Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Операционные системы и системное программирование

Отчёт

к лабораторной работе №7

на тему

Средства обмена данными (Windows). Изучение и использованием средств

обмена данными и совместного доступа.

Выполнил: студент гр.153502 Кирзнер А.П.

Проверил: Гриценко Н.Ю.

Минск 2023

**СОДЕРЖАНИЕ**

1 Теоретические сведения 3

2 Результат выполнения программы 4

Список использованных источников 6

Приложение А 7

# 1 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Операционная система *Windows* предоставляет механизмы для облегчения связи и обмена данными между приложениями. В совокупности действия, выполняемые этими механизмами, называются межпроцессными коммуникациями (*IPC*). Некоторые формы *IPC* облегчают разделение труда между несколькими специализированными процессами. Другие формы *IPC* облегчают разделение труда между компьютерами в сети.

Обычно приложения могут использовать *IPC*, классифицированные как клиенты или серверы. Клиент – это приложение или процесс, который запрашивает услугу у какого-либо другого приложения или процесса. Сервер – это приложение или процесс, который отвечает на запрос клиента. Многие приложения действуют и как клиент, и как сервер, в зависимости от ситуации. Например, приложение для обработки текста может выступать в качестве клиента при запросе сводной таблицы производственных затрат у приложения для работы с электронными таблицами, действующего в качестве сервера. Приложение для работы с электронными таблицами, в свою очередь, может выступать в качестве клиента, запрашивая последние уровни запасов из приложения автоматического управления запасами.

Windows поддерживает следующие механизмы *IPC*:

– буфер обмена;

– КОМ;

– Копирование данных;

– ДДЕ;

– Сопоставление файлов;

– Почтовые ящики;

– Трубы;

– ПКП;

– сокеты *Windows*.

*Windows Sockets* – это независимый от протокола интерфейс. Он использует преимущества коммуникационных возможностей базовых протоколов.

Сокеты Windows основаны на сокетах, впервые популяризированных *Berkeley Software Distribution* (*BSD*). Приложение, использующее сокеты *Windows*, может взаимодействовать с другой реализацией сокета в системах других типов. Однако не все поставщики транспортных услуг поддерживают все доступные варианты.

*Windows Sockets* – это независимый от протокола интерфейс, способный поддерживать текущие и новые сетевые возможности [1].

# 2 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ

В рамках данной лабораторной работы было создано приложение, позволяющее обмениваться сообщениями между подключёнными пользователями. Для этого было создано два проекта: сервер и клиент. Для демонстрации выполнения работы необходимо запустить один сервер и минимум два клиента.

При запуске приложения-сервера при успешном создании и настройки сокета выводится сообщение о том, что сервер запущен. В противном случае программа выведет сообщение об ошибке и завершит свою работу. Во время работы сервера выводятся сообщения о подключении пользователей и сообщения с ником пользователей, которое данное сообщение отправили (рисунок 1.1).

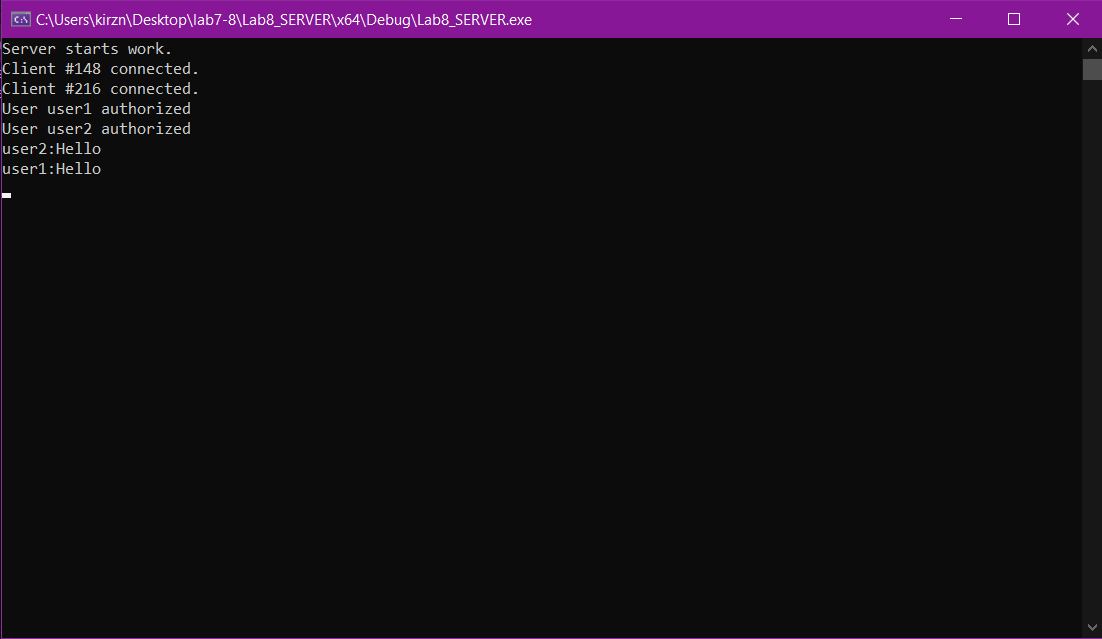


Рисунок 1.1 – Вывод программы во время работы приложения-сервера

При запуске приложения-клиента программа первоочерёдно пытается подключится к запущенному серверу. В случае возникновения ошибки подключения программа выведет сообщение об ошибке и завершит своё выполнение. Если же подключение удалось установить, то выводится соответствующее сообщение, а также запрашивается ник, по которому другие пользователи смогут идентифицировать его в общем чате. После ввода ника приложение переходит в режим прослушивания: при наборе текста и нажатии кнопки «*Enter*» сообщение отправляется всем пользователям, кроме того, от кого было отправлено сообщение, а также при получении нового сообщения оно выводится вместе с ником того человека, который его отправил (рисунок 1.2).

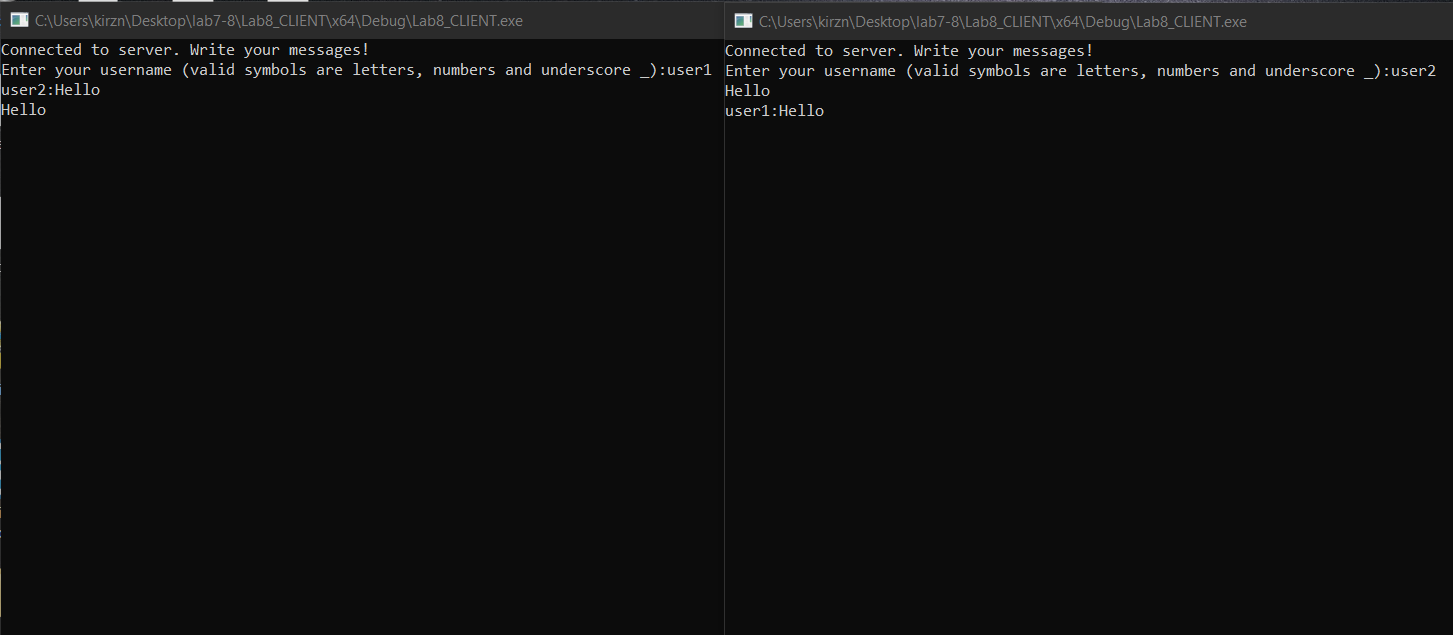


Рисунок 1.2 – Вывод программ во время работы приложений-клиентов

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Межпроцессные коммуникации [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/en-us/windows/win32/ipc/interprocess-communications#using-dde-for-ipc>.

.

# ПРИЛОЖЕНИЕ А

**(обязательное)**

**Исходный код программы**

**Файл server.cpp**

#include <iostream>

#include <WinSock2.h>

#include <WS2tcpip.h>

#include <thread>

#include <sstream>

#include <list>

#include <map>

WSAData data;

addrinfo hints;

addrinfo\* serverInfo = nullptr;

SOCKET listenSocket;

bool itWorks = true;

std::map<SOCKET, std::string> connectionsInfo;

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

void setHints()

{

ZeroMemory(&hints, sizeof(hints));

hints.ai\_family = AF\_INET;

hints.ai\_socktype = SOCK\_STREAM;

hints.ai\_protocol = IPPROTO\_TCP;

hints.ai\_flags = AI\_PASSIVE;

}

void clear(const std::string& message, addrinfo\* info = nullptr, SOCKET\* socket = nullptr)

{

std::cout << message;

if (info) {

freeaddrinfo(info);

}

if (socket) {

closesocket(\*socket);

}

}

void sendMessageToClients(const char\* message, SOCKET from)

{

for (auto& pair : connectionsInfo) {

if (pair.first != from) {

send(pair.first, message, strlen(message), 0);

}

}

}

void receiveMessageFromClient(SOCKET client)

{

char receivebuffer[256];

ZeroMemory(receivebuffer, 256);

while (true) {

ZeroMemory(receivebuffer, 256);

if (recv(client, receivebuffer, 256, 0) > 0) {

if (connectionsInfo[client] == "") {

connectionsInfo[client] = receivebuffer;

std::cout << "User " << connectionsInfo[client] << " authorized\n";

}

else {

std::stringstream ss;

ss << connectionsInfo[client] << ':' << receivebuffer << '\n';

std::cout << ss.str();

sendMessageToClients(ss.str().c\_str(), client);

}

}

else {

std::cout << "Connection closed or error occured when receiving from " << connectionsInfo[client] << "\n";

return;

}

std::this\_thread::sleep\_for(std::chrono::milliseconds(50));

}

}

int main()

{

int result = WSAStartup(MAKEWORD(2, 2), &data);

if (result != 0) {

std::cout << "Startup failed with code " << result << '\n';

return 1;

}

setHints();

result = getaddrinfo(NULL, "80", &hints, &serverInfo);

if (result != 0) {

std::stringstream ss;

ss << "getaddrinfo error. Code : " << result << '\n';

clear(ss.str());

return 0;

}

listenSocket = socket(serverInfo->ai\_family, serverInfo->ai\_socktype, serverInfo->ai\_protocol);

if (listenSocket == INVALID\_SOCKET) {

clear("Connection ListenSocket failed.\n", serverInfo);

return 0;

}

result = bind(listenSocket, serverInfo->ai\_addr, serverInfo->ai\_addrlen);

if (result == SOCKET\_ERROR) {

clear("binding socket failed.\n", serverInfo, &listenSocket);

return 0;

}

result = listen(listenSocket, SOMAXCONN);

if (result == SOCKET\_ERROR) {

clear("listen failed.\n", serverInfo, &listenSocket);

return 0;

}

std::cout << "Server starts work.\n";

SOCKET socket;

while (true) {

socket = accept(listenSocket, nullptr, nullptr);

if (socket != INVALID\_SOCKET) {

connectionsInfo[socket] = "";

std::cout << "Client #" << socket << " connected.\n";

std::thread read\_thread(&receiveMessageFromClient, socket);

read\_thread.detach();

}

std::this\_thread::sleep\_for(std::chrono::milliseconds(50));

}

for (auto& pair : connectionsInfo) {

closesocket(pair.first);

}

WSACleanup();

}

**Файл client.cpp**

#include <iostream>

#include <WinSock2.h>

#include <WS2tcpip.h>

#include <thread>

#include <sstream>

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

WSAData data;

addrinfo hints;

addrinfo\* serverInfo = nullptr;

SOCKET connectSocket;

std::thread receive\_thread;

void setHints()

{

ZeroMemory(&hints, sizeof(hints));

hints.ai\_family = AF\_INET;

hints.ai\_socktype = SOCK\_STREAM;

hints.ai\_protocol = IPPROTO\_TCP;

}

bool validateUsername(const std::string& s)

{

for (const char c : s) {

if (!isalpha(c) && !isdigit(c) && c != '\_')

return false;

}

return true;

}

void sendMessages() {

std::string username;

bool valid\_username = false;

while (!valid\_username) {

std::cout << "Enter your username (valid symbols are letters, numbers and underscore \_):";

std::cin >> username;

valid\_username = validateUsername(username);

if (!valid\_username) {

std::cout << "\033[1;31mInvalid username!\033[0m\n";

}

}

send(connectSocket, username.c\_str(), username.length(), 0);

std::string message;

while (true) {

std::cin >> message;

if (send(connectSocket, message.c\_str(), message.length(), 0) == SOCKET\_ERROR) {

std::cout << "Error occured when seding message to server.";

}

}

}

std::string colors[] = {"\n\033[1;32m", "\n\033[1;33m", "\n\033[1;34m", "\n\033[1;35m", "\n\033[1;36m"};

std::string getColor() {

int random = rand() % 5;

return colors[random];

}

void receiveMessageFromServer()

{

char receivebuffer[256];

while (true) {

ZeroMemory(receivebuffer, 256);

if (recv(connectSocket, receivebuffer, 256, 0) != SOCKET\_ERROR)

{

std::cout << receivebuffer;

}

}

}

void clear(const std::string& message, addrinfo\* info, SOCKET\* socket = nullptr)

{

std::cout << message << "\n";

if (info)

freeaddrinfo(info);

if (socket)

closesocket(\*socket);

WSACleanup();

}

int main() {

int res = WSAStartup(MAKEWORD(2, 2), &data);

if (res != 0)

{

std::cout << "Startup failed with code " << res << '\n';

return 1;

}

setHints();

res = getaddrinfo("127.0.0.1", "80", &hints, &serverInfo);

if (res != 0)

{

clear("getaddrinfo error. Code : " + std::to\_string(res) + "\n", nullptr);

return false;

}

connectSocket = socket(serverInfo->ai\_family, serverInfo->ai\_socktype, serverInfo->ai\_protocol);

if (connectSocket == INVALID\_SOCKET)

{

clear("Connection ConnectSocket failed.\n", serverInfo);

return 1;

}

res = connect(connectSocket, serverInfo->ai\_addr, serverInfo->ai\_addrlen);

if (res == SOCKET\_ERROR)

{

clear("Unable connect to server.\n", serverInfo, &connectSocket);

return false;

}

std::cout << "Connected to server. Write your messages!\n";

std::thread sendMessagesThread(&sendMessages);

std::thread receiveMessagesThread(&receiveMessageFromServer);

sendMessagesThread.join();

receiveMessagesThread.join();

clear("", serverInfo, &connectSocket);

WSACleanup();

}