Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет

к лабораторной работе №4
«Виртуальная файловая систицма/proc»

Студент: Батбилэг Н.

Группа: ИУ7-71Б

Оценка (баллы) _____

Преподаватель: Рязанова Н.Ю.

Москва.
2021 г.

1.1 В пользовательском режиме вывести на экран информацию об окружении процесса.
В листинге 1 представлен код вывода информации об окружении процесса.

Листинг 1. prog.c

```c
#include <stdio.h>

#define BUF_SIZE 512

int main()
{
    FILE *f = fopen("/proc/self/environ", "r");
    int len = 0;
    char buf[BUF_SIZE];
    while ((len = fread(buf, 1, BUF_SIZE, f)) > 0)
    {
        for (int i = 0; i < len; i++)
            if (buf[i] == 0)
                buf[i] = 10;
            buf[len - 1] = 0;
            printf("%s\n", buf);
        }
    fclose(f);
}
```

показать корневой каталог, который создался в файловой системе.

Показать в виде дерева каталогов

На рисунках 1, 2 представлен результат работы программы (информация об окружении процесса)

```
dalai@dalai-Inspiron-5567:~/Documents/BMSTU/labs/lab_04$ gcc prog.c
dalai@dalai-Inspiron-5567:~/Documents/BMSTU/labs/lab_04$ ls -a
.  ..  a.out  lab_4_part2.c  prog.c
dalai@dalai-Inspiron-5567:~/Documents/BMSTU/labs/lab_04$ ./a.out
SHELL=/bin/bash
SESSION_MANAGER=local/dalai-Inspiron-5567:@/tmp/.ICE-unix/2921,unix/dalai-Inspiron-5567:/tmp/
.ICE-unix/2921
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
NVM_INC=/home/dalai/.nvm/versions/node/v16.5.0/include/node
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
LC_ADDRESS=mn_MN
GNOME_SHELL_SESSION_MODE=ubuntu
LC_NAME=mn_MN
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
LC_MONETARY=mn_MN
SSH_AGENT_P
D=2880
GTK_MODULES=gail:atk-bridge
PWD=/home/dalai/Documents/BMSTU/labs/lab_04
LOGNAME=dalai
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=x11
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
XAUTHORITY=/run/user/1000/gdm/Xauthority
WINDOWPATH=2
HOME=/home/dalai
USERNAME=dalai
IM_CONFIG_PHASE=1
LC_PAPER=mn_MN
LANG=en_US.UTF-8
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=
40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31
*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01
;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;
31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01
;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear
=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.
```

Рисунок 1. Информация об окружении процесса (часть 1).

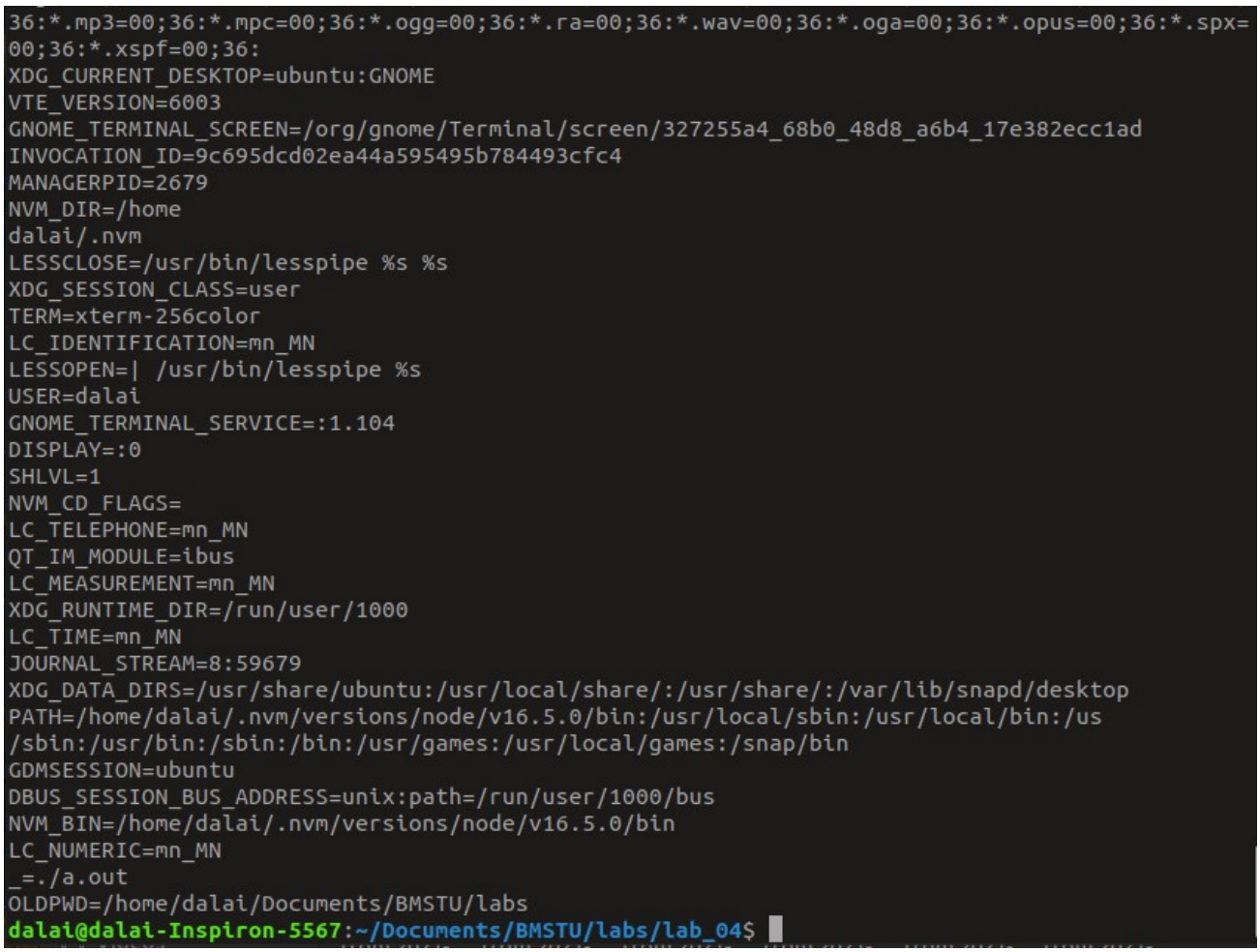

Рисунок 2. Информация об окружении процесса (часть 2).

1.2 В пользовательском режиме вывести на экран информацию о состоянии процесса.
В листинге 2 представлен код вывода информации о состоянии процесса. Для наглядности в вывод была добавлена информация о том, что именно выводится в каждой конкретной строке (pid, comm и т. д.)

Листинг 2. stat.c

```c
#include <stdio.h>
#include <string.h>
#define BUF_SIZE 512
int main()
{
    char *arr[52] = {"pid", "comm", "state", "ppid", "pgrp",
    "session", "tty_nr", "tpgid",
    "flags", "minflt", "cminflt", "majflt", "cmajflt", "utime",
    "stime", "cutime",
    "cstime", "priority", "nice", "num_threads", "itrealvalue",
    "starttime", "vsize",
    "rss", "rsslim", "startcode", "endcode", "startstack",
    "kstkesp", "kstkeip", "signal",
    "blocked", "sigignore", "sigcatch", "wchan", "nswap",
    "cnswap", "exit_signal",
    "processor", "rt_priority", "policy",
    "delayacct_blkio_ticks", "guest_time",
    "cguest_time", "start_data", "end_data", "start_brk",
    "arg_start", "arg_end",
    "env_start", "env_end", "exit_code"};
    FILE *f = fopen("/proc/self/stat", "r");
    char buf[BUF_SIZE];
    fread(buf, 1, BUF_SIZE, f);
    char *pch = strtok(buf, " ");
    printf("stat: \n");
    int i = 0;
    while(pch != NULL)
    {
        printf("%s = %s\n", arr[i++], pch);
        pch = strtok(NULL, " ");
    }
    fclose(f);
}
```

На рисунках 3, 4 представлен результат работы программы (информации о состоянии процесса)

```
dalai@dalai-Inspiron-5567:~/Documents/BMSTU/labs/lab_04$ ./stat.out
stat:
pid = 15406
comm = (stat.out)
state = R
ppid = 5290
pgrp = 15406
session = 5290
tty_nr = 34816
tpgid = 15406
flags = 4194304
minflt = 71
cminflt = 0
majflt = 0
cmajflt = 0
utime = 0
stime = 0
cutime = 0
cstime = 0
priority = 20
nice = 0
num_threads = 1
itrealvalue = 0
starttime = 2711219
vsize = 2555904
rss = 145
rsslim = 18446744073709551615
startcode = 94162968576000
endcode = 94162968576965
startstack = 140722987189856
kstkesp = 0
kstkeip = 0
signal = 0
blocked = 0
sigignore = 0
sigcatch = 0
wchan = 0
nswap = 0
cnswap = 0
exit_signal = 17
processor = 3
rt_priority = 0
policy = 0
```

Рисунок 3. Информация о состоянии процесса (часть 1).

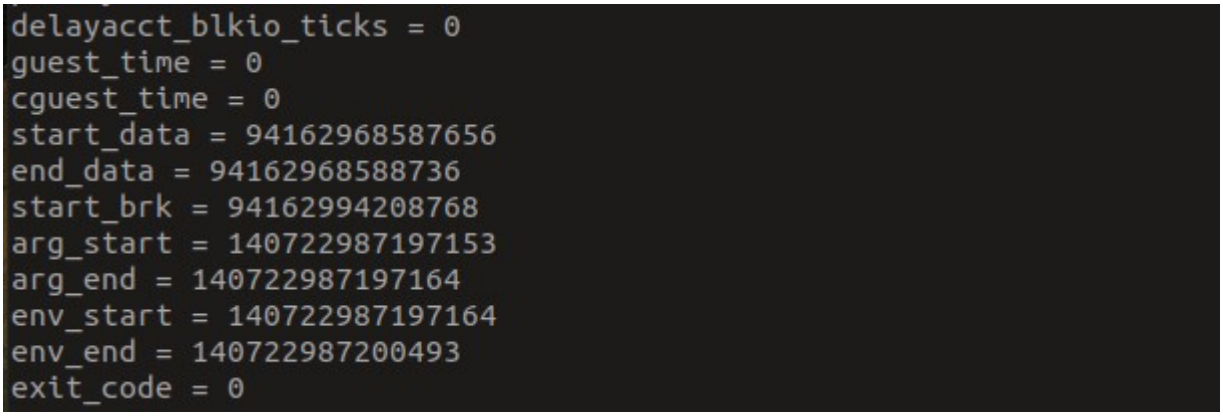

Рисунок 4. Информация о состоянии процесса (часть 2).

1.3 Вывести информацию о файле cmdfile и директории fd.
В листинге 3 представлен код вывода информации о файле cmdfile и директории fd.

Листинг 3. Cmdfile и fd.

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
#define BUFF_SIZE 512
#define PID_LEN 30

int main()
{
    char path[PID_LEN];
    sprintf(path, "/proc/%d/fd", getpid());
    printf("fd:\n\n");

    pid_t pid = fork();

    if (pid == 0) {
        if (execlp("/bin/ls", "ls", "-l", path, NULL) == -1) {
        printf("Error with \"%s\"!\n", path);
        }
    } else {
        int status = 0;
        pid_t wpid = wait(&status);
    }
```

На рисунке 5 представлен результат программы (информация о cmdfile и fd)

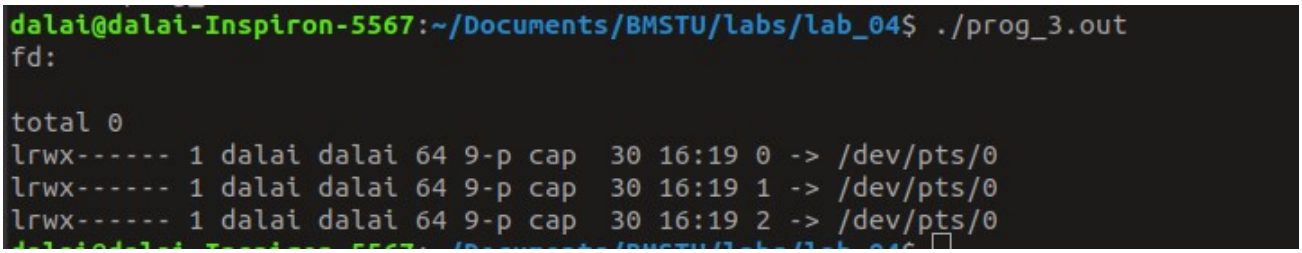

Рисунок 5. Информация о cmdfile и fd

2. Написать программу — загружаемый модуль ядра (LKM) — которая поддерживает чтение из пространства пользователя и запись в пространство пользователя. В программе необходимо создать поддиректорию и символическую ссылку.

В листинге 4 представлен код загружаемого модуля ядра fortune.c

```c
#include <linux/string.h>
#include <linux/module.h>
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/proc_fs.h>
#include <linux/vmalloc.h>
#include <asm/uaccess.h>
#include <linux/uaccess.h>

#define OK 0

#define MAX_COOKIE_LEN PAGE_SIZE
#define BUF_SIZE 100
#define FILE_NAME "fortune"
#define DIR_NAME "fortune_dir"
#define SYMLINK_NAME "fortune_symlink"


MODULE_LICENSE ("BSD");
MODULE_AUTHOR ("Nomuundalai B.");

static struct proc_dir_entry *proc_entry;

ssize_t fortune_write (struct file *filp, const char __user *buf, unsigned long count,
                                                                  loff_t *offp);

ssize_t fortune_read(struct file *filp, char __user *buf, unsigned long count,
                                                                  loff_t *offp);

struct file_operations fops = {
    .owner = THIS_MODULE,
    .read = fortune_read,
    .write = fortune_write,
};

static char *cookie_pot;

```

```c
// Управляющие индексы
static int cookie_index;
static int next_fortune;
static char tmp_buf[BUF_SIZE];

// Запись
ssize_t fortune_write (struct file *filp, const char __user *buf, size_t len,loff_t *offp)
{
    int space_available = (MAX_COOKIE_LEN - cookie_index) + 1;
    if (len > space_available) {
    printk(KERN_INFO "fortune: cookie_pot is full!\n");
    }
    // Копирует из пространства пользователя в буфер ядра
    if (copy_from_user(&cookie_pot[cookie_index], buf, len)){
        return -EFAULT;
    }
    cookie_index += len;
    cookie_pot[cookie_index - 1] = 0;

    return len;
}

// Чтение
ssize_t fortune_read (struct file *filp, char __user *buf, size_t count, loff_t *offp) {
    int len = 0;
    if (!cookie_index || *offp > 0) {
        return 0;
    }

    len = sprintf(tmp_buf, "%s\n", &cookie_pot[next_fortune]);
    // Копирует из пространства ядра в пространство пользователя
    copy_to_user(buf, tmp_buf, len);
    next_fortune += len;
    *offp += len;

    return len;
}

// Инициализация
int init_fortune_module(void) {
    int ret = OK;
    // Выделение пространства для "cookie pot"
    cookie_pot = (char *)vmalloc(MAX_COOKIE_LEN);
    if (!cookie_pot) {
        ret = -ENOMEM;
    }
    else {
        //Очистка пространства "cookie pot"
        memset(cookie_pot, 0, MAX_COOKIE_LEN);
        proc_entry = proc_create("fortune", 0666, NULL, &fops);
        if(!proc_entry) {
            ret = -ENOMEM;
            vfree(cookie_pot);
            printk(KERN_INFO "fortune: Couldn't create proc
            entry\n");
        }
        else {
            // Управляющие индексы, устанавливаются в начальное положение
            cookie_index = 0;
            next_fortune = 0;
```

```
            proc_mkdir(DIR_NAME, NULL);
            proc_symlink(SYMLINK_NAME, NULL, "/proc/" FILE_NAME);
            printk(KERN_INFO "fortune: Module loaded.\n");
        }
    }
    return ret;
}

void cleanup_fortune_module(void) {
    remove_proc_entry(SYMLINK_NAME, NULL);
    remove_proc_entry(FILE_NAME, NULL);
    remove_proc_entry(DIR_NAME, NULL);

    if (cookie_pot) {
        vfree(cookie_pot);
    }

    printk(KERN_INFO "fortune: Module unloaded.\n");
}

module_init(init_fortune_module);
module_exit(cleanup_fortune_module);
```

Данный исходный текст собирался с помощью makefile. Листинг его кода представлен ниже.

Листинг 5. Makefile.

```
ifneq ($(KERNELRELEASE),)
        obj-m := fortune.o
else
        CURRENT = $(shell uname -r)
        KDIR = /lib/modules/$(CURRENT)/build
        PWD = $(shell pwd)

default:
        $(MAKE) -C $(KDIR) M=$(PWD) modules
# sudo make clean

clean:
        rm -rf .tmp_versions
        rm .fortune.*
        rm *.o
        rm *.mod.c
        rm *.symvers
        rm *.order

endif
```

На рисунке 6 представлен процесс сборки и загрузки модуля ядра fortune.