



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»

**Лабораторная работа № 4**

Дисциплина Операционные системы

Тема Виртуальная файловая система /proc

Студент Ильясов И. М.

Группа ИУ7-63Б

Оценка (баллы) \_\_\_\_\_

Преподаватель Рязанова Н. Ю.

Москва, 2020 г.

## Задание 1.

- В пользовательском режиме вывести на экран информацию об окружении процесса с комментариями;
- В пользовательском режиме вывести на экран информацию о состоянии процесса с комментариями;
- Вывести информацию из файла `cmdline` и директории `fd`.

Ниже на рисунке 1 приведен результат вывода на экран информации об окружении процесса (`/proc/self/environ`).

```
-----/proc/self/environ-----
CLUTTER_IM_MODULE=xtn
LS_COLORS=rs=0:di=01;34:ln=01;36:rnh=00:pt=40;33:son=01;35:dov=01;35:bd=40;33:01:cd=40;33:01:or=40;31:01:ml=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:* rj=01;31:* txx=01;31:* l=01;31:* l4=01;31:* lz=01;31:* lma=01;31:* tlz=01;31:* txx=01;31:* tzo=01;31:* t7z=01;31:* zip=01;31:* z=01;31:* Z=01;31:* dz=01;31:* gz=01;31:* lz=01;31:* lzo=01;31:* lz=01;31:* lzo=01;31:* xz=01;31:* zst=01;31:* tzt=01;31:* bzip=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:* cpt=01;31:* 7z=01;31:* rz=01;31:* cab=01;31:* xim=01;31:* xum=01;31:* dw=01;31:* esd=01;31:* jpe=01;35:* jpg=01;35:* njpg=01;35:* gif=01;35:* bmp=01;35:* pbm=01;35:* ppm=01;35:* tga=01;35:* xbm=01;35:* xpm=01;35:* ttf=01;35:* ttf=01;35:* png=01;35:* svga=01;35:* svga=01;35:* mqv=01;35:* pqa=01;35:* mov=01;35:* mpg=01;35:* mpeg=01;35:* mva=01;35:* webm=01;35:* ogm=01;35:* mpa=01;35:* mpa=01;35:* vob=01;35:* qt=01;35:* mov=01;35:* a
sfx=01;35:* rrm=01;35:* rrvb=01;35:* flc=01;35:* avi=01;35:* flv=01;35:* gl=01;35:* dl=01;35:* xcf=01;35:* xwd=01;35:* yuv=01;35:* cgm=01;35:* enf=01;35:* ogv=01;35:* ogx=01;35:* aac=00;36:* au=00;36:* flac=00;36:* m4
e=00;36:* mid=00;36:* mtd=00;36:* mka=00;36:* mp3=00;36:* nps=00;36:* ogg=00;36:* ra=00;36:* wav=00;36:* oga=00;36:* opus=00;36:* spx=00;36:* xspf=00;3:
LESSCLOSE=/usr/bin/lesspipe ks ks
XDG_MENU_PREFIX=gnome-
LANG=en_US.UTF-8
WINDOWID=2192
DISPLAY=:0
OLDPWD=/home/parallels/OS/lab_04
INVOCATION_ID=b976723b941aa7e49aeb3d85e751d08
GNOME_SHELL_SESSION_MODE=ubuntu
COLORTERM=truecolor
USER=parallels
XC_VTNR=2
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XDG_SESSION_ID=3
USER=parallels
DESKTOP_SESSION=ubuntu
GTK_IM_MODULE=xtn
TEXTDOMAIN=ibus:/usr/share/locale/
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/8edf7120_00b0_4107_8dcb_3b2506bee5ed
PWD=/home/parallels/OS/lab_04/task1
HOME=/home/parallels
JOURNAL_STREAM=9:34043
TEXTDOMAIN=ntn-config
SSH_AGENT_PID=2318
QT_ACCESSIBILITY=1
XDG_SESSION_TYPE=x11
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share/:/usr/share/:/var/lib/snapd/desktop
XDG_SESSION_DESKTOP=ubuntu
DBUS_STARTER_ADDRESS=unix:path=/run/user/1000/bus,guid=b976723b941aa7e49aeb3d85e751d08
GTK_MODULES=gall:atk-bridge
WINDOWPATH=2
TERM=xterm-256color
SHELL=/bin/bash
VTE_VERSION=5202
QT_IM_MODULE=xtn
MODIFIERS=@=ibus
IM_CONFIG_PHASE=2
DBUS_STARTER_BUS_TYPE=session
XDG_CURRENT_DESKTOP=ubuntu:GNOME
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
GNOME_TERMINAL_SERVICE=1.70
XDG_SEAT=seat0
SHLV=1
CONNECTION=ubuntu
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
LOGNAME=parallels
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus,guid=b976723b941aa7e49aeb3d85e751d08
XDG_RUNTIME_DIR=/run/user/1000
XAUTHORITY=/run/user/1000/gdm/Authority
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
SESSION_MANAGER=local/parallels-Parallels-Virtual-Platform:@/tmp/.ICE-unix/2223,unix/parallels-Parallels-Virtual-Platform:/tmp/.ICE-unix/2223
LESSOPEN=| /usr/bin/lesspipe ks
GTK_IM_MODULE=ibus
_=/usr/bin/ls
```

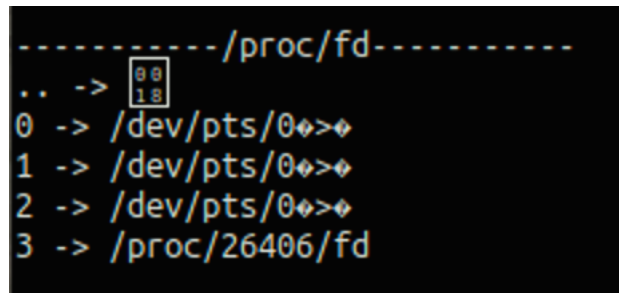
Рисунок 1. Информация об окружении процесса (`/proc/self/environ`).

На рисунке 2 приведен результат вывода на экран информации об состоянии процесса (/proc/self/stat).

```
-----/proc/stat-----
pid ==> 26833
comm ==> (1.exe)
state ==> R
ppid ==> 19211
pgrp ==> 26833
session ==> 19211
tty_nr ==> 34816
tpgid ==> 26833
flags ==> 4194304
minflt ==> 70
cmiflt ==> 0
majflt ==> 0
cmajflt ==> 0
utime ==> 0
stime ==> 0
cutime ==> 0
cstime ==> 0
priority ==> 20
nice ==> 0
num_threads ==> 1
itrealvalue ==> 0
starttime ==> 716802
vsize ==> 4620288
rss ==> 193
rsslim ==> 18446744073709551615
startcode ==> 94355229310976
endcode ==> 94355229317288
startstack ==> 140726370128272
kstkesp ==> 0
kstkeip ==> 0
signal ==> 0
blocked ==> 0
sigignore ==> 0
sigcatch ==> 0
wchan ==> 0
nswap ==> 0
cnswap ==> 0
exit_signal ==> 17
processor ==> 0
rt_priority ==> 0
policy ==> 0
delayacct_blkio_ticks ==> 0
guest_time ==> 0
cguest_time ==> 0
start_data ==> 94355231415632
end_data ==> 94355231416768
start_brk ==> 94355240660992
arg_start ==> 140726370136000
```

Рисунок 2. Информация об состоянии процесса (/proc/self/stat).

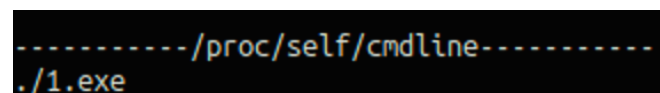
На рисунке 3 приведен результат вывода на экран содержания директории fd (/proc/self/fd).



```
-----/proc/fd-----
.. -> 00
    18
0 -> /dev/pts/0>>
1 -> /dev/pts/0>>
2 -> /dev/pts/0>>
3 -> /proc/26406/fd
```

Рисунок 3. Содержание директории fd (/proc/self/fd).

На рисунке 4 приведен результат вывода на экран содержания файла cmdline (/proc/self/cmdline).



```
-----/proc/self/cmdline-----
./1.exe
```

Рисунок 4. Содержание файла cmdline (/proc/self/cmdline).

Далее приведен листинг программы из задания №1:

```
#include <stdio.h>
#include <string.h>
#include <dirent.h>
#include <unistd.h>

#define BUF_SIZE 0x100
#define SUCCESS 0
#define FILE_ERROR -1
#define FREAD_ERROR -2
#define SPRINTF_ERROR -3
#define READLINK_ERROR -4
#define OPEN_DIR_ERROR -5
#define CLOSE_DIR_ERROR -6

char *attr[] = {"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"};

int print_file(char *name)
{
    char buf[BUF_SIZE];
    int len = 0;
    FILE *f = NULL;
    f = fopen(name, "r");

    if (!f)
        return FILE_ERROR;

    printf("\n-----%s-----\n", name);

    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", buf); }

    if (fclose(f) != 0)
        return FILE_ERROR;

    printf("\n");
    return SUCCESS;
}

int print_stat()
{
    char buf[BUF_SIZE];
    FILE *f = NULL;
    char *pch = NULL;
    int i = 0;

    f = fopen("/proc/self/stat", "r");
    if (!f)
        return FILE_ERROR;
    if (fread(buf, 1, BUF_SIZE, f) <= 0)
        return FREAD_ERROR;

    printf("\n-----stat-----\n");

    pch = strtok(buf, " ");

    while (pch != NULL)
    {
        printf("%s ==> %s\n", attr[i], pch);
        pch = strtok(NULL, " ");
        i++;
    }

    if (fclose(f) != 0)
        return FILE_ERROR;

    return SUCCESS;
}

int print_fd()
{
    struct dirent *dirp = NULL;
    DIR *dp = NULL;
    char str[BUF_SIZE];
    char path[BUF_SIZE];

    dp = opendir("/proc/self/fd");

    if (!dp)
        return OPEN_DIR_ERROR;

    printf("\n-----fd-----\n");

    while ((dirp = readdir(dp)) != NULL)
    {
        if ((strcmp(dirp->d_name, ".") != 0) && (strcmp(dirp->d_name, "..") != 0))
        {

```

```

        if (sprintf(path, "%s%s", "/proc/self/fd/", dirp->d_name) < 0)
            return SPRINTF_ERROR;

        readlink(path, str, BUF_SIZE);
        printf("%s -> %s\n", dirp->d_name, str);
    }
}

if (closedir(dp) < 0)
    return CLOSE_DIR_ERROR;

return SUCCESS;
}

int main(int argc, char **argv)
{
    int err = 0;

    if ((err = print_file("/proc/self/environ")))
        return err;
    if ((err = print_stat()))
        return err;
    if ((err = print_fd()))
        return err;
    if ((err = print_file("/proc/self/cmdline")))
        return err;

    return err;
}

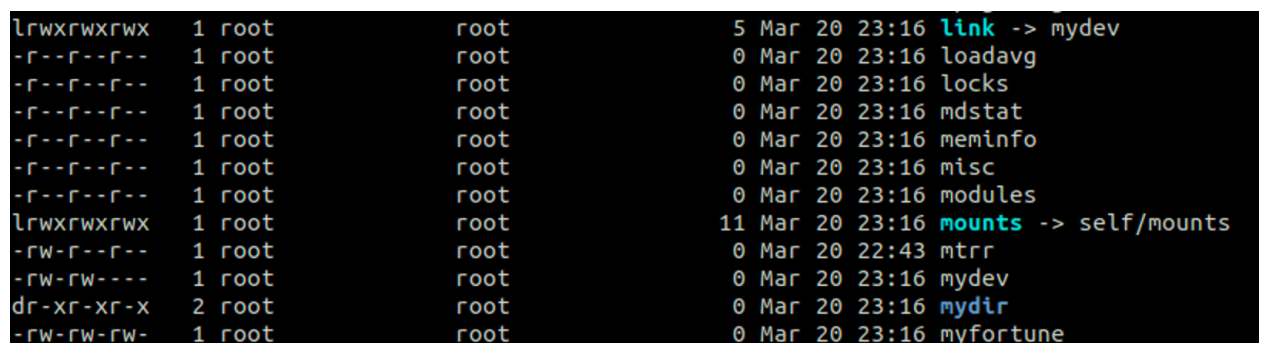
```

## Задание 2.

Написать программу – загружаемый модуль ядра (LKM) – которая поддерживает чтение из пространства пользователя и запись в пространство пользователя. После загрузки модуля пользователь может загружать в него строки с помощью команды `echo`, а затем автоматически считывать их с помощью команды `cat`.

В программе необходимо создать поддиректорию и символическую ссылку.

На рисунке 5 представлен частичный вывод команды `ls` для демонстрации создания файла (`mydev`), символической ссылки (`link`) и директории (`mydir`) в файловой системе `proc`.



```

lrwxrwxrwx  1 root      root           5 Mar 20 23:16 link -> mydev
-r--r--r--  1 root      root            0 Mar 20 23:16 loadavg
-r--r--r--  1 root      root            0 Mar 20 23:16 locks
-r--r--r--  1 root      root            0 Mar 20 23:16 mdstat
-r--r--r--  1 root      root            0 Mar 20 23:16 meminfo
-r--r--r--  1 root      root            0 Mar 20 23:16 misc
-r--r--r--  1 root      root            0 Mar 20 23:16 modules
lrwxrwxrwx  1 root      root          11 Mar 20 23:16 mounts -> self/mounts
-rw-r--r--  1 root      root            0 Mar 20 22:43 mtrr
-rw-rw----  1 root      root            0 Mar 20 23:16 mydev
dr-xr-xr-x  2 root      root            0 Mar 20 23:16 mydir
-rw-rw-rw-  1 root      root            0 Mar 20 23:16 myfortune

```

Рисунок 5. Частичный вывод `ls`.

Далее приведен листинг модуля, где создается файл, символическая ссылка и директория в файловой системе proc.

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

#define BUFSIZE 100

MODULE_LICENSE("Dual BSD/GPL");
MODULE_AUTHOR("Ilyasov Idris");

static struct proc_dir_entry * ent;

static ssize_t mywrite(struct file *file, const char __user *ubuf, size_t count, loff_t *ppos)
{
    printk(KERN_DEBUG "write handler\n");
    return -1;
}

static ssize_t myread(struct file *file, char __user *ubuf, size_t count, loff_t *ppos)
{
    printk(KERN_DEBUG "read handler\n");
    return 0;
}

static struct file_operations myops =
{
    .owner = THIS_MODULE,
    .read = myread,
    .write = mywrite
};

static int simple_init(void)
{
    ent = proc_create("mydev", 0660, NULL, &myops);
    ent = proc_symlink("link", NULL, "mydev");
    ent = proc_mkdir("mydir", NULL);
    return 0;
}

static void simple_cleanup(void)
{
    proc_remove(ent);
}

module_init(simple_init);
module_exit(simple_cleanup);
```

Ниже представлен результат работы программы, в которой данные передаются из пространства пользователя и из пространства ядра в пространство пользователя.

```
parallels@parallels-Parallels-Virtual-Platform:~/OS/lab_04/task2$ sudo insmod main.ko
[sudo] password for parallels:
parallels@parallels-Parallels-Virtual-Platform:~/OS/lab_04/task2$ echo "Success is an individual proposition. Thomas Watson" > /proc/myfortune
parallels@parallels-Parallels-Virtual-Platform:~/OS/lab_04/task2$ echo "If a man does his best, what else is there? Gen. Patton" > /proc/myfortune
parallels@parallels-Parallels-Virtual-Platform:~/OS/lab_04/task2$ echo "Cats: All your base are belong to us. Zero Wing" > /proc/myfortune
parallels@parallels-Parallels-Virtual-Platform:~/OS/lab_04/task2$ cat /proc/myfortune
Success is an individual proposition. Thomas Watson
parallels@parallels-Parallels-Virtual-Platform:~/OS/lab_04/task2$ cat /proc/myfortune
If a man does his best, what else is there? Gen. Patton
parallels@parallels-Parallels-Virtual-Platform:~/OS/lab_04/task2$
```

Рисунок 6. Пример результата работы программы, в которой данные передаются из пространства пользователя и из пространства ядра в пространство пользователя.

Далее приведен листинг программы, в которой данные передаются из пространства пользователя и из пространства ядра в пространство пользователя.

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

#define MAX_COOKIE_LENGTH PAGE_SIZE

MODULE_LICENSE("Dual BSD/GPL");
MODULE_AUTHOR("Ilyasov Idris");

static struct proc_dir_entry *proc_entry;
char *cookie_pot;
int cookie_index;
int next_fortune;
char buf[256];

static ssize_t fortune_write(struct file *file, const char __user *ubuf, size_t count, loff_t *ppos)
{
    if (count > MAX_COOKIE_LENGTH - cookie_index + 1)
    {
        printk(KERN_DEBUG "Big count\n");
        return -ENOSPC;
    }

    if (copy_from_user(cookie_pot + cookie_index, ubuf, count))
    {
        printk(KERN_DEBUG "Copy_from_user error\n");
        return -EFAULT;
    }

    cookie_index += count;
    cookie_pot[cookie_index - 1] = 0;
    printk(KERN_DEBUG "Good write\n");

    return count;
}

static ssize_t fortune_read(struct file *file, char __user *ubuf, size_t count, loff_t *ppos)
{
    int len = 0;

    if (*ppos > 0)
```



```

    return 0;

if (next_fortune >= cookie_index)
{
    next_fortune = 0;
}

if (cookie_index > 0)
{
    len = sprintf(buf, "%s\n", cookie_pot + next_fortune);
    copy_to_user(ubuf, buf, len);
    next_fortune += len;
    ubuf += len;
    printk(KERN_DEBUG "Len: %d\n", len);
    *ppos += len;
}

return len;
}

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

static int simple_init(void)
{
    cookie_pot = (char *)vmalloc(MAX_COOKIE_LENGTH);

    if (!cookie_pot)
        return -ENOMEM;

    memset(cookie_pot, 0, MAX_COOKIE_LENGTH);
    proc_entry = proc_create("myfortune", 0666, NULL, &myops);

    if (proc_entry == NULL)
    {
        vfree(cookie_pot);
        printk(KERN_DEBUG "fortune: Couldn't create proc entry\n");
        return -ENOMEM;
    }

    cookie_index = 0;
    next_fortune = 0;
    printk(KERN_DEBUG "fortune: Init\n");

    proc_symlink("link", NULL, "mydev");
    proc_mkdir("mydir", NULL);

    return 0;
}

static void simple_cleanup(void)
{
    proc_remove(proc_entry);
    vfree(cookie_pot);
    printk(KERN_DEBUG "fortune: Clean\n");
}

module_init(simple_init);
module_exit(simple_cleanup);

```