

Учреждение образования  
«Белорусский государственный университет  
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
к курсовому проекту  
на тему  
«Утилита перехвата сетевого трафика»  
БГУИР КР 1-40 02 01 302 ПЗ

Студентка:

Балталиня К.А.

Руководитель:

Глоба А.А.

Минск 2022

## Оглавление

ВВЕДЕНИЕ .....	4
1 ОБЗОР ЛИТЕРАТУРЫ .....	5
1.1 Обзор предметной области .....	5
1.2 Обзор существующих аналогов.....	5
1.2.1 Tcpdump .....	5
1.2.2 Wireshark .....	6
1.2.3 Aircrack-ng .....	7
1.3 Постановка задачи .....	7
2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ .....	8
2.1 Постановка задачи .....	8
2.2 Модуль пользовательского интерфейса .....	8
2.3 Модуль работы приложения .....	8
2.4 Модуль перехвата трафика .....	8
2.5 Модуль обработки трафика.....	9
2.6 Модуль записи данных .....	9
3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ.....	10
3.1 struct sockaddr_in .....	10
3.2 struct sockaddr .....	10
4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ.....	11
4.1 Функция обработки пакета .....	11
5 ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ .....	13
5.1 Пользовательский ввод.....	13
5.2 Проверка прав суперпользователя .....	13
6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ .....	14
6.1 Системные требования .....	14
6.2 Использование приложения.....	14
ЗАКЛЮЧЕНИЕ .....	16
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ .....	17

ПРИЛОЖЕНИЕ А .....	18
ПРИЛОЖЕНИЕ Б .....	19
ПРИЛОЖЕНИЕ В.....	20

## ВВЕДЕНИЕ

Перехватом данных по сети считается получение любой информации с удаленного компьютерного устройства. Она может состоять из личных сведений пользователя, его сообщений, записей о посещаемости веб-сайтов. Захват данных может осуществляться программами-шпионами или при помощи сетевых sniffers, а также позволяет:

- Обнаружить вирусный трафик, наличие которого увеличивает загрузку сетевого оборудования и каналов связи.
- Выявить в сети вредоносное и несанкционированное ПО, например, сетевые сканеры, флудеры, троянские программы и другие.
- Перехватить любой пользовательский трафик с целью получения паролей и другой информации.
- Локализовать неисправность сети или ошибку конфигурации сетевых агентов.

Трафик подразделяется на:

- Исходящий (информация, поступающая во внешнюю сеть).
- Входящий (информация, поступающая из внешней сети).
- Внутренний (информация находится в пределах определенной сети, чаще всего локальной).
- Внешний (информация находится за пределами определенной сети, чаще всего – интернет-трафик).

Данными из трафика можно завладеть разными способами: прослушиванием интерфейса сети, подключением устройства перехвата в разрыв канала, созданием ветки трафика и ее дублированием на sniffer, путем проведения атаки.

Данный курсовой проект представляет собой такую утилиту командной строки, которая будет осуществлять захват трафика.

# **1 ОБЗОР ЛИТЕРАТУРЫ**

## **1.1 Обзор предметной области**

Сниффером называют программу или компьютерную технику, перехватывающую и анализирующую трафик, который проходит через сеть. Сниффер позволяет подключаться к веб-сессии и осуществлять операции от имени владельца компьютера.

Сетевой трафик – объем информации, передаваемой через компьютерную сеть за определенный период времени. Количество трафика измеряется как в пакетах, так и в битах, байтах и т.д.

Пакет данных – комплексный термин, описывающий любой набор двоичных данных, отправляемых по сетевому каналу.

Протокол – это набор правил, описывающих формат назначения кадров, пакетов или сообщений, которыми обмениваются объекты одного ранга внутри уровня.

В 1983 году стек протоколов TCP/IP стал официальным стандартом для всего интернета.

TCP/IP (Transmission Control Protocol и Internet Protocol – сетевая модель передачи данных, представленных в цифровом виде. Модель описывает способ передачи данных от источника информации к получателю, обеспечивает сквозную передачу данных, определяющую, как данные должны пакетироваться, обрабатываться, передаваться, маршрутизироваться и приниматься.

## **1.2 Обзор существующих аналогов**

Целью данной курсовой работы является создание корректно работающего приложения. Рассмотрим существующие аналоги, их преимущества, недостатки и функционал.

### **1.2.1 Tcpdump**

Tcpdump – утилита UNIX позволяющая перехватывать и анализировать сетевой трафик, проходящий через компьютер, на котором запущена данная программа.

Для выполнения программы требуется наличие прав суперпользователя и прямой доступ к устройству.

Основные назначения tcpdump: отладка сетевых приложений, отладка сети и сетевой конфигурации в целом.

## 1.2.2 Wireshark

Wireshark – программа-анализатор трафика для компьютерных сетей Ethernet и некоторых других. Имеет графический пользовательский интерфейс.

Функциональность схожа с возможностями программы tcpdump, но имеет больше возможностей по сортировке и фильтрации информации, а также вышеупомянутый графический пользовательский интерфейс.

Существуют версии для большинства UNIX-подобных систем, в том числе GNU/Linux, Solaris, FreeBSD, macOS, а также для Windows.

Основные достоинства инструмента :

- Поддержка большого количества сетевых протоколов (в том числе протоколов IP-телефонии);

- Детальная система фильтров сетевых пакетов;

- Возможность восстановления протоколов TCP;

Существенных недостатков не выявлено.

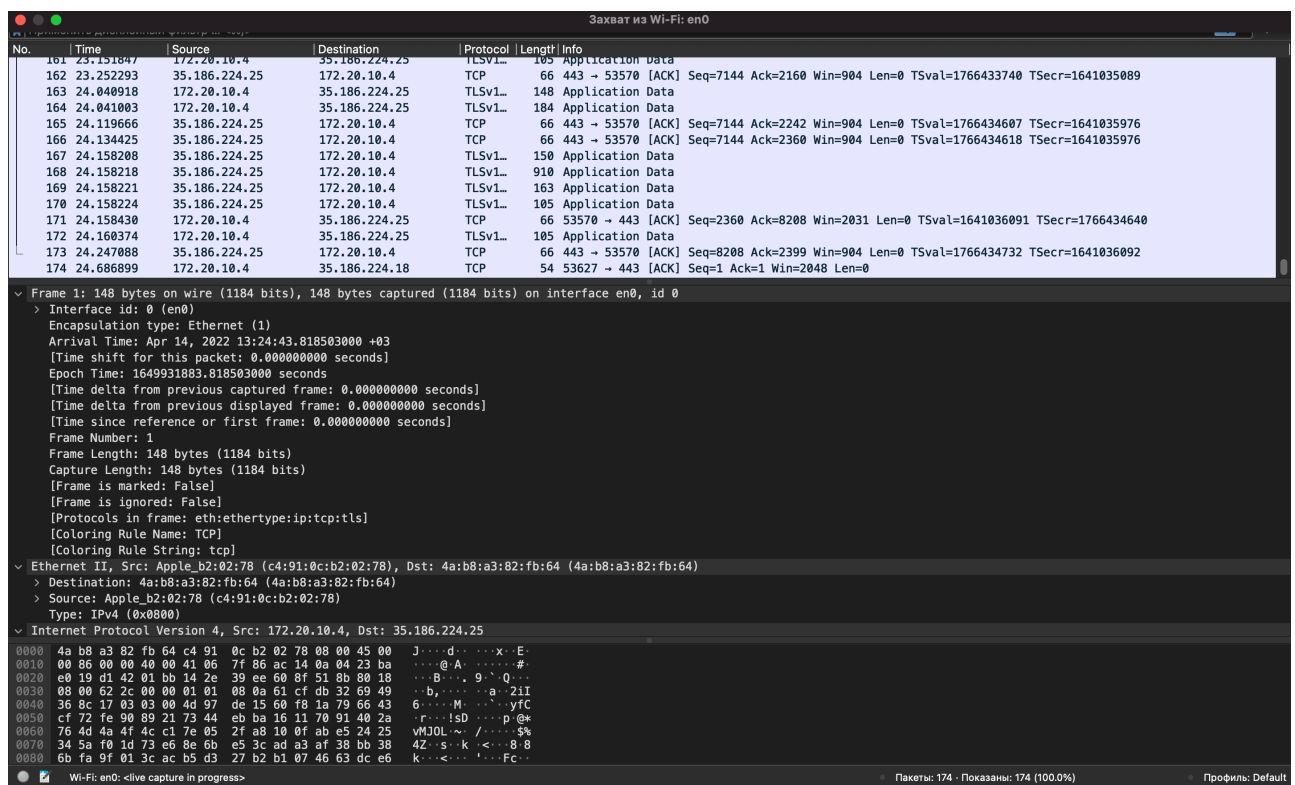


Рисунок 1.2 – Скриншот программы Wireshark

Wireshark разработан компанией “The Wireshark team”. Приложение написано на языке C++ и C. Последняя версия была выпущена 23 марта 2022 года.

### 1.2.3 Aircrack-ng

Aircrack-ng – набор программ, предназначенных для обнаружения беспроводных сетей, перехвата передаваемого через беспроводные сети трафика, аудита WEP и WPA/WPA2-PSK ключей шифрования (проверка стойкости), в том числе пентеста беспроводных сетей.

Программа работает с любыми беспроводными сетевыми адаптерами, драйвер которых поддерживает режим мониторинга.

Программа работает в операционных системах Windows, UNIX, Linux, Mac OS X. Версия для UNIX-подобных операционных систем имеет значительно большую функциональность и поддерживает больше беспроводных адаптеров, чем Windows-версия.

```
CH  0  ][ BAT 100%  ][ GPS    0.000    0.000    0.000    0.00  ][ 2006-05-08 13:04

BSSID                PWR Beacons  # Data  CH  MB  ENC  ESSID
00:0F:CC:39:8A:BC    65      780      39   7  54. WEP  Jungnetz

BSSID                STATION            PWR  Packets  Probes
(not associated)     00:12:79:40:90:65   73      23
(not associated)     00:15:00:45:0E:FF   80     148 ANIT,geziistanbul,AIRTIES
```

Рисунок 1.3 – Скриншот программы Aircrack-ng

Приложение написано на языке C. Последняя версия была выпущена 9 декабря 2018 года.

### 1.3 Постановка задачи

После рассмотрения аналогов можно сказать, что все они обладают большим количеством функций, которые невозможно реализовать в курсовом проекте за данный период времени. Поэтому были выбраны несколько ключевых возможностей, которые будут выполнены в рамках одного семестра:

- Программа должна иметь понятный интерфейс.
- В программе должна быть реализована обработка известных протоколов.
- Полученная информация должна сохраняться в файл.

## **2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ**

Сначала следует поставить конкретные функциональные требования разрабатываемой программе, разбить утилиту на модули и функциональные блоки. Данный подход в большинстве упростит понимание проектирования, сможет помочь устранить проблемы в архитектуре и обеспечить гибкость каждого из модулей. После того, как аналоги рассмотрены и проведён краткий экскурс в основные понятия, можно поставить конкретные цели разрабатываемому программному обеспечению.

### **2.1 Постановка задачи**

Для постановки конкретных функциональных требований нужно понять, какие функции должен выполнять наш сниффер. При запуске программного обеспечения от имени администратора, так как программа создает сокеты, которые требуют корневого доступа, должен производиться перехват сетевого трафика. Для этого нам нужно создать сокет и получить все входящие пакеты. В таком случае буфер будет содержать данные, которые будут проанализированы и собраны. Следующая задача – прочитать захваченный пакет, проанализировать его и представить пользователю в читаемом виде. Перехваченные данные будем записывать с файл log.txt.

### **2.2 Модуль пользовательского интерфейса**

Блок пользовательского интерфейса предназначен для взаимодействия пользователя с приложением. Основан на представлении и визуализации данных.

Для данного проекта используется консольный доступ.

### **2.3 Модуль работы приложения**

Блок работы приложения необходим для взаимодействия пользователя с программой и является неотъемлемой его частью. В данном блоке осуществляются следующие операции:

- Создание либо открытие файла для записи обработанных пакетов.
- Инициализация сокета.
- Обработка действий пользователя.

### **2.4 Модуль перехвата трафика**



Блок перехвата трафика предназначен для получения информации о сетевых потоках. Данная информация в дальнейшем используется в других модулях программы.

## **2.5 Модуль обработки трафика**

В данном блоке происходит обработка перехваченного трафика. Подсчитывается количество tcp, udp, icmp, igmp и других полученных протоколов. Вызывается функция записи данных в файл.

## **2.6 Модуль записи данных**

Блок записи данных предназначен для сохранения информации о полученных пакетах в текстовый файл под названием log.txt.

### **3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ**

В данном разделе описывается функционирование и структура разрабатываемого приложения. Структурная схема вынесена в приложение “А”.

#### **3.1 struct sockaddr\_in**

Описывает сокет для работы с протоколами IP.

Поля:

- sin\_family всегда равно AF\_INET.
- sin\_port содержит номер порта, который намерен занять процесс, если значение этого поля равно нулю, то ОС сама выделит свободный номер порта для сокета.
- sin\_addr содержит IP адрес к которому будет привязан сокет.

#### **3.2 struct sockaddr**

Данная структура используется для обработки адресов сетевого трафика. Предназначена для ОС .

Поля:

- sa\_family принимает значение AF\_xxx.
- Sa\_data содержит адрес протокола.

## 4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

В данном разделе будет рассмотрен алгоритм, используемый при разработке данного ПО.

### 4.1 Функция обработки пакета

Структурная схема представлена в Приложении А.

Шаг 1. Начало алгоритма.

Шаг 2. Инициализируем указатель на структуру.

```
struct iphdr *iph = (struct iphdr*)(buffer + sizeof(struct ethhdr));
```

Шаг 3. Инкрементируем переменную total.

```
++total;
```

Шаг 4. В конструкции switch-case с помощью \*iph проверяем протокол, печатаем его содержимое.

```
switch(iph->protocol)
{
```

```
    case 1:
```

Отвечает за Internet Control Message Protocol – протокол управления сообщениями в Интернете. Используется IP – устройствами, чтобы информировать другие IP-устройства о действиях и ошибках в сети.

Инкрементируем переменную icmp для отслеживания количества таких перехваченных протоколов.

```
    ++icmp;
```

С помощью следующей функции распечатываем содержимое пакета.

```
    print_icmp_packet(buffer, size);
```

Выходим из case 1.

```
    break;
```

```
    case 2:
```

Отвечает за Internet Group Management Protocol – протокол управления групповой передачей данных в сетях, основанных на протоколе IP. IGMP используется маршрутизаторами и IP-узлами для организации сетевых устройств в группы.

Инкрементируем переменную igmp для отслеживания количества таких перехваченных протоколов.

```
    ++igmp;
```

Выходим из case 2.

```
    break;
```

```
    case 6:
```

Отвечает за Transmission Control Protocol – один из основных протоколов передачи данных интернета. Прежде, чем начать обмен данными, ТСР протоколу требуется установить соединение между двумя хостами.

Инкрементируем переменную tcp для отслеживания количества таких перехваченных протоколов.

```
++tcp;
```

С помощью следующей функции распечатываем содержимое пакета.

```
print_tcp_packet(buffer, size);
```

Выходим из case 6.

```
break;
```

```
case 17:
```

Отвечает за User Datagram Protocol. Обеспечивает доставку дейтограмм, но не требует подтверждения их получения.

Инкрементируем переменную udp для отслеживания количества таких перехваченных протоколов.

```
++udp;
```

С помощью следующей функции распечатываем содержимое пакета.

```
print_udp_packet(buffer, size);
```

Выходим из case 17.

```
break;
```

```
default:
```

Собирает данные о количестве иных перехваченных протоколов (прим. ARP).

```
++other;
```

Выходим из case default.

```
break;
```

```
}
```

Шаг 5. С помощью функции printf выводим количество перехваченных пакетов.

```
printf("TCP : %d UDP : %d ICMP : %d IGMP : %d Others : %d Total : %d\n", tcp, udp, icmp, igmp, others, total);
```

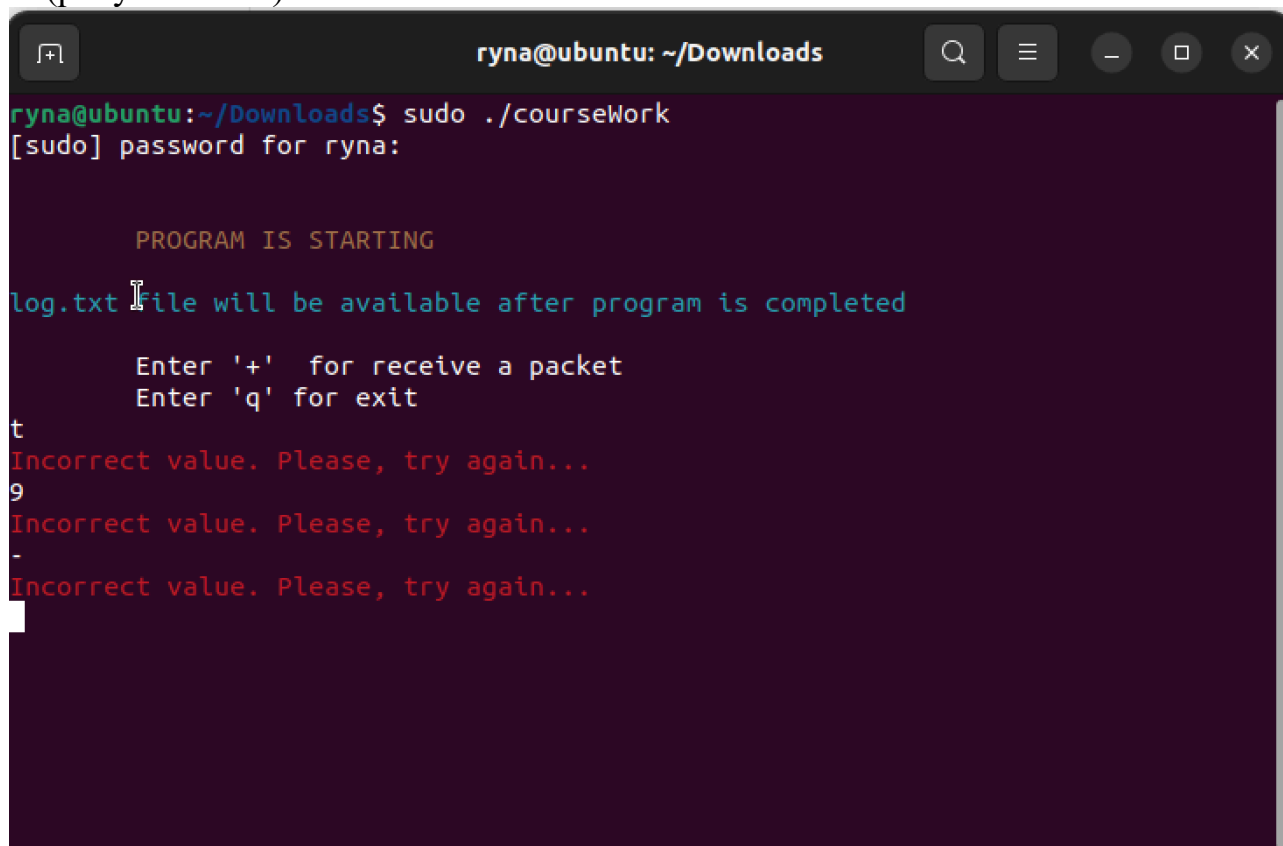
Шаг 8. Конец алгоритма.

## 5 ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ

В данном разделе описывается функциональное тестирование программы.

### 5.1 Пользовательский ввод

В программе реализована проверка пользовательского ввода. Пользователь должен ввести '+' для перехвата сетевого трафика либо 'q' для выхода из программы. При некорректном вводе выводится сообщение об ошибке (рисунок 5.1.1).



```
ryna@ubuntu: ~/Downloads
ryna@ubuntu:~/Downloads$ sudo ./courseWork
[sudo] password for ryna:

    PROGRAM IS STARTING

log.txt file will be available after program is completed

    Enter '+' for receive a packet
    Enter 'q' for exit

t
Incorrect value. Please, try again...
9
Incorrect value. Please, try again...
-
Incorrect value. Please, try again...
```

Рисунок 5.1.1 – Обработка некорректного ввода

### 5.2 Проверка прав суперпользователя

Данный курсовой проект реализован с помощью сокетов, потому для его запуска требуются права суперпользователя. В случае ошибки выведется сообщение об ошибке.

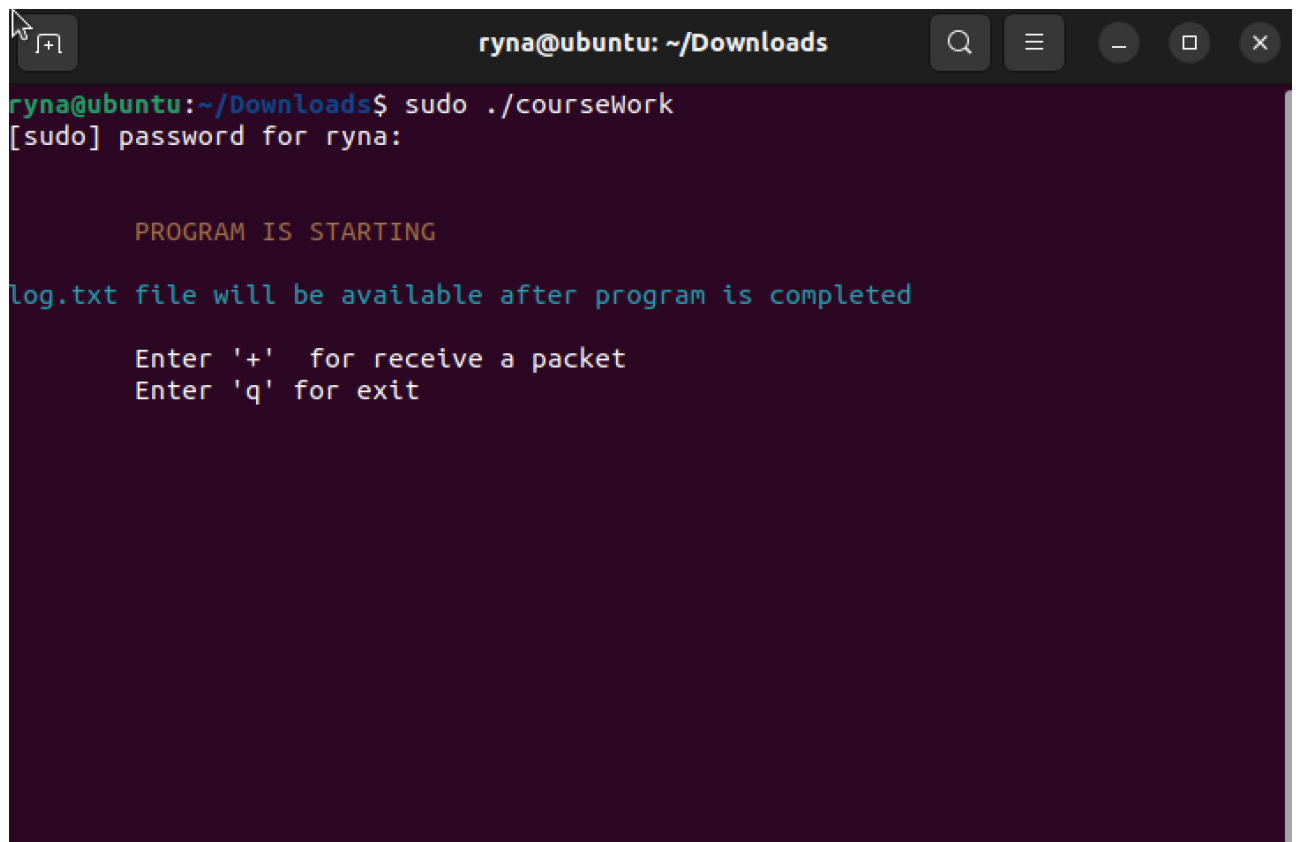
## 6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

### 6.1 Системные требования

Требования для пользования программным обеспечением: ОС – Linux.  
Программу необходимо запускать от имени администратора.

### 6.2 Использование приложения

При запуске приложения выводится окно, в котором предложен функционал для дальнейшей работы с программой. Необходимо ввести один из предложенных символов, иначе будет выведено сообщение об ошибке (рисунок 6.2.1).

A screenshot of a terminal window titled 'ryna@ubuntu: ~/Downloads'. The terminal shows the command 'sudo ./courseWork' being executed. The prompt changes to '[sudo] password for ryna:' and then back to 'ryna@ubuntu:~/Downloads\$'. The program output includes 'PROGRAM IS STARTING', 'log.txt file will be available after program is completed', and instructions to 'Enter '+' for receive a packet' and 'Enter 'q' for exit'.

```
ryna@ubuntu: ~/Downloads
ryna@ubuntu:~/Downloads$ sudo ./courseWork
[sudo] password for ryna:

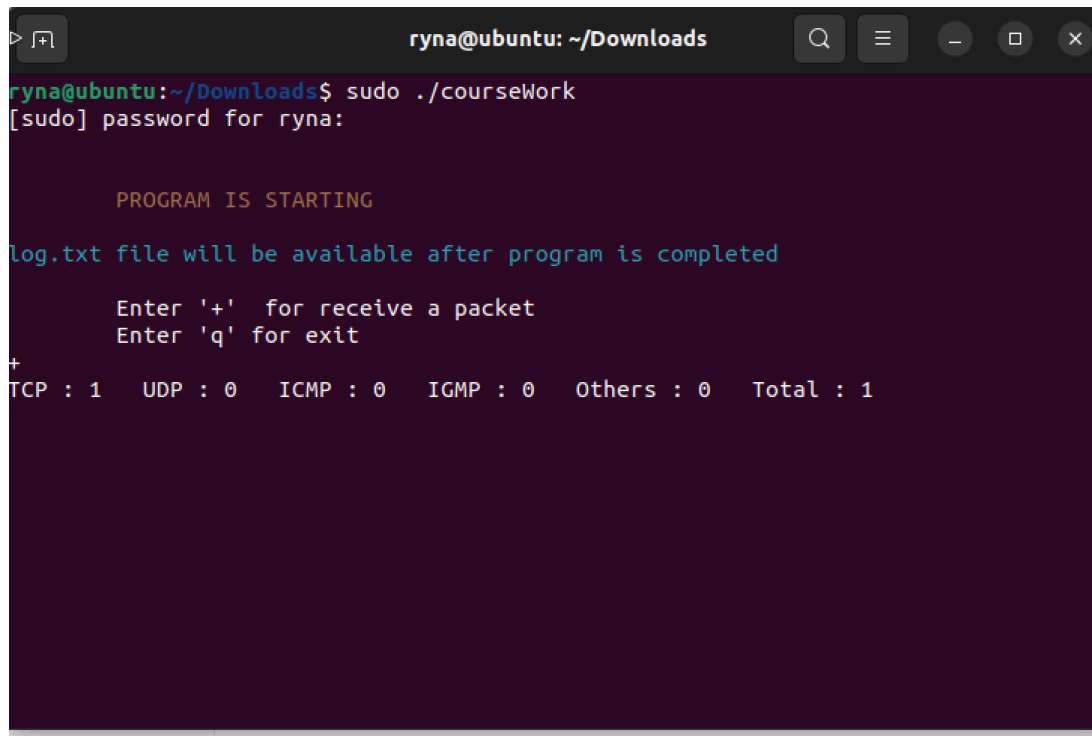
PROGRAM IS STARTING

log.txt file will be available after program is completed

Enter '+' for receive a packet
Enter 'q' for exit
```

Рисунок 6.2.1 – Запуск программы

После выбора дальнейшего действия будет произведен либо перехват сетевого трафика (рисунок 6.2.2) , либо выход из программы (6.2.3).

A terminal window titled 'ryna@ubuntu: ~/Downloads' showing the execution of a program. The user runs 'sudo ./courseWork' and enters a password. The program outputs 'PROGRAM IS STARTING' and 'log.txt file will be available after program is completed'. It then prompts the user to 'Enter '+' for receive a packet' or 'Enter 'q' for exit'. The user enters '+', and the program displays a network statistics summary: 'TCP : 1 UDP : 0 ICMP : 0 IGMP : 0 Others : 0 Total : 1'.

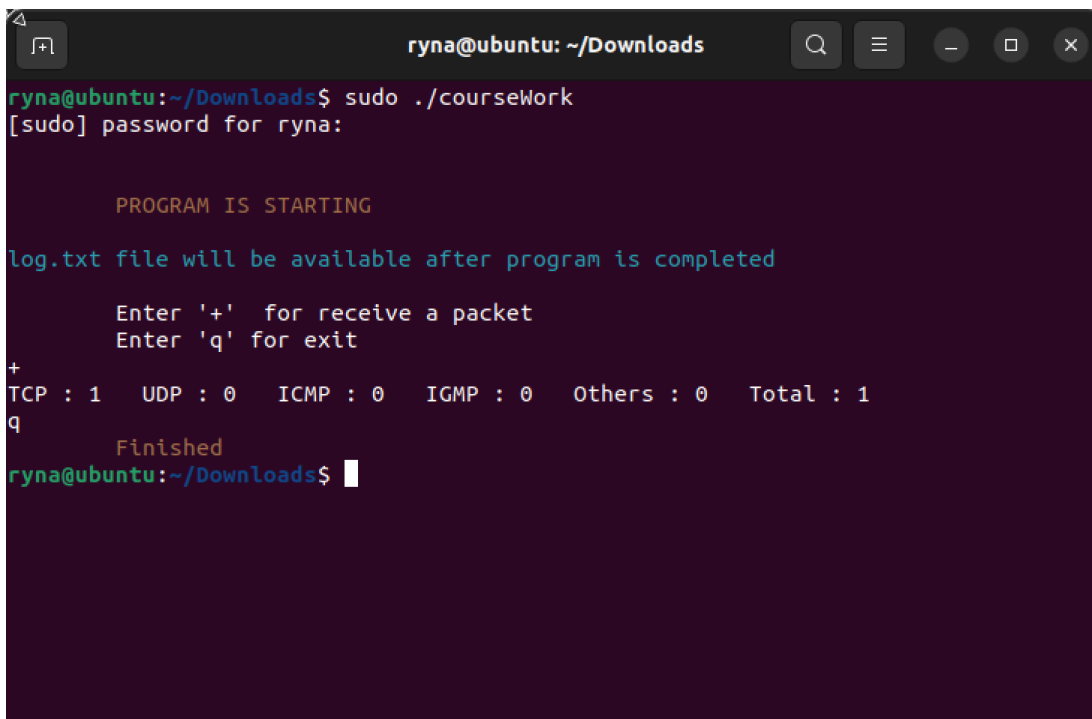
```
ryna@ubuntu:~/Downloads$ sudo ./courseWork
[sudo] password for ryna:

    PROGRAM IS STARTING

log.txt file will be available after program is completed

    Enter '+' for receive a packet
    Enter 'q' for exit
+
TCP : 1  UDP : 0  ICMP : 0  IGMP : 0  Others : 0  Total : 1
```

Рисунок 6.2.2 – Перехват сетевого трафика

A terminal window titled 'ryna@ubuntu: ~/Downloads' showing the execution of the same program. The user runs 'sudo ./courseWork' and enters a password. The program outputs 'PROGRAM IS STARTING' and 'log.txt file will be available after program is completed'. It then prompts the user to 'Enter '+' for receive a packet' or 'Enter 'q' for exit'. The user enters '+', and the program displays the same network statistics summary as in the previous image. Then, the user enters 'q', and the program outputs 'Finished' before returning to the shell prompt.

```
ryna@ubuntu:~/Downloads$ sudo ./courseWork
[sudo] password for ryna:

    PROGRAM IS STARTING

log.txt file will be available after program is completed

    Enter '+' for receive a packet
    Enter 'q' for exit
+
TCP : 1  UDP : 0  ICMP : 0  IGMP : 0  Others : 0  Total : 1
q
    Finished
ryna@ubuntu:~/Downloads$
```

Рисунок 6.2.3 – Выход из программы

## **ЗАКЛЮЧЕНИЕ**

В результате выполнения курсового проекта разработана утилита перехвата сетевого трафика, а также были реализованы основные поставленные задачи.

В ходе написания данной утилиты была изучена работа с сокетами на ОС Linux, изучено устройство сетевых протоколов TCP, UDP, IGMP, ICMP и взаимодействие с ними на языке C. Также для выполнения проекта была использована работа с процессами ОС Linux и работа с текстовыми файлами.

К достоинствам программы относятся простой пользовательский интерфейс, малый объем оперативной памяти.



## СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

- [1] Э. Таненбаум, Д.Уэзеролл. Компьютерные сети. 5-е издание, 2019 год.
- [2] Э. Таненбаум, Т. Остин. Архитектура компьютера. 6-е издание, 2013 год.
- [3] Программирование сокетов в Linux [Электронный курс]. – Режим доступа: <https://rdsn.org/article/unix/sockets.xml>.
- [4]. Программирование сокетов – CodeNet [Электронный курс]. – Режим доступа: <http://www.codenet.ru/progr/cpp/Sockets.php>.

**ПРИЛОЖЕНИЕ А**  
*(обязательное)*

**Структурная схема**

**ПРИЛОЖЕНИЕ Б**  
*(обязательное)*

Блок-схема алгоритма

**ПРИЛОЖЕНИЕ В**  
*(обязательное)*

Ведомость документов