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»

Выполнил:

студент гр. 153504

Михалевич M.П.

Проверил:

Гриценко Н.Ю.

Минск 2023

СОДЕРЖАНИЕ

[1 Цели работы 3](#_Toc147863804)

[2 Kраткие теоретические сведения 4](#_Toc147863805)

[3 Результаты выполнения лабораторной работы 5](#_Toc147863806)

[Вывод 6](#_Toc147863807)

[Список использованных источников 7](#_Toc147863808)

[Приложение А 8](#_Toc147863809)

## 1 ЦЕЛИ РАБОТЫ

Целью данной лабораторной работы является разработка клиент-серверное приложение для обмена текстовыми сообщениями с использованием TCP сокетов.

## 2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Win32 API, также известный как Windows API, — это комплекс функций и интерфейсов, предложенный Microsoft для создания программ на Windows. Этот инструментарий дает программистам возможности управления окнами, обработки событий, доступа к файлам и системному реестру, а также многого другого. Win32 API ключевой для разработки на Windows, предоставляя детальный контроль над приложениями.[1]

Средства обмена данными в Windows, включая Windows Sockets (Winsock), облегчают взаимодействие между процессами и устройствами. Winsock используется для разработки сетевых приложений с поддержкой TCP/IP, позволяя приложениям взаимодействовать с веб-сервисами и другими сетевыми ресурсами. Механизмы многозадачности и потоков обеспечивают эффективное выполнение множества задач и используются для оптимизации работы приложений. Эти инструменты важны для создания функциональных и взаимодействующих приложений в Windows.

В Windows работа с многозадачностью и потоками является ключевым элементом для параллельного выполнения задач в рамках одного приложения. Это улучшает эффективность использования системных ресурсов и способствует быстродействию приложений. Таким образом, инструменты для обмена данными и организации совместного доступа играют важную роль в разработке программ на Windows, обеспечивая взаимодействие приложений с другими программами, устройствами и сетевыми ресурсами, и гарантируя при этом безопасность и эффективность обработки данных.

В процессе выполнения данной лабораторной работы, были применены следующие теоретические сведения:

1. Программирование сокетов в Windows (Winsock API): Использование функций socket, bind, listen, accept и connect для установления соединений TCP.
2. Многопоточность: Создание и управление потоками с помощью стандартной библиотеки C++ (std::thread) для одновременной обработки нескольких клиентских соединений.
3. Обработка сетевых событий: Прием и отправка данных через сокеты с использованием функций send и recv.
4. Использование структур сетевых адресов (sockaddr\_in): Установка параметров сетевого соединения, таких как IP-адрес и порт.
5. Управление ошибками сети: Использование WSAGetLastError для диагностики ошибок сети.

## 3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В ходе лабораторной работы было разработано клиент-серверное приложение для обмена текстовыми сообщениями. Результат работы показан на рис. 3.1.

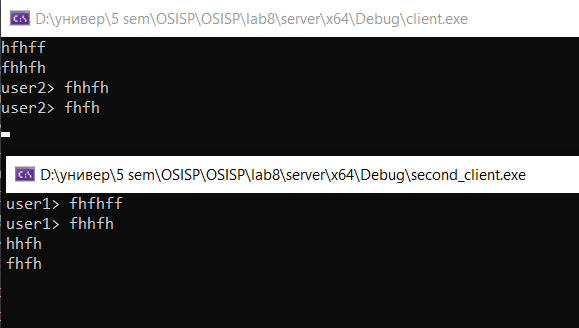


Рисунок 3.1 – Результат работы программы

## ВЫВОД

В ходе выполнения лабораторной работы была разработано клиент-серверное приложение для обмена текстовыми сообщениями с использованием TCP сокетов.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Щупак Ю. Win32 API. Разработка приложений для Windows. – СПб: Питер, 2008. – 592 с.: ип.
2. [Электронный ресурс]. – Режим доступа: [https://learn.microsoft.com/ru-ru/windows/win32/api – Дата доступа 23.10.2023](https://learn.microsoft.com/ru-ru/windows/win32/api%20–%20Дата%20доступа%2023.10.2023)
3. [Электронный ресурс]. – Режим доступа:  [https://learn.microsoft.com/ru-ru/cpp/parallel/multithreading-with-c-and-win32?view=msvc-170 – Дата доступа 24.10.2023](%20https://learn.microsoft.com/ru-ru/cpp/parallel/multithreading-with-c-and-win32?view=msvc-170%20–%20Дата%20доступа%2024.10.2023)

## ПРИЛОЖЕНИЕ А

**(обязательное)**

**Листинг кода**

﻿#include <WinSock2.h>

#include <iostream>

#include <thread>

#include <vector>

#include <string>

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

const int BUF\_SIZE = 4096;

std::vector<SOCKET> clientSockets; // Вектор для хранения сокетов клиентов

std::vector<std::thread> threads; // Вектор для хранения потоков

// Обработчик клиента

void handleClient(SOCKET clientSocket) {

char buffer[BUF\_SIZE];

while (true) {

ZeroMemory(buffer, BUF\_SIZE);

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

if (bytesReceived <= 0) {

// Обработка ошибок или закрытие соединения

break;

}

// Пересылка сообщения другим клиентам

for (const auto& otherClientSocket : clientSockets) {

if (otherClientSocket != clientSocket) {

send(otherClientSocket, buffer, bytesReceived, 0);

}

}

}

closesocket(clientSocket);

}

int main() {

WSADATA wsaData;

SOCKET serverSocket;

sockaddr\_in serverAddr;

sockaddr\_in clientAddr;

int clientAddrSize = sizeof(clientAddr);

WSAStartup(MAKEWORD(2, 2), &wsaData);

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

serverAddr.sin\_family = AF\_INET;

serverAddr.sin\_port = htons(54000);

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

bind(serverSocket, (sockaddr\*)&serverAddr, sizeof(serverAddr));

listen(serverSocket, 2);

// Принимаем два соединения и создаем для них потоки

for (int i = 0; i < 2; i++) {

SOCKET clientSocket = accept(serverSocket, (sockaddr\*)&clientAddr, &clientAddrSize);

if (clientSocket == INVALID\_SOCKET) {

std::cerr << "Error accepting client: " << WSAGetLastError() << std::endl;

continue;

}

clientSockets.push\_back(clientSocket);

threads.push\_back(std::thread(handleClient, clientSocket));

}

// Ожидаем завершения работы потоков

for (auto& t : threads) {

if (t.joinable()) {

t.join();

}

}

closesocket(serverSocket);

WSACleanup();

return 0;

}

﻿#include <WinSock2.h>

#include <ws2tcpip.h>

#include <iostream>

#include <thread>

#include <string>

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

const int BUF\_SIZE = 4096;

// Функция для обработки входящих сообщений

void receiveMessages(SOCKET clientSocket) {

char buffer[BUF\_SIZE];

while (true) {

ZeroMemory(buffer, BUF\_SIZE);

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

if (bytesReceived > 0) {

std::cout << "user2> " << std::string(buffer, 0, bytesReceived) << std::endl;

}

// В реальном приложении здесь должна быть проверка на ошибки и закрытие соединения при необходимости

}

}

int main() {

WSADATA wsaData;

SOCKET clientSocket;

sockaddr\_in serverAddr;

// Инициализация Winsock

WSAStartup(MAKEWORD(2, 2), &wsaData);

// Создание клиентского сокета

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

// Задание адреса и порта сервера

serverAddr.sin\_family = AF\_INET;

serverAddr.sin\_port = htons(54000); // Порт сервера

inet\_pton(AF\_INET, "127.0.0.1", &serverAddr.sin\_addr); // IP сервера

// Подключение к серверу

connect(clientSocket, (sockaddr\*)&serverAddr, sizeof(serverAddr));

// Запуск потока для приема сообщений

std::thread receiveThread(receiveMessages, clientSocket);

// Основной цикл для отправки сообщений

std::string userInput;

while (true) {

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

if (userInput == "exit") {

break; // Выход из цикла, если пользователь ввел "exit"

}

send(clientSocket, userInput.c\_str(), userInput.size() + 1, 0);

}

// Закрытие потока и сокета

receiveThread.detach();

closesocket(clientSocket);

WSACleanup();

return 0;

}