Министерство цифрового развития, связи
и массовых коммуникаций Российской Федерации

Сибирский государственный университет
телекоммуникаций и информатики

Кафедра прикладной математики и кибернетики

КУРСОВАЯ РАБОТА

По дисциплине: «Операционные системы»

Выполнили:

Студенты 3 курса группы ИП-111
Корнилов А.А.,
Попов М.И.,
Толкач А.А.

Проверил:

Профессор кафедры ПМиК
Малков Е.А.

Новосибирск, 2023

Задание: В качестве задания была выбрана работа на оценку хорошо, написание программы анализатора трафика «сниффера» и к ней был сделан режим Linux демона.

Выполнение работы:

В группе было проведено распределение работы по написанию программы:

- Попов Мирон – общая работы программы, разработка выбора режима демона или интерактивного;
- Корнилов Андрей – парсинг данных из структур и в них;
- Толкач Илья – вывод данных на экран.

Программа имеет два режима работы, при запуске с указанием аргумента -i программа выводит информацию о перехваченных пакетах в терминал, если указать аргумент -d программа запуститься в режиме демона и выведет на экран свой PID для последующего выключения программы, иначе выведет сообщение о необходимости выбора режимы и завершит работу. Программа работает только с правами суперпользователя (sudo).

```
#ifndef COURSE
#define COURSE

#include<netinet/in.h>
#include<errno.h>
#include<netdb.h>
#include<time.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<netinet/ip_icmp.h>
#include<netinet/udp.h>
#include<netinet/tcp.h>
#include<netinet/ip.h>
#include<netinet/if_ether.h>
#include<net/ethernet.h>
#include<sys/socket.h>
#include<arpa/inet.h>
#include<sys/ioctl.h>
#include<sys/time.h>
#include<sys/types.h>
#include<unistd.h>

FILE *logfile; //место для вывода данных
struct sockaddr_in source, dest; //структуры для хранения IP-адресов отправителя и
получателя
```

```
int tcp = 0, udp = 0, others = 0, total = 0, i, j; //подсчет пакетов

void ProcessPacket(unsigned char *, int); //обработка пакета, выбор функции для
обработки
void print_ip_header(unsigned char *, int); //вывод заголовка ip
void print_tcp_packet(unsigned char *, int); //вывод содержимого TCP пакета
void print_udp_packet(unsigned char *, int); //вывод содержимого UDP пакета
void PrintData(unsigned char *, int); // вывод RAW данных
void DaemonMode(); //Основная функция программы
#endif
```

Листинг 1 – заголовочный файл course.h

```
#include "course.h"

void ProcessPacket(unsigned char *buffer, int size) { //Получаем заголовок пакета,
выбираем как обработать
    struct iphdr *iph = (struct iphdr *) (buffer + sizeof(struct ethhdr));
    ++total;
    switch (iph->protocol){ //Выбираем протокол
        case 6:  //TCP
            ++tcp;
            print_tcp_packet(buffer, size);
            break;
        case 17: //UDP
            ++udp;
            print_udp_packet(buffer, size);
            break;
        default: //Другие
            ++others;
            break;
    }
    if (logfile != stdout) printf("TCP : %d   UDP : %d   Другие : %d   Всего :
%d\r", tcp, udp, others, total);
}
void print_ethernet_header(unsigned char *Buffer, int Size) {
    struct ethhdr *eth = (struct ethhdr *) Buffer; //достаем из буфера в ethhdr
структуру
    time_t realtime = time(NULL);

    fprintf(logfile, "\nЗаголовок интернет пакета\n");
    fprintf(logfile, "  Время получения    - %s",  asctime(gmtime(&realtime)));
    fprintf(logfile, "  Адрес получателя   - %.2X-%.2X-%.2X-%.2X-%.2X-%.2X \n", eth-
>h_dest[0], eth->h_dest[1],
            eth->h_dest[2], eth->h_dest[3], eth->h_dest[4], eth->h_dest[5]);
    fprintf(logfile, "  Адрес отправителя - %.2X-%.2X-%.2X-%.2X-%.2X-%.2X \n", eth-
>h_source[0], eth->h_source[1],
            eth->h_source[2], eth->h_source[3], eth->h_source[4], eth-
>h_source[5]);
    fprintf(logfile, "  Протокол           - %u \n", (unsigned short) eth->h_proto);
}
void print_ip_header(unsigned char *Buffer, int Size) {
    print_ethernet_header(Buffer, Size);
    unsigned short iphdrlen;
```

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

    iphdrlen = iph->ihl * 4;
    memset(&source, 0, sizeof(source));
    source.sin_addr.s_addr = iph->saddr;
    memset(&dest, 0, sizeof(dest));
    dest.sin_addr.s_addr = iph->daddr;

    fprintf(logfile, "\n");
    fprintf(logfile, "IP Заголовок\n");
    fprintf(logfile, "  Версия IP протокола          - IPv%d\n", (unsigned int)
iph->version);
    fprintf(logfile, "  Длинна IP заголовка          - %d Байт\n",((unsigned int)
(iph->ihl)) * 4);
    fprintf(logfile, "  Тип сервиса                  - %d\n", (unsigned int) iph-
>tos);
    fprintf(logfile, "  Полная длинна заголовка IP   - %d Байт\n", ntohs(iph-
>tot_len));
    fprintf(logfile, "  Индификатор                  - %d\n", ntohs(iph->id));
    fprintf(logfile, "  TTL                          - %d\n", (unsigned int) iph-
>ttl);
    fprintf(logfile, "  Протокол                     - %d\n", (unsigned int) iph-
>protocol);
    fprintf(logfile, "  Чексумма                     - %d\n", ntohs(iph->check));
    fprintf(logfile, "  IP Отправителя               - %s\n",
inet_ntoa(source.sin_addr));
    fprintf(logfile, "  IP Назначения                - %s\n",
inet_ntoa(dest.sin_addr));
}
void print_tcp_packet(unsigned char *Buffer, int Size) {
    unsigned short iphdrlen;
    struct iphdr *iph = (struct iphdr *) (Buffer + sizeof(struct ethhdr));
    iphdrlen = iph->ihl * 4;

    struct tcphdr *tcph = (struct tcphdr *) (Buffer + iphdrlen + sizeof(struct
ethhdr));
    int header_size = sizeof(struct ethhdr) + iphdrlen + tcph->doff * 4;

    fprintf(logfile, "\n----------------------------------------------------------
");

    fprintf(logfile, "\nTCP Пакет:\n");
    print_ip_header(Buffer, Size);
    fprintf(logfile, "\nTCP Заголовок\n");
    fprintf(logfile, "  Порт отправителя     - %d\n", ntohs(tcph->source));
    fprintf(logfile, "  Порт назначения      - %d\n", ntohs(tcph->dest));
    fprintf(logfile, "  Номер пакета         - %u\n", ntohl(tcph->seq));
    fprintf(logfile, "  Номер потверждения   - %u\n", ntohl(tcph->ack_seq));
    fprintf(logfile, "  Длинна заголовка     - %d BYTES\n",(unsigned int) tcph-
>doff * 4);
    fprintf(logfile, "  Важность             - %d\n", (unsigned int) tcph->urg);
    fprintf(logfile, "  Потверждающий флаг   - %d\n", (unsigned int) tcph->ack);
    fprintf(logfile, "  Push флаг            - %d\n", (unsigned int) tcph->psh);
    fprintf(logfile, "  Reset флаг           - %d\n", (unsigned int) tcph->rst);
    fprintf(logfile, "  Флаг синхронизации   - %d\n", (unsigned int) tcph->syn);
```

```
    fprintf(logfile, "  Флаг конца пакета    - %d\n", (unsigned int) tcph->fin);
    fprintf(logfile, "  Размер окна          - %d\n", ntohs(tcph->window));
    fprintf(logfile, "  Чек-сумма            - %d\n", ntohs(tcph->check));
    fprintf(logfile, "  Указатель важности   - %d\n", tcph->urg_ptr);
    fprintf(logfile, "\n");

    fprintf(logfile, "  RAW Заголовок пакета\n");
    PrintData(Buffer, iphdrlen);

    fprintf(logfile, "  RAW Заголовок TCP\n");
    PrintData(Buffer + iphdrlen, tcph->doff * 4);

    fprintf(logfile, "  RAW Данные\n");
    PrintData(Buffer + header_size, Size - header_size);

    fprintf(logfile, "-----------------------------------------------------------
");
}
void print_udp_packet(unsigned char *Buffer, int Size) {
    unsigned short iphdrlen;
    struct iphdr *iph = (struct iphdr *) (Buffer + sizeof(struct ethhdr));
    iphdrlen = iph->ihl * 4;

    struct udphdr *udph = (struct udphdr *) (Buffer + iphdrlen + sizeof(struct
ethhdr));
    int header_size = sizeof(struct ethhdr) + iphdrlen + sizeof udph;

    fprintf(logfile, "\n-----------------------------------------------------------
");

    fprintf(logfile, "\nUDP Пакет:\n");
    print_ip_header(Buffer, Size);
    fprintf(logfile, "\nUDP Заголовок\n");
    fprintf(logfile, "  Порт отправителя    - %d\n", ntohs(udph->source));
    fprintf(logfile, "  Порт назначения     - %d\n", ntohs(udph->dest));
    fprintf(logfile, "  Длинна пакета       - %d\n", ntohs(udph->len));
    fprintf(logfile, "  Чек-сумма           - %d\n", ntohs(udph->check));
    fprintf(logfile, "\n");

    fprintf(logfile, "  RAW Заголовок пакета:\n");
    PrintData(Buffer, iphdrlen);

    fprintf(logfile, "  RAW Заголовок UDP\n");
    PrintData(Buffer + iphdrlen, sizeof udph);

    fprintf(logfile, "  RAW Данные\n");
    PrintData(Buffer + header_size, Size - header_size); //Двигаем указатель на
след. данные и сокращаем строку

    fprintf(logfile, "-----------------------------------------------------------
");
}
void PrintData(unsigned char *data, int Size) {
    int i, j;
    for (i = 0; i < Size; i++) { //Проходится по каждому байту в буфере данных
```

```
        // Вывод символов ASCII в конце каждой строки
        if (i != 0 && i % 16 == 0){   //конец линии
            fprintf(logfile, "         ");
            for (j = i - 16; j < i; j++) {
                if (data[j] >= 32 && data[j] <= 128)
                    fprintf(logfile, "%c", (unsigned char) data[j]); //проверка на
символ или цифру
                else fprintf(logfile, "."); //в другом случае печатаем точку
            }
            fprintf(logfile, "\n");
        }

        if (i % 16 == 0) fprintf(logfile, "   ");
        fprintf(logfile, " %02X", (unsigned int) data[i]); //Выводит
шестнадцатеричное значение каждого байта, разделенное пробелом

        if (i == Size - 1) {  //печатаем последний пакет
            // Вывод оставшихся символов ASCII в последней строке
            for (j = 0; j < 15 - i % 16; j++) fprintf(logfile, "   "); //ЕЩЕ
ПРОБЕЛОВ!!!!
            fprintf(logfile, "         ");

            for (j = i - i % 16; j <= i; j++){
                if (data[j] >= 32 && data[j] <= 128){
                    fprintf(logfile, "%c", (unsigned char) data[j]);
                } else fprintf(logfile, ".");
            }
            fprintf(logfile, "\n");
        }
    }
}
void DaemonMode(){
    int saddr_size,data_size;
    struct sockaddr saddr;
    unsigned char *buffer = (unsigned char *) malloc(65536); // буфер для
полученных данных

    if (logfile == NULL) {
        perror("Не удалось создать/открыть файл >^<");
        exit(0);
    }

    int sock_raw = socket(AF_PACKET, SOCK_RAW, htons(ETH_P_ALL));
    if (sock_raw < 0) {
        printf("Запустите программу с sudo! \n");
        perror("Ошибка открытия сокета :-(");
        exit(0);
    }

    printf("Запуск...\n");
    while (1) {
        saddr_size = sizeof saddr;
        data_size = recvfrom(sock_raw, buffer, 65536, 0, &saddr, (socklen_t * ) &
saddr_size); //получаем сообщение из сокета
```

```
        if (data_size < 0) {
            printf("Не получилось получить данные из сокета\n");
            exit(0);
        }
        ProcessPacket(buffer, data_size);
    }
}

int main(int argc, char **argv) {
    if (argc < 2){
        printf("Для запуска в режиме демона sudo ./course -d\n");
        perror("Для запуска в режиме программы sudo ./course -i\n");
        return 0;
    }
    pid_t parpid;

    if (strcmp(argv[1],"-i")==0) {
        logfile=stdout;
        DaemonMode();
    }
    else if (strcmp(argv[1],"-d")==0){
        if((parpid=fork())<0){
            printf("\nНеудалось создать дочерний процесс");
            exit(1);
        }
        else if (parpid!=0) exit(0);
        fprintf(stdout,"Для выключения демона напишите sudo kill
%d\n",(int)getpid());
        setsid();
        logfile = fopen("log.txt", "w");
        DaemonMode();
    }
    else{
        printf("Для запуска в режиме демона sudo ./course -d\n");
        perror("Для запуска в режиме программы sudo ./course -i\n");
        return 0;
    }
    return 0;
}
```

Листинг 2 – файл программы course.c

1. Функция ProcessPacket
   - Обрабатывает полученный сетевой пакет, определяет протокол (TCP, UDP, другие) и вызывает соответствующую функцию для дополнительной обработки.
   - Параметры:
     - buffer: Указатель на буфер данных пакета.
     - size: Размер пакета.
2. Функция print_ethernet_header

- Выводит информацию об Ethernet-заголовке пакета.
- Параметры:
    - Buffer: Указатель на буфер данных пакета.
    - Size: Размер пакета.

3. Функция print_ip_header
    - Выводит информацию об IP-заголовке пакета, используя print_ethernet_header для вывода информации об Ethernet-заголовке.
    - Параметры:
        - Buffer: Указатель на буфер данных пакета.
        - Size: Размер пакета.
4. Функции print_tcp_packet и print_udp_packet
    - Выводят информацию о TCP- и UDP-пакетах соответственно. Используют print_ip_header для вывода информации об IP-заголовке.
    - Параметры:
        - Buffer: Указатель на буфер данных пакета.
        - Size: Размер пакета.
5. Функция PrintData
    - Выводит содержимое буфера данных в формате шестнадцатеричного представления, а также символов ASCII.
    - Параметры:
        - data: Указатель на буфер данных.
        - Size: Размер данных.
6. Функция DaemonMode
    - Запускает программу в режиме демона, открывает сокет для захвата сетевого трафика и вызывает ProcessPacket для обработки каждого полученного пакета.
7. Функция main
    - Основная функция программы. В зависимости от аргументов командной строки (-i или -d), программа либо выполняет взаимодействие в режиме интерактивной программы, либо запускает себя в виде демона.
8. Другие переменные и структуры данных:
    - total, tcp, udp, others: Переменные для подсчета статистики по протоколам.
    - logfile: Указатель на файл, в который выводится информация о сетевом трафике.

- source, dest: Структуры для хранения IP-адресов отправителя и получателя.
- parpid: Идентификатор процесса.

Программа использует структуры данных из заголовочных файлов <netinet/ip.h>, <netinet/tcp.h>, <netinet/udp.h> и <netinet/ether.h>, что указывает на использование сетевых библиотек для работы с данными протоколов.

Команда компиляции и результат работы:

```
miron@DESKTOP-UMC1Q46:/mnt/u/Documents/B    ВУЗ/OS/course$    gcc    course.c
course.h -o course
miron@DESKTOP-UMC1Q46:/mnt/u/Documents/B ВУЗ/OS/course$ sudo ./course -i
[sudo] пароль для miron:
Запуск...


------------------------------------------------------------
UDP Пакет:


Заголовок интернет пакета
  Время получения   - Mon Dec 11 06:58:53 2023
  Адрес получателя  - FF-FF-FF-FF-FF-FF
  Адрес отправителя - 00-15-5D-13-7E-C5
  Протокол          - 8


IP Заголовок
  Версия IP протокола          - IPv4
  Длинна IP заголовка          - 20 Байт
  Тип сервиса                  - 0
  Полная длинна заголовка IP   - 229 Байт
  Индификатор                  - 60761
  TTL                          - 128
  Протокол                     - 17
  Чексумма                     - 17533
  IP Отправителя               - 172.24.80.1
  IP Назначения                - 172.24.95.255
```

```
UDP Заголовок
  Порт отправителя     - 138
  Порт назначения      - 138
  Длинна пакета        - 209
  Чек-сумма            - 33779

  RAW Заголовок пакета:
    FF FF FF FF FF FF 00 15 5D 13 7E C5 08 00 45 00          ........].~...E.
    00 E5 ED 59                                              ...Y
  RAW Заголовок UDP
    00 00 80 11 44 7D AC 18                                  ..€.D}..
  RAW Данные
    11 02 EF DE AC 18 50 01 00 8A 00 BB 00 00 20 45          ......P....... E
    45 45 46 46 44 45 4C 46 45 45 50 46 41 43 4E 46          EEFFDELFEEPFACNF
    46 45 4E 45 44 44 42 46 42 44 45 44 47 43 41 00          FENEDDBFBDEDGCA.
    20 46 48 45 50 46 43 45 4C 45 48 46 43 45 50 46           FHEPFCELEHFCEPF
    46 46 41 43 41 43 41 43 41 43 41 43 41 43 41 42          FFACACACACACACAB
    4E 00 FF 53 4D 42 25 00 00 00 00 00 00 00 00 00          N..SMB%.........
    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00          ................
    00 00 11 00 00 21 00 00 00 00 00 00 00 00 00 E8          .....!..........
    03 00 00 00 00 00 00 00 00 21 00 56 00 03 00 01          .........!.V....
    00 00 00 02 00 32 00 5C 4D 41 49 4C 53 4C 4F 54          .....2.\MAILSLOT
    5C 42 52 4F 57 53 45 00 01 00 80 FC 0A 00 44 45          \BROWSE...€...DE
    53 4B 54 4F 50 2D 55 4D 43 31 51 34 36 00 0A 00          SKTOP-UMC1Q46...
    03 10 00 00 0F 01 55 AA 00                               ......U..
```

Листинг 3 – запуск в интерактивном режиме

На лист. 3 демонстрируется запуск программы с параметром -i , запуск в интерактивном режиме. Перехваченные пакеты выводятся на экран.

```
miron@DESKTOP-UMC1Q46:/mnt/u/Documents/B ВУЗ/OS/course$ gcc course.c course.h -o course
miron@DESKTOP-UMC1Q46:/mnt/u/Documents/B ВУЗ/OS/course$ sudo ./course -d
Для выключения демона напишите sudo kill 2093
```

Листинг 4 – запуск в режиме демона

```
------------------------------------------------------------
UDP Пакет:

Заголовок интернет пакета
  Время получения   - Mon Dec 11 07:03:24 2023
  Адрес получателя  - 01-00-5E-7F-FF-FA
  Адрес отправителя - 00-15-5D-13-7E-C5
  Протокол          - 8

IP Заголовок
  Версия IP протокола          - IPv4
  Длинна IP заголовка          - 20 Байт
  Тип сервиса                  - 0
  Полная длинна заголовка IP   - 201 Байт
  Индификатор                  - 5922
  TTL                          - 1
  Протокол                     - 17
  Чексумма                     - 46574
  IP Отправителя               - 172.24.80.1
  IP Назначения                - 239.255.255.250

UDP Заголовок
  Порт отправителя    - 62850
  Порт назначения     - 1900
  Длинна пакета       - 181
  Чек-сумма           - 55103

  RAW Заголовок пакета:
    01 00 5E 7F FF FA 00 15 5D 13 7E C5 08 00 45 00         ..^•....].~...E.
    00 C9 17 22                                             ..."
  RAW Заголовок UDP
    00 00 01 11 B5 EE AC 18                                 ........
  RAW Данные
    4D 2D 53 45 41 52 43 48 20 2A 20 48 54 54 50 2F         M-SEARCH * HTTP/
    31 2E 31 0D 0A 48 4F 53 54 3A 20 32 33 39 2E 32         1.1..HOST: 239.2
    35 35 2E 32 35 35 2E 32 35 30 3A 31 39 30 30 0D         55.255.250:1900.
    0A 4D 41 4E 3A 20 22 73 73 64 70 3A 64 69 73 63         .MAN: "ssdp:disc
    6F 76 65 72 22 0D 0A 4D 58 3A 20 31 0D 0A 53 54         over"..MX: 1..ST
    3A 20 75 72 6E 3A 64 69 61 6C 2D 6D 75 6C 74 69         : urn:dial-multi
    73 63 72 65 65 6E 2D 6F 72 67 3A 73 65 72 76 69         screen-org:servi
    63 65 3A 64 69 61 6C 3A 31 0D 0A 55 53 45 52 2D         ce:dial:1..USER-
    41 47 45 4E 54 3A 20 47 6F 6F 67 6C 65 20 43 68         AGENT: Google Ch
    72 6F 6D 65 2F 31 32 31 2E 30 2E 36 31 36 37 2E         rome/121.0.6167.
    38 20 57 69 6E 64 6F 77 73 0D 0A 0D 0A                  8 Windows....
------------------------------------------------------------
------------------------------------------------------------
UDP Пакет:

Заголовок интернет пакета
  Время получения   - Mon Dec 11 07:03:25 2023
  Адрес получателя  - 01-00-5E-7F-FF-FA
  Адрес отправителя - 00-15-5D-13-7E-C5
  Протокол          - 8
...
```

Листинг 5 – содержимое log.txt

При запуске программы в режиме демона (лис. 4) программа выводит свой PID, и для завершения просит написать команду kill. Пока демон запущен программа записывает данные перехваченных пакетов в log.txt файл (лис. 5).