hwnd;

WCHAR saveFolderPath[MAX\_PATH] = L"C:\\Users\\Professional\\Desktop";

SOCKET clientSocket;

LRESULT CALLBACK WndProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam);

void ConnectServer();

void ReceiveFile();

void OpenFile(HWND hwnd);

void SelectDirectory(HWND hwnd);

void SocketListener();

void SendFile(const wchar\_t\* filePath);

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) {

hInst = hInstance;

WNDCLASS wc = {};

wc.lpfnWndProc = WndProc;

wc.hInstance = hInstance;

wc.lpszClassName = L"MyWindowClass";

if (!RegisterClass(&wc)) {

MessageBox(NULL, L"Ошибка регистрации класса окна.", L"Error", MB\_ICONEXCLAMATION | MB\_OK);

return 0;

}

hwnd = CreateWindow(

L"MyWindowClass", L"FileSender", WS\_OVERLAPPEDWINDOW | WS\_VISIBLE,

CW\_USEDEFAULT, CW\_USEDEFAULT, 400, 300,

NULL, NULL, hInstance, NULL

);

if (hwnd == NULL) {

MessageBox(NULL, L"Ошибка создания окна.", L"Error", MB\_ICONEXCLAMATION | MB\_OK);

return 0;

}

std::thread socketListenerThread(SocketListener);

socketListenerThread.detach();

MSG msg = {};

while (GetMessage(&msg, NULL, 0, 0)) {

TranslateMessage(&msg);

DispatchMessage(&msg);

}

closesocket(clientSocket);

WSACleanup();

return msg.wParam;

}

LRESULT CALLBACK WndProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam) {

switch (msg) {

case WM\_CREATE:

// Создание кнопок

hwndButtonSelectDir = CreateWindow(

L"BUTTON", L"Выбрать директорию", WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

100, 100, 200, 30, hwnd, (HMENU)2, hInst, NULL

);

hwndButtonSendFile = CreateWindow(

L"BUTTON", L"Отправить файл", WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

100, 150, 200, 30, hwnd, (HMENU)3, hInst, NULL

);

ConnectServer();

break;

case WM\_COMMAND:

switch (LOWORD(wParam)) {

case 2:

SelectDirectory(hwnd);

break;

case 3:

OpenFile(hwnd);

break;

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hwnd, msg, wParam, lParam);

}

return 0;

}

void SelectDirectory(HWND hwnd) {

BROWSEINFO browseInfo = { 0 };

browseInfo.hwndOwner = hwnd;

browseInfo.pidlRoot = NULL;

browseInfo.pszDisplayName = NULL;

browseInfo.lpszTitle = L"Выберите директорию";

browseInfo.ulFlags = BIF\_RETURNONLYFSDIRS | BIF\_NEWDIALOGSTYLE;

LPITEMIDLIST pidl = SHBrowseForFolder(&browseInfo);

if (pidl != NULL) {

SHGetPathFromIDList(pidl, saveFolderPath);

CoTaskMemFree(pidl);

}

}

void ConnectServer() {

WSADATA wsaData;

if (WSAStartup(MAKEWORD(2, 2), &wsaData) == SOCKET\_ERROR) {

MessageBox(hwnd, L"Ошибка инициализации WinSock", L"Error", MB\_OK | MB\_ICONERROR);

return;

}

clientSocket = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

if (clientSocket == INVALID\_SOCKET) {

MessageBox(hwnd, L"Ошибка создания сокета", L"Error", MB\_OK | MB\_ICONERROR);

WSACleanup();

return;

}

sockaddr\_in serverAddr;

serverAddr.sin\_family = AF\_INET;

if (inet\_pton(AF\_INET, "127.0.0.1", &serverAddr.sin\_addr) != 1) {

MessageBox(hwnd, L"Некорректный IP-адрес сервера", L"Error", MB\_OK | MB\_ICONERROR);

WSACleanup();

return;

}

serverAddr.sin\_port = htons(8080);

if (connect(clientSocket, (sockaddr\*)&serverAddr, sizeof(serverAddr)) == SOCKET\_ERROR) {

MessageBox(hwnd, L"Ошибка соединения с сервером", L"Error", MB\_OK | MB\_ICONERROR);

closesocket(clientSocket);

WSACleanup();

return;

}

MessageBox(hwnd, L"Соеденинение с сервером успешно установлено", L"Уведомление", MB\_OK);

}

void OpenFile(HWND hwnd) {

OPENFILENAME ofn;

wchar\_t szFile[260];

ZeroMemory(&ofn, sizeof(ofn));

ofn.lStructSize = sizeof(ofn);

ofn.hwndOwner = hwnd;

ofn.lpstrFile = szFile;

ofn.lpstrFile[0] = '\0';

ofn.nMaxFile = sizeof(szFile) / sizeof(szFile[0]);

ofn.lpstrFilter = L"All Files (\*.\*)\0\*.\*\0";

ofn.nFilterIndex = 1;

ofn.Flags = OFN\_PATHMUSTEXIST | OFN\_FILEMUSTEXIST;

if (GetOpenFileName(&ofn) == TRUE) {

SendFile(szFile);

}

}

void SendFile(const wchar\_t\* filePath) {

std::ifstream file(filePath, std::ios::binary);

if (!file) {

MessageBox(hwnd, L"Ошибка открытия файла для отправки", L"Error", MB\_OK | MB\_ICONERROR);

return;

}

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

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

file.seekg(0, std::ios::beg);

if (send(clientSocket, reinterpret\_cast<const char\*>(&fileSize), sizeof(fileSize), 0) <= 0) {

MessageBox(hwnd, L"Ошибка отправки размера файла", L"Error", MB\_OK | MB\_ICONERROR);

file.close();

return;

}

const wchar\_t\* fileName = wcsrchr(filePath, L'\\');

if (!fileName) {

fileName = filePath;

}

else {

fileName++;

}

if (send(clientSocket, reinterpret\_cast<const char\*>(fileName), sizeof(wchar\_t) \* MAX\_PATH, 0) <= 0) {

MessageBox(hwnd, L"Ошибка отправки имени файла", L"Error", MB\_OK | MB\_ICONERROR);

file.close();

return;

}

char buffer[1024];

while (!file.eof()) {

file.read(buffer, sizeof(buffer));

if (send(clientSocket, buffer, file.gcount(), 0) <= 0) {

MessageBox(hwnd, L"Ошибка отправки файла", L"Error", MB\_OK | MB\_ICONERROR);

file.close();

return;

}

}

file.close();

MessageBox(hwnd, L"Файл отправлен успешно", L"Success", MB\_OK | MB\_ICONINFORMATION);

}

void SocketListener() {

while (true) {

char buffer[1024];

int bytesRead = recv(clientSocket, buffer, sizeof(buffer), 0);

if (bytesRead > 0) {

buffer[bytesRead] = '\0';

if (strcmp(buffer, "REQUEST\_FILE\_TRANSFER") == 0) {

ReceiveFile();

}

}

}

}

void ReceiveFile() {

std::streamsize fileSize;

int bytesRead = recv(clientSocket, reinterpret\_cast<char\*>(&fileSize), sizeof(fileSize), 0);

if (bytesRead <= 0) {

MessageBox(hwnd, L"Ошибка получения размера файла", L"Error", MB\_OK | MB\_ICONERROR);

return;

}

wchar\_t filePath[MAX\_PATH];

bytesRead = recv(clientSocket, reinterpret\_cast<char\*>(filePath), sizeof(filePath), 0);

if (bytesRead <= 0) {

MessageBox(hwnd, L"Ошибка получения имени файла", L"Error", MB\_OK | MB\_ICONERROR);

return;

}

std::wstring fullFilePath = saveFolderPath;

fullFilePath += L"\\";

fullFilePath += filePath;

std::ofstream receivedFile(fullFilePath, std::ios::binary);

char buffer[1024];

while (fileSize > 0) {

bytesRead = recv(clientSocket, buffer, sizeof(buffer), 0);

if (bytesRead <= 0) {

MessageBox(hwnd, L"Ошибка получения содержимого файла", L"Error", MB\_OK | MB\_ICONERROR);

return;

}

receivedFile.write(buffer, bytesRead);

fileSize -= bytesRead;

}

receivedFile.close();

MessageBox(hwnd, L"Файл получен успешно", L"Success", MB\_OK | MB\_ICONINFORMATION);

}

Листинг 2 – Файл Server.cpp

#include <iostream>

#include <winsock2.h>

#include <fstream>

#include <vector>

#include <thread>

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

void HandleClient(SOCKET clientSocket, std::vector<SOCKET>& clients, const std::wstring& serverFolderPath);

void BroadcastFile(const wchar\_t\* filePath, const std::vector<SOCKET>& clients, SOCKET senderClient);

int main() {

setlocale(LC\_ALL, "Russian");

WSADATA wsaData;

if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0) {

std::cerr << "Ошибка инициализации WinSock" << std::endl;

return 1;

}

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

if (serverSocket == INVALID\_SOCKET) {

std::cerr << "Ошибка создания сокета" << std::endl;

WSACleanup();

return 1;

}

sockaddr\_in serverAddr;

serverAddr.sin\_family = AF\_INET;

serverAddr.sin\_addr.s\_addr = INADDR\_ANY;

serverAddr.sin\_port = htons(8080);

if (bind(serverSocket, (sockaddr\*)&serverAddr, sizeof(serverAddr)) == SOCKET\_ERROR) {

std::cerr << "Ошибка связывания сокета и адреса" << WSAGetLastError() << std::endl;

closesocket(serverSocket);

WSACleanup();

return 1;

}

if (listen(serverSocket, SOMAXCONN) == SOCKET\_ERROR) {

std::cerr << "При прослушивании прорта 8080 произошла ошибка" << WSAGetLastError() << std::endl;

closesocket(serverSocket);

WSACleanup();

return 1;

}

std::cout << "Сервер прослушивает порт 8080" << std::endl;

std::vector<SOCKET> clients;

std::wstring serverFolderPath = L"M:\\ServerFiles\\";

while (true) {

SOCKET clientSocket = accept(serverSocket, nullptr, nullptr);

if (clientSocket == INVALID\_SOCKET) {

std::cerr << "Ошибка подключение клиента" << WSAGetLastError() << std::endl;

closesocket(serverSocket);

WSACleanup();

return 1;

}

std::cout << "Подключен клиент." << std::endl;

clients.push\_back(clientSocket);

std::thread(HandleClient, clientSocket, std::ref(clients), serverFolderPath).detach();

}

closesocket(serverSocket);

WSACleanup();

return 0;

}

void HandleClient(SOCKET clientSocket, std::vector<SOCKET>& clients, const std::wstring& serverFolderPath) {

while (true) {

std::streamsize fileSize;

int bytesRead = recv(clientSocket, reinterpret\_cast<char\*>(&fileSize), sizeof(fileSize), 0);

if (bytesRead <= 0) {

std::cerr << "Ошибка получения размера файла или отключение клиента" << std::endl;

return;

}

wchar\_t fileName[MAX\_PATH];

bytesRead = recv(clientSocket, reinterpret\_cast<char\*>(fileName), sizeof(fileName), 0);

if (bytesRead <= 0) {

std::cerr << "Ошибка получения имени файла" << std::endl;

return;

}

std::wstring fullFilePath = serverFolderPath + L"\\" + fileName;

std::ofstream receivedFile(fullFilePath.c\_str(), std::ios::binary);

while (fileSize > 0) {

char buffer[1024];

bytesRead = recv(clientSocket, buffer, sizeof(buffer), 0);

if (bytesRead <= 0) {

std::cerr << "Ошибка получения содержимого файла" << std::endl;

return;

}

receivedFile.write(buffer, bytesRead);

fileSize -= bytesRead;

}

receivedFile.close();

std::cout << "Файл получен успешно" << std::endl;

BroadcastFile(fileName, clients, clientSocket);

}

}

void BroadcastFile(const wchar\_t\* filePath, const std::vector<SOCKET>& clients, SOCKET senderClient) {

std::wstring serverFolderPath = L"M:\\ServerFiles\\";

std::wstring NewFilePath = serverFolderPath + filePath;

std::ifstream file(NewFilePath, std::ios::binary);

if (!file) {

std::cerr << "Ошибка открытия файла для транслирования" << std::endl;

return;

}

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

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

file.seekg(0, std::ios::beg);

const char\* request = "REQUEST\_FILE\_TRANSFER";

for (SOCKET clientSocket : clients) {

if (clientSocket != senderClient) {

send(clientSocket, request, strlen(request), 0);

}

}

for (SOCKET clientSocket : clients) {

if (clientSocket != senderClient) {

send(clientSocket, reinterpret\_cast<const char\*>(&fileSize), sizeof(fileSize), 0);

send(clientSocket, reinterpret\_cast<const char\*>(filePath), sizeof(wchar\_t) \* MAX\_PATH, 0);

}

}

char buffer[1024];

while (!file.eof()) {

file.read(buffer, sizeof(buffer));

for (SOCKET clientSocket : clients) {

if (clientSocket != senderClient) {

send(clientSocket, buffer, file.gcount(), 0);

}

}

}

file.close();

std::cout << "Файл транслирован." << std::endl;

}