

Часть 1

Содержимое файла `/proc/[pid]/environ`

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

Листинг 1: Программа для вывода информации об окружении процесса (`environ`)

```
1 #include <stdio.h>
2
3 #define BUF_SIZE 0x1000
4
5 int main()
6 {
7     char buffer[BUF_SIZE];
8     int len, i;
9     FILE *f;
10
11     f = fopen("/proc/self/environ", "r");
12
13     while ((len = fread(buffer, 1, BUF_SIZE, f)) > 0)
14     {
15         for (i = 0; i < len; i++)
16             if (buffer[i] == 0)
17                 buffer[i] = 10;
18         buffer[len] = 0;
19         printf("%s", buffer);
20     }
21
22     fclose(f);
23     return 0;
24 }
```

```

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;4
2:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lzh=01;31:*.lzm=01;31:*.tlz=01;31:*.txz=01;3
1:*.tzo=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:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzt=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:*.m
jpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=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:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;3
5:*.m4v=01;35:*.mp4v=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:*.enf=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=0
0;36:
LC_MEASUREMENT=ru_RU.UTF-8
LESSCLOSE=/usr/bin/lesspipe %s %s
LC_PAPER=ru_RU.UTF-8
LC_MONETARY=ru_RU.UTF-8
XDG_MENU_PREFIX=gnome-
LANG=en_US.UTF-8
DISPLAY=:0
GNOME_SHELL_SESSION_MODE=ubuntu
COLORTERM=truecolor
USERNAME=olga
XDG_VTNR=2
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
LC_NAME=ru_RU.UTF-8
XDG_SESSION_ID=2
USER=olga
DESKTOP_SESSION=ubuntu
QT4_IM_MODULE=xim
TEXTDOMAINDIR=/usr/share/locale/
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/5acb9a06_5aa9_4074_95cd_a7c066760ff9
PWD=/home/olga/Documents/lab4
HOME=/home/olga
TEXTDOMAIN=im-config
SSH_AGENT_PID=1899
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
LC_ADDRESS=ru_RU.UTF-8
GJS_DEBUG_OUTPUT=stderr
LC_NUMERIC=ru_RU.UTF-8
GTK_MODULES=gail:atk-bridge
WINDOWPATH=2
VTE_VERSION=5202
TERM=xterm-256color
SHELL=/bin/bash
QT_IM_MODULE=ibus
XMODIFIERS=@im=ibus
IM_CONFIG_PHASE=2
XDG_CURRENT_DESKTOP=ubuntu:GNOME
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
GNOME_TERMINAL_SERVICE=:1.67
XDG_SEAT=seat0
SHLVL=1
LC_TELEPHONE=ru_RU.UTF-8
GDMSESSION=ubuntu
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
LOGNAME=olga
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
XDG_RUNTIME_DIR=/run/user/1000
XAUTHORITY=/run/user/1000/gdm/Xauthority
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
LC_IDENTIFICATION=ru_RU.UTF-8
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
SESSION_MANAGER=local/olga-VirtualBox:@/tmp/.ICE-unix/1731,unix/olga-VirtualBox:/tmp/.ICE-unix/1731
LESSOPEN=| /usr/bin/lesspipe %s
GTK_IM_MODULE=ibus
LC_TIME=ru_RU.UTF-8
OLDPWD=/home/olga/Documents
_=/a.out

```

Рис. 1: Вывод информации об окружении процесса

Содержимое файла `/proc/[pid]/stat`

Листинг 2: Программа для вывода информации о процессе (stat)

```
1 #include <stdio.h>
2 #include <string.h>
3
4 #define BUF_SIZE 0x1000
5
6 static char* stat_parameters[] = {"pid", "comm", "state", "ppid",
7     "pgrp", "session",
8     "tty_nr", "tpgid", "flags", "minflt", "cminflt", "majflt", "cmajflt",
9     "utime",
10    "stime", "cutime", "cstime", "priority", "nice", "num_threads",
11    "itrealvalue",
12    "starttime", "vsize", "rss", "rsslim", "startcode", "endcode",
13    "startstack",
14    "kstkesp", "kstkeip", "signal", "blocked", "sigignore", "sigcatch",
15    "wchan",
16    "nswap", "cnswap", "exit_signal", "processor", "rt_priority", "policy",
17    "delayacct_blkio_ticks", "guest_time", "cguest_time", "start_data",
18    "end_data",
19    "start_brk", "arg_start", "arg_end", "env_start", "env_end",
20    "exit_code"};
21
22 int main()
23 {
24     char buffer[BUF_SIZE];
25     FILE *f;
26
27     f = fopen("/proc/self/stat", "r");
28
29     fread(buffer, 1, BUF_SIZE, f);
30     char *pch = strtok(buffer, "_");
31
32     int i = 0;
33     while (pch != NULL)
34     {
35         printf("%s_=", stat_parameters[i]);
36         i++;
37         printf("%s\n", pch);
38         pch = strtok(NULL, "_");
39     }
40
41     fclose(f);
42     return 0;
43 }
```

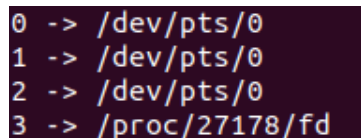
```
pid = 27082
comm = (a.out)
state = R
ppid = 26573
pgrp = 27082
session = 26573
tty_nr = 34816
tpgid = 27082
flags = 4194304
minflt = 70
cminflt = 0
majflt = 0
cmajflt = 0
utime = 0
stime = 0
cutime = 0
cstime = 0
priority = 20
nice = 0
num_threads = 1
itrealvalue = 0
starttime = 1423832
vsize = 4616192
rss = 180
rsslim = 18446744073709551615
startcode = 94433194008576
endcode = 94433194013096
startstack = 140722673505760
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 = 94433196113288
end_data = 94433196114368
start_brk = 94433214140416
arg_start = 140722673513258
arg_end = 140722673513266
env_start = 140722673513266
env_end = 140722673516528
exit_code = 0
```

Рис. 2: Вывод информации о процессе

Содержимое директории /proc/[pid]/fd

Листинг 3: Программа для вывода содержимого директории fd

```
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <string.h>
4 #include <dirent.h>
5
6 #define BUF_SIZE 0x1000
7
8 int main()
9 {
10     struct dirent *dirp;
11     DIR *dp;
12
13     char str[BUF_SIZE];
14     char path[BUF_SIZE];
15
16     dp = opendir("/proc/self/fd/");
17
18     while ((dirp = readdir(dp)) != NULL)
19     {
20         if ((strcmp(dirp->d_name, ".") != 0) && (strcmp(dirp->d_name,
21             "..") != 0))
22         {
23             sprintf(path, "%s%s", "/proc/self/fd/", dirp->d_name);
24             readlink(path, str, BUF_SIZE);
25             printf("%s->%s\n", dirp->d_name, str);
26         }
27     }
28     closedir(dp);
29     return 0;
30 }
```



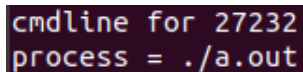
```
0 -> /dev/pts/0
1 -> /dev/pts/0
2 -> /dev/pts/0
3 -> /proc/27178/fd
```

Рис. 3: Содержимое директории fd

Содержимое файла `/proc/[pid]/cmdline`

Листинг 4: Программа для вывода содержимого `cmdline`

```
1 #include <stdio.h>
2 #include <unistd.h>
3
4 #define BUF_SIZE 0x1000
5
6 int main()
7 {
8     char buffer[BUF_SIZE];
9     FILE *f;
10    int len;
11
12    f = fopen("/proc/self/cmdline", "r");
13
14    len = fread(buffer, 1, BUF_SIZE, f);
15    buffer[--len] = 0;
16
17    printf("cmdline_for_%d\nprocess_=%s\n", getpid(), buffer);
18
19    fclose(f);
20    return 0;
21 }
```



```
cmdline for 27232
process = ./a.out
```

Рис. 4: Содержимое `cmdline`

Часть 2

Задание: написать загружаемый модуль ядра, создать файл в файловой системе `proc`, `sysmlink`, `subdir`. Используя соответствующие функции, передать данные из пространства пользователя в пространство ядра (введенные данные вывести в файл ядра) и из пространства ядра в пространство пользователя.

Для передачи данных в ядро из режима пользователя и из ядра в режим пользователя нужны специальные функции `copy_from_user` и `copy_to_user`. В Linux память сегментирована, указатель ссылается не на уникальную позицию в памяти, а на позицию в сегменте (адресация "сегмент-смещение"). Фактически само ядро и каждый из процессов располагаются в своих собственных изолированных адресных пространствах. При выполнении обычной программы адресация происходит автоматически в соответствии с принятой в системе.

Если выполняется код ядра и необходимо получить доступ к сегменту кода ядра, то нужен буфер. Но когда хотим передать информацию между процессом, запущенным в режиме пользователя, и ядром, то соответствующая функция ядра получит указатель на буфер процесса.

Листинг 5: Листинг программы

```
1 #include <linux/module.h>
2 #include <linux/init.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 #include <asm/uaccess.h>
9
10 #define BUF_SIZE PAGE_SIZE
11
12 MODULE_LICENSE("GPL");
13 MODULE_AUTHOR("Kondrashova");
14 MODULE_DESCRIPTION("lab4_proc");
15
16 ssize_t fortune_read(struct file *file, char *buf, size_t count, loff_t
    *f_pos);
17 ssize_t fortune_write(struct file *file, const char *buf, size_t count,
    loff_t *f_pos);
18 int fortune_init(void);
19 void fortune_exit(void);
20
21 struct file_operations fops = {
22     .owner = THIS_MODULE,
23     .read = fortune_read,
24     .write = fortune_write,
25 };
26
```

```

27 static char *buffer;
28 static struct proc_dir_entry *proc_file, *dir, *symlink;
29 int read_index, write_index;
30
31 ssize_t fortune_read(struct file *file, char *buf, size_t count, loff_t
    *f_pos)
32 {
33     int len;
34
35     if (write_index == 0 || *f_pos > 0)
36         return 0;
37
38     if (read_index >= write_index)
39         read_index = 0;
40
41     len = copy_to_user(buf, &buffer[read_index], count);
42     read_index += len;
43     *f_pos += len;
44
45     return len;
46 }
47
48 ssize_t fortune_write(struct file *file, const char *buf, size_t count,
    loff_t *f_pos)
49 {
50     int free_space = (BUF_SIZE - write_index) + 1;
51
52     if (count > free_space)
53     {
54         printk(KERN_INFO "Buffer_is_full\n");
55         return -ENOSPC;
56     }
57
58     if (copy_from_user(&buffer[write_index], buf, count))
59     {
60         return -EFAULT;
61     }
62
63     write_index += count;
64     buffer[write_index-1] = 0;
65
66     return count;
67 }
68
69
70 int fortune_init(void)
71 {
72     buffer = (char*)vmalloc(BUF_SIZE);
73
74     if (!buffer)
75     {
76         printk(KERN_INFO "Not_enough_memory\n");

```



```

77         return -ENOMEM;
78     }
79
80     memset(buffer, 0, BUF_SIZE);
81     proc_file = proc_create("fortune", 0666, NULL, &fops);
82
83     if (!proc_file)
84     {
85         vfree(buffer);
86         printk(KERN_INFO "Cannot_create_fortune_file.\n");
87         return -ENOMEM;
88     }
89
90     dir = proc_mkdir("fortune_dir", NULL);
91     symlink = proc_symlink("fortune_symlink", NULL,
92         "/proc/fortune_dir");
93
94     read_index = 0;
95     write_index = 0;
96
97     printk(KERN_INFO "Fortune_module_loaded.\n");
98     return 0;
99 }
100
101 void fortune_exit(void)
102 {
103     remove_proc_entry("fortune", NULL);
104     remove_proc_entry("fortune_dir", NULL);
105     remove_proc_entry("fortune_symlink", NULL);
106
107     if (buffer)
108         vfree(buffer);
109
110     printk(KERN_INFO "Fortune_module_unloaded.\n");
111 }
112
113 module_init(fortune_init);
114 module_exit(fortune_exit);

```

Сборка и загрузка модуля ядра:

```
make -C /lib/modules/5.3.0-45-generic/build M=/home/olga/Documents/4 modules
make[1]: Entering directory '/usr/src/linux-headers-5.3.0-45-generic'
  CC [M]  /home/olga/Documents/4/fortune.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/olga/Documents/4/fortune.mod.o
  LD [M]  /home/olga/Documents/4/fortune.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.3.0-45-generic'
sudo make clean
rm -rf .tmp_versions
rm .fortune.*
rm *.o
rm *.mod.c
rm *.symvers
rm *.order
olga@olga-VirtualBox:~/Documents/4$ sudo insmod fortune.ko
olga@olga-VirtualBox:~/Documents/4$ sudo dmesg | tail -1
[ 4814.880499] Fortune module loaded.
```

Проверка создания файла, директории и символической ссылки:

```
olga@olga-VirtualBox:~/Documents/4$ ls -l /proc | grep fortune
-rw-rw-rw- 1 root      root           0 anp  2 23:21 fortune
dr-xr-xr-x 2 root      root           0 anp  2 23:21 fortune_dir
lrwxrwxrwx 1 root      root          17 anp  2 23:21 fortune_symlink -> /proc/fortune_dir
```

Отправка и получения сообщения:

```
olga@olga-VirtualBox:~/Documents/4$ echo "This is message" > /proc/fortune
olga@olga-VirtualBox:~/Documents/4$ cat /proc/fortune
This is message
```

Выгрузка модуля ядра:

```
olga@olga-VirtualBox:~/Documents/4$ sudo rmmod fortune
olga@olga-VirtualBox:~/Documents/4$ sudo dmesg | tail -2
[ 4814.880499] Fortune module loaded.
[ 4915.016402] Fortune module unloaded.
```