

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический  
университет имени Н.Э. Баумана»  
(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 / прос
Студент	Куприй А. А.
Группа	ИУ7-63Б
Преподаватель	Рязанова Н.Ю.

Москва, 2020 г.

## 1 Практическая часть

### 1.1 Задание №1

Используя виртуальную файловую систему `proc` вывести информацию об окружении процесса, информацию, характеризующую состояние процесса, содержание директории `fd` и файла `cmdline`.

Листинг 1.1 — Вывод содержимого `environ`

```
1 void print_environ()
2 {
3     char buf[BUF_SIZE] = {0};
4     int len, i;
5     FILE *fd = fopen("/proc/self/environ", "r");
6     if (fd == NULL)
7     {
8         printf("%s", strerror(errno));
9         exit(errno);
10    }
11
12    while ((len = fread(buf, 1, BUF_SIZE, fd)) > 0)
13    {
14        for (i = 0; i < len; i++)
15            if (buf[i] == 0)
16            {
17                buf[i] = 10;
18            }
19        buf[len] = 0;
20        printf("%s", buf);
21    }
22
23    fclose(fd);
24 }
```

Листинг 1.2 — Вывод содержимого файла для `stat` и `cmdline`

```
1 void print_file(char *path)
2 {
3     char buf[BUF_SIZE] = {0};
4     char *pch;
5     int i = 0;
6     FILE *f = fopen(path, "r");
7     if (f == NULL)
8     {
9         printf("%s", strerror(errno));
10        exit(errno);
11    }
```

```

11     }
12
13     fread(buf, 1, BUF_SIZE, f);
14     pch = strtok(buf, " ");
15
16     while (pch)
17     {
18         if (strcmp("/proc/self/stat", path) == 0)
19         {
20             printf("%s=", param_stat[i++]);
21         }
22         printf("%s\n", pch);
23         pch = strtok(NULL, " ");
24     }
25
26     fclose(f);
27 }

```

### Листинг 1.3 — Вывод содержимого директории

```

1 void print_fd()
2 {
3     struct dirent *dirp;
4     DIR *dp;
5     char path[BUF_SIZE];
6     char str[BUF_SIZE];
7
8     if (!(dp = opendir("/proc/self/fd/")))
9     {
10         printf("%s", strerror(errno));
11         exit(errno);
12     }
13
14     while ((dirp = readdir(dp)))
15     {
16         if ((strcmp(dirp->d_name, ".") != 0) && (strcmp(dirp->d_name, "..") !=
17             0))
18         {
19             sprintf(path, "%s%s", "/proc/self/fd/", dirp->d_name);
20             readlink(path, str, BUF_SIZE);
21             printf("%s -> %s\n", dirp->d_name, str);
22         }
23     }
24
25     if (closedir(dp) < 0)
26     {
27         printf("%s", strerror(errno));
28     }
29 }

```

```
27     exit(errno);
28 }
29 }
```

Результат:



The screenshot shows a terminal window with a dark background. The title bar at the top indicates the user is 'san\_sanchez@LEX' and the current directory is '~/workspace/OC/sem\_2/lab\_04/source/task\_1'. The command being executed is './task\_1'. The output is a list of environment variables. The variables include display-related settings like CLUTTER\_IM\_MODULE, COLORTERM, DBUS\_SESSION\_BUS\_ADDRESS, and desktop session information like DESKTOP\_SESSION, GDMSESSION, and GIO\_LAUNCHED\_DESKTOP\_FILE. It also lists locale settings (LANG, LC\_\*), GTK settings, and various path-related variables. The output is truncated at the bottom with a red line.

```
san_sanchez@LEX ~/workspace/OC/sem_2/lab_04/source/task_1 ./task_1 [120/124]
environ:
CLUTTER_IM_MODULE=xim
COLORTERM=truecolor
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
DEFAULTS_PATH=/usr/share/gconf/ubuntu.default.path
DESKTOP_AUTOSTART_ID=10d51606ffaf918bf9158907274733416000000016600007
DESKTOP_SESSION=ubuntu
DISPLAY=:0
GDMSESSION=ubuntu
GIO_LAUNCHED_DESKTOP_FILE=/etc/xdg/autostart/org.gnome.SettingsDaemon.MediaKeys.desktop
GIO_LAUNCHED_DESKTOP_FILE_PID=1922
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
GTK_IM_MODULE=ibus
GTK_MODULES=gail:atk-bridge
HOME=/home/san_sanchez
IM_CONFIG_PHASE=2
LANG=en_US.UTF-8
LANGUAGE=en_US:en
LC_ADDRESS=ru_RU.UTF-8
LC_IDENTIFICATION=ru_RU.UTF-8
LC_MEASUREMENT=ru_RU.UTF-8
LC_MONETARY=ru_RU.UTF-8
LC_NAME=ru_RU.UTF-8
LC_NUMERIC=ru_RU.UTF-8
LC_PAPER=ru_RU.UTF-8
LC_TELEPHONE=ru_RU.UTF-8
LC_TIME=ru_RU.UTF-8
LESS=-R
LOGNAME=san_sanchez
LSCOLORS=Gxfxcxdxbxegedabagacad
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:*.t
ar=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;3
1:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst=0
1;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;3
1:*.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
```

Рисунок 1.1 — Скришот вывода информации об окружении процесса (часть 1).

```

PATH=/home/san_sanchez/workspace/datagrip-2019.2.5/DataGrip-2019.2.5/bin:/opt/m[66/124]
ls/bin:/home/san_sanchez/bin:/usr/local/bin:/home/san_sanchez/workspace/datagrip-2019.2
.5/DataGrip-2019.2.5/bin:/opt/mssql-tools/bin:/home/san_sanchez/bin:/usr/local/bin:/hom
e/san_sanchez/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/
usr/games:/usr/local/games:/snap/bin:/home/san_sanchez/.fzf/bin
PWD=/home/san_sanchez/workspace/OC/sem_2/lab_04/source/task_1
QT4_IM_MODULE=xim
QT_ACCESSIBILITY=1
QT_IM_MODULE=ibus
SESSION_MANAGER=local/LEX:@/tmp/.ICE-unix/1660,unix/LEX:/tmp/.ICE-unix/1660
SHELL=/usr/bin/zsh
SHLVL=2
SSH_AGENT_PID=1738
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
S_COLORS=auto
TERM=screen-256color
TERMINATOR_DBUS_NAME=net.tenshu.Terminator20x1a6021154d881c
TERMINATOR_DBUS_PATH=/net/tenshu/Terminator2
TERMINATOR_UUID=urn:uuid:a8abd3aa-434c-4736-bf95-bb431352207d
TEXTDOMAIN=im-config
TEXTDOMAINDIR=/usr/share/locale/
TMUX=/tmp/tmux-1000/default,2175,0
TMUX_PANE=%1
USER=san_sanchez
USERNAME=san_sanchez
VTE_VERSION=5202
WINDOWPATH=2
XAUTHORITY=/run/user/1000/gdm/Xauthority
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_CURRENT_DESKTOP=ubuntu:GNOME
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share:/usr/share:/var/lib/snapd/desktop
XDG_MENU_PREFIX=gnome-
XDG_RUNTIME_DIR=/run/user/1000
XDG_SEAT=seat0
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_ID=3
XDG_SESSION_TYPE=x11
XDG_VTNR=2
XMODIFIERS=@im=ibus
ZSH=/home/san_sanchez/.oh-my-zsh
_=/home/san_sanchez/workspace/OC/sem_2/lab_04/source/task_1/./task_1

```

Рисунок 1.2 — Скришот вывода информации об окружении процесса (часть 2).

```

cmdline:
./task_1

```

Рисунок 1.3 — Скришот вывода содержимого файла cmdline.

```

fd:
0 -> /dev/pts/21)
1 -> /dev/pts/21)
2 -> /dev/pts/21)
3 -> /proc/793/fd

```

Рисунок 1.4 — Скришот вывода содержимого директории fd.

```
stat:
pid=10335
comm=(task_1)
state=R
ppid=4187
pgrp=10335
session=4187
tty_nr=34821
tpgid=10335
flags=4194304
minflt=97
cminflt=0
majflt=1
cmajflt=0
utime=0
stime=0
cutime=0
cstime=0
priority=20
nice=0
num_threads=1
itrealvalue=0
starttime=36511999
vsize=4624384
rss=192
rsslim=18446744073709551615
startcode=94465334353920
```

Рисунок 1.5 — Скришот вывода состояния процесса. (часть 1)

```
startcode=94465334353920
endcode=94465334360440
startstack=140726574953312
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
delayacct_blkio_ticks=0
guest_time=0
cguest_time=0
start_data=94465336458552
end_data=94465336459712
start_brk=94465350205440
arg_start=140726574956492
arg_end=140726574956501
env_start=140726574956501
env_end=140726574960623
exit_code=0
```

Рисунок 1.6 — Скришот вывода состояния процесса. (часть 2)

## 1.2 Задание №2

Написать программу – загружаемый модуль ядра (LKM) – которая поддерживает чтение из пространства пользователя и запись в пространство пользователя из пространства ядра.

Листинг 1.4 — Текст программы второго задания

```
1 #include <linux/module.h>
2 #include <linux/kernel.h>
3 #include <linux/proc_fs.h>
4 #include <linux/string.h>
5 #include <linux/vmalloc.h>
6 #include <linux/fs.h>
7 #include <linux/uaccess.h>
8
9 MODULE_LICENSE("GPL");
10 MODULE_AUTHOR("Kupry");
11
12 static struct proc_dir_entry *proc_entry;
13 static struct proc_dir_entry *proc_dir;
14 static struct proc_dir_entry *proc_link;
15
16 static char *buffer;
17 static int buf_index, next_fortune;
18
19 static ssize_t fortune_write(struct file *filp, const char __user *buf,
20                             size_t len, loff_t *offp);
21 static ssize_t fortune_read(struct file *filp, char __user *buf, size_t
22                             count, loff_t *offp);
23
24 static int fortune_init(void)
25 {
26     static struct file_operations fileops =
27     {
28         .owner = THIS_MODULE,
29         .read = fortune_read,
30         .write = fortune_write
31     };
32
33     buffer = (char *)vmalloc(PAGE_SIZE);
34     if (!buffer)
35     {
36         printk(KERN_ERR "fortune: no memory for create buffer\n");
37         return -ENOMEM;
38     }
39 }
```

```

37
38     memset( buffer , 0, PAGE_SIZE);
39
40     proc_entry = proc_create("fortune", 0644, NULL, &fileops);
41     if (proc_entry == NULL)
42     {
43         vfree( buffer );
44         printk(KERN_ERR "fortune: couldn't create proc entry\n");
45         return -ENOMEM;
46     }
47
48     proc_dir = proc_mkdir("fortune_dir", NULL);
49     if (!proc_dir)
50     {
51         vfree( buffer );
52         printk(KERN_ERR "fortune: Couldn't create proc dir, symlink\n");
53         return -ENOMEM;
54     }
55
56     proc_link = proc_symlink("fortune_symlink", NULL, "/proc/fortune_dir");
57     if (!proc_link)
58     {
59         vfree( buffer );
60         printk(KERN_ERR "fortune: Couldn't create proc dir, symlink\n");
61         return -ENOMEM;
62     }
63
64     buf_index = 0;
65     next_fortune = 0;
66
67     printk(KERN_INFO "fortune: module loaded.\n");
68     return 0;
69 }
70
71 static ssize_t fortune_read(struct file *filp, char __user *buf, size_t
count, loff_t *offp)
72 {
73     if (*offp > 0)
74     {
75         return 0;
76     }
77
78     if (next_fortune >= buf_index)
79     {
80         next_fortune = 0;
81     }

```



```
82
83     count = copy_to_user(buf, &buffer[next_fortune], count);
84     next_fortune += count;
85     *offp += count;
86
87     return count;
88 }
89
90 static ssize_t fortune_write(struct file *filp, const char __user *buf,
91                             size_t len, loff_t *offp)
92 {
93     int available_space = PAGE_SIZE - buf_index + 1;
94
95     if (len > (size_t)available_space)
96     {
97         printk(KERN_NOTICE "fortune: there isn't enough memory\n");
98         return -ENOSPC;
99     }
100
101     if (copy_from_user(&buffer[buf_index], buf, len))
102     {
103         printk(KERN_NOTICE "fortune: copy_to_user failed\n");
104         return -EFAULT;
105     }
106
107     buf_index += len;
108     buffer[buf_index - 1] = 0;
109
110     return len;
111 }
112
113 static void fortune_exit(void)
114 {
115     proc_remove(proc_entry);
116     proc_remove(proc_dir);
117     proc_remove(proc_link);
118
119     vfree(buffer);
120
121     printk(KERN_INFO "fortune: module unloaded.\n");
122 }
123
124 module_init(fortune_init);
125 module_exit(fortune_exit);
```

Результат:

```
san_sanchez@LEX ~/workspace/0C/sem_2/lab_04/source/task_2
make -C /lib/modules/4.15.0-99-generic/build M=/home/san_sanchez/workspace/0C/sem_2/lab_04/source/task_2 modules
make[1]: Entering directory '/usr/src/linux-headers-4.15.0-99-generic'
CC [M] /home/san_sanchez/workspace/0C/sem_2/lab_04/source/task_2/md.o
Building modules, stage 2.
MODPOST 1 modules
LD [M] /home/san_sanchez/workspace/0C/sem_2/lab_04/source/task_2/md.ko
```

Рисунок 1.7 — Скришот сборки модуля.

```
san_sanchez@LEX ~/workspace/0C/sem_2/lab_04/source/task_2
[sudo] password for san_sanchez:
sudo insmod md.ko
```

Рисунок 1.8 — Скришот загрузки модуля.

```
root@LEX:~# ls -l /proc | grep fortune
-rw-r--r-- 1 root root 0 мая 10 05:06 fortune
dr-xr-xr-x 2 root root 0 мая 10 05:06 fortune_dir
lrwxrwxrwx 1 root root 13 мая 10 05:06 fortune_symlink -> /proc/fortune
```

Рисунок 1.9 — Скриншот созданных директории, файла и символической ссылки.

```
root@LEX:~# echo "Hello, world!" >> /proc/fortune
root@LEX:~# cat /proc/fortune
Hello, world
```

Рисунок 1.10 — Скриншот работы файла.

```
root@LEX:~# cat /proc/fortune_symlink
Hello, world
```

Рисунок 1.11 — Скриншот работы ссылки.

## ОБОСНОВАНИЕ ИСПОЛЬЗОВАНИЯ СПЕЦИАЛЬНЫХ ФУНКЦИИ

В Linux каждый процесс имеет собственное изолированное адресное пространство, то есть указатель ссылается не на уникальную позицию в физической памяти, а на позицию в адресном пространстве процесса.

Во время выполнения обычной программы адресация происходит автоматически, если же выполняется код ядра и необходимо получить доступ к странице кода ядра, то необходим буфер.

Когда мы хотим передавать информацию между процессом и кодом ядра, то соответствующая специальная функция (`copy_from_user`, `copy_to_user`) получает указатель на буфер процесса.