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 №7

на тему

Средства обмена данными (Windows). Изучение и использование средств обмена данными и совместного доступа

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 средствах обмена данными *Windows*, научиться использовать средства обмена данными и совместного доступа. Для достижения цели будет создано приложение для обмена текстовыми сообщениями между клиентами по локальной сети с использованием сокетов.

1. **ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Средства обмена данными и совместного доступа в операционной системе *Windows* предоставляют разработчикам средства для обеспечения взаимодействия между различными процессами и потоками, а также средства синхронизации для безопасного доступа к ресурсам. Ниже представлены основные сведения по этой теме.

Процесс – это изолированная программа, которая выполняется в собственном виртуальном адресном пространстве. Процессы не могут напрямую обмениваться данными.

Поток – это наименьшая единица выполнения внутри процесса. Потоки в одном процессе могут легко обмениваться данными.

События позволяют одному потоку/процессу уведомить другой поток/процесс о наступлении определенного события. В *Windows*, события могут быть использованы для синхронизации и сигнализации между потоками.

Семафоры – это счетчики, которые можно использовать для контроля доступа к общим ресурсам. Они позволяют ограничить количество потоков, которые могут одновременно получить доступ к ресурсу.

Критическая секция – это участок кода, который может быть выполнен только одним потоком одновременно. Она может быть использована для защиты общих ресурсов.

Мьютексы – это объекты синхронизации, которые могут быть использованы для обеспечения взаимного исключения между потоками.

Windows предоставляет различные механизмы межпроцессного взаимодействия, включая именованные каналы, сокеты, *COM* (*Component Object* *Model*) и другие.

Именованные каналы позволяют процессам обмениваться данными через именованные каналы. Это эффективный способ *IPC* для локальных и удаленных соединений.

Сокеты обеспечивают сетевой уровень межпроцессного взаимодействия и могут использоваться для обмена данными между процессами на разных компьютерах.

*Windows* предоставляет средства синхронизации, такие как критические секции и мьютексы, для обеспечения безопасного совместного доступа к общим ресурсам, таким как файлы, разделяемая память и другие объекты.

Разделяемая память позволяет нескольким процессам совместно использовать участок памяти для обмена данными.

Файловые мапы позволяют отображать файлы в память процесса для совместного доступа.

В *Windows*, существует множество объектов синхронизации, таких как события, семафоры и мьютексы, которые могут быть использованы для синхронизации действий различных процессов и потоков.

Разработчики могут использовать *API Windows* для создания и управления потоками, процессами, средствами синхронизации и межпроцессного взаимодействия.

Документация и примеры кода *Windows SDK* предоставляют информацию о работе с средствами обмена данными и совместного доступа в приложениях под управлением *Windows*.

Важно следить за соблюдением правил синхронизации и обеспечения безопасности при разработке приложений, чтобы избежать состояний гонки и других проблем, связанных с совместным доступом к ресурсам.

Обмен данных и совместный доступ – важные аспекты при разработке приложений под *Windows*, и знание средств и методов, предоставляемых операционной системой, позволяет разработчикам создавать стабильные и масштабируемые приложения.

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).

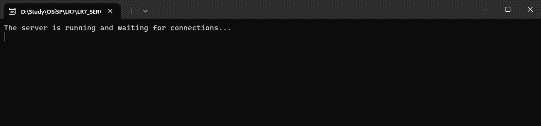


Рисунок 1 – Серверное окно приложения

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

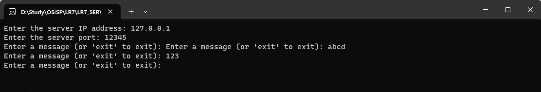


Рисунок 2 – Клиентское окно приложения

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

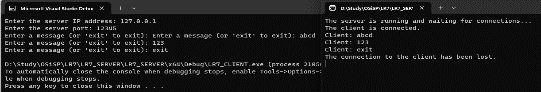


Рисунок 3 – Взаимодействие между серверной и клиентской частью

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.

## ЗАКЛЮЧЕНИЕ

В данной лабораторной работе были изучены и применены основы сетевого программирования и средства обмена данными в операционной системе Windows. Процесс разработки клиент-серверного приложения для обмена текстовыми сообщениями с использованием TCP сокетов позволил ознакомиться с ключевыми аспектами сетевого взаимодействия и многозадачности.

Изучение и использование функций, предоставляемых Winsock API для создания сокетов клиента и сервера.

Реализация серверной части приложения, включая привязку к определенному IP-адресу и порту, прослушивание подключений и обработку входящих запросов от клиентов.

Разработка клиентской части приложения, которая взаимодействует с сервером, подключаясь к нему по указанному IP-адресу и порту.

Реализация многопоточности для обработки нескольких клиентов одновременно, что позволяет серверу эффективно обслуживать множество подключений.

Применение средств синхронизации, таких как критические секции и события, для обеспечения безопасного совместного доступа к ресурсам при обмене данными между клиентами и сервером.

Разработка приложения для обмена текстовыми сообщениями, что представляет собой пример простого сетевого протокола.

Заключение по данной лабораторной работе подчеркивает важность понимания и применения средств сетевого программирования, средств синхронизации и совместного доступа, а также навыков работы с клиент-серверными приложениями в Windows. Полученные знания и опыт полезны при разработке сетевых приложений, а также при создании простых сетевых протоколов и обмене данными через сеть.

# 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 Lab7\_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 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;

}

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;

}