



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 система/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

```
1 #include <stdio.h>
2
3 #define BUF_SIZE 512
4
5 int main()
6 {
7     FILE *f = fopen("/proc/self/environ", "r");
8     int len = 0;
9     char buf[BUF_SIZE];
10    while ((len = fread(buf, 1, BUF_SIZE, f)) > 0)
11    {
12        for (int i = 0; i < len; i++)
13            if (buf[i] == 0)
14                buf[i] = 10;
15        buf[len - 1] = 0;
16        printf("%s\n", buf);
17    }
18    fclose(f);
19 }
```

показать корневой каталог, который создан 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_PID=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).

```
36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=
00;36:*.xspf=00;36:
XDG_CURRENT_DESKTOP=ubuntu:GNOME
VTE_VERSION=6003
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/327255a4_68b0_48d8_a6b4_17e382ecc1ad
INVOCATION_ID=9c695dcd02ea44a595495b784493cfc4
MANAGERPID=2679
NVM_DIR=/home
dalai/.nvm
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
LC_IDENTIFICATION=mn_MN
LESSOPEN=| /usr/bin/lesspipe %s
USER=dalai
GNOME_TERMINAL_SERVICE=:1.104
DISPLAY=:0
SHLVL=1
NVM_CD_FLAGS=
LC_TELEPHONE=mn_MN
QT_IM_MODULE=ibus
LC_MEASUREMENT=mn_MN
XDG_RUNTIME_DIR=/run/user/1000
LC_TIME=mn_MN
JOURNAL_STREAM=8:59679
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share:/usr/share:/var/lib/snapd/desktop
PATH=/home/dalai/.nvm/versions/node/v16.5.0/bin:/usr/local/sbin:/usr/local/bin:/us
/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
NVM_BIN=/home/dalai/.nvm/versions/node/v16.5.0/bin
LC_NUMERIC=mn_MN
_=/a.out
OLDPWD=/home/dalai/Documents/BMSTU/labs
dalai@dalai-Inspiron-5567:~/Documents/BMSTU/labs/lab_04$
```

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

```
1 #include <stdio.h>
2 #include <string.h>
3 #define BUF_SIZE 512
4 int main()
5 {
6     char *arr[52] = {"pid", "comm", "state", "ppid", "pgrp",
7                     "session", "tty_nr", "tpgid",
8                     "flags", "minflt", "cminflt", "majflt", "cmajflt", "utime",
9                     "stime", "cutime",
10                    "cstime", "priority", "nice", "num_threads", "itrealvalue",
11                    "starttime", "vsize",
12                    "rss", "rsslim", "startcode", "endcode", "startstack",
13                    "kstkesp", "kstkeip", "signal",
14                    "blocked", "sigignore", "sigcatch", "wchan", "nswap",
15                    "cnsnap", "exit_signal",
16                    "processor", "rt_priority", "policy",
17                    "delayacct_blkio_ticks", "guest_time",
18                    "cguest_time", "start_data", "end_data", "start_brk",
19                    "arg_start", "arg_end",
20                    "env_start", "env_end", "exit_code"};
21     FILE *f = fopen("/proc/self/stat", "r");
22     char buf[BUF_SIZE];
23     fread(buf, 1, BUF_SIZE, f);
24     char *pch = strtok(buf, " ");
25     printf("stat: \n");
26     int i = 0;
27     while(pch != NULL)
28     {
29         printf("%s = %s\n", arr[i++], pch);
30         pch = strtok(NULL, " ");
31     }
32     fclose(f);
33 }
```

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).

```
delayacct_blkio_ticks = 0
guest_time = 0
cguest_time = 0
start_data = 94162968587656
end_data = 94162968588736
start_brk = 94162994208768
arg_start = 140722987197153
arg_end = 140722987197164
env_start = 140722987197164
env_end = 140722987200493
exit_code = 0
```

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.

```
1  #include <stdio.h>
2  #include <unistd.h>
3  #include <sys/wait.h>
4  #define BUFF_SIZE 512
5  #define PID_LEN 30
6
7  int main()
8  {
9      char path[PID_LEN];
10     sprintf(path, "/proc/%d/fd", getpid());
11     printf("fd:\n\n");
12
13     pid_t pid = fork();
14
15     if (pid == 0) {
16         if (execlp("/bin/ls", "ls", "-l", path, NULL) == -1) {
17             printf("Error with \"%s\"!\n", path);
18         }
19     } else {
20         int status = 0;
21         pid_t wpid = wait(&status);
22     }
23 }
```

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

```
dalai@dalai-Inspiron-5567:~/Documents/BMSTU/labs/lab_04$ ./prog_3.out
fd:

total 0
lrwx----- 1 dalai dalai 64 9-p cap 30 16:19 0 -> /dev/pts/0
lrwx----- 1 dalai dalai 64 9-p cap 30 16:19 1 -> /dev/pts/0
lrwx----- 1 dalai dalai 64 9-p cap 30 16:19 2 -> /dev/pts/0
```

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

```
1 #include <linux/string.h>
2 #include <linux/module.h>
3 #include <linux/init.h>
4 #include <linux/kernel.h>
5 #include <linux/proc_fs.h>
6 #include <linux/vmalloc.h>
7 #include <asm/uaccess.h>
8 #include <linux/uaccess.h>
9
10 #define OK 0
11
12 #define MAX_COOKIE_LEN PAGE_SIZE
13 #define BUF_SIZE 100
14 #define FILE_NAME "fortune"
15 #define DIR_NAME "fortune_dir"
16 #define SYMLINK_NAME "fortune_symlink"
17
18
19 MODULE_LICENSE ("BSD");
20 MODULE_AUTHOR ("Nomuundalai B.");
21
22 static struct proc_dir_entry *proc_entry;
23
24 ssize_t fortune_write (struct file *filp, const char __user *buf, unsigned long count,
25                                     loff_t *offp);
26
27 ssize_t fortune_read(struct file *filp, char __user *buf, unsigned long count,
28                                     loff_t *offp);
29
30 struct file_operations fops = {
31     .owner = THIS_MODULE,
32     .read = fortune_read,
33     .write = fortune_write,
34 };
35
36 static char *cookie_pot;
37
```



```

38 // Управляющие индексы
39 static int cookie_index;
40 static int next_fortune;
41 static char tmp_buf[BUF_SIZE];
42
43 // Запись
44 ssize_t fortune_write (struct file *filp, const char __user *buf, size_t len, loff_t *offp)
45 {
46     int space_available = (MAX_COOKIE_LEN - cookie_index) + 1;
47     if (len > space_available) {
48         printk(KERN_INFO "fortune: cookie_pot is full!\n");
49     }
50     // Копирует из пространства пользователя в буфер ядра
51     if (copy_from_user(&cookie_pot[cookie_index], buf, len)){
52         return -EFAULT;
53     }
54     cookie_index += len;
55     cookie_pot[cookie_index - 1] = 0;
56
57     return len;
58 }
59
60 // Чтение
61 ssize_t fortune_read (struct file *filp, char __user *buf, size_t count, loff_t *offp) {
62     int len = 0;
63     if (!cookie_index || *offp > 0) {
64         return 0;
65     }
66
67     len = sprintf(tmp_buf, "%s\n", &cookie_pot[next_fortune]);
68     // Копирует из пространства ядра в пространство пользователя
69     copy_to_user(buf, tmp_buf, len);
70     next_fortune += len;
71     *offp += len;
72
73     return len;
74 }
75
76 // Инициализация
77 int init_fortune_module(void) {
78     int ret = OK;
79     // Выделение пространства для "cookie pot"
80     cookie_pot = (char *)vmalloc(MAX_COOKIE_LEN);
81     if (!cookie_pot) {
82         ret = -ENOMEM;
83     }
84     else {
85         //Очистка пространства "cookie pot"
86         memset(cookie_pot, 0, MAX_COOKIE_LEN);
87         proc_entry = proc_create("fortune", 0666, NULL, &fops);
88         if(!proc_entry) {
89             ret = -ENOMEM;
90             vfree(cookie_pot);
91             printk(KERN_INFO "fortune: Couldn't create proc
92             entry\n");
93         }
94         else {
95             // Управляющие индексы, устанавливаются в начальное положение
96             cookie_index = 0;
97             next_fortune = 0;

```

```

98         proc_mkdir(DIR_NAME, NULL);
99         proc_symlink(SYMLINK_NAME, NULL, "/proc/" FILE_NAME);
100        printk(KERN_INFO "fortune: Module loaded.\n");
101    }
102 }
103 return ret;
104 }
105
106 void cleanup_fortune_module(void) {
107     remove_proc_entry(SYMLINK_NAME, NULL);
108     remove_proc_entry(FILE_NAME, NULL);
109     remove_proc_entry(DIR_NAME, NULL);
110
111     if (cookie_pot) {
112         vfree(cookie_pot);
113     }
114
115     printk(KERN_INFO "fortune: Module unloaded.\n");
116 }
117
118 module_init(init_fortune_module);
119 module_exit(cleanup_fortune_module);

```

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

Листинг 5. Makefile.

```

1  ifneq ($(KERNELRELEASE),)
2      obj-m := fortune.o
3  else
4      CURRENT = $(shell uname -r)
5      KDIR = /lib/modules/$(CURRENT)/build
6      PWD = $(shell pwd)
7
8  default:
9      $(MAKE) -C $(KDIR) M=$(PWD) modules
10 # sudo make clean
11
12 clean:
13     rm -rf .tmp_versions
14     rm .fortune.*
15     rm *.o
16     rm *.mod.c
17     rm *.symvers
18     rm *.order
19
20 endif

```

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