

Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Операционные системы и системное программирование

К ЗАЩИТЕ ДОПУСТИТЬ

_____ Д. В. Басак

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту на
тему

КРОСПЛАТФОРМЕННЫЙ СКАНЕР СЕТЕВЫХ ПОРТОВ

БГУИР КП 1-40 02 01 324 ПЗ

Студент: _____ группы 150503, Ходосевич М.А.

Руководитель: _____ ассистент каф. ЭВМ, Басак Д.В.

Минск 2023
Учреждение образования

«Белорусский государственный университет информатики и
радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ЭВМ

_____ Б. В. Никульшин

(подпись)

_____ 2023г.

ЗАДАНИЕ

по курсовому проектированию

Студенту Ходосевичу Матвею Александровичу

1. Тема проекта Кроссплатформенный сканер сетевых портов
2. Срок сдачи студентом законченного проекта 15 мая 2023 г.
3. Исходные данные к проекту Язык программирования C++. Операционная система Linux.
4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)
Введение. 1. Обзор литературы. 2. Системное проектирование. 3. Функциональное проектирование. 4. Разработка программных модулей. 5. Руководство пользователя. Заключение. Список использованных источников.
5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков) 1. Схема структурная 2. Схемы алгоритмов.
6. Консультат по проекту Басак Д.В.
7. Дата выдачи задания 18 февраля 2023 г.

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и трудоемкости отдельных этапов):

раздел 1 к 1 марта 2023 г. – 15 %;

разделы 2,3 к 1 апреля 2023 г. – 50 %;

разделы 4,5 к 1 мая 2023 г. – 80 %;

оформление пояснительной записки и графического материала к 15 мая 2023 г. – 100 %

Защита курсового проекта с 29 мая 2023 г. по 9 июня 2023 г.

РУКОВОДИТЕЛЬ

Д.В. Басак

(подпись)

Задание принял к исполнению

М.А.Ходосевич

(дата и подпись студента)

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ ИСПОЛЬЗУЕМЫХ СОКРАЩЕНИЙ	5
ВВЕДЕНИЕ	6
1 ОБЗОР ЛИТЕРАТУРЫ	8
1.1 Обзор аналогов	8
1.2 Постановка задачи	8
2 СИСТЕМОНОЕ ПРОЕКТИРОВАНИЕ	10
2.1 Блок настройки	10
2.2 Блок сканирования портов	10
2.3 Блок записи в файл	11
2.4 Структура выходных данных	11
3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ	12
3.1 Файл «main.cpp»	13
3.2 Файл «Makefile»	14
4 РАЗРАБОТКА ПРОГРАМНЫХ МОДУЛЕЙ	16
4.1 Алгоритм работы функции main():	16
4.2 Алгоритм работы функции scanOnePort(int portToCheck):	17
5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ	18
ЗАКЛЮЧЕНИЕ	21
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ	22
ПРИЛОЖЕНИЕ А	23
ПРИЛОЖЕНИЕ Б	24
ПРИЛОЖЕНИЕ В	25
ПРИЛОЖЕНИЕ Г	26
ПРИЛОЖЕНИЕ Д	27

ПЕРЕЧЕНЬ ИСПОЛЬЗУЕМЫХ СОКРАЩЕНИЙ

TCP – Transmission Control Protocol.

UDP – User Datagram Protocol.

Nmap – Network Mapper.

API – Application programming interface.

IDE – International Development Enterprises.

ВВЕДЕНИЕ

Сетевые порты являются ключевыми элементами в области сетевого взаимодействия. Они позволяют установить соединение между различными компьютерами и обеспечивают передачу данных между ними. В современных компьютерных системах использование сетевых портов является неотъемлемой частью множества приложений и сервисов.

Каждый компьютер в сети имеет множество портов, которые используются для обработки входящих и исходящих сетевых пакетов. Сетевые порты делятся на две категории: TCP и UDP. Протокол TCP обеспечивает надежную и упорядоченную передачу данных, а протокол UDP - более простую и быструю передачу без гарантий доставки и упорядочивания пакетов. Каждый протокол использует свой диапазон портов: TCP порты варьируются от 0 до 65535, а UDP порты - также от 0 до 65535. Понимание состояния и доступности сетевых портов имеет важное значение для администрирования сети, обеспечения безопасности и отладки сетевых приложений.

Программа сканирования сетевых портов, описанная в данной пояснительной записке, разработана для операционной системы Linux. Linux является широко распространённой и мощной операционной системой с открытым исходным кодом, которая обеспечивает надёжность, гибкость и безопасность в сетевых средах.

Программа использует возможности Linux для создания сокетов, установления соединений, отправки и приема данных по протоколу TCP. Операционная система Linux предоставляет обширный набор сетевых API и инструментов, которые позволяют программам взаимодействовать с сетевыми ресурсами, включая сканирование портов.

В целом, Linux является платформой выбора для разработки программы сканирования сетевых портов благодаря своим возможностям, надежности и безопасности. Операционная система предоставляет необходимые средства для реализации сканирования портов и обеспечения эффективной работы в сетевой среде.

Разработка данной программы сканирования сетевых портов была выполнена на языке программирования C++. C++ был выбран в качестве языка разработки из-за его преимуществ в области системного программирования и работы с сетевыми операциями. В C++ можно легко использовать функциональность, предоставляемую операционной системой Linux для создания сокетов, установления соединений и выполнения сетевых операций. Это делает C++ подходящим выбором для разработки сетевых приложений, включая программы сканирования портов.

Кроме того, в процессе разработки данной программы сканирования сетевых портов использовалась интегрированная среда разработки (IDE) под названием CLion, разработанная компанией JetBrains.

CLion предоставляет удобную среду для написания, отладки и сборки программ на языке C++. CLion является мощным инструментом разработки с широким набором функций, включая подсветку синтаксиса, автодополнение, статический анализ кода, интегрированную систему сборки, отладчик и другие инструменты, упрощающие процесс разработки и повышающие продуктивность разработчика.

1 ОБЗОР ЛИТЕРАТУРЫ

1.1 Обзор аналогов

При разработке программы сканирования сетевых портов, я обратился к различным источникам информации и использовал опыт существующих аналогичных программ. Некоторыми из них являются:

1. Nmap: Nmap - это мощный и распространенный инструмент для сканирования сетевых портов. Он предоставляет широкий набор функций для обнаружения открытых портов, определения сервисов, анализа сетевой инфраструктуры и обнаружения уязвимостей.

2. Netcat: Netcat (или nc) - это утилита командной строки, которая предоставляет возможность создания и управления сетевыми соединениями, включая сканирование портов. Она позволяет отправлять и получать данные через сетевые порты и может быть использована для проверки доступности портов на удаленных хостах.

3. Angry IP Scanner: Angry IP Scanner - это простой в использовании сканер сетевых портов и IP-адресов. Он предоставляет возможность сканировать порты на множестве хостов одновременно и выводить результаты в удобном виде.

Обращение к существующим аналогичным программам помогло мне получить представление о функциональности, возможностях и пользовательском опыте, которые я хотел реализовать в своей программе сканирования сетевых портов. При этом я стремился создать программу, которая сочетает простоту использования, гибкость и надежность для удовлетворения потребностей пользователей в области сетевого сканирования портов.

1.2 Постановка задачи

Программа должна иметь возможность сканировать определенный диапазон портов на удаленном хосте и определять, являются ли они открытыми или закрытыми. Это позволяет пользователям проверить доступность определенных портов на удаленном сервере.

Помимо обнаружения открытых и закрытых портов, программа должна выводить информацию о состоянии порта, указывая, является ли он свободным или занятым. Это позволяет пользователям определить, используется ли порт определенным сервисом или приложением.

Программа должна предоставлять пользователю возможность настройки различных параметров сканирования, таких как диапазон портов, IP-адрес удаленного хоста, использование TCP, а также возможность сохранения результатов сканирования в файл.

Цель программы состоит в том, чтобы предоставить пользователям удобный и надежный инструмент для сканирования сетевых портов, который поможет им проверить доступность определенных портов на удаленных хостах и получить информацию о состоянии портов в сети.

2 СИСТЕМОНОЕ ПРОЕКТИРОВАНИЕ

Данная глава посвящена системному проектированию моей программы сканера сетевых портов. Программу можно разделить на два основных блока: блок настройки и блок сканирования портов.

Блоки в программе могут быть связаны между собой для создания полноценного функционала сканера сетевых портов.

Блок настройки и блок сканирования: блок настройки может предоставлять пользователю возможность указать параметры сканирования, такие как диапазон портов. Затем эти настройки передаются в блок сканирования, который осуществляет фактическое сканирование портов с использованием указанных параметров.

Блок сканирования и блок выходных данных: блок сканирования может предоставлять результаты сканирования в блок выходных данных для отображения пользователю.

Возможность использовать блоки отдельно или в комбинации зависит от конкретных требований программы и функционала, который необходимо реализовать. В некоторых случаях можно использовать только один блок, например, если требуется только сканирование портов без записи результатов. Однако, для полноценного функционала и удобства использования, рекомендуется связывать и комбинировать различные блоки в программе.

Структурная схема представлена в приложении А.

2.1 Блок настройки

В данном блоке происходит настройка программы перед сканированием портов. Блок настройки выполняет чтение и обработку параметров, валидацию, настройку опций вывода и передачу параметров сканирования в блок сканирования портов. Он также обрабатывает результаты сканирования в соответствии с выбранными опциями.

2.2 Блок сканирования портов

В этом блоке реализуется сам процесс сканирования портов. Блок сканирования выполняет установку соединения с целевыми портами и анализирует результаты для определения состояния каждого порта (открыт или закрыт). Для каждого порта в заданном диапазоне производится подключение с использованием сокетов. Это позволяет проверить доступность порта. После подключения к порту производится проверка статуса подключения. Если подключение успешно, то порт считается свободным, иначе - занятым.

Полученные результаты сканирования каждого порта сохраняются для последующего отображения или записи в файл.

2.3 Блок записи в файл

Блок записи в файл является важной частью программы сканера сетевых портов, поскольку позволяет сохранить результаты сканирования для дальнейшего использования.

2.4 Блок выходных данных

Выходные данные программы сканера сетевых портов представляют собой результаты сканирования каждого порта. Возможны два статуса для каждого порта: свободный или занятый. Результаты могут быть выведены на экран в виде таблицы или сохранены в файл для дальнейшего использования.

3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

Для разработки программы сканера сетевых портов были использованы следующие библиотеки:

<iostream> и <string>: эти стандартные библиотеки C++ используются для обработки ввода-вывода и работы со строками. В программе используются для вывода сообщений, ввода значений, работы с текстовыми данными и преобразования чисел в строки и наоборот.

<netinet/in.h>: данная библиотека содержит определения структур и констант, необходимых для работы с сетевыми протоколами. В программе используется для настройки и работы с сетевыми адресами и портами. В частности, используется следующая структура и функции: struct sockaddr_in. Эта структура определяет адрес сокета для протокола IPv4. Она содержит поля, такие как sin_family для указания семейства адресов, sin_addr для указания IP-адреса и sin_port для указания порта.

<sys/socket.h>: данная библиотека содержит определения функций и структур для работы с сокетами. В программе используется для создания и управления сокетами. Некоторые из функций, используемых в программе: socket(). Функция socket () создает новый сокет с указанными параметрами домена, типа и протокола. В программе используется для создания сокета семейства AF_INET, типа SOCK_STREAM (TCP) для установления соединения. connect(): устанавливает соединение с удаленным хостом по указанному адресу. В программе используется для установки соединения с удаленным хостом с помощью созданного сокета. Также используется функция close(), которая закрывает сокет после работы с ним.

<arpa/inet.h>: эта библиотека содержит функции для преобразования IP-адресов и сетевых порядков данных. В программе используется для преобразования IP-адреса из текстового формата в двоичный и наоборот, а также для преобразования порта из хостового порядка в сетевой порядок и наоборот. inet_addr(): преобразует IP-адрес из текстового формата в двоичный формат, который может быть использован в структуре sockaddr_in. inet_ntoa(): Преобразует IP-адрес из двоичного формата в текстовый формат. htons(): Преобразует порт из хостового порядка в сетевой порядок (если необходимо). Это нужно для правильной передачи порта в структуру sockaddr_in.

<unistd.h>: данная библиотека содержит функции, специфичные для UNIX-подобных систем, для работы с файловыми дескрипторами, процессами, директориями и другими системными вызовами. В программе используется функция close(), которая закрывает файловый дескриптор после завершения работы с сокетом.

<getopt.h>: эта библиотека предоставляет функции для разбора аргументов командной строки. В программе используется функция getopt(), которая

позволяет обрабатывать опции командной строки и их значения. Она позволяет программе принимать опции, такие как -s, -e, -f, -T, -p, -h, и определять соответствующие значения и действия.

<vector>: библиотека предоставляет класс vector, который представляет динамический массив элементов. В программе используется для хранения информации о портах и их статусе.

<cstdlib> и <cstring>: эти стандартные библиотеки C++ содержат функции для работы с преобразованиями строк в числа (atoi()) и для работы со строками (strcmp(), strcpy() и т.д.). В программе они используются для преобразования аргументов командной строки в числа и для работы с символьными данными.

Библиотеки предоставляют необходимые функции и структуры данных, которые позволяют программе сканера сетевых портов выполнять сетевые операции, управлять сокетами, анализировать аргументы командной строки, обрабатывать ввод-вывод и манипулировать данными в программе.

3.1 Файл «main.cpp»

В данном файле содержится все функции нашей программы. Ниже разберем для чего нужны данные функции.

В данном коде используется перечисление (enum) Status и структура Port, чтобы представить информацию о статусе порта.

Перечисление Status определяет два возможных значения: FREE (свободный) и BUSY (занятый). Оно используется для обозначения состояния порта, то есть указывает, доступен ли порт для использования или занят другим процессом или сервисом.

```
enum Status {  
    FREE, BUSY  
};
```

Рис.3.1.1 – Перечисляемый тип Status

Структура Port состоит из двух полей: number (номер порта) и status (статус порта). Она используется для хранения информации о порте, включая его номер и текущий статус (свободный или занятый). При сканировании портов, функции и алгоритмы могут использовать структуру Port, чтобы сохранять информацию о каждом отсканированном порте, его номере и статусе (свободный или занятый). Это позволяет удобно организовывать и обрабатывать данные о портах в программе.

```

Struct Port {
    int number;
    Status status;
};

```

Рис.3.1.2 – Структура Port

`bool scanOnePort(int portToCheck)` – выполняет сканирование одного конкретного порта для проверки его доступности. Принимает в качестве аргумента `portToCheck` номер порта, который нужно проверить.

`Vector<Port> scanAllPorts(int startPort, int endPort)` – выполняет сканирование диапазона портов. Принимает аргументы `startPort` и `endPort`, которые определяют диапазон портов для сканирования.

`Void writeToFile(const vector<Port>& ports, const string& filename)` – записывает результаты сканирования в файл. Принимает аргументы `ports`, содержащий информацию о сканированных портах, и `filename`, указывающий имя файла, в который нужно записать результаты.

`Void printHelp()` – выводит справочную информацию о программе и доступные опции командной строки.

`printTable(vector<Port> ports)` – выводит результаты сканирования в виде таблицы. Принимает вектор `ports`, содержащий информацию о сканированных портах. Функция выводит заголовок таблицы и затем в цикле выводит информацию о каждом порту.

`Int main(int argc, char *argv[])` – это главная функция программы. Принимает аргументы командной строки `argc` и `argv`. Внутри функции объявляются переменные для параметров командной строки, флагов и других необходимых значений. В зависимости от установленных флагов выполняются различные действия: вывод справки, сканирование одного порта, сканирование диапазона портов, вывод результатов в консоль или запись результатов в файл.

3.2 Файл «Makefile»

Makefile - это текстовый файл, который содержит инструкции для сборки программы. Он используется в процессе автоматической сборки программного проекта и управления зависимостями между исходными файлами.

Основная цель Makefile состоит в том, чтобы определить, какие файлы должны быть скомпилированы и какие команды компиляции нужно выполнить для каждого файла. В результате Makefile позволяет компилировать проект с помощью единственной команды `make`, что упрощает процесс сборки и обновления программы.

```
CC = g++
CFLAGS = -std=c++11

all: portscanner

portscanner: main.cpp
    $(CC) $(CFLAGS) -o portscanner main.cpp

clean:
    rm -f portscanner
```

Рис.3.2.1 – Структура Makefile

4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

Для более наглядного отражения принципа работы и понимания происходящего были составлены блок-схемы функций `scanAllPorts(int startPort, int endPort)` и `writeToFile(const vector<Port>& ports, const string& filename)`, которые представлены в приложениях Б и В соответственно.

Далее пошагово рассмотрим другие программные модули.

4.1 Алгоритм работы функции `main()`:

Шаг 1: Начало.

Шаг 2: Объявление переменных для хранения значений параметров командной строки и установка значений по умолчанию:

- `opt` - переменная для хранения текущего опции командной строки.
- `startPort` - переменная для хранения начального порта сканирования (по умолчанию 1).
- `endPort` - переменная для хранения конечного порта сканирования (по умолчанию 65535).
- `helpFlag` - флаг для указания на запрос справки (по умолчанию `false`).
- `tableFlag` - флаг для указания на вывод результата в виде таблицы (по умолчанию `false`).
- `portToCheck` - переменная для хранения порта, который нужно проверить (по умолчанию -1).
- `writeFile` - флаг для указания на запись результата в файл (по умолчанию `false`).
- `scanningOnePort` - флаг для указания на сканирование одного порта (по умолчанию `false`).
- `filename` - строка для хранения имени файла для записи результата (по умолчанию пустая строка).

Шаг 3: Начало цикла обработки параметров командной строки с помощью функции `getopt`.

Шаг 4: Получение очередной опции командной строки и проверка ее значения.

Шаг 5: Проверка флага `helpFlag`.

Шаг 6: Проверка флага `scanningOnePort`.

Шаг 7: Вызов функции `scanAllPorts(startPort, endPort)` для сканирования всех портов и сохранения результатов в вектор `ports`.

Шаг 8: Проверка флага `tableFlag`.

Шаг 9: Проверка флага `writeFile`.

Шаг 10: Возврат значения 0.

Шаг 11: Конец.

4.2 Алгоритм работы функции `scanOnePort(int portToCheck)` :

Шаг 1: Начало.

Шаг 2: Получение порта для проверки из параметра функции `scanOnePort` и сохранение его в переменной `port`.

Шаг 3: Создание сокета с помощью функции `socket(AF_INET, SOCK_STREAM, 0)` и сохранение дескриптора сокета в переменной `sockfd`.

Шаг 4: Создание структуры `sockaddr_in` для хранения информации об адресе.

Шаг 5: Заполнение полей структуры `addr`.

Шаг 6: Вызов функции `connect(sockfd, (struct sockaddr *)&addr, sizeof(addr))` для установления соединения с указанным адресом и портом.

Шаг 7: Проверка результата установки соединения (`status`):

Если соединение установлено успешно (`status == 0`): переходим к шагу 8.

Если не удалось установить соединение (`status == 1`): переходим к шагу 9.

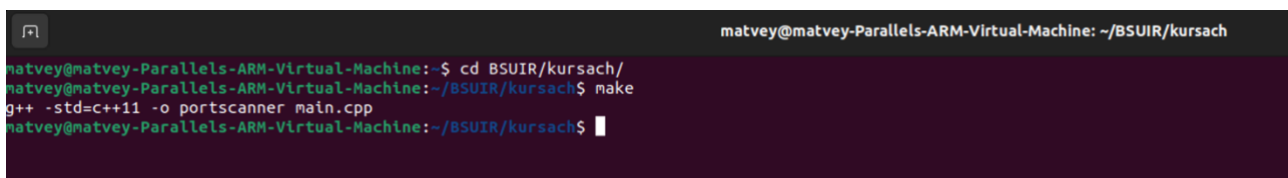
Шаг 8: Закрытие сокета с помощью `close(sockfd)` и возврат 1 (порт открыт).

Шаг 9: Возврат значения 0 (указывает на закрытый порт).

Шаг 10: Конец.

5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

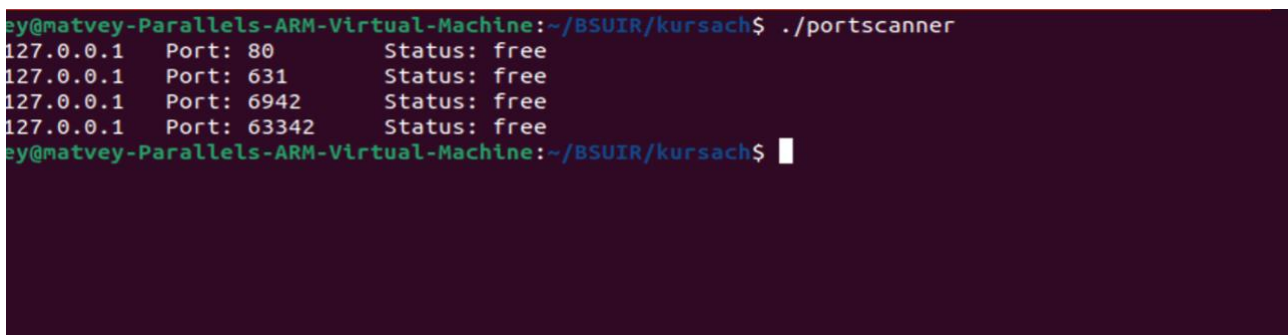
Чтобы запустить данную программу сперва требуется скомпилировать проект. Для начала откроем терминал и перейдем в папку с нашим проектом с помощью команды «cd» и введем make для сборки нашего проекта. Если make не найдем, требуется его установить с помощью команды «sudo apt-get install make», после чего следует повторить сборку нашего проекта. Сборка проекта:



```
matvey@matvey-Parallels-ARM-Virtual-Machine: ~/BSUIR/kursach
matvey@matvey-Parallels-ARM-Virtual-Machine:~$ cd BSUIR/kursach/
matvey@matvey-Parallels-ARM-Virtual-Machine:~/BSUIR/kursach$ make
g++ -std=c++11 -o portscanner main.cpp
matvey@matvey-Parallels-ARM-Virtual-Machine:~/BSUIR/kursach$
```

Рис.5.1 – Сборка проекта

Исполняемый файл можно запустить следующим способом:



```
matvey@matvey-Parallels-ARM-Virtual-Machine:~/BSUIR/kursach$ ./portscanner
127.0.0.1 Port: 80 Status: free
127.0.0.1 Port: 631 Status: free
127.0.0.1 Port: 6942 Status: free
127.0.0.1 Port: 63342 Status: free
matvey@matvey-Parallels-ARM-Virtual-Machine:~/BSUIR/kursach$
```

Рис.5.2 –Запуск исполняемого файла

На рисунке 5.2 мы видим, что открытых портов только два.

С помощью флага «-h» можно просмотреть, какие флаги имеет программа:



```
matvey@matvey-Parallels-ARM-Virtual-Machine:~/BSUIR/kursach$ ./portscanner -h
Usage: portscanner [OPTIONS]

Options:
  -h                Show this help message
  -s <port>         Start scanning from port (default: 1)
  -e <port>         End scanning at port (default: 65535)
  -p <port>         Scanning one port
  -T                Print results as a table
  -f <name file>    Write to file
```

Рис.5.3 – Вывод результата операции «-h»

Чтобы проверить только один порт, можно воспользоваться флагом «-p» и указать, какой порт требуется проверить:

```
matvey@matvey-Parallels-ARM-Virtual-Machine:~/BSUIR/kursach$ ./portscanner -p 80
port to check: 80
+-----+-----+-----+
| IP address | Port | Status |
+-----+-----+-----+
| 127.0.0.1  | 80   | free   |
+-----+-----+-----+
```

Рис.5.4 – Результат проверки одного порта

Следующая операция, которую мы можем использовать – это указания диапазона для проверки портов. С помощью флага «-s» мы указываем начальный порт, флаг «-e» позволяет указывать конечный порт. Результат работы данной операции:

```
matvey@matvey-Parallels-ARM-Virtual-Machine:~/BSUIR/kursach$ ./portscanner -s 80 -e 650 -T
+-----+-----+-----+
| IP address | Port | Status |
+-----+-----+-----+
| 127.0.0.1  | 80   | free   |
| 127.0.0.1  | 631  | free   |
+-----+-----+-----+
```

Рис.5.5 – Проверка выбранного диапазона

Здесь необязательно использовать указание начального и конечного порта, мы можем указать только тот порт, с которого начинать или каким заканчивать.

Чтобы продемонстрировать работу данных флагов выведем сначала состояние всех портов, затем выведем ограничение с помощью флага «-e» и затем проверим работу флага «-s»:

```
matvey@matvey-Parallels-ARM-Virtual-Machine:~/BSUIR/kursach$ ./portscanner
IP: 127.0.0.1 Port: 80 Status: free
IP: 127.0.0.1 Port: 631 Status: free
IP: 127.0.0.1 Port: 6942 Status: free
IP: 127.0.0.1 Port: 63342 Status: free
matvey@matvey-Parallels-ARM-Virtual-Machine:~/BSUIR/kursach$ ./portscanner -e 7000
IP: 127.0.0.1 Port: 80 Status: free
IP: 127.0.0.1 Port: 631 Status: free
IP: 127.0.0.1 Port: 6942 Status: free
matvey@matvey-Parallels-ARM-Virtual-Machine:~/BSUIR/kursach$ ./portscanner -s 100
IP: 127.0.0.1 Port: 631 Status: free
IP: 127.0.0.1 Port: 6942 Status: free
IP: 127.0.0.1 Port: 63342 Status: free
```

Рис.5.6 – Проверка работы флагов «-s», «-e»

На рисунке 5.6 мы видим, что сначала выводятся все открытые порты, затем мы запускаем программу, используя флаг «-e» 7000, тем самым делаем ограничение

на последний порт и видим, что выводятся порты до 7000. После чего мы используем флаг «-s» 100, то есть указываем от какого порта начинать. Программа успешно отработала.

Также есть флаг «-T», которые выводит наши порты в таблице:

```
matvey@matvey-Parallels-ARM-Virtual-Machine:~/BSUIR/kursach$ ./portscanner -T
```

IP address	Port	Status
127.0.0.1	80	free
127.0.0.1	631	free
127.0.0.1	6942	free
127.0.0.1	63342	free

Рис.5.7 – Проверка работы флага «-T»

Следующая операция, которая есть в данной работе – это запись в файл с помощью флага «-f <имя файла, куда записывается результат>»:

```
matvey@matvey-Parallels-ARM-Virtual-Machine:~/BSUIR/kursach$ ./portscanner -f test
IP: 127.0.0.1 Port: 80 Status: free
IP: 127.0.0.1 Port: 631 Status: free
writing to file....
```

Рис.5.8 – Проверка работы флага «-f»

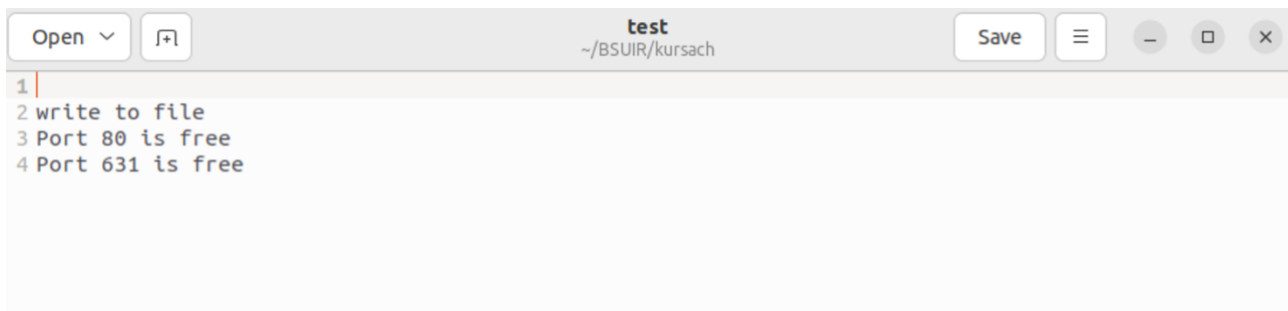


Рис.5.9 – Файл «test» после записи открытых портов

ЗАКЛЮЧЕНИЕ

В заключение, в рамках данной курсовой работы была разработана программа сканирования сетевых портов под операционную систему Linux. Программа позволяет пользователю сканировать диапазон портов на заданном IP-адресе и определять их доступность.

В процессе разработки программы были использованы принципы языка программирования C++ и объектно-ориентированного программирования. Использование сокетов и протокола TCP позволило осуществить сканирование портов и определить их статус (свободен или занят).

Программа обладает гибким функционалом, позволяя пользователю настраивать параметры сканирования, выводить результаты в табличной форме и сохранять их в файл.

Разработанная программа позволяет пользователям проводить сканирование портов для обнаружения открытых или закрытых сетевых сервисов. Она может быть полезной как для системных администраторов, так и для разработчиков и тестировщиков сетевых приложений.

В результате выполнения данной курсовой работы был получен полноценный инструмент для сканирования сетевых портов, который может быть использован для анализа и обеспечения безопасности сетей и систем.

Данный проект может быть усовершенствован в следующих направлениях:

- добавление графического интерфейса;
- добавление многопоточности;
- добавление дополнительных протоколов для сканирования;

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

- [1] Страуструп, Б. Язык программирования С++ / Б. Страуструп; специальное издание. Пер. с англ. – СПб. : BHV, 2008. – 1098 с.
- [2] Объектно-ориентированное программирование в С++ / Роберт Лафоре; г. пер. с англ. – Санкт-Петербург, 2019. – 1152 с.
- [3] Приемы объектно-ориентированного проектирования. Паттерны проектирования / Джон Валлисидес, Эрик Гамма, Р. Хелм – Питер Мейл, 2003.

ПРИЛОЖЕНИЕ А

(обязательное)

Структурная схема

ПРИЛОЖЕНИЕ Б

(обязательное)

Схема алгоритма метода

ПРИЛОЖЕНИЕ В

(обязательное)

Схема алгоритма метода

ПРИЛОЖЕНИЕ Г

(обязательное)

Код программы

ПРИЛОЖЕНИЕ Д

(обязательное)

Ведомость документов