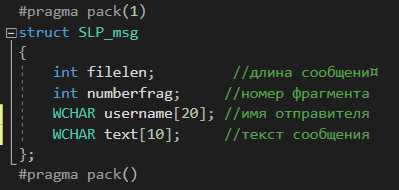
1. 1. студента группы ИТ – 32  
      Курбатовой Софьи Андреевны

|  |  |  |  |
| --- | --- | --- | --- |
| Выполнение: |  | Защита |  |

* + 1. Разработка сетевых приложений
       1. **Цель работы** : получение навыков в проектировании прикладных протоколов для передачи данных и реализации этих протоколов в приложениях Windows на языке C/C++ с применением Windows Sockets.
       2. Содержание работы
       3. 1. Разработать прикладной протокол для передачи текстовых сообщений c помощью транспортного протокола UDP. Предусмотреть возможность разбиения больших сообщений на отдельные фрагменты.
       4. 2. Разработать в Visual C++ приложение Windows для обмена сообщениями по протоколу, разработанному в п. 1. Отправка сообщений должна производиться как широковещательно, так и с указанием адреса назначения (имя или IP-адрес).
       5. 3. Разработать прикладной протокол для передачи файлов c помощью транспортного протокола TCP.
       6. 4. Разработать в Visual C++ два приложения Windows для копирования файлов по протоколу, разработанному в п. 3:
       7. − серверное приложение, которое принимает файлы от клиентских приложений и сохраняет их на диске;
       8. − клиентское приложение, которое должно подключаться к серверному приложению для отправки файлов. Предусмотреть возможность одновременного копирования нескольких файлов.
       9. 5. Протестируйте работу приложений, разработанных в п. 2 и 4, на нескольких компьютерах под управлением Windows.
          1. Ход работы

1. В соответствии с определенным в п.1. задании был разработан прикладной протокол для передачи текстовых сообщений с помощью транспортного протокола UDP. Структура пакета определена на рисунке 4.1. :



Структура фрагмента: UDP

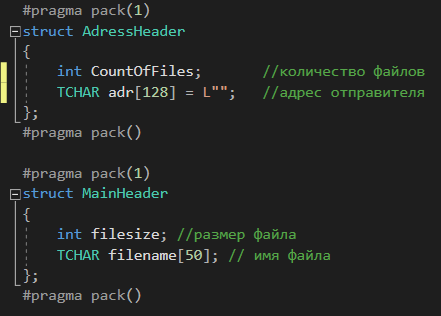
Поле filelen содержит общую длину передаваемого сообщения, которая в дальнейшем будет использована для определения количества переданных пакетов.

Поле numberfrag содержит номер передаваемого фрагмента. Каждый отправляемый фрагмент перед отправкой получает номер, начиная с 0. Получение сообщения будет производиться до тех пор, пока не будет получен последний фрагмент.

Поле username содержит в себе имя отправителя, которое будет выведено на экран вместе с полученным сообщением.

Поле text содержит в себе часть сообщения фиксированной длины. Таким образом зная длину всего сообщения и размер этого поля можно определить какое количество пакетов должно быть получено.

В соответствии с определенным в п.3. задании был разработан прикладной протокол для передачи файлов с помощью транспортного протокола TCP. Структуры описаны на рисунке



Структуры фрагмента: заголовок и данные

Структура AddressHeader будет отправлена серверу перед началом отправки файлов. В содержимом структуры определены адрес отправителя и передаваемое количество файлов, которое используется при получении файлов сервером.

Структура MainHeader будет отправлена серверу после отправки первой структуры. Её содержимое включает в себя поля filesize, в котором будет записан размер передаваемого файла, а в поле filename определено его имя. Получив структуру сервер сможет создать файл в соответствии с содержимым ее полей. А затем выполнит запись полученных данных в созданный файл. Процесс будет выполняться до тех пор, пока количество полученных сегментов не станет равным содержимому поля CountOfFiles из структуры AddressHeader.

2. Блок-схемы алгоритмов представлены на рисунках:

3.

* 1. Листинг 1. Приложение для обмена сообщениями с помощью UDP

//данные для отправки

#define IDC\_EDIT\_MESSAGES 2001

#define IDC\_EDIT\_TEXT 2002

#define IDC\_EDIT\_USERNAME 2003

#define IDC\_CLEAR\_ALL 2004 //кнопка чтобы удалить весь отправленный текст

//переключение адреса

#define IDC\_IPADDR 2005 //чтобы ввести целевой IP

#define IDC\_CONNECT 2006

#define IDC\_DISCONNECT 2007

#define MAX\_TEXT 1024

//размеры текста с сообщением

#define MAX\_MESSAGE\_SIZE 255

#define MAX\_USERNAME\_SIZE 20

#define MAX\_USERMESSAGE\_SIZE 532

#define FRAFMENT\_PACK\_SIZE 10

/\*Дескрипторы\*/

HWND hwnd = NULL; // дескриптор главного окна

/\*Оконные процедуры\*/

LRESULT CALLBACK MainWindowProcess(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam);

BOOL PreTranslateMessage(LPMSG lpMsg);//для передачи сообщений при нажатии на Enter

/\*Обработчики WM\_CREATE, WM\_DESTROY, WM\_SIZE\*/

BOOL OnCreate(HWND hwnd, LPCREATESTRUCT lpCreateStruct);

void OnDestroy(HWND hwnd);

void OnCommand(HWND hwnd, int id, HWND hwndCtl, UINT codeNotify);

HANDLE hThread; //поток для приема сообщений от экземпляров приложения

unsigned \_stdcall ThreadFunc(LPVOID lParam); //чтение сообщений здесь

//пакет для передачи

#pragma pack(1)

struct SLP\_msg

{

int filelen; //длина сообщени¤

int numberfrag; //номер фрагмента

WCHAR username[20]; //имя отправителя

WCHAR text[10]; //текст сообщения

};

#pragma pack()

//работа с сокетами

SOCKET sockets;

sockaddr\_in sockSin = { 0 };

sockaddr\_in sockSout = { 0 };

sockaddr\_in sockSoin = { 0 };

void SendText(SLP\_msg msg, LPCTSTR Send\_Data, unsigned int Send\_Data\_Size);//отправка сообщений

void StartChat(HWND hwnd, LPCTSTR Message);//вывод имени отправителя и сообщения на экран

int WINAPI \_tWinMain(HINSTANCE hInstance, HINSTANCE, LPTSTR CmdLine, int CmdShow)

{

/\*Регистрация оконного класса и обработка ошибки\*/

WNDCLASSEX wcex = { sizeof(WNDCLASSEX) };

wcex.style = CS\_HREDRAW | CS\_VREDRAW | CS\_DBLCLKS;

wcex.lpfnWndProc = MainWindowProcess; // оконная процедура

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(NULL, IDI\_APPLICATION);

wcex.hCursor = LoadCursor(NULL, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)CreateSolidBrush(RGB(0, 100, 256));

wcex.lpszClassName = TEXT("MainWindowProcess"); // имя класса

wcex.hIconSm = LoadIcon(NULL, IDI\_APPLICATION);

if (0 == RegisterClassEx(&wcex))

{

return -1;

}

/\*---------------------------------------------\*/

LoadLibrary(TEXT("ComCtl32.dll"));

LoadLibrary(TEXT("Ws2\_32.dll"));//for sockets

/\*Создание главного файла и обработка ошибки \*/

hwnd = CreateWindowEx(0, TEXT("MainWindowProcess"), TEXT("Chat"),

WS\_OVERLAPPEDWINDOW, CW\_USEDEFAULT, 0, 900, 600, NULL, NULL, hInstance, NULL);

if (hwnd == NULL) { return -1; }

/\*--------------------------------------------------\*/

ShowWindow(hwnd, CmdShow); // отображаем главное окно

/\*Цикл обработки сообщений\*/

MSG msg;

BOOL RetRes;

while ((RetRes = GetMessage(&msg, NULL, 0, 0)) != FALSE)

{

if (RetRes == -1)

{

//Error editing

}

else if (!PreTranslateMessage(&msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return (int)msg.wParam;

}

/\*Процедура главного окна\*/

LRESULT CALLBACK MainWindowProcess(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam)

{

switch (msg)

{

HANDLE\_MSG(hWnd, WM\_CREATE, OnCreate);

HANDLE\_MSG(hWnd, WM\_DESTROY, OnDestroy);

HANDLE\_MSG(hWnd, WM\_COMMAND, OnCommand);

}

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

}

/\*Обработчик сообщений\*/

BOOL PreTranslateMessage(LPMSG Msg)

{

/\*Переменные\*/

WCHAR Message[MAX\_MESSAGE\_SIZE]; //сообщение

WCHAR UserName[MAX\_USERNAME\_SIZE] = \_T("");//имя отправителя

WCHAR UserMessage[MAX\_USERMESSAGE\_SIZE] = \_T("");//сообщение + имя отправителя

//объявим структуру с пакетом

SLP\_msg msg;

memset(msg.text, NULL, 10);

memset(msg.username, NULL, 20);

DWORD symbols, symb\_user; //количество символов в сообщении

if ((WM\_KEYDOWN == Msg->message) && (VK\_RETURN == Msg->wParam)) // нажата клавиша Enter

{

HWND hwndCtl = GetDlgItem(hwnd, IDC\_EDIT\_TEXT);

HWND hwndUser = GetDlgItem(hwnd, IDC\_EDIT\_USERNAME);

if (GetFocus() == hwndCtl) // поле ввода обладает фокусом клавиатуры

{

/\*Чтобы можно было отправить многострочный текст\*/

/\*CTRL+ENTER\*/

if (GetKeyState(VK\_SHIFT) < 0) // нажата клавиша SHIFT

{

Edit\_ReplaceSel(hwndCtl, \_T("\r\n"));

}

else

{

symbols = Edit\_GetText(hwndCtl, Message, \_countof(Message));// копируем текст из поля ввода

symb\_user = Edit\_GetText(hwndUser, UserName, \_countof(UserName));// копируем текст из поля ввода

if (symbols > 0)

{

// очищаем поле ввода

Edit\_SetText(hwndCtl, NULL);

//скопируем введенные данные в структуру

StringCchCat(msg.username,20, UserName);

StartChat(hwnd,Message);//отобрзим отправляемый текст у себя

SendText(msg, Message, wcslen(Message));

}

}

return TRUE;

}

}

return FALSE;

}

BOOL OnCreate(HWND hwnd, LPCREATESTRUCT lpCreateStruct)

{

DWORD dwStyle = WS\_CHILD | WS\_VISIBLE | WS\_VSCROLL | WS\_HSCROLL | ES\_MULTILINE | ES\_AUTOHSCROLL | ES\_AUTOVSCROLL;

// создаём поле вывода сообщений

CreateWindowEx(WS\_EX\_STATICEDGE, TEXT("Edit"), TEXT(""), dwStyle | ES\_READONLY,

10, 10, 490, 250, hwnd, (HMENU)IDC\_EDIT\_MESSAGES, lpCreateStruct->hInstance, NULL);

//Для username

CreateWindowEx(0, TEXT("Static"), TEXT("User: "), WS\_CHILD | WS\_VISIBLE | SS\_SIMPLE,

10, 270, 40, 40, hwnd, NULL, lpCreateStruct->hInstance, NULL);

CreateWindowEx(WS\_EX\_CLIENTEDGE, TEXT("Edit"), TEXT(""), WS\_CHILD | WS\_VISIBLE,

50, 270, 450, 40, hwnd, (HMENU)IDC\_EDIT\_USERNAME, lpCreateStruct->hInstance, NULL);

CreateWindowEx(0, TEXT("Button"), TEXT("Clear History"), WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

520, 10, 200, 30, hwnd, (HMENU)IDC\_CLEAR\_ALL, lpCreateStruct->hInstance, NULL);

// создаём поле ввода

HWND hwndCtl = CreateWindowEx(WS\_EX\_CLIENTEDGE, TEXT("Edit"), TEXT(""), dwStyle,

10, 320, 490, 140, hwnd, (HMENU)IDC\_EDIT\_TEXT, lpCreateStruct->hInstance, NULL);

//Для работы с адресами

// создаём поле ввода IP-адреса

CreateWindowEx(0, TEXT("SysIPAddress32"), NULL, WS\_CHILD | WS\_VISIBLE,

520, 50, 200, 30, hwnd, (HMENU)IDC\_IPADDR, lpCreateStruct->hInstance, NULL);

//кнопка для переключения

CreateWindowEx(0, TEXT("Button"), TEXT("Connect"), WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

730, 50, 100, 30, hwnd, (HMENU)IDC\_CONNECT, lpCreateStruct->hInstance, NULL);

//кнопка для переключения

CreateWindowEx(0, TEXT("Button"), TEXT("Disconnect"), WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

730, 90, 100, 30, hwnd, (HMENU)IDC\_DISCONNECT, lpCreateStruct->hInstance, NULL);

// задаём ограничение на ввод текста

Edit\_LimitText(hwndCtl, MAX\_MESSAGE\_SIZE);

//Инициализация библиотеки WinSock

WSADATA WsaData;

int err = WSAStartup(MAKEWORD(2,2),&WsaData);

// Открытие сокета

sockets = socket(AF\_INET, SOCK\_DGRAM,NULL);

//заполним структур, чтобы через можно было получать данные

sockSout.sin\_family = AF\_INET; // AF\_INET = IPv4 addresses

sockSout.sin\_port = htons(7581); // Little to big endian conversion

sockSout.sin\_addr.s\_addr = htonl(INADDR\_ANY);//0.0.0.0

//выполним ассоциирование сокета

err = bind(sockets, (sockaddr\*)&sockSout, sizeof(sockSout));

if (err != SOCKET\_ERROR)

{

hThread = (HANDLE)\_beginthreadex(NULL, 0, ThreadFunc, NULL, 0, NULL);

}

return TRUE;

}

void OnDestroy(HWND hwnd)

{

// уведомляем о завершении приложения,

closesocket(sockets);

WSACleanup();

PostQuitMessage(0); // отправляем сообщение WM\_QUIT

}

void OnCommand(HWND hwnd, int id, HWND hwndCtl, UINT codeNotify)

{

if (BN\_CLICKED == codeNotify)

{

HINSTANCE hInstance = GetWindowInstance(hwnd);

switch (id)

{

case IDC\_CLEAR\_ALL:

{

HWND hwndCtl = GetDlgItem(hwnd, IDC\_EDIT\_MESSAGES);

Edit\_SetText(hwndCtl, NULL);// очищаем поле ввода

}

break;

case IDC\_CONNECT:

{

// получим IP-адрес из поля ввода

const char bufferNameIP[25] = { 0 };

GetDlgItemTextA(hwnd, IDC\_IPADDR, (LPSTR)bufferNameIP, \_countof(bufferNameIP));

if (bufferNameIP != NULL)

{

sockaddr\_in sin = { 0 };

sockSin.sin\_family = AF\_INET;

sockSin.sin\_port = htons(7581);

sockSin.sin\_addr.s\_addr = inet\_addr(bufferNameIP);//"192.168.56.104" "192.168.56.1"

MessageBox(hwnd, L"IP-connect Enable", L"Details", MB\_OK);

}

else

{

MessageBox(hwnd, L"Null IP", L"Details", MB\_OK);

}

}

break;

case IDC\_DISCONNECT:

{

// очистим поле ввода IP-адреса

SendDlgItemMessage(hwnd, IDC\_IPADDR, IPM\_CLEARADDRESS,0,0);

BOOL optval = TRUE;

int optlen = sizeof(optval);

int err = setsockopt(sockets, SOL\_SOCKET, SO\_BROADCAST, (char\*)&optval, optlen);

sockSin.sin\_family = AF\_INET;

sockSin.sin\_port = htons(7581);

sockSin.sin\_addr.s\_addr = htonl(INADDR\_BROADCAST);

MessageBox(hwnd, L"BroadCast Enable", L"Details", MB\_OK);

}

break;

}

}

}

// ----------------------------------------------------------------------------------------------

void SendText(SLP\_msg msg, LPCTSTR Send\_Data, unsigned int Send\_Data\_Size)

{

if ((Send\_Data != NULL) && (Send\_Data\_Size > 0))

{

int packnum = 0;//номер пакета

msg.filelen = Send\_Data\_Size;

for (int i = 0; i < Send\_Data\_Size; i+=10)

{

WCHAR frag\_pack[FRAFMENT\_PACK\_SIZE] = L""; //инициализация фрагмента пакета

memcpy(frag\_pack, &Send\_Data[i], sizeof(frag\_pack));//скопируем данные в фрагмента пакета

StringCchCat(msg.text,sizeof(msg.text), frag\_pack);//скопируем данные в отправялемый пакет

msg.numberfrag = packnum;//запишем номер пакета

int result = sendto(sockets,(const char\*)&msg, sizeof(msg), NULL,(struct sockaddr\*)&sockSin, sizeof(sockSin));

packnum++;//увеличим номер пакета для следующей отправки

ZeroMemory(msg.text, sizeof(msg.text));

}

}

}

void StartChat(HWND hwnd, LPCTSTR Message)

{

MessageBox(hwnd, L"LET'S GO CHAT!", L"CHAT", MB\_OK);

WCHAR UserMessage[MAX\_USERMESSAGE\_SIZE] = \_T("");

HWND hwndCtl = GetDlgItem(hwnd, IDC\_EDIT\_MESSAGES);

StringCchCat(UserMessage, \_countof(UserMessage), \_T("\r\n"));

StringCchCat(UserMessage, \_countof(UserMessage), Message);

Edit\_SetSel(hwndCtl, Edit\_GetTextLength(hwndCtl), -1);// устанавливаем курсор в конец поля вывода

SetFocus(hwnd);

Edit\_ReplaceSel(hwndCtl, UserMessage);// вставляем текст в поле вывода

}

unsigned \_\_stdcall ThreadFunc(void\* lParam)

{

/\*Переменные\*/

WCHAR Message[MAX\_MESSAGE\_SIZE] = \_T(""); //сообщение

WCHAR UserName[MAX\_USERNAME\_SIZE] = \_T(""); //имя отправителя

WCHAR UserMessage[MAX\_USERMESSAGE\_SIZE] = \_T(""); //имя отправителя+сообщение

SLP\_msg recived\_msg = {0};

int socket\_len = sizeof(sockSout);

for (;;)

{

//Начнем прием сообщений через сокет

int result = recvfrom(sockets,(char\*)&recived\_msg,sizeof(recived\_msg),NULL,(struct sockaddr\*)&sockSout, &socket\_len);

if(result != SOCKET\_ERROR)

{

int struct\_size = sizeof(recived\_msg);//узнаем размер структуры

int reseived\_size = recived\_msg.filelen / \_countof(recived\_msg.text); //количество пакетов

/\*Собираем сообщение\*/

for (int i = 0; i < MAX\_MESSAGE\_SIZE; i++)

{

if (i == recived\_msg.numberfrag\*10)

{

for (int j = 0; j < 10 ; j++)

{

Message[i+j] = recived\_msg.text[j];

}

}

}

int recived = wcslen(Message);//не используется. Для просмотра размера при отладке

//если определенный через размер номер равен номеру пакета, значит сообщение было получено верно

if (reseived\_size == recived\_msg.numberfrag)

{

/\*Заполнение массивов для вывода имени отправителя и сообщения\*/

StringCchCat(UserName, 20, recived\_msg.username);

StringCchCat(UserName, \_countof(UserName), \_T(":"));

StringCchCat(UserMessage, \_countof(UserMessage), UserName);

StringCchCat(UserMessage, \_countof(UserMessage), Message);

StartChat(hwnd, UserMessage);

/\*Освобождение ресурсов и т.д.\*/

memset(UserName, NULL, 255);

memset(UserMessage, NULL, 255);

memset(Message, NULL, 255);

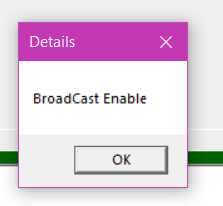
}

}

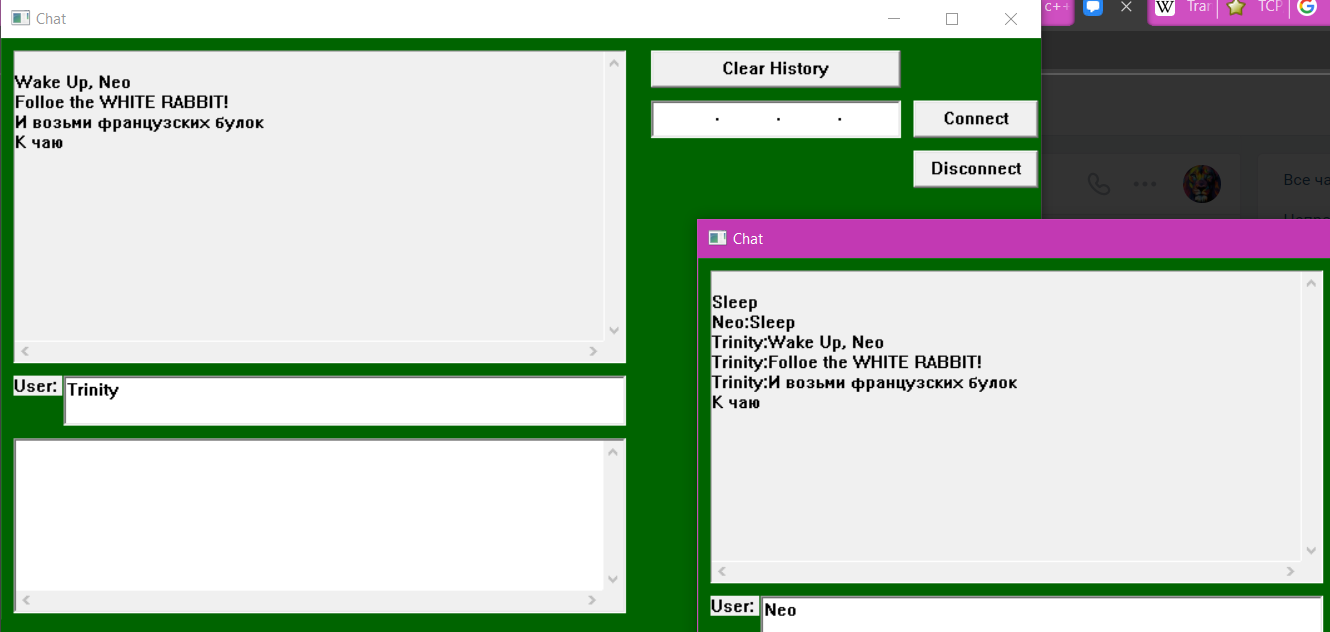
}

return(0);

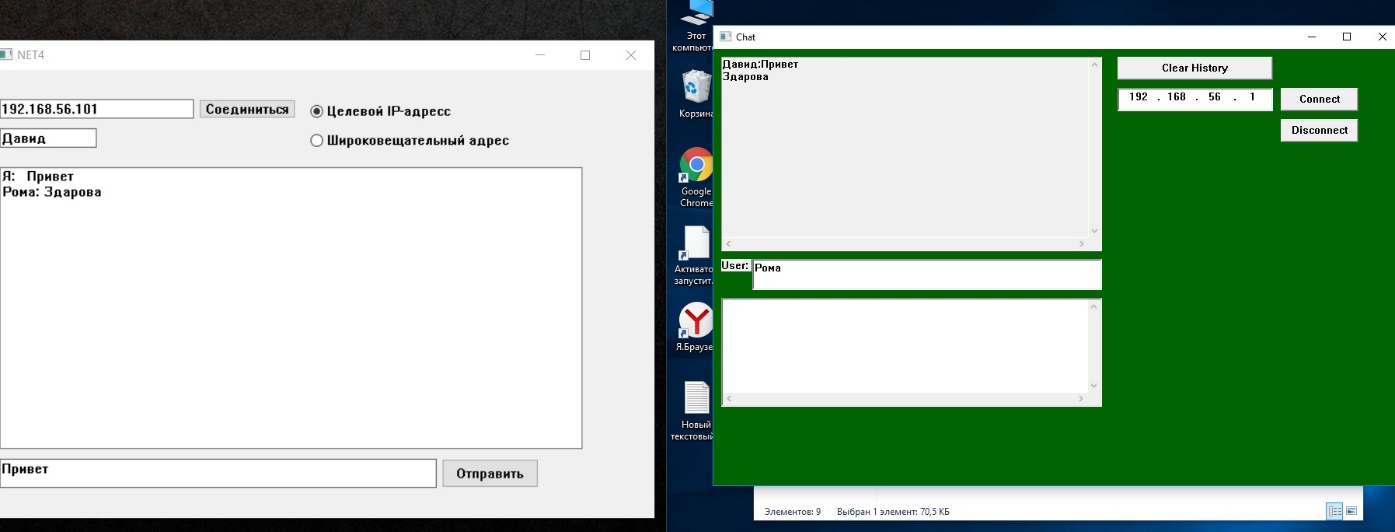
}

* + - 1. 4. Демонстрация работы приложения 1 на рисунках :
      2. 

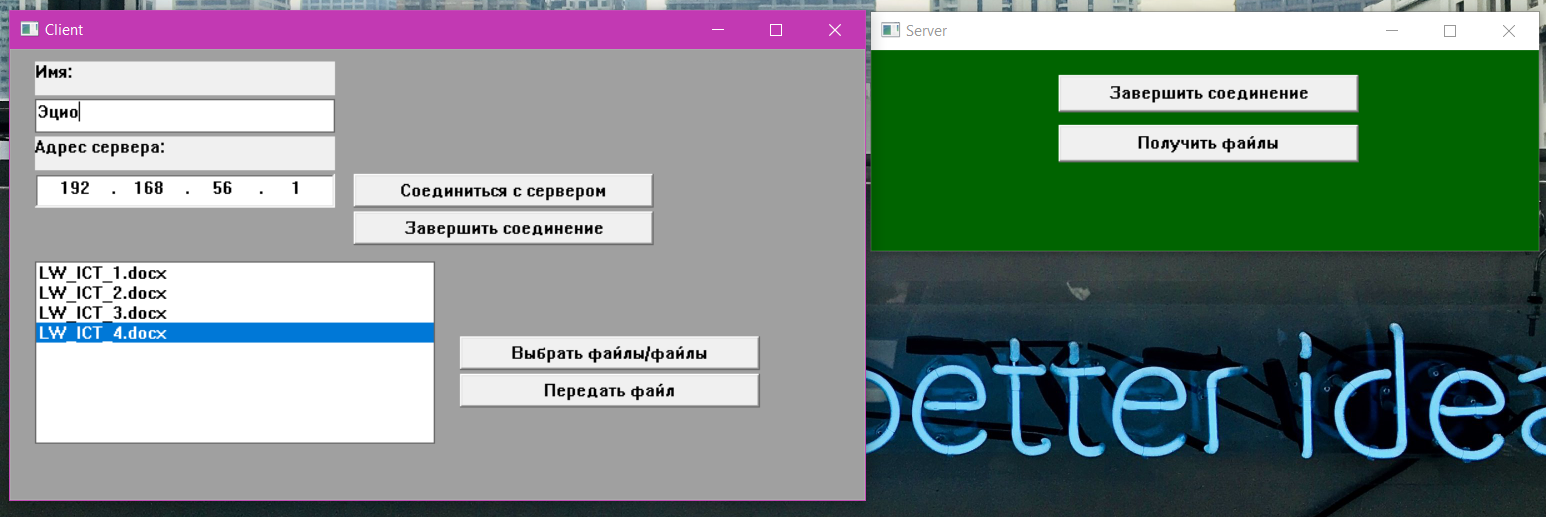
Нажатие на кнопку Disconnect включает широковещательную передачу

****

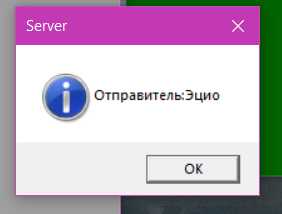
Пример отправки сообщений

* + - 1. 

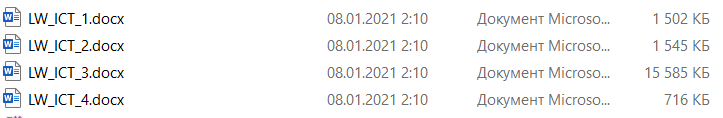
Отправка сообщений на целевой IP-адрес

* + - 1. Демонстрация работа приложения 2 продемонстрирована на рисунках :
      2. 

Вид приложения перед началом отправки файлов

* + - 1. 

Информация об отправителе

* + - 1. 

Переданные файлы

**Вывод:** Таким образом, в ходе выполнения лабораторной работы были получены навыки в проектировании прикладных протоколов для передачи данных и реализации этих протоколов в приложениях Windows на языке C/C++ с применением Windows Sockets