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 №8

на тему

**ИНТЕРФЕЙС СОКЕТОВ И ОСНОВЫ СЕТЕВОГО ПРОГРАММИРОВАНИЯ (WINDOWS).**

**ПРОГРАММИРОВАНИЕ ВЗАИМОДЕЙСТВИЯ ЧЕРЕЗ**

**СЕТЬ С ИСПОЛЬЗОВАНИЕМ ИНТЕРФЕЙСА**

**СОКЕТОВ. РЕАЛИЗАЦИЯ СЕТЕВЫХ ПРОТОКОЛОВ: СОБСТВЕННЫХ ИЛИ СТАНДАРТНЫХ**

Студент З. Э. Карась

Преподаватель Н. Ю. Гриценко

Минск 2023

СОДЕРЖАНИЕ

[1 Цель работы 3](#_Toc146752068)

[2 Теоретические сведения 4](#_Toc146752069)

[3 Результат выполнения 5](#_Toc146752070)

[Заключение 6](#_Toc146752071)

[Список использованных источников 7](#_Toc146752072)

[Приложение А (обязательное) Листинг кода 8](#_Toc146752073)

1. **ЦЕЛЬ РАБОТЫ**

Целью выполнения лабораторной работы является проектирование и

написание многопользовательского приложения для обмена изображениями

в локальной сети с использованием сокетов. Приложение позволяет

как отправлять изображения от клиента к серверу.

1. **ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Сетевое программирование – это область программирования, связанная с разработкой приложений, способных обмениваться данными через компьютерные сети. Один из ключевых инструментов для реализации сетевого взаимодействия – это сокеты.

Сокет (*socket*) – это конечная точка для обмена данными между процессами в компьютерной сети. Сокеты предоставляют абстракцию для взаимодействия между процессами, работающими на разных узлах сети, используя протоколы транспортного уровня, такие как *TCP* или *UDP)* [1]*.*

*TCP* и *UDP* – это два основных протокола транспортного уровня, используемых в сетевом программировании.

*TCP (Transmission Control Protocol)* – обеспечивает устойчивое, упорядоченное и надежное соединение между двумя точками в сети. Применяется в ситуациях, где важна доставка данных без потерь и в правильном порядке.

*UDP (User Datagram Protocol) -* обеспечивает простую и неупорядоченную передачу данных. Применяется в сценариях, где допускаются потери данных, но критически важна минимальная задержка [2].

*Windows Sockets API (Winsock)* – это стандартный интерфейс для разработки сетевых приложений под операционной системой Windows. Он предоставляет функции и структуры данных, необходимые для работы с сокетами.

Инициализация *Winsock* выполняется с использованием функции *WSAStartup*, а завершение - с использованием функции *WSACleanup*. Эти функции обеспечивают правильную инициализацию и очистку ресурсов, связанных с работой сокетов.

Сокет создается с использованием функции *socket.* После создания сокета, необходимо задать параметры, такие как адрес и порт, с которыми он будет ассоциирован. Эти параметры настраиваются с использованием структуры *sockaddr\_in*.

Для установки соединения с удаленным сервером используется функция *connect.* Она требуется для клиентских приложений и выполняет установку соединения с сервером по указанному адресу и порту.

Функции *send* и *recv* используются для отправки и приема данных через сокеты. Эти функции позволяют передавать байты данных между клиентом и сервером [3].

1. **РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ**

В результате работы было создано два приложения: сервер и клиент.

Реализация серверного приложения (рисунок 1):

1. Создан серверный сокет, привязанный к определенному порту, ожидающий подключений от клиента.
2. Сервер способен делать и сохранять фотографию с веб-камеры.
3. Сервер получаю фотографию от клиента сохраняет ее в папке.

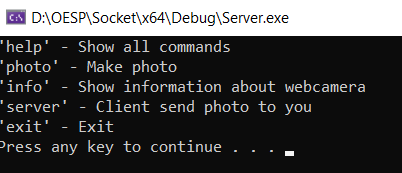


Рисунок 1 – Окно сервера

Реализация клиентского приложения (рисунок 2):

Создано клиентское приложение, которое ждет сообщения от сервера, принимает его, делает фотографию, сохраняет ее и отправляет серверу.

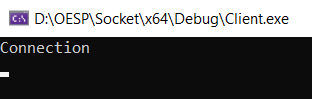


Рисунок 2 – Запущенный клиент

# ЗАКЛЮЧЕНИЕ

В результате выполнения лабораторной работы были изучен и освоен интерфейс сокетов и основы сетевого программирования Windows с использованием Win32 C++ API. Помимо этого, для закрепления изученных навыков было разработано многопользовательское приложение, использующее вышеперечисленные технологии для обмена сообщениями в локальной сети.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Программирование сетевых задач [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Программирование_сетевых_задач>.
2. Протоколы TCP и UDP [Электронный ресурс]. – Режим доступа: https://professorweb.ru/my/csharp/web/level1/1\_4.php.
3. Использование Winsock [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/ru-ru/windows/win32/winsock/using-winsock>.

# ПРИЛОЖЕНИЕ А (обязательное) Листинг кода

Листинг 1 – Client.cpp

#pragma once

#define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS

#define \_SILENCE\_EXPERIMENTAL\_FILESYSTEM\_DEPRECATION\_WARNING

#define VIDEO\_FOLDER\_NAME "./Video/" // папка для видео

#define PHOTO\_FOLDER\_NAME "./Photo/" // папка для фото

#define VIDEO\_FORMAT ".avi" // расширение видео

#define PHOTO\_FORMAT ".jpg" // расширение фото

#define PHOTO\_DELAY 5000 // задержка между фото

#include <iostream>

#include <iomanip>

#include <experimental/filesystem>

#include <string>

#include <set>

#include <thread>

#include <atomic>

#include <locale.h>

#include <fstream>

#include <vector>

#include <cmath>

#include <opencv2\opencv.hpp>

#include <windows.h>

#include <vfw.h>

#include <setupapi.h>

#include <devguid.h>

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

#pragma comment(lib, "Wsock32.lib")

#pragma comment(lib, "setupapi.lib")

#pragma comment(lib, "vfw32.lib")

using namespace std;

using namespace cv;

void makePhoto()

{

Mat image;

VideoCapture camera;

camera.open(0);

if (!camera.isOpened()) {

cout << "Error: failed to open webcam" << endl;

exit(EXIT\_FAILURE);

}

camera >> image;

imwrite("./Photo/photo.jpg", image);

camera.release();

}

int main() {

string operation = "";

WSADATA wsaData;

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

std::cerr << "WSAStartup failed." << std::endl;

return 1;

}

// Создаем сокет

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

if (clientSocket == INVALID\_SOCKET) {

std::cerr << "Error when creating a socket." << std::endl;

WSACleanup();

return 1;

}

// Задаем параметры сервера

sockaddr\_in serverAddr;

serverAddr.sin\_family = AF\_INET;

serverAddr.sin\_addr.s\_addr = inet\_addr("192.168.0.100"); // IP адрес сервера

serverAddr.sin\_port = htons(12345); // Порт сервера

while (true)

{

Sleep(5000);

int flag = 0;

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

{

std::cerr << "Error connecting to the server." << std::endl;

closesocket(clientSocket);

WSACleanup();

flag = 1;

}

if (flag) {

continue;

}

else {

break;

}

}

std::cout << "Connection" << std::endl;

int bytesReceived;

char buffer[102400];

while (true) {

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

operation = buffer;

// getline(cin, operation);

if (operation == "exit") {

return 0;

}

else if (operation == "photo") {

makePhoto();

}

else if (operation == "connect") {

// Подключаемся к серверу

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

std::cerr << "Error connecting to the server." << std::endl;

closesocket(clientSocket);

WSACleanup();

return 1;

}

}

else if (operation == "send") {

std::ifstream inputFile("./Photo/photo.jpg", std::ios::binary);

std::ofstream outputFile("output.txt");

char message[102400];

int bytesSent;

if (inputFile.is\_open()) {

char byte;

int i = 0;

for (; inputFile.get(byte); i++) {

message[i] = byte;

}

message[i] = '\0';

bytesSent = send(clientSocket, message, i, 0);

if (bytesSent == SOCKET\_ERROR) {

std::cerr << "Error sending the message." << std::endl;

break;

}

inputFile.close();

outputFile.close();

}

else {

std::cerr << "Failed to open files." << std::endl;

}

std::cout << "sended" << std::endl;

}

}

}

Листинг 2 – Server.cpp

#include "Server.h"

Server::~Server() {

closesocket(clientSocket);

closesocket(serverSocket);

WSACleanup();

}

int Server::openServer() {

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

std::cerr << "WSAStartup failde!" << std::endl;

return 1;

}

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

if (serverSocket == INVALID\_SOCKET) {

std::cerr << "Invalid." << std::endl;

WSACleanup();

return 2;

}

serverAddr.sin\_family = AF\_INET;

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

serverAddr.sin\_port = htons(12345);

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

std::cerr << "Invalid." << std::endl;

closesocket(serverSocket);

WSACleanup();

return 4;

}

return 0;

}

int Server::connectServer() {

if (listen(serverSocket, 1) == SOCKET\_ERROR) {

std::cerr << "Invalid." << std::endl;

closesocket(serverSocket);

WSACleanup();

return 1;

}

clientSocket = accept(serverSocket, (struct sockaddr\*)&clientAddr, &clientAddrLen);

if (clientSocket == INVALID\_SOCKET) {

std::cerr << "Invalid." << std::endl;

closesocket(serverSocket);

WSACleanup();

return 2;

}

return 0;

}

int Server::sendMessage(string message) {

int bytesReceived;

bytesReceived = send(clientSocket, message.c\_str(), message.size() + 1, 0);

if (bytesReceived <= 0) {

std::cout << "Invalid." << std::endl;

}

return bytesReceived;

}

int Server::getMessage(int \*length, char \*message) {

int bytesReceived;

sendMessage("send");

bytesReceived = recv(clientSocket, message, 1024, 0);

if (bytesReceived <= 0) {

std::cout << "Error when receiving data." << std::endl;

}

\*length = bytesReceived;

return 0;

}

int Server::getPhoto() {

int bytesReceived;

char photo[1024 \* 80];

ofstream outputFile("./webcum/output.jpg", ios::out | ios::binary | ios::trunc);

sendMessage("send");

bytesReceived = recv(clientSocket, photo, 1024 \* 80, 0);

if (bytesReceived <= 0) {

std::cout << "Error when receiving data." << std::endl;

}

for (int i = 0; i < bytesReceived; i++) {

outputFile << photo[i];

}

outputFile.close();

return 0;

}