Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп'ютерних технологій, автоматики та метрології Спеціальність «Комп’ютерна Інженерія»

Кафедра СКС



**Звіт**

з лабораторної роботи №5

з дисципліни «Системне програмне забезпечення»

на тему «Робота з мережевими вводом-виводом, сокети»

**Варіант №3**

Виконав:

студент групи KI-307

Ващишин І.І.

Прийняв:  
Олексів М.В.

ЛЬВІВ 2025

**Мета:** Навчитися розробляти програми керування мережними засобами операційних систем.

**Теоретичні відомості**

1. Загальні принципи мережної підтримки

**Мережа** - набір комп'ютерів або апаратних пристроїв (вузлів), пов'язаних між собою каналами зв'язку, які можуть передавати інформацію один одному. Мережа має конкретну фізичну структуру (топологію), усі вузли підключають до мережі із використанням апаратного забезпечення. Мережа об'єднує обмежену кількість вузлів.

Під **інтернетом** (з малої літери) розуміють сукупність мереж, які використовують один і той самий набір мережних протоколів — правил, що визначають формат даних для пересилання мережею. Фізична структура окремих мереж, які входять до складу інтернету, може різнитися. Такі різнорідні мережі пов'язують одну з одною маршрутизатори, які переадресовують пакети з однієї мережі в іншу, залежно від їхньої адреси призначення і при цьому перетворюють пакети між форматами відповідних мереж. Маршрутизатори підтримують міжмережну взаємодію.

**Інтернет** (з великої літери)— це сукупність пов'язаних між собою інтернетів, відкритих для публічного доступу, які використовують визначений набір протоколів і охоплюють увесь світ.

* 1. Рівні мережної архітектури і мережні сервіси

Функції забезпечення зв'язку між вузлами є складними. Для спрощення їхньої реалізації використовують багаторівневий підхід - вертикальний розподіл мережних функцій і можливостей. Він дає змогу приховувати складність реалізації функцій зв'язку: кожен рівень приховує від вищих рівнів деталі реалізації своїх функцій та функцій, реалізованих на нижчих рівнях.

**Мережний сервіс** — це набір операцій, які надає рівень мережної архітектури для використання на вищих рівнях. Сервіси визначено як частину специфікації інтерфейсу рівня.

Розрізняють сервіси, **орієнтовані на з'єднання і без з'єднань, або дейтаграмні сервіси**:

**Сервіси, орієнтовані на з'єднання**, реалізують три фази взаємодії із верхнім рівнем: встановлення з'єднання, передавання даних і розрив з'єднання. При цьому передавання даних на верхніх рівнях здійснюють у вигляді неперервного потоку байтів.

**Дейтаграмні** сервіси реалізують пересилання незалежних повідомлень, які можуть переміщатися за своїми маршрутами і приходити у пункт призначення в іншому порядку.

**Завдання:**

Розробити програми роботи керування мережними засобами ОС. Для програмної реалізації задач використовувати середовище програмування   
Visual Studio. Програми розробляти у консольному режимі.

**Код програм:  
Код клієнта:**  
#include <iostream>

#include <winsock2.h>

#include <ws2tcpip.h>

#include <string>

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

const int PORT = 77777;

const std::string SERVER\_ADDRESS = "127.0.0.1";

int main() {

WSADATA wsData;

if (WSAStartup(MAKEWORD(2, 2), &wsData) != 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;

serverAddress.sin\_port = htons(PORT);

if (inet\_pton(AF\_INET, SERVER\_ADDRESS.c\_str(), &(serverAddress.sin\_addr)) <= 0) {

std::cerr << "An error occurred while converting the server address" << std::endl;

closesocket(clientSocket);

WSACleanup();

return 1;

}

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

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

closesocket(clientSocket);

WSACleanup();

return 1;

}

std::cout << "Connected to the server" << std::endl;

std::string input\_payload;

std::cout << "Enter array size: ";

std::getline(std::cin, input\_payload);

// Відправляємо рядки на сервер

send(clientSocket, input\_payload.c\_str(), input\_payload.size(), 0);

// Отримуємо результат від сервера

char buffer[1024];

int bytesReceived = recv(clientSocket, buffer, 1024, 0);

if (bytesReceived == SOCKET\_ERROR) {

std::cerr << "Error in receiving data from server" << std::endl;

}

else {

buffer[bytesReceived] = '\0';

std::cout << "Result from server: " << buffer << std::endl;

}

closesocket(clientSocket);

WSACleanup();

return 0;

}

**Код сервера:**  
#include <iostream>

#include <string>

#include <winsock2.h>

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

const int PORT = 77777;

int findMax(int \*arr, int size) {

int max = arr[0];

for (int i = 1; i < size; i++) {

if (max < arr[i])

max = arr[i];

}

return max;

}

bool isInteger(const std::string &str) {

for (char c : str) {

if (!isdigit(c) && c != '+' && c != '-') {

return false;

}

}

return true;

}

int main() {

WSADATA wsData;

if (WSAStartup(MAKEWORD(2, 2), &wsData) != 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(PORT);

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

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

SOCKET\_ERROR) {

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

closesocket(serverSocket);

WSACleanup();

return 1;

}

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

std::cerr

<< "An error occurred when setting standby mode for connections"

<< std::endl;

closesocket(serverSocket);

WSACleanup();

return 1;

}

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

while (true) {

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

if (clientSocket == INVALID\_SOCKET) {

std::cerr << "Error receiving connection" << std::endl;

closesocket(serverSocket);

WSACleanup();

return 1;

}

std::cout << "Connected to client" << std::endl;

char buffer[1024];

std::string str1;

// Отримуємо рядки від клієнта

int bytesReceived = recv(clientSocket, buffer, 1024, 0);

if (bytesReceived == SOCKET\_ERROR) {

std::cerr << "Error in receiving data from client" << std::endl;

} else {

buffer[bytesReceived] = '\0';

str1 = buffer;

}

std::cout << "Received strings from client: " << str1 << std::endl;

std::string result;

if (isInteger(str1) || std::stoi(str1) < 0) {

int size = std::stoi(str1);

int half = size;

size \*= size;

int \*dynamicArray = (int \*)malloc(size \* sizeof(int));

std::cout << "array is:\n";

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

dynamicArray[i] = rand() % 100;

std::cout << dynamicArray[i] << " ";

if (i % half == half - 1)

std::cout << "\n";

}

int max = findMax(dynamicArray, size);

std::cout << "\nmax is: " << max << "\n";

result = std::to\_string(max);

} else {

result = "please, provide valid number, that greater than 0";

}

send(clientSocket, result.c\_str(), result.size(), 0);

closesocket(clientSocket);

std::cout << "The connection with the client is closed" << std::endl;

}

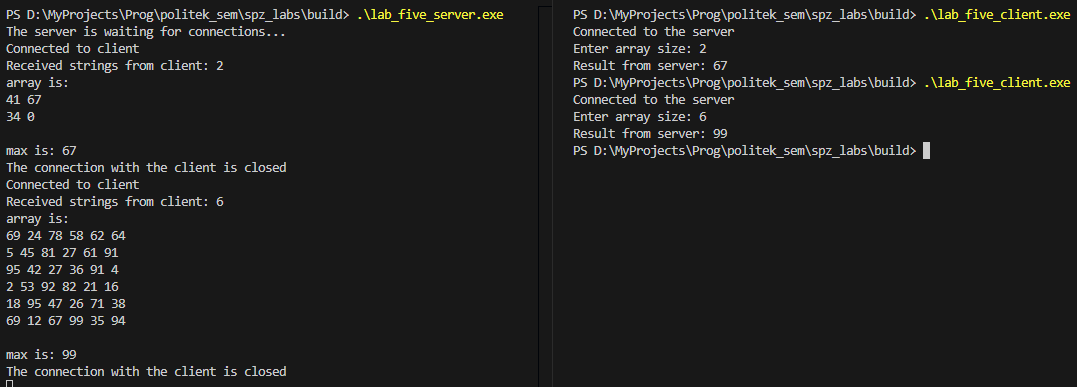
closesocket(serverSocket);

WSACleanup();

return 0;

}

**Результат виконання програми:**

**Висновок:** під час виконання даної лабораторної роботи я навчився розробляти програми керування мережними засобами операційних систем.