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

ОТЧЁТ

к лабораторной работе №5

на тему

ЭЛЕМЕНТЫ СЕТЕВОГО ПРОГРАММИРОВАНИЯ

Выполнил: студент гр.253505 Косяков М.М.

Проверил: ассистент кафедры информатики Гриценко Н.Ю.

Минск 2024

**СОДЕРЖАНИЕ**

[1 Формулировка задачи 3](#_Toc178093096)

[2 Краткие теоритические сведения 4](#_Toc178093097)

[3 Описание функций программы 5](#_Toc178093098)

[3.1 Ввод состояния для фильтрации 5](#_Toc178093099)

[3.2 Вывод портов 5](#_Toc178093100)

[Заключение 6](#_Toc178093101)

[Список использованных источников 7](#_Toc178093102)

[Приложение А (обязательное) Исходный код программы 8](#_Toc178093103)

1 ФОРМУЛИРОВКА ЗАДАЧИ

В рамках этой лабораторной работы ставлю перед собой цель углубить и закрепить навыки работы с элементами сетевого программирования в операционной системе Windows. В ходе выполнения работы необходимо изучить основы построения и функционирования сетей, стеков протоколов и программных интерфейсов.

Особое внимание будет уделено организации взаимодействия между процессами в сетях, которая является одной из основных задач взаимодействия процессов. Также предстоит изучить работу с иерархической моделью взаимодействия открытых систем, уровнями, протоколами и интерфейсами. Предстоит ознакомиться с программным интерфейсом сокетов, типами, структурами данных, функциями API.

Для выполнения лабораторной работы по созданию утилиты, использующей сетевые протоколы необходимо реализовать приложение, которое будет выводить порты, соответствующие заданному состоянию. Функциональность достаточна минимальная для демонстрации, интерфейс произвольный. Программа будет написана на языке C с использованием WinAPI.

В качестве задачи необходимо реализовать аналог сетевой утилиты, а для этого:

– ввести состояние для фильтрации;

– реализовать вывод портов, соответствующие заданному состоянию.

В результате выполнения этой лабораторной работы будут не только получены теоретические знания работы с элементами сетевого программирования в операционной системе Windows, но и практический опыт в разработке приложений, которые могут эффективно использовать ресурсы компьютера, а также корректно и безопасно взаимодействовать с сетью. Понимание концепций построения и функционирования сетей, стеков протоколов, программных интерфейсов, сокетов станет необходимым для создания высокопроизводительных программных решений.

2 КРАТКИЕ ТЕОРИТИЧЕСКИЕ СВЕДЕНИЯ

Интернет— компьютерная сеть, объединяющая миллионы компьютеров в единую информационную систему [1].

Физический канал — это реальный канал связи, который представляет собой физическую инфраструктуру (например, провода, оптоволокно, беспроводные средства связи), через который передаются данные между устройствами.

Виртуальный канал — это абстрактный канал передачи данных, который создается на основе физического канала. Виртуальный канал обладает своими свойствами и характеристиками, которые могут отличаться от физического канала. Виртуальный канал передачи данных имитирует файл или поток ввода-вывода для прикладных программ.

Коммутация — распределение имеющихся физических каналов для создания виртуальных.

Маска подсети— битовая маска для определения по IP-адресу адреса подсети и адреса узла этой подсети [2].

Порт— это сетевая абстракция, представляющая собой число, записываемое в заголовках протоколов транспортного уровня модели OSI.

Топологией компьютерной сети называют физическое расположение компьютеров сети друг относительно друга и способ соединения их линиями связи.

Стек протоколов в сети — набор протоколов, обслуживающих различные уровни взаимодействия.

Модель OSI — сетевая модель стека сетевых протоколов OSI/ISO. Посредством данной модели различные сетевые устройства могут взаимодействовать друг с другом. Модель определяет различные уровни взаимодействия систем. Каждый уровень выполняет определённые функции при таком взаимодействии.

Модель TCP/IP также является многоуровневой сетевой моделью, но это четырехуровневая модель. IP— маршрутизируемый протокол сетевого уровня стека TCP/IP.Объединяет сегменты сети в единую сеть, обеспечивая доставку пакетов данных между любыми узлами сети через произвольное число промежуточных узлов (маршрутизаторов). TCP— один из основных протоколов передачи данных. Пакеты в TCP называются сегментами. Реализации TCP обычно встроены в ядра ОС. UDP— более простой (и менее надежный) протокол передачи данных.С UDP компьютерные приложения могут посылать сообщения другим хостам по IP-сети без необходимости предварительного сообщения для установки специальных путей передачи данных.

3 ОПИСАНИЕ ФУНКЦИЙ ПРОГРАММЫ

Согласно формулировке задачи, были спроектированы следующие функции программы [3]:

– ввод состояния для фильтрации;

– вывод портов;

– завершение работы программы;

**3.1 Ввод состояния для фильтрации**

Для начала обработки программы следует ввести состояние для фильтрации портов (рисунок 3.1).

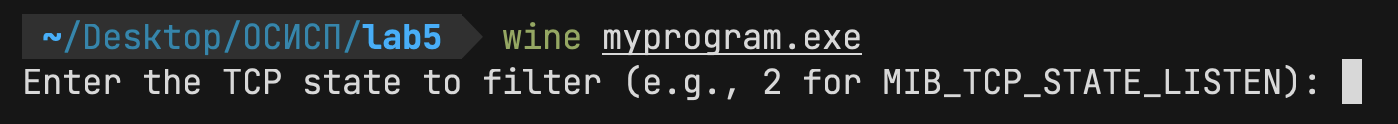


Рисунок 3.1 – Ввод состояния для фильтрации

**3.2 Вывод портов**

Программа использует Windows Sockets 2 API и IP Helper API для получения и вывода списка TCP-портов, находящихся в состоянии прослушивания (listening ports) в зависимости от указанного фильтра состояния TCP. После инициализации Winsock программа запрашивает у пользователя состояние TCP для фильтрации, а затем выводит порты, соответствующие заданному состоянию. В конце работы освобождаются выделенные ресурсы и закрывается Winsock (рисунок 3.2).

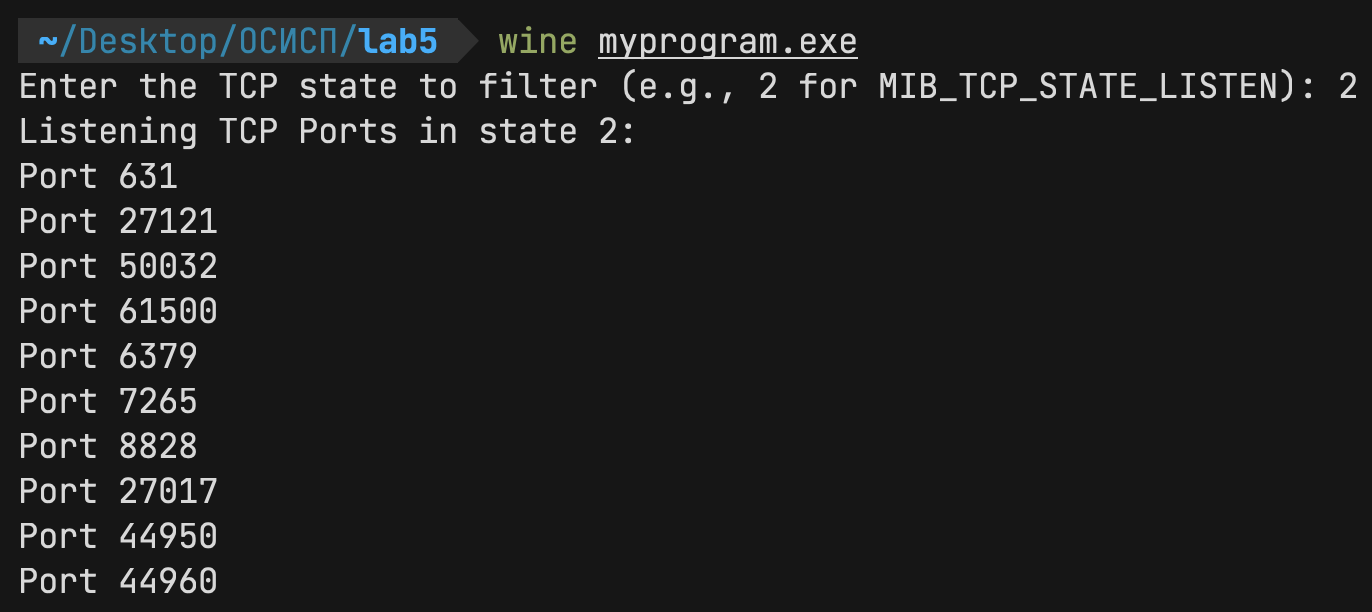


Рисунок 3.2 – Сообщение о выводе портов

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной лабораторной работы были изучены и закреплены навыки программирования приложений для операционной системы Windows с использованием элементов сетевого программирования. Мы погрузились в основы построения и функционирования сетей, стеков протоколов, программных интерфейсов и научились их реализовывать в среде Windows с помощью WinAPI.

Основными аспектами, с которыми мы ознакомились, стали сеть, протоколы, модель OSI, интерфейсы, сокеты. Мы также освоили типичные проблемы, возникающие при организации взаимодействия между процессами в сетях, модели для их описания и пути их решения, что сыграет важную роль в разработке производительных приложений. Изучили принципы работы с иерархической моделью взаимодействия открытых систем, уровнями, протоколами и интерфейсами.

В ходе работы над созданием утилиты, использующей сетевые протоколы, мы реализовали приложение, которое выводит порты, соответствующие заданному состоянию.

Эта лабораторная работа не только позволила углубить наши теоретические знания в области взаимодействия процессов, но и предоставила ценный практический опыт в разработке эффективных приложений, способных эффективно использовать ресурсы компьютера.

Понимание концепций построения и функционирования сетей, стеков протоколов, программных интерфейсов, сокетов, а также навыки управления ими стали ключевыми в создании высокопроизводительных программных решений.

Выполнение этой лабораторной работы позволило нам не только расширить нашу базу знаний, но и приобрести практические навыки, которые будут полезны в дальнейшей разработке программного обеспечения для Windows.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Build desktop Windows apps using the Win32 API [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/en-us/windows/win32/

[2] Основные сообщения ОС Windows (Win32 API). Программирование в ОС Windows. Лекция 1. – Режим доступа: https://www.youtube.com/watch?v=wTArIolxch0

[3] Разработка приложений с помощью WinAPI. – Режим доступа: https://shorturl.at/BDJW8

ПРИЛОЖЕНИЕ А

(обязательное)

**Исходный код программы**

#include <stdio.h>

#include <winsock2.h> // Windows Sockets 2 API

#include <iphlpapi.h> // IP Helper API

#pragma comment(lib, "iphlpapi.lib") // Link IP Helper API library

#pragma comment(lib, "ws2\_32.lib") // Link Windows Sockets 2 library

// Function to print listening ports based on a filter state

void printListeningPorts(int filterState)

{

PMIB\_TCPTABLE2 pTcpTable;

DWORD dwSize = 0;

DWORD dwRetVal;

pTcpTable = (MIB\_TCPTABLE2 \*)malloc(sizeof(MIB\_TCPTABLE2)); // Allocate memory for TCP table

if (pTcpTable == NULL)

{

printf("Error allocating memory\n");

return;

}

// Retrieve TCP table to get listening ports

dwRetVal = GetTcpTable2(pTcpTable, &dwSize, TRUE);

if (dwRetVal == ERROR\_INSUFFICIENT\_BUFFER)

{

free(pTcpTable);

pTcpTable = (MIB\_TCPTABLE2 \*)malloc(dwSize);

if (pTcpTable == NULL)

{

printf("Error allocating memory\n");

return;

}

}

dwRetVal = GetTcpTable2(pTcpTable, &dwSize, TRUE);

if (dwRetVal != NO\_ERROR)

{

printf("GetTcpTable2 failed with %d\n", dwRetVal);

free(pTcpTable);

return;

}

printf("Listening TCP Ports in state %d:\n", filterState);

for (DWORD i = 0; i < pTcpTable->dwNumEntries; i++)

{

if (pTcpTable->table[i].dwState == filterState)

{

printf("Port %d\n", ntohs((u\_short)pTcpTable->table[i].dwLocalPort));

}

}

if (pTcpTable != NULL)

{

free(pTcpTable);

}

}

int main()

{

WSADATA wsaData;

if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0) // Initialize Winsock

{

printf("WSAStartup failed\n");

return 1;

}

int filterState;

printf("Enter the TCP state to filter (e.g., 2 for MIB\_TCP\_STATE\_LISTEN): ");

scanf("%d", &filterState);

printListeningPorts(filterState);

WSACleanup(); // Clean up Winsock

return 0;

}

// netstat -an | grep LISTEN