



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»

ОТЧЁТ

По лабораторной работе №4

По курсу: «Операционные системы»

Тема: «Виртуальная файловая система /proc»

Студент: Керимов А. Ш.

Группа: ИУ7-64Б

Преподаватель: Рязанова Н. Ю.

Москва

2020

Часть 1

Задание. Используя виртуальную файловую систему **proc**, вывести информацию об окружении процесса, о состоянии процесса, информацию из файла **cmdline** и директории **fd**.

В листинге 1 представлена программа **proc.c**, выводящая информацию о процессе. На рисунках 1–3 продемонстрирована работа программы.

Листинг 1: task1/proc.c

```
1 #include <errno.h>
2 #include <dirent.h>
3 #include <linux/limits.h>
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <string.h>
7 #include <unistd.h>
8
9 #define BUF_SIZE 4096
10
11 FILE *open_proc_file(int pid, const char *filename);
12 void close_proc_file(FILE *file);
13 void print_proc_file(FILE *file, char delim);
14 void print_proc_environ(FILE *file, int pid);
15 void print_proc_cmdline(FILE *file, int pid);
16 void print_proc_stat(FILE *file);
17 void print_proc_fd(int pid);
18
19 int main(int argc, char *argv[])
20 {
21     if (argc > 2) {
22         fprintf(stderr, "Usage: %s [pid=self]\n", argv[0]);
23         return EXIT_FAILURE;
24     }
25
26     const int pid = argc == 1 ? getpid() : atoi(argv[1]);
27
28     FILE *file = NULL;
29
30     file = open_proc_file(pid, "environ");
31     print_proc_environ(file, pid);
32     close_proc_file(file);
33
34     file = open_proc_file(pid, "stat");
35     print_proc_stat(file);
36     close_proc_file(file);
37
38     file = open_proc_file(pid, "cmdline");
39     print_proc_cmdline(file, pid);
40     close_proc_file(file);
41
42     print_proc_fd(pid);
43 }
44
45 /*
46  * Открытие файла /proc/<pid>/<filename>
47  */
48 FILE *open_proc_file(int pid, const char *filename)
49 {
50     char pathname[PATH_MAX];
51     snprintf(pathname, sizeof pathname, "/proc/%d/%s", pid, filename);
52
53     FILE *file = fopen(pathname, "r");
54     if (file == NULL) {
55         fprintf(stderr, "fopen(\"%s\", \"%r\"): %s\n", pathname, strerror(errno));
56         exit(EXIT_FAILURE);
57     }
58
59     return file;
60 }
61
62 /*
63  * Закрытие файла
64  */
```

```

65 void close_proc_file(FILE *file)
66 {
67     fclose(file);
68 }
69
70 /*
71  * Вывод на экран файла file с заменой символа '\0' на символ delim
72  */
73 void print_proc_file(FILE *file, char delim)
74 {
75     char buf[BUF_SIZE];
76     size_t len;
77
78     while ((len = fread(buf, 1, BUF_SIZE, file)) > 0) {
79         for (char *c = buf; c < buf + len; ++c) {
80             if (*c == '\0') {
81                 *c = delim;
82             }
83         }
84         buf[len - 1] = '\0';
85         printf("%s", buf);
86     }
87 }
88
89 /*
90  * Вывод информации об окружении процесса
91  */
92 void print_proc_environ(FILE *file, int pid)
93 {
94     printf("=== Environment list of the process with id %d:\n", pid);
95     print_proc_file(file, '\n');
96 }
97
98 /*
99  * Вывод информации из файла cmdline
100  */
101 void print_proc_cmdline(FILE *file, int pid)
102 {
103     printf("\n=== Command line for the process with id %d:\n", pid);
104     print_proc_file(file, ' ');
105 }
106
107 /*
108  * Вывод информации о состоянии процесса
109  */
110 void print_proc_stat(FILE *file)
111 {
112     static char *fields[] = {
113         "pid", "comm", "state", "ppid", "pgrp", "session", "tty_nr", "tpgid",
114         "flags", "minflt", "cmiflt", "majflt", "cmajflt", "utime", "stime",
115         "cutime", "cstime", "priority", "nice", "num_threads", "itrealvalue",
116         "starttime", "vsize", "rss", "rsslim", "startcode", "endcode",
117         "startstack", "kstkesp", "kstkeip", "signal", "blocked", "sigignore",
118         "sigcatch", "wchan", "nswap", "cnswap", "exit_signal", "processor",
119         "rt_priority", "policy", "delayacct_blkio_ticks", "guest_time",
120         "cguest_time", "start_data", "end_data", "start_brk", "arg_start",
121         "arg_end", "env_start", "env_end", "exit_code", NULL
122     };
123
124     char buf[BUF_SIZE];
125     const size_t len = fread(buf, 1, BUF_SIZE, file);
126     buf[len - 1] = '\0';
127
128     printf("\n\n=== State of the process:\n");
129     for (
130         char *pch = strtok(buf, " "), **pfield = fields;
131         pch && *pfield;
132         pch = strtok(NULL, " "), ++pfield
133     ) {
134         printf("%-21s %s\n", *pfield, pch);
135     }
136 }
137
138 /*
139  * Вывод информации из директории fd
140  */
141 void print_proc_fd(int pid)
142 {
143     char dirname[PATH_MAX];

```

```

144     snprintf(dirname, sizeof dirname, "/proc/%d/fd", pid);
145
146     DIR *dir = opendir(dirname);
147     if (dir == NULL) {
148         fprintf(stderr, "opendir(\"%s\"): %s\n", dirname, strerror(errno));
149         exit(EXIT_FAILURE);
150     }
151
152     printf("\n\n=== List of open files for the process with id %d:\n", pid);
153     struct dirent *entry = NULL;
154     while ((entry = readdir(dir)) != NULL) {
155         if (strcmp(entry->d_name, ".") == 0
156             || strcmp(entry->d_name, "..") == 0) {
157             continue;
158         }
159
160         char path[PATH_MAX], str[PATH_MAX];
161         snprintf(path, sizeof path, "%s/%s", dirname, entry->d_name);
162
163         const int n = readlink(path, str, sizeof str);
164         str[n] = '\0';
165
166         printf("\t%-17s -> %s\n", path, str);
167     }
168
169     closedir(dir);
170 }

```

```

user@user-Lenovo-ideapad-310-15ISK: ~/bmstu/O...
user@user-Lenovo-ideapad-310-15ISK:~/bmstu/OS/sem06/Lab04/task1$ ./proc.out
=== Environment list of the process with id 17531:
SHELL=/bin/bash
SESSION_MANAGER=local/user-Lenovo-ideapad-310-15ISK:@/tmp/.ICE-unix/1935,unl
x/user-Lenovo-ideapad-310-15ISK:/tmp/.ICE-unix/1935
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GTK_IM_MODULE=ibus
QT4_IM_MODULE=ibus
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
LC_ADDRESS=ru_RU.UTF-8
GNOME_SHELL_SESSION_MODE=ubuntu
LC_NAME=ru_RU.UTF-8
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
LC_MONETARY=ru_RU.UTF-8
SSH_AGENT_PID=1839
GTK_MODULES=gall:atk-bridge
PWD=/home/user/bmstu/OS/sem06/Lab04/task1
NODE_VERSION=v12.9.1
XDG_SESSION_DESKTOP=ubuntu
LOGNAME=user
XDG_SESSION_TYPE=x11
NODE_PATH=/home/user/.npm-global/bin
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
XAUTHORITY=/run/user/1000/gdm/Xauthority
WINDOWPATH=2
HOME=/home/user
USERNAME=user
IM_CONFIG_PHASE=1
LC_PAPER=ru_RU.UTF-8
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;0
1: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:*.lзма=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;3
1:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wlm=01;31:*.s
wm=01;31:*.dwm=01;31:*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjp
eg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=0
1;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;3
5:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:
*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.m
p4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;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;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;
36:*.midi=00;36:*.mka=00;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

```

```

user@user-Lenovo-ideapad-310-15ISK: ~/bmstu/O...
: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:*.lзма=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;3
1:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wlm=01;31:*.s
wm=01;31:*.dwm=01;31:*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjp
eg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=0
1;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;3
5:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:
*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.m
p4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;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;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;
36:*.midi=00;36:*.mka=00;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=5802
NODE_ROOT=/usr/local/lib/nodejs/node-v12.9.1-linux-x64
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/7cd25918_c4fe_412c_9e9e_5dc
7aafd335e
INVOCATION_ID=f8e95777918c4fd4b5fd9cbb81976c36
MANAGERPID=1725
GOROOT=/usr/local/go
CLUTTER_IM_MODULE=ibus
NVM_DIR=/home/user/.nvm
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
LC_IDENTIFICATION=ru_RU.UTF-8
DEFAULTS_PATH=/usr/share/gconf/ubuntu.default.path
LESSOPEN=| /usr/bin/lesspipe %s
USER=user
GNOME_TERMINAL_SERVICE=:1.203
DISPLAY=:0
SHLVL=1
LC_TELEPHONE=ru_RU.UTF-8
QT_IM_MODULE=ibus
LC_MEASUREMENT=ru_RU.UTF-8
XDG_RUNTIME_DIR=/run/user/1000
LC_TIME=ru_RU.UTF-8
JOURNAL_STREAM=9:40812
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share:/usr/share:/var/lib/snapd
/desktop
PATH=/home/user/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin
:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/usr/local/lib/nodejs/node
-v12.9.1-linux-x64/bin:/home/user/.npm-global/bin:/usr/local/go/bin
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
LC_NUMERIC=ru_RU.UTF-8
GOPATH=/home/user/go
OLDPWD=/home/user/bmstu/OS/sem06/Lab04
_=./proc.out

```

Рис. 1: Информация об окружении процесса

```
user@user-Lenovo-ideapad-310-15ISK: ~/bmstu/O...
=== State of the process:
pid             17531
comm            (proc.out)
state           R
ppid           15111
pgrp           17531
session        15111
tty_nr         34816
tpgid          17531
flags          4194304
minflt         80
cmnflt         0
majflt         0
cmajflt        0
utime          0
stime          0
cutime         0
cstime         0
priority       20
nice           0
num_threads    1
itrealvalue    0
starttime      4199251
vsize          2539520
rss            209
rsslim         18446744073709551615
startcode      94321731805184
endcode        94321731812053
startstack     140724461712928
kstkesp        0
kstkeip        0
signal         0
blocked        0
sigignore      0
sigcatch       0
wchan          0
nswap          0
cnsnap         0
exit_signal    17
processor      3
rt_priority    0
policy         0
delayacct_blkio_ticks 0
guest_time     0
cguest_time    0
start_data     94321731820840
end_data       94321731822024
start_brk      94321737007104
arg_start      140724461715880
arg_end        140724461715891
env_start      140724461715891
env_end        140724461719533
exit_code      0

=== Command line for the process with id 17531:
./proc.out
```

Рис. 2: Информация о состоянии процесса

```
user@user-Lenovo-ideapad-310-15ISK: ~/bmstu/O...
guest_time     0
cguest_time    0
start_data     94321731820840
end_data       94321731822024
start_brk      94321737007104
arg_start      140724461715880
arg_end        140724461715891
env_start      140724461715891
env_end        140724461719533
exit_code      0

=== Command line for the process with id 17531:
./proc.out

=== List of open files for the process with id 17531:
/proc/17531/fd/0 -> /dev/pts/0
/proc/17531/fd/1 -> /dev/pts/0
/proc/17531/fd/2 -> /dev/pts/0
/proc/17531/fd/3 -> /proc/17531/fd
```

Рис. 3: Информация из файла `cmdline` и директории `fd`

Часть 2

Задание. Написать загружаемый модуль ядра, создать файл в файловой системе **proc**, поддиректорию и символическую ссылку. Используя соответствующие функции, передать данные из пространства пользователя в пространство ядра (введённые данные вывести в файл ядра) и из пространства ядра в пространство пользователя. Продемонстрировать это.

В листинге 2 содержится код программы загружаемого модуля. На рисунке 4 продемонстрирована работа программы. На рисунке 5 показаны созданные в **/proc** файлы после загрузки модуля.

Листинг 2: task2/fortune.c

```
1 #include <linux/init.h>
2 #include <linux/module.h>
3 #include <linux/kernel.h>
4 #include <linux/proc_fs.h>
5 #include <linux/string.h>
6 #include <linux/vmalloc.h>
7 #include <linux/uaccess.h>
8
9 MODULE_LICENSE("GPL");
10 MODULE_AUTHOR("Kerimov A. IU7-64b");
11 MODULE_DESCRIPTION("LKM fortune");
12
13 #define MAX_COOKIE_LENGTH PAGE_SIZE
14
15 #define FORTUNE_DIRNAME "fortune_dir"
16 #define FORTUNE_FILENAME "fortune_file"
17 #define FORTUNE_SYMLINK "fortune"
18 #define FORTUNE_FILEPATH FORTUNE_DIRNAME "/" FORTUNE_FILENAME
19
20 static struct proc_dir_entry *fortune_dir = NULL;
21 static struct proc_dir_entry *fortune_file = NULL;
22 static struct proc_dir_entry *fortune_symlink = NULL;
23
24 static char *cookie_pot; // Хранилище наших фортунок
25 static int cookie_index; // Индекс для добавления печеньки
26 static int next_fortune; // Индекс для чтения печеньки
27
28 static char tmp[MAX_COOKIE_LENGTH];
29
30 ssize_t fortune_read(struct file *filep, char __user *buf, size_t count, loff_t *offp)
31 {
32     int len;
33
34     if (*offp > 0 || cookie_index == 0)
35         return 0;
36
37     if (next_fortune >= cookie_index)
38         next_fortune = 0;
39
40     len = snprintf(tmp, MAX_COOKIE_LENGTH, "%s\n", &cookie_pot[next_fortune]);
41     if (copy_to_user(buf, tmp, len)) {
42         printk(KERN_ERR "=== fortune: copy_to_user error\n");
43         return -EFAULT;
44     }
45     next_fortune += len;
46     *offp += len;
47
48     return len;
49 }
50
51 ssize_t fortune_write(struct file *file, const char __user *buf, size_t len, loff_t *offp)
52 {
53     if (len > MAX_COOKIE_LENGTH - cookie_index + 1) {
54         printk(KERN_ERR "=== fortune: cookie_pot overflow error\n");
55         return -ENOSPC;
56     }
57
58     if (copy_from_user(&cookie_pot[cookie_index], buf, len)) {
59         printk(KERN_ERR "=== fortune: copy_to_user error\n");
60         return -EFAULT;
61     }
62 }
```

```

62
63     cookie_index += len;
64     cookie_pot[cookie_index - 1] = '\0';
65
66     return len;
67 }
68
69 static struct file_operations fops = {
70     owner: THIS_MODULE,
71     read: fortune_read,
72     write: fortune_write
73 };
74
75 static void cleanup(void)
76 {
77     if (fortune_symlink) remove_proc_entry(FORTUNE_SYMLINK, NULL);
78     if (fortune_file) remove_proc_entry(FORTUNE_FILENAME, fortune_dir);
79     if (fortune_dir) remove_proc_entry(FORTUNE_DIRNAME, NULL);
80     if (cookie_pot) vfree(cookie_pot);
81 }
82
83 static int shutdown_enomem(const char *s)
84 {
85     cleanup();
86     printk(KERN_ERR "=== fortune: %s error\n", s);
87     return -ENOMEM;
88 }
89
90 static int __init fortune_init(void)
91 {
92     if (!(cookie_pot = vmalloc(MAX_COOKIE_LENGTH)))
93         return shutdown_enomem("vmalloc");
94     memset(cookie_pot, 0, MAX_COOKIE_LENGTH);
95
96     if (!(fortune_dir = proc_mkdir(FORTUNE_DIRNAME, NULL)))
97         return shutdown_enomem("proc_create");
98     if (!(fortune_file = proc_create(FORTUNE_FILENAME, 0666, fortune_dir, &fops)))
99         return shutdown_enomem("proc_create");
100     if (!(fortune_symlink = proc_symlink(FORTUNE_SYMLINK, NULL, FORTUNE_FILEPATH)))
101         return shutdown_enomem("proc_symlink");
102
103     cookie_index = 0;
104     next_fortune = 0;
105
106     printk(KERN_INFO "=== fortune: module loaded\n");
107     return 0;
108 }
109
110 static void __exit fortune_exit(void)
111 {
112     cleanup();
113     printk(KERN_INFO "=== fortune: module unloaded\n");
114 }
115
116 module_init(fortune_init)
117 module_exit(fortune_exit)

```

```
user@user-Lenovo-ideapad-310-15ISK: ~/bmstu/OS/sem0...
$ sudo insmod fortune.ko
$ echo "Success is an individual proposition.  Thomas Watson" > /proc/fortune
$ echo "If a man does his best, what else is there?  Gen. Patton" > /proc/fortune
$ echo "Cat: All your base are belong to us.  Zero Wing" > /proc/fortune
$ cat /proc/fortune
Success is an individual proposition.  Thomas Watson
$ cat /proc/fortune
If a man does his best, what else is there?  Gen. Patton
$ cat /proc/fortune
Cat: All your base are belong to us.  Zero Wing
$ cat /proc/fortune
Success is an individual proposition.  Thomas Watson
$
```

Рис. 4: Демонстрация работы модуля **fortune**

```
user@user-Lenovo-ideapad-310-15ISK: ~/bmstu/OS/sem06/lab04/task2
user@user-Lenovo-ideapad-310-15ISK:~/bmstu/OS/sem06/lab04/task2$ sudo insmod fortune.ko
user@user-Lenovo-ideapad-310-15ISK:~/bmstu/OS/sem06/lab04/task2$ ls -al /proc | grep fortune
lrwxrwxrwx   1 root      root           24 anp 17 20:40 fortune -> fortune_dir/fortune_file
dr-xr-xr-x   2 root      root           0 anp 17 20:40 fortune_dir
user@user-Lenovo-ideapad-310-15ISK:~/bmstu/OS/sem06/lab04/task2$ ls -al /proc/fortune_dir
total 0
dr-xr-xr-x   2 root root 0 anp 17 20:40 .
dr-xr-xr-x 266 root root 0 anp 17 2020 ..
-rw-rw-rw-   1 root root 0 anp 17 20:41 fortune_file
user@user-Lenovo-ideapad-310-15ISK:~/bmstu/OS/sem06/lab04/task2$
```

Рис. 5: Созданные файлы в **/proc** после загрузки модуля