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 И. А. Тиханёнок

Преподаватель Н. Ю. Гриценко

Минск 2023

СОДЕРЖАНИЕ

[1 Цель работы 3](#_Toc146752068)

[2 Теоретические сведения 4](#_Toc146752069)

[3 Результат выполнения 5](#_Toc146752070)

[Заключение 5](#_Toc146752071)

[Список использованных источников 7](#_Toc146752072)

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

1. **ЦЕЛЬ РАБОТЫ**

Цель работы изучить и применить на практике знания об интерфейсах сокетов и основах сетевого программирования *Windows*, научиться использовать взаимодействия через сеть с использованием интерфейса сокетов, а также использовать сетевые протоколы. Для достижения цели будет создано клиент-серверное приложение для обмена текстовыми сообщениями с использованием TCP сокетов.

1. **ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Сокеты - это абстракция, предоставляемая операционной системой, которая позволяет приложениям взаимодействовать через сеть. В *Windows* для сетевого программирования с использованием сокетов предоставляются функции и структуры из библиотеки *Winsock* (*Windows* *Sockets*), которая является стандартом для разработки сетевых приложений. Протоколы, поддерживаемые *Winsock*, включают *TCP*/*IP*, *UDP*, и другие.

Библиотека *Winsock* предоставляет набор функций и структур, необходимых для создания сетевых приложений в среде *Windows*. Она позволяет приложениям создавать сокеты, устанавливать соединения, отправлять и получать данные через сеть, а также управлять сетевыми настройками.

В *Winsock* существуют разные типы сокетов, включая: *SOCK\_STREAM* (соксеты *TCP*): Представляют собой потоковые сокеты, которые обеспечивают надежную и упорядоченную передачу данных. Они используют протокол *TCP*. *SOCK\_DGRAM* (соксеты *UDP*): Представляют датаграммные сокеты, которые обеспечивают ненадежную передачу данных, но без установления соединения. Они используют протокол *UDP*. *SOCK\_RAW* (сокеты *RAW*): Позволяют приложению работать на низком уровне с сетевыми пакетами. Они обеспечивают полный доступ к сетевому стеку.

Для создания сокета в Windows используется функция *socket* (). Она принимает параметры, такие как домен (*AF\_INET* для *IPv4*), тип сокета (*SOCK\_STREAM* или *SOCK\_DGRAM*), и протокол (обычно 0, чтобы выбрать соответствующий протокол для домена и типа).

Для клиентского приложения в сокетах *TCP* используется функция *connect* (), чтобы установить соединение с сервером. Для сервера используется функция *bind* (), чтобы связать сокет с конкретным адресом и портом, и *listen* (), чтобы начать прослушивание входящих соединений.

После успешного установления соединения, вы можете использовать функции *send* () и *recv* () для отправки и приема данных через сокета.

После завершения работы с сокетом, его следует закрыть с использованием функции *closesocket* ().

Для создания многопоточных серверов, способных обслуживать несколько клиентов одновременно, обычно используются многопоточные или асинхронные подходы. Каждое новое входящее соединение может обрабатываться в отдельном потоке или асинхронно.

С использованием интерфейса сокетов и сетевого программирования в *Windows*, вы можете разрабатывать клиент-серверные приложения для обмена данными через сеть, а также реализовывать сетевые протоколы, как стандартные, так и собственные.

1. **РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ**

В результате выполнения лабораторной работы было реализовано приложение для обмена текстовыми сообщениями между клиентами по локальной сети с использованием сокетов (Рисунок 1).

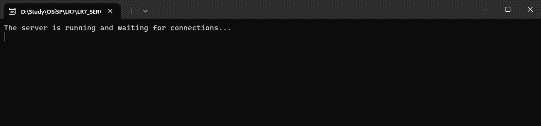


Рисунок 1 – Серверное окно приложения

Также у нас есть клиентское окно приложения, которое позволяет взаимодействовать с сервером (Рисунок 2).

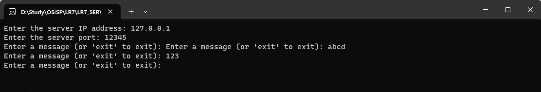


Рисунок 2 – Клиентское окно приложения

Далее будет представлена реализация взаимодействия между клиентом и сервером (Рисунок 3).

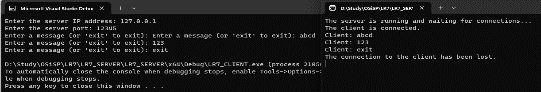


Рисунок 3 – Взаимодействие между серверной и клиентской частью

Наше приложение позволяет пользователям использовать наш самописный сервер для отправки различных сообщений, также наш сервер обрабатывает сразу несколько пользователей, используя многопоточность, для одного пользователя один поток. Сервер и клиент взаимодействуют через сокетов, обеспечивая передачу текстовых сообщений по локальной сети.

## ЗАКЛЮЧЕНИЕ

В данной лабораторной работе были изучены и применены основы сетевого программирования в среде *Windows* с использованием сокетов. Мы разработали клиент-серверное приложение, позволяющее обмениваться текстовыми сообщениями через сеть с использованием протокола *TCP*. Клиентское приложение было адаптировано для ввода адреса сервера и порта, что обеспечивает более гибкую настройку подключения.

Данная лабораторная работа позволила ознакомиться с основами сетевого программирования, сокетами, и применить полученные знания для разработки сетевого приложения. Мы также рассмотрели создание серверов, способных обслуживать несколько клиентов одновременно, что поднимает понимание сетевых взаимодействий на новый уровень.

Изучение и практическое применение интерфейса сокетов и сетевого программирования оказалось полезным для будущих разработчиков, позволяя создавать множество приложений, работающих через сеть и обменивающихся данными.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Щупак Ю. *Win32 API*. Разработка приложений для *Windows*. ─ СПБ: Питер, 2008. ─ 592 с.: ип.
2. Создание классических приложений для *Windows* с использованием *API Win32* [Электронный ресурс]. ─ Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/api

# ПРИЛОЖЕНИЕ А (обязательное) Листинг кода

Листинг 1 – Файл Lab8\_SERVER.cpp

#include <iostream>

#include <Winsock2.h>

#include <vector>

#include <thread>

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

void HandleClient(SOCKET clientSocket) {

char buffer[1024];

int bytesReceived;

while (true) {

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

if (bytesReceived <= 0) {

std::cerr << "The connection to the client has been lost." << std::endl;

break;

}

buffer[bytesReceived] = '\0';

std::cout << "Client: " << buffer << std::endl;

// Отправка ответа клиенту

send(clientSocket, buffer, bytesReceived, 0);

}

closesocket(clientSocket);

}

int main() {

WSADATA wsaData;

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

std::cerr << "Failed to initialize Winsock." << std::endl;

return 1;

}

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

if (serverSocket == INVALID\_SOCKET) {

std::cerr << "Failed to create socket." << std::endl;

WSACleanup();

return 1;

}

sockaddr\_in serverAddress;

serverAddress.sin\_family = AF\_INET;

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

serverAddress.sin\_addr.s\_addr = INADDR\_ANY; // Принимаем подключения на всех доступных интерфейсах

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

std::cerr << "Failed to bind socket." << std::endl;

closesocket(serverSocket);

WSACleanup();

return 1;

}

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

std::cerr << "Failed to listen for connections." << std::endl;

closesocket(serverSocket);

WSACleanup();

return 1;

}

std::cout << "The server is running and waiting for connections..." << std::endl;

std::vector<std::thread> clientThreads;

while (true) {

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

if (clientSocket == INVALID\_SOCKET) {

std::cerr << "Failed to accept connection." << std::endl;

continue;

}

std::cout << "The client is connected." << std::endl;

clientThreads.emplace\_back(HandleClient, clientSocket);

}

closesocket(serverSocket);

WSACleanup();

for (auto& thread : clientThreads) {

thread.join();

}

return 0;

}

Листинг 2 – Файл Lab8\_Client.cpp

#include <iostream>

#include <Winsock2.h>

#include <string>

#include <WS2tcpip.h>

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

int main() {

WSADATA wsaData;

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

std::cerr << "Failed to initialize Winsock." << std::endl;

return 1;

}

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

if (clientSocket == INVALID\_SOCKET) {

std::cerr << "Failed to create socket." << std::endl;

WSACleanup();

return 1;

}

sockaddr\_in serverAddress;

serverAddress.sin\_family = AF\_INET;

// Ввод адреса сервера и порта с клавиатуры

std::string serverIP;

int serverPort;

std::cout << "Enter the server IP address: ";

std::cin >> serverIP;

std::cout << "Enter the server port: ";

std::cin >> serverPort;

if (inet\_pton(AF\_INET, serverIP.c\_str(), &serverAddress.sin\_addr) != 1) {

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

closesocket(clientSocket);

WSACleanup();

return 1;

}

serverAddress.sin\_port = htons(serverPort);

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

std::cerr << "Failed to connect to the server." << std::endl;

closesocket(clientSocket);

WSACleanup();

return 1;

}

std::string message;

while (true) {

std::cout << "Enter a message (or 'exit' to exit): ";

std::getline(std::cin, message);

send(clientSocket, message.c\_str(), static\_cast<int>(message.size()), 0);

if (message == "exit") {

break;

}

}

closesocket(clientSocket);

WSACleanup();

return 0;

}

Листинг 1 – Файл Lab7\_CLIENT.cpp

#include <iostream>

#include <Winsock2.h>

#include <string>

#include <WS2tcpip.h>

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

int main() {

WSADATA wsaData;

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

std::cerr << "Failed to initialize Winsock." << std::endl;

return 1;

}

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

if (clientSocket == INVALID\_SOCKET) {

std::cerr << "Failed to create socket." << std::endl;

WSACleanup();

return 1;

}

sockaddr\_in serverAddress;

serverAddress.sin\_family = AF\_INET;

// Ввод адреса сервера и порта с клавиатуры

std::string serverIP;

int serverPort;

std::cout << "Enter the server IP address: ";

std::cin >> serverIP;

std::cout << "Enter the server port: ";

std::cin >> serverPort;

if (inet\_pton(AF\_INET, serverIP.c\_str(), &serverAddress.sin\_addr) != 1) {

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

closesocket(clientSocket);

WSACleanup();

return 1;

}

serverAddress.sin\_port = htons(serverPort);

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

std::cerr << "Failed to connect to the server." << std::endl;

closesocket(clientSocket);

WSACleanup();

return 1;

}

std::string message;

while (true) {

std::cout << "Enter a message (or 'exit' to exit): ";

std::getline(std::cin, message);

send(clientSocket, message.c\_str(), static\_cast<int>(message.size()), 0);

if (message == "exit") {

break;

}

}

closesocket(clientSocket);

WSACleanup();

return 0;

}