



**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  
По курсу «Операционные системы»**

**Тема** Виртуальная файловая система /proc

**Студент** Неклепаева А. Н.

**Группа** ИУ7-63б

**Преподаватель** Рязанова Н. Ю.

Москва.  
2020 г.

### Задание № 1:

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

Листинг программы к заданию 1

```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <dirent.h>
#include <unistd.h>

#define BUF_SIZE 0x100

int read_env()
{
    char buf[BUF_SIZE];
    int len, i;
    FILE *f = fopen("/proc/self/environ", "r");

    if (!f)
    {
        printf("env: fopen err\n");
        return -1;
    }

    while ((len = fread(buf, 1, BUF_SIZE, f)) > 0)
    {
        for (i = 0; i < len; i++)
            if (buf[i] == 0)
                buf[i] = 10;
        buf[len - 1] = 0;
        printf("%s", buf);
    }

    if (fclose(f))
    {
        printf("env: fclose err\n");
        return -1;
    }

    return 0;
}

int read_cmdline()
{
    char buf[BUF_SIZE];
    int len, i;
    FILE *f = fopen("/proc/self/cmdline", "r");

    if (!f)
```

```

{
    printf("cmdline: fopen err\n");
    return -1;
}

while ((len = fread(buf, 1, BUF_SIZE, f)) > 0)
{
    for (i = 0; i < len; i++)
        if (buf[i] == 0)
            buf[i] = 10;
    buf[len - 1] = 0;
    printf("%s", buf);
}

if (fclose(f))
{
    printf("cmdline: fclose err\n");
    return -1;
}

return 0;
}

int read_stat()
{
    char *records[] = {"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",
"cnsnap", "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 i;
    char buf[BUF_SIZE];
    FILE *f = fopen("/proc/self/stat", "r");

    if (!f)
    {
        printf("stat: fopen err\n");
        return -1;
    }

    fread(buf, 1, BUF_SIZE, f);
    char *pch = strtok(buf, " ");

    i = 0;
    while (pch != NULL)
    {
        printf("%s = %s\n", records[i], pch);
        pch = strtok(NULL, " ");
        i++;
    }

    if (fclose(f))
    {
        printf("stat: fclose err\n");
        return -1;
    }
}

```

```

    }
    return 0;
}

int read_fd()
{
    struct dirent *dirp;
    DIR *dp;
    char str[BUF_SIZE];
    char path[BUF_SIZE];
    dp = opendir("/proc/self/fd");//открыть директорию

    if (!dp)
    {
        printf("fd: opendir err\n");
        return -1;
    }

    while ((dirp = readdir(dp)) != NULL)//читаем директорию
    {
        if ((strcmp(dirp->d_name, ".") != 0) &&
            (strcmp(dirp->d_name, "..") != 0))
        {
            sprintf(path, "%s%s", "/proc/self/fd/", dirp->d_name);
            readlink(path, str, BUF_SIZE);
            printf("%s -> %s\n", dirp->d_name, str);
        }
    }

    if (closedir(dp))
    {
        printf("fd: closedir err\n");
        return -1;
    }

    return 0;
}

int main(int argc, char *argv[])
{
    printf("ENVIRON\n");
    if (read_env())
        return -1;

    printf("\n\nCMDLINE\n");
    if (read_cmdline())
        return -1;

    printf("\n\nSTAT\n");
    if (read_stat())
        return -1;

    printf("\n\nFD\n");
    if (read_fd())
        return -1;

    return 0;
}

```

Рис 1: вывод на экран информации об окружении процесса

```
anastasia@anastasia-Swift-SF314-54G:~/bmstu/sem_6/os/lab_04/1$ ./a
ENVIRON
CLUTTER_IM_MODULE=xim
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:*.ha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.t
zo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.l
z=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:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.
esd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pg
m=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;
35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.mv=01;35:*.mkv=01;35:*.
webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=0
1;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.
dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;3
6:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.o
gg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;3:
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_MENU_PREFIX=gnome-
LANG=ru_RU.UTF-8
MANAGERPID=2318
DISPLAY=:0
INVOCATION_ID=edfa222d763048b8814f5b10bda68fbc
GNOME_SHELL_SESSION_MODE=ubuntu
COLORTERM=truecolor
USERNAME=anastasia
XDG_VTNR=2
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
S_COLORS=auto
XDG_SESSION_ID=2
USER=anastasia
DESKTOP_SESSION=ubuntu
QT4_IM_MODULE=xim
TEXTDOMAINDIR=/usr/share/locale/
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/14e59314_717c_b64a_aedccde225f7
DEFAULTS_PATH=/usr/share/gconf/ubuntu.default.path
PWD=/home/anastasia/bmstu/sem_6/os/lab_04/1
HOME=/home/anastasia
JOURNAL_STREAM=9:39674
TEXTDOMAIN=im-config
SSH_AGENT_PID=2449
QT_ACCESSIBILITY=1
LIBVIRT_DEFAULT_URI=qemu:///system
XG_SESSION_TYPE=x11
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share:/usr/share:/var/lib/flatpak/desktop
XDG_SESSION_DESKTOP=ubuntu
DBUS_STARTER_ADDRESS=unix:path=/run/user/1000/bus,guid=df4522eda7fd724435a18a055e7abca3
GTK_MODULES=gail:atk-bridge
WINDOWPAH=2
TERM=xterm-256color
SHELL=/bin/bash
VTE_VERSION=5202
QT_IM_MODULE=ibus
XMODIFIERS=@im=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=:1.14
XDG_SEAT=seat0
SHLVL=1
GDMSESSION=ubuntu
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
LOGNAME=anastasia
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus,guid=df4522eda7fd724435a18a055e7abca3
XDG_RUNTIME_DIR=/run/user/1000
XAUTHORITY=/run/user/1000/gm/Xauthority
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
PATH=/home/anastasia/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/game
s:/usr/local/games:/snap/bin:/home/anastasia/.dotnet/tools:/opt/mssql-tools/bin
SESSION_MANAGERlocal/anastasia-Swift-SF314-54G:@/tmp/.ICE-unix/2353,unix/anastasia-Swift-SF314-54G:/tm
p/.ICE-unix/2353
LESSOPEN=| /usr/bin/lesspipe %s
GTK_IM_MODULE=ibus
_=./a
```

Рис 2: вывод на экран содержания файла cmdline

```
CMDLINE
./a
```

Рис 3: вывод на экран информации о состоянии процесса

```
STAT
pid = 4432
comm = (a)
state = R
ppid = 4420
pgrp = 4432
session = 4420
tty_nr = 34817
tpgid = 4432
flags = 4194304
minflt = 74
cmnflt = 0
majflt = 0
cmajflt = 0
utime = 0
stime = 0
cutime = 0
cstime = 0
priority = 20
nice = 0
num_threads = 1
itrealvalue = 0
starttime = 155622
vsize = 4620288
rss = 187
rsslim = 18446744073709551615
startcode = 94829593735168
endcode = 94829593741768
startstack = 140724610809936
kstkesp = 0
kstkeip = 0
signal = 0
blocked = 0
sigignore = 0
sigcatch = 0
wchan = 0
nswap = 0
cnsnap = 0
exit_signal = 17
processor = 1
rt_priority = 0
policy = 0
delayacct_blkio_ticks = 0
guest_time = 0
cguest_time = 0
start_data = 94829595839832
end_data = 94829595840960
start_brk = 94829624168448
arg_start = 140724610814543
arg_end = 14072p?p
```

Рис 4: вывод на экран содержания директории fd

```
FD
0 -> /dev/pts/1?<?V
1 -> /dev/pts/1?<?V
2 -> /dev/pts/1?<?V
3 -> /proc/4432/fdV
```

## 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.

### Листинг к заданию 2

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

#define BUFSIZE 100
#define MAX_COOKIE_LENGTH PAGE_SIZE

MODULE_LICENSE("Dual BSD/GPL");
MODULE_AUTHOR("Neklepaeva A. N.");

static struct proc_dir_entry *proc_entry;
static struct proc_dir_entry *proc_link;
static struct proc_dir_entry *proc_dir;
static char *cookie_pot = NULL;
int cookie_index;
int next_fortune;

static ssize_t fortune_write(struct file *file, const char __user *ubuf, size_t
count, loff_t *ppos)
{
    int free_space = (MAX_COOKIE_LENGTH - next_fortune) + 1;

    if (count > free_space)
    {
        printk(KERN_INFO "Cookie pot full.\n");
        return -ENOSPC;
    }

    if (copy_from_user(&cookie_pot[cookie_index], ubuf, count))
    {
        return -EFAULT;
    }

    cookie_index += count;
    cookie_pot[cookie_index - 1] = 0;

    return count;
}

static ssize_t fortune_read(struct file *file, char __user *ubuf, size_t count,
loff_t *ppos)
{
    int len;
    char buf[256];

    if (cookie_index == 0 || *ppos > 0)
```

```

    {
        return 0;
    }

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

    len = sprintf(buf, "%s\n", &cookie_pot[next_fortune]);
    copy_to_user(ubuf, buf, len);
    next_fortune += len;
    *ppos += len;

    return len;
}

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

static int __init fortune_init(void)
{
    cookie_pot = (char *)vmalloc(MAX_COOKIE_LENGTH);
    if (!cookie_pot)
    {
        printk(KERN_INFO "Not enough memory for the cookie pot.\n");
        return -ENOMEM;
    }

    memset( cookie_pot, 0, MAX_COOKIE_LENGTH );

    proc_entry = proc_create("fortune", 0666, NULL, &myops);
    proc_dir = proc_mkdir("mydir", NULL);

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

    proc_link = proc_symlink("link", NULL, "fortune");

    cookie_index = 0;
    next_fortune = 0;

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

    return 0;
}

static void __exit fortune_exit(void)
{
    remove_proc_entry("fortune", NULL);
    remove_proc_entry("mydir", NULL);
    remove_proc_entry("link", NULL);

    if (cookie_pot)
        vfree(cookie_pot);
}

```



```

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

module_init(fortune_init);
module_exit(fortune_exit);

```

Рис 5: демонстрация работы «fortune cookie» LKM, демонстрация передачи данных из пространства пользователя и из пространства ядра в пространство ПОЛЬЗОВАТЕЛЯ

```

anastasia@anastasia-Swift-SF314-54G:~/bnstu/sem_6/os/lab_04/2$ sudo insmod ./md.ko
anastasia@anastasia-Swift-SF314-54G:~/bnstu/sem_6/os/lab_04/2$ echo "first message" > /proc/fortune
anastasia@anastasia-Swift-SF314-54G:~/bnstu/sem_6/os/lab_04/2$ echo "second message" > /proc/fortune
anastasia@anastasia-Swift-SF314-54G:~/bnstu/sem_6/os/lab_04/2$ echo "third message" > /proc/fortune
anastasia@anastasia-Swift-SF314-54G:~/bnstu/sem_6/os/lab_04/2$ cat /proc/fortune
first message
anastasia@anastasia-Swift-SF314-54G:~/bnstu/sem_6/os/lab_04/2$ cat /proc/fortune
second message
anastasia@anastasia-Swift-SF314-54G:~/bnstu/sem_6/os/lab_04/2$ cat /proc/fortune
third message

```

Рис 6: вывод ls -al /proc , демонстрация создания файла fortune, символической ссылки link и директории mydir в файловой системе proc

-rw-rw-rw-	1	root	root	0	map 25 07:03	filesystems
dr-xr-xr-x	6	root	root	0	map 25 05:06	fs
-r--r--r--	1	root	root	0	map 25 07:03	interrupts
-r--r--r--	1	root	root	0	map 25 07:03	iomem
-r--r--r--	1	root	root	0	map 25 07:03	ioports
dr-xr-xr-x	32	root	root	0	map 25 05:06	irq
-r--r--r--	1	root	root	0	map 25 07:03	kallsyms
-r-----	1	root	root	140737477881856	map 25 07:03	kcore
-r--r--r--	1	root	root	0	map 25 07:03	keys
-r--r--r--	1	root	root	0	map 25 07:03	key-users
-r-----	1	root	root	0	map 25 05:06	kmsg
-r-----	1	root	root	0	map 25 07:03	kpagecgroup
-r-----	1	root	root	0	map 25 07:03	kpagecount
-r-----	1	root	root	0	map 25 07:03	kpageflags
lrwxrwxrwx	1	root	root	7	map 25 07:03	link -> fortune
-r--r--r--	1	root	root	0	map 25 07:03	loadavg
-r--r--r--	1	root	root	0	map 25 07:03	locks
-r--r--r--	1	root	root	0	map 25 07:03	mdstat
-r--r--r--	1	root	root	0	map 25 07:03	meminfo
-r--r--r--	1	root	root	0	map 25 07:03	misc
-r--r--r--	1	root	root	0	map 25 07:03	modules
lrwxrwxrwx	1	root	root	11	map 25 07:03	mounts -> self/mounts
-rw-r--r--	1	root	root	0	map 25 05:06	mtrr
dr-xr-xr-x	2	root	root	0	map 25 07:03	mydir