НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ КОРПОРАЦИЯ ИТМО

Факультет программной инженерии и компьютерной техники

Операционные системы

Лабораторная работа №2

Выполнил
студент

Нестеров Иван Алексеевич

Группа P33302

Преподаватель:

Осипов Святослав Владимирович

г. Санкт-Петербург

2022

Цель задания:

Разработать комплекс программ на пользовательском уровне и уровне ярда, который собирает информацию на стороне ядра и передает информацию на уровень пользователя, и выводит ее в удобном для чтения человеком виде. Программа на уровне пользователя получает на вход аргумент(ы) командной строки (не адрес!), позволяющие идентифицировать из системных таблиц необходимый путь до целевой структуры, осуществляет передачу на уровень ядра, получает информацию из данной структуры и распечатывает структуру в стандартный вывод. Загружаемый модуль ядра принимает запрос через указанный в задании интерфейс, определяет путь до целевой структуры по переданному запросу и возвращает результат на уровень пользователя.

Интерфейс передачи между программой пользователя и ядром и целевая структура задается преподавателем.

Согласно варианту, выданы:

1. debugfs - отладочная файловая система /sys/kernel/debug, передача параметров через запись в файл;
2. Целевые структуры, заданные именем структуры в заголовочных файлах Linux: vm_area_struct и net_device.

Ход работы:

Был написан следующий код. Кроме того, для удобства чтения он опубликован на GitHub.

Makefile

```
obj-m = kernelmodule.o

all:
	make -C /lib/modules/$(shell uname -r)/build M=$(shell pwd) modules
	gcc client.c -o client
	sudo insmod kernelmodule.ko

clean:
	make -C /lib/modules/$(shell uname -r)/build M=$(shell pwd) clean
	sudo rmmod kernelmodule.ko
	rm client
```

kernelmodule.h

```
#define DEBUGFS_DIR_NAME "os_lab2"
#define DEBUGFS_VMA_FILE_NAME "debugfs_vma"
#define DEBUGFS_VMA_ARG_FILE_NAME "debugfs_vma_arg"
#define DEBUGFS_VMA_FILE_PATH "/sys/kernel/debug/os_lab2/debugfs_vma"
#define DEBUGFS_VMA_ARG_FILE_PATH
"/sys/kernel/debug/os_lab2/debugfs_vma_arg"
#define DEBUGFS_NDS_FILE_NAME "debugfs_nds"
#define DEBUGFS_NDS_FILE_PATH "/sys/kernel/debug/os_lab2/debugfs_nds"
#define BUFFER_SIZE 10000
#define KERNEL_ERROR_MSG "Kernel module error\ndmesg for more information\n"
```

kernelmodule.c

```
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/module.h>
```

```c
#include <linux/mm_types.h>
#include <linux/sched/mm.h>
#include <linux/pid.h>
#include <linux/sched.h>
#include <linux/mount.h>
#include <linux/vmalloc.h>
#include <linux/fs.h>
#include <linux/debugfs.h>
#include <linux/uaccess.h>
#include <linux/netdevice.h>
#include "kernelmodule.h"

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Ivan Nesterov");
MODULE_DESCRIPTION("Kernel module for getting vm_area_struct and net_device
structs information via debugfs driver");
MODULE_VERSION("1.0");

/* debugfs directory and files */
static struct dentry *debugfs_dir;
static struct dentry *debugfs_vma_file;
static struct dentry *debugfs_vma_arg_file;
static struct dentry *debugfs_nds_file;

/* needed structs and arguments */
static struct vm_area_struct *vma;
static struct net_device *dev;
static pid_t pid_num;

static char buffer_k[BUFFER_SIZE] = {NULL};
static bool init_error = false;

/* function for getting task_struct */
static struct task_struct* get_ts(pid_t nr) {
    return get_pid_task(find_get_pid(nr), PIDTYPE_PID);
}

/* function for getting vm_area_struct */
static struct vm_area_struct* get_vma(struct task_struct *ts) {
    struct mm_struct* mm = get_task_mm(ts);
    struct vm_area_struct* vma = mm -> mmap;
    return vma;
}


/* utility function for sending message to user */
static void to_user(const char __user *buff, size_t count, loff_t *fpos) {
    count = strlen(buffer_k) + 1;
    copy_to_user(buff, buffer_k, count);
    *fpos += count;
}

/* overrode function to send vm_area_struct to user when he tries to read
debugfs file */
static ssize_t vma_to_user(struct file* filp, char __user* buff, size_t
count, loff_t* fpos) {
    if (*fpos > 0)  {
        return -EFAULT;
    }
    if (init_error) {
        sprintf(buffer_k, KERNEL_ERROR_MSG);
    } else {
        printk(KERN_INFO "Trying to get vm_area_struct...\n");
        struct task_struct* ts = get_ts(pid_num);
        if (ts == NULL) {
            printk(KERN_WARNING "Wrong PID\n");
            sprintf(buffer_k, KERNEL_ERROR_MSG);
        } else {
            vma = get_vma(ts);
            if (vma == NULL) {
                printk(KERN_WARNING "Cannot get vm_area_struct\n");
                sprintf(buffer_k, KERNEL_ERROR_MSG);
            } else {
                char vma_to_user[BUFFER_SIZE];
                while (vma) {
                    char current_vma_info[200];
                    char filename[50];
                    if (vma->vm_file) {
                        strcpy(filename, vma->vm_file->f_path.dentry-
>d_iname);
                    } else {
                        strcpy(filename, "[ anon ]");
                    }
                    sprintf(current_vma_info, "%#lx - %#lx, flags = %lu,
pgoff = %lu, mapped file: %s\n",
                    vma -> vm_start, vma -> vm_end, vma -> vm_flags, vma ->
vm_pgoff, filename);
                    strcat(vma_to_user, current_vma_info);
                    vma = vma->vm_next;
                }
                sprintf(buffer_k, vma_to_user);
            }
        }

    }
    to_user(buff, count, fpos);
    return *fpos;
}

/* overrode function to get PID from user when he tries to write it to
debugfs file */
static ssize_t pid_from_user(struct file* filp, const char __user* buff,
size_t count, loff_t* fpos) {
    if (BUFFER_SIZE < count) {
        printk(KERN_WARNING "Buffer size is not enough\n");
        sprintf(buffer_k, KERNEL_ERROR_MSG);
        to_user(buff, count, fpos);
        return -EFAULT;
    }
    if (*fpos > 0)  {
        return -EFAULT;
    }
    printk(KERN_INFO "Trying to get PID from user...\n");
    copy_from_user(buffer_k, buff, count);
    if (kstrtoint(buffer_k, 10, &pid_num) != 0) {
        printk(KERN_WARNING "PID is not decimal\n");
        return -EFAULT;
    }
```

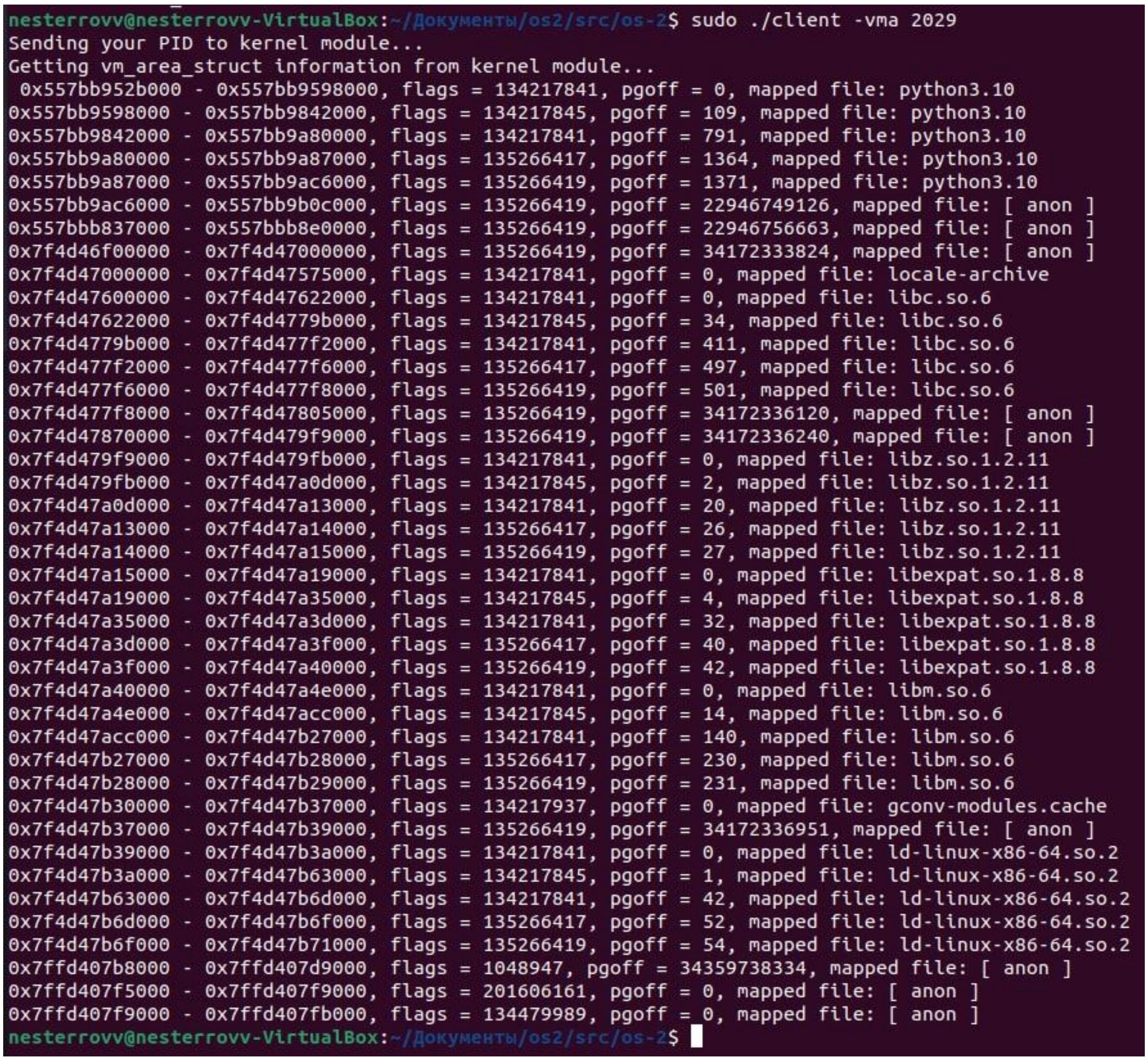
```c
    if (sscanf(buffer_k, "%d", &pid_num) < 1) {
        printk(KERN_WARNING "Cannot read PID properly\n");
        return -EFAULT;
    }
    *fpos = strlen(buffer_k);
    return *fpos;
}

/* overrode function to send net_device struct to user when he tries to read
debugfs file */
static ssize_t nds_to_user(struct file* filp, char __user* buff, size_t
count, loff_t* fpos ) {
    if (*fpos > 0) {
        return -EFAULT;
    }
    if (init_error) {
        sprintf(buffer_k, KERNEL_ERROR_MSG);
    } else {
        printk(KERN_INFO "Trying to get net_device struct...\n");
        read_lock(&dev_base_lock);
        printk(KERN_INFO "Reading...\n");
        dev = first_net_device(&init_net);
        if (dev == NULL) {
            printk(KERN_WARNING "Cannot get net_device struct\n");
            sprintf(buffer_k, KERNEL_ERROR_MSG);
        } else {
            char data_for_user[5000];
            while (dev) {
                printk(KERN_INFO "found [%s]\n", dev->name);
                char nds_info[1000];
                sprintf(nds_info, "Found network device:\n\tname:
%s\n\tmem_start: %#lx\n\tmem_end: %#lx\n\tbase_addr: %#lx\n\tirq:
%d\n\tstate: %lu\n\tMAC address: %pMF\n\tbroadcast address: %pMF\n\tflags:
%d\n\tmtu: %d\n\tmin_mtu: %d\n\tmax_mtu: %d\n\ttype: %hu\n\tmin_header_len:
%d\n\tname_assign_type: %d\n\tgroup: %d\n\tneeded_headroom:
%hu\n\tneeded_tailroom: %hu\n", dev->name, dev->mem_start, dev->mem_end,
dev->base_addr, dev->irq, dev->state, dev->dev_addr, dev->broadcast, dev-
>flags, dev->mtu, dev->min_mtu, dev->max_mtu, dev->type, dev-
>min_header_len, dev->name_assign_type, dev->group, dev->needed_headroom,
dev->needed_tailroom);
                strcat(data_for_user, nds_info);
                dev = next_net_device(dev);
            }
            sprintf(buffer_k, data_for_user);
        }
        read_unlock(&dev_base_lock);
    }
    to_user(buff, count, fpos);
    return *fpos;
}

/* overriding read and write functions for debugfs files */
static const struct file_operations vma_file_op = {
        .read = vma_to_user
};
static const struct file_operations vma_arg_file_op = {
        .write = pid_from_user
};
static const struct file_operations nds_file_op = {
        .read = nds_to_user
};

/* module initialization */
static int __init kernel_module_init(void) {
    printk(KERN_INFO "Initializing kernel module...\n");
    debugfs_dir = debugfs_create_dir(DEBUGFS_DIR_NAME, NULL);
    if (!debugfs_dir) {
        printk(KERN_WARNING "Cannot create debugfs directory\n");
        init_error = true;
        return -1;
    }
    debugfs_vma_file = debugfs_create_file(DEBUGFS_VMA_FILE_NAME, 0666,
debugfs_dir, NULL, &vma_file_op);
    if (!debugfs_vma_file) {
        printk(KERN_WARNING "Cannot create debugfs file\n");
        init_error = true;
        return -1;
    }
    debugfs_vma_arg_file = debugfs_create_file(DEBUGFS_VMA_ARG_FILE_NAME,
0666, debugfs_dir, NULL, &vma_arg_file_op);
    if (!debugfs_vma_arg_file) {
        printk(KERN_WARNING "Cannot create debugfs arg file\n");
        init_error = true;
        return -1;
    }
    debugfs_nds_file = debugfs_create_file(DEBUGFS_NDS_FILE_NAME, 0666,
debugfs_dir, NULL, &nds_file_op);
    if (!debugfs_nds_file) {
        printk(KERN_WARNING "Cannot create debugfs file\n");
        init_error = true;
        return -1;
    }
    printk(KERN_INFO "Kernel module initialized successfully\n");
    return 0;
}

/* module cleanup */
static void __exit kernel_module_exit( void ) {
    debugfs_remove(debugfs_vma_file);
    debugfs_remove(debugfs_vma_arg_file);
    debugfs_remove(debugfs_nds_file);
    debugfs_remove(debugfs_dir);
}

/* registering module init and exit functions */
module_init(kernel_module_init);
module_exit(kernel_module_exit);
```

client.c

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include "kernelmodule.h"

static void print_program_usage() {
    printf("Command: sudo ./client [-vma|-netdev] <PID>\n");
}
```

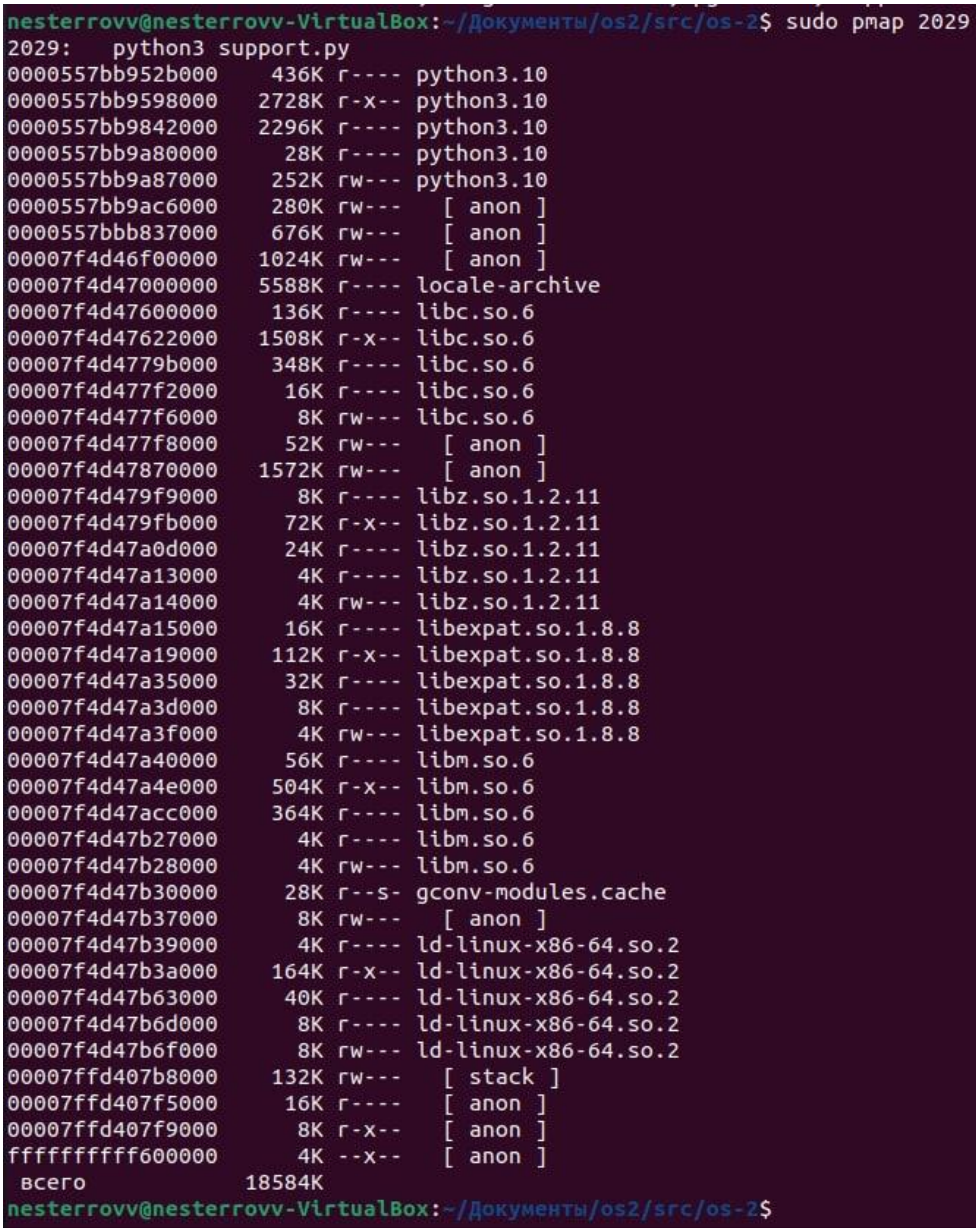
```c
int main( int argc, char *argv[] ) {
    FILE *vma_file = fopen(DEBUGFS_VMA_FILE_PATH, "r");
    FILE *vma_arg_file = fopen(DEBUGFS_VMA_ARG_FILE_PATH, "w");
    FILE *nds_file = fopen(DEBUGFS_NDS_FILE_PATH, "r");
    if ( !vma_file || !nds_file || !vma_arg_file ) {
        printf("Cannot open files for debugfs driver\n");
        return -1;
    }
    if ( argc < 2 || argc > 3 ) {
        print_program_usage();
        return -1;
    }
    char *line = NULL;
    size_t length = 0;
    if (strcmp(argv[1], "-vma") == 0 ) {
        printf("Sending your PID to kernel module...\n");
        char buf[1024];
        sprintf(buf, "%s", argv[2]);
        fwrite(&buf, 1, sizeof(buf), vma_arg_file);
        printf("Getting vm_area_struct information from kernel module...\n
");
        while (getline(&line, &length, vma_file) != -1) {
            printf("%s", line);
        }
    } else if (strcmp(argv[1], "-netdev") == 0) {
        printf("Getting net_device struct information from kernel
module...\n ");
        while (getline(&line, &length, nds_file) != -1) {
            printf("%s", line);
        }
    } else {
        print_program_usage();
        return -1;
    }
    fclose(vma_file);
    fclose(nds_file);
    fclose(vma_arg_file);

    return 0;
}
```

По смыслу задания вывод данных из цепочки vm_area_struct’ов подразумевает получение данных, схожих с данными, выдаваемыми утилитой pmap. Сравним данные:



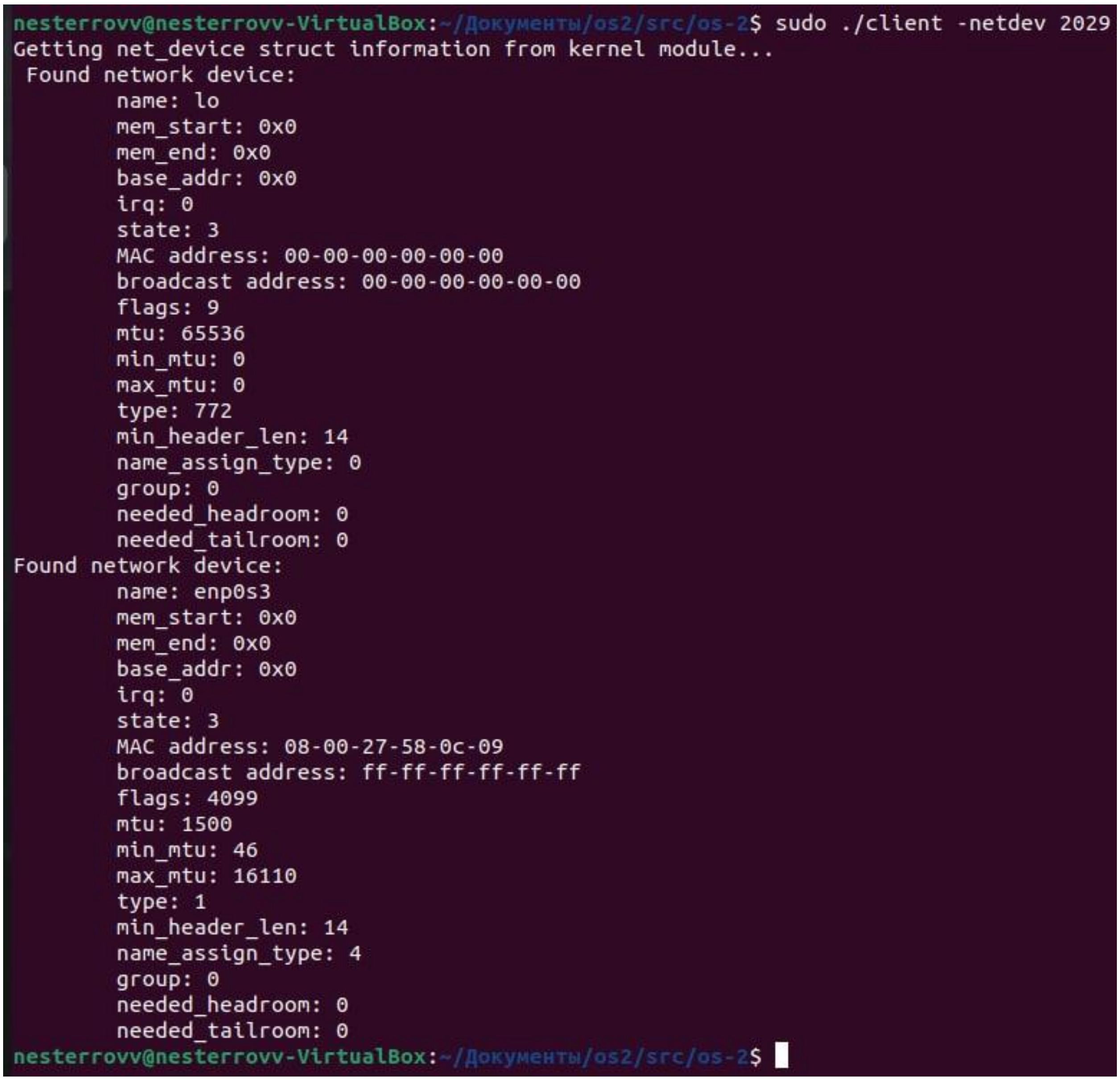
```
nesterrovv@nesterrovv-VirtualBox:~/Документы/os2/src/os-2$ sudo ./client -vma 2029
Sending your PID to kernel module...
Getting vm_area_struct information from kernel module...
 0x557bb952b000 - 0x557bb9598000, flags = 134217841, pgoff = 0, mapped file: python3.10
0x557bb9598000 - 0x557bb9842000, flags = 134217845, pgoff = 109, mapped file: python3.10
0x557bb9842000 - 0x557bb9a80000, flags = 134217841, pgoff = 791, mapped file: python3.10
0x557bb9a80000 - 0x557bb9a87000, flags = 135266417, pgoff = 1364, mapped file: python3.10
0x557bb9a87000 - 0x557bb9ac6000, flags = 135266419, pgoff = 1371, mapped file: python3.10
0x557bb9ac6000 - 0x557bb9b0c000, flags = 135266419, pgoff = 22946749126, mapped file: [ anon ]
0x557bbb837000 - 0x557bbb8e0000, flags = 135266419, pgoff = 22946756663, mapped file: [ anon ]
0x7f4d46f00000 - 0x7f4d47000000, flags = 135266419, pgoff = 34172333824, mapped file: [ anon ]
0x7f4d47000000 - 0x7f4d47575000, flags = 134217841, pgoff = 0, mapped file: locale-archive
0x7f4d47600000 - 0x7f4d47622000, flags = 134217841, pgoff = 0, mapped file: libc.so.6
0x7f4d47622000 - 0x7f4d4779b000, flags = 134217845, pgoff = 34, mapped file: libc.so.6
0x7f4d4779b000 - 0x7f4d477f2000, flags = 134217841, pgoff = 411, mapped file: libc.so.6
0x7f4d477f2000 - 0x7f4d477f6000, flags = 135266417, pgoff = 497, mapped file: libc.so.6
0x7f4d477f6000 - 0x7f4d477f8000, flags = 135266419, pgoff = 501, mapped file: libc.so.6
0x7f4d477f8000 - 0x7f4d47805000, flags = 135266419, pgoff = 34172336120, mapped file: [ anon ]
0x7f4d47870000 - 0x7f4d479f9000, flags = 135266419, pgoff = 34172336240, mapped file: [ anon ]
0x7f4d479f9000 - 0x7f4d479fb000, flags = 134217841, pgoff = 0, mapped file: libz.so.1.2.11
0x7f4d479fb000 - 0x7f4d47a0d000, flags = 134217845, pgoff = 2, mapped file: libz.so.1.2.11
0x7f4d47a0d000 - 0x7f4d47a13000, flags = 134217841, pgoff = 20, mapped file: libz.so.1.2.11
0x7f4d47a13000 - 0x7f4d47a14000, flags = 135266417, pgoff = 26, mapped file: libz.so.1.2.11
0x7f4d47a14000 - 0x7f4d47a15000, flags = 135266419, pgoff = 27, mapped file: libz.so.1.2.11
0x7f4d47a15000 - 0x7f4d47a19000, flags = 134217841, pgoff = 0, mapped file: libexpat.so.1.8.8
0x7f4d47a19000 - 0x7f4d47a35000, flags = 134217845, pgoff = 4, mapped file: libexpat.so.1.8.8
0x7f4d47a35000 - 0x7f4d47a3d000, flags = 134217841, pgoff = 32, mapped file: libexpat.so.1.8.8
0x7f4d47a3d000 - 0x7f4d47a3f000, flags = 135266417, pgoff = 40, mapped file: libexpat.so.1.8.8
0x7f4d47a3f000 - 0x7f4d47a40000, flags = 135266419, pgoff = 42, mapped file: libexpat.so.1.8.8
0x7f4d47a40000 - 0x7f4d47a4e000, flags = 134217841, pgoff = 0, mapped file: libm.so.6
0x7f4d47a4e000 - 0x7f4d47acc000, flags = 134217845, pgoff = 14, mapped file: libm.so.6
0x7f4d47acc000 - 0x7f4d47b27000, flags = 134217841, pgoff = 140, mapped file: libm.so.6
0x7f4d47b27000 - 0x7f4d47b28000, flags = 135266417, pgoff = 230, mapped file: libm.so.6
0x7f4d47b28000 - 0x7f4d47b29000, flags = 135266419, pgoff = 231, mapped file: libm.so.6
0x7f4d47b30000 - 0x7f4d47b37000, flags = 134217937, pgoff = 0, mapped file: gconv-modules.cache
0x7f4d47b37000 - 0x7f4d47b39000, flags = 135266419, pgoff = 34172336951, mapped file: [ anon ]
0x7f4d47b39000 - 0x7f4d47b3a000, flags = 134217841, pgoff = 0, mapped file: ld-linux-x86-64.so.2
0x7f4d47b3a000 - 0x7f4d47b63000, flags = 134217845, pgoff = 1, mapped file: ld-linux-x86-64.so.2
0x7f4d47b63000 - 0x7f4d47b6d000, flags = 134217841, pgoff = 42, mapped file: ld-linux-x86-64.so.2
0x7f4d47b6d000 - 0x7f4d47b6f000, flags = 135266417, pgoff = 52, mapped file: ld-linux-x86-64.so.2
0x7f4d47b6f000 - 0x7f4d47b71000, flags = 135266419, pgoff = 54, mapped file: ld-linux-x86-64.so.2
0x7ffd407b8000 - 0x7ffd407d9000, flags = 1048947, pgoff = 34359738334, mapped file: [ anon ]
0x7ffd407f5000 - 0x7ffd407f9000, flags = 201606161, pgoff = 0, mapped file: [ anon ]
0x7ffd407f9000 - 0x7ffd407fb000, flags = 134479989, pgoff = 0, mapped file: [ anon ]
nesterrovv@nesterrovv-VirtualBox:~/Документы/os2/src/os-2$
```

Рисунок 1. Вывод данных цепочки структур vm_area_struct на уровень пользователя



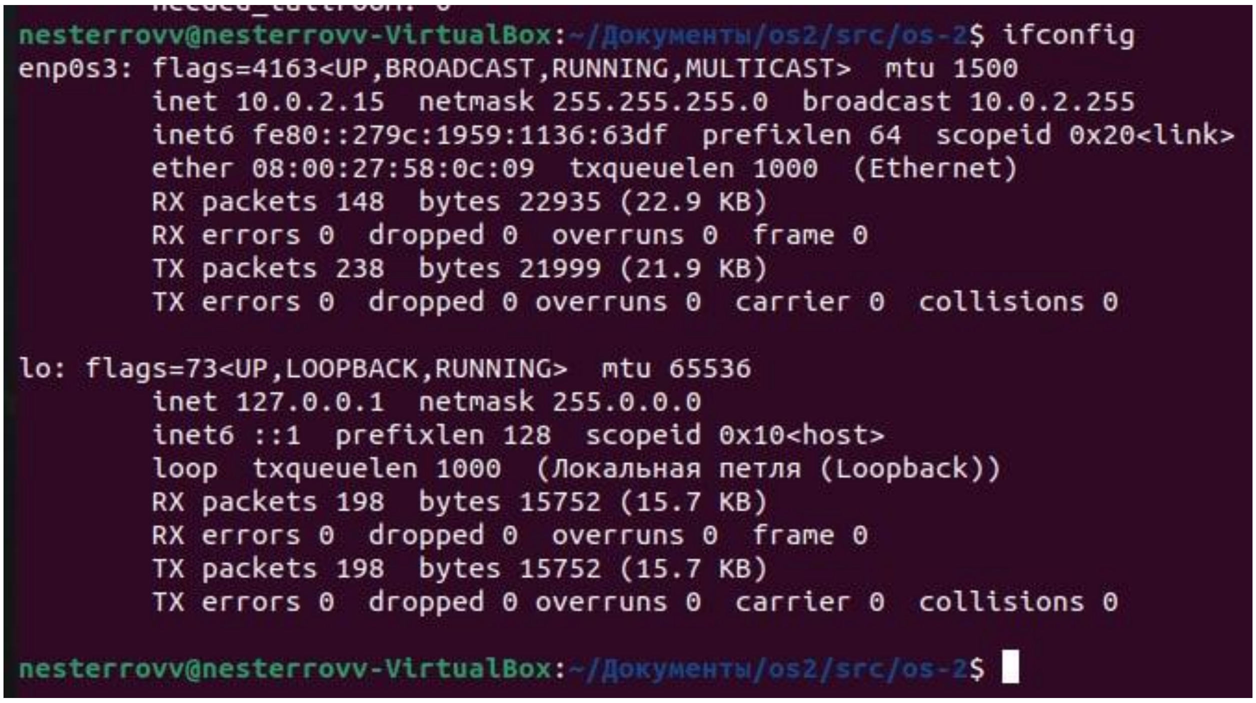
```
nesterrovv@nesterrovv-VirtualBox:~/Документы/os2/src/os-2$ sudo pmap 2029
2029:   python3 support.py
0000557bb952b000    436K r---- python3.10
0000557bb9598000   2728K r-x-- python3.10
0000557bb9842000   2296K r---- python3.10
0000557bb9a80000     28K r---- python3.10
0000557bb9a87000    252K rw--- python3.10
0000557bb9ac6000    280K rw---   [ anon ]
0000557bbb837000    676K rw---   [ anon ]
00007f4d46f00000   1024K rw---   [ anon ]
00007f4d47000000   5588K r---- locale-archive
00007f4d47600000    136K r---- libc.so.6
00007f4d47622000   1508K r-x-- libc.so.6
00007f4d4779b000    348K r---- libc.so.6
00007f4d477f2000     16K r---- libc.so.6
00007f4d477f6000      8K rw--- libc.so.6
00007f4d477f8000     52K rw---   [ anon ]
00007f4d47870000   1572K rw---   [ anon ]
00007f4d479f9000      8K r---- libz.so.1.2.11
00007f4d479fb000     72K r-x-- libz.so.1.2.11
00007f4d47a0d000     24K r---- libz.so.1.2.11
00007f4d47a13000      4K r---- libz.so.1.2.11
00007f4d47a14000      4K rw--- libz.so.1.2.11
00007f4d47a15000     16K r---- libexpat.so.1.8.8
00007f4d47a19000    112K r-x-- libexpat.so.1.8.8
00007f4d47a35000     32K r---- libexpat.so.1.8.8
00007f4d47a3d000      8K r---- libexpat.so.1.8.8
00007f4d47a3f000      4K rw--- libexpat.so.1.8.8
00007f4d47a40000     56K r---- libm.so.6
00007f4d47a4e000    504K r-x-- libm.so.6
00007f4d47acc000    364K r---- libm.so.6
00007f4d47b27000      4K r---- libm.so.6
00007f4d47b28000      4K rw--- libm.so.6
00007f4d47b30000     28K r--s- gconv-modules.cache
00007f4d47b37000      8K rw---   [ anon ]
00007f4d47b39000      4K r---- ld-linux-x86-64.so.2
00007f4d47b3a000    164K r-x-- ld-linux-x86-64.so.2
00007f4d47b63000     40K r---- ld-linux-x86-64.so.2
00007f4d47b6d000      8K r---- ld-linux-x86-64.so.2
00007f4d47b6f000      8K rw--- ld-linux-x86-64.so.2
00007ffd407b8000    132K rw---   [ stack ]
00007ffd407f5000     16K r----   [ anon ]
00007ffd407f9000      8K r-x--   [ anon ]
ffffffffff600000      4K --x--   [ anon ]
 всего            18584K
nesterrovv@nesterrovv-VirtualBox:~/Документы/os2/src/os-2$
```

Рисунок 2. Выводимые утилитой pmap данные



```
nesterrovv@nesterrovv-VirtualBox:~/Документы/os2/src/os-2$ sudo ./client -netdev 2029
Getting net_device struct information from kernel module...
 Found network device:
        name: lo
        mem_start: 0x0
        mem_end: 0x0
        base_addr: 0x0
        irq: 0
        state: 3
        MAC address: 00-00-00-00-00-00
        broadcast address: 00-00-00-00-00-00
        flags: 9
        mtu: 65536
        min_mtu: 0
        max_mtu: 0
        type: 772
        min_header_len: 14
        name_assign_type: 0
        group: 0
        needed_headroom: 0
        needed_tailroom: 0
Found network device:
        name: enp0s3
        mem_start: 0x0
        mem_end: 0x0
        base_addr: 0x0
        irq: 0
        state: 3
        MAC address: 08-00-27-58-0c-09
        broadcast address: ff-ff-ff-ff-ff-ff
        flags: 4099
        mtu: 1500
        min_mtu: 46
        max_mtu: 16110
        type: 1
        min_header_len: 14
        name_assign_type: 4
        group: 0
        needed_headroom: 0
        needed_tailroom: 0
nesterrovv@nesterrovv-VirtualBox:~/Документы/os2/src/os-2$
```

Рисунок 3. Вывод данных из набора струтур net_device



```
nesterrovv@nesterrovv-VirtualBox:~/Документы/os2/src/os-2$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::279c:1959:1136:63df  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:58:0c:09  txqueuelen 1000  (Ethernet)
        RX packets 148  bytes 22935 (22.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 238  bytes 21999 (21.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Локальная петля (Loopback))
        RX packets 198  bytes 15752 (15.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 198  bytes 15752 (15.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

nesterrovv@nesterrovv-VirtualBox:~/Документы/os2/src/os-2$
```

Рисунок 4. Выводимые утилитой ifconfig данные

Вывод: в ходе работы были рассмотрены некоторые структуры из заголовочных файлов Linux, а также разработать комплекс программ на пользовательском уровне и уровне ярда, который собирает информацию на стороне ядра о структурах vm_area_struct и net_device и передает информацию на уровень пользователя при помощи интерфейса debugfs, и выводит ее в удобном для чтения человеком виде.