



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

Тема Виртуальная файловая система /proc

Студент Сушина А.Д.

Группа ИУ7-616

Оценка (баллы) _____

Преподаватель Рязанова Н.Ю.

Москва.
2020 г

Оглавление

1 Задание на лабораторную работу.....	3
2 Листинг программ.....	3
3 Результаты работы программы.....	7

1 Задание на лабораторную работу

Цель лабораторной работы: изучить работу файловой системы /proc.

Часть 1.

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

Часть 2.

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. Продемонстрировать это.

2 Листинг программ

На листингах 1-2 представлен код программ для 1 и второй части лабораторной работы.

Листинг 1. Часть 1. Вывод информации об окружении, состоянии процесса и информацию из cmdline и директории fd.

```
1. static int i = 0;

2. void statOutput(char *buf)
3. {
4.     int len = strlen(buf);
5.     int currentName = 0;
6.     char *pch = strtok(buf, " ");
7.
8.     while (pch != NULL && i < 51)
9.     {
10.         printf("\n%15s:\t %s", outputNames[i], pch);
11.         pch = strtok(NULL, " ");
12.         i++;
13.     }
14. }

15. void simpleOutput(char *buf)
16. {
17.     printf("%s\n", buf);
18. }

19. void read_one_file(char* filename, void (*print_func)(char*))
20. {
21.     char buf[BUF_SIZE];
22.     int i, len;
23.     FILE *f = fopen(filename, "r");
24.     while ((len = fread(buf, 1, BUF_SIZE, f)) > 0)
```

```

25.     {
26.         for (i = 0; i < len; i++)
27.             if( buf[i] == 0)
28.                 buf[i] = 10;
29.         buf[len - 1] = 0;
30.         print_func(buf);
31.     }
32.     fclose(f);
33. }

34. int main(int argc, char *argv[])
35. {
36.     printf("\n_____ \n");
37.     printf("STAT \n\n");
38.     read_one_file("/proc/self/stat", statOutput);
39.     printf("\n_____ \n");
40.     printf("ENVIRON\n\n");
41.     read_one_file("/proc/self/environ", simpleOutput);
42.     printf("\n_____ \n");
43.     printf("CMDLINE\n\n");
44.     read_one_file("/proc/self/cmdline", simpleOutput);
45.     printf("\n_____ \n");
46.     printf("FD\n\n");
47.     execl("/bin/ls", "ls", "/proc/self/fd", NULL);
48.     return 0;
49. }

```

Листинг 2. Часть 2. Модуль ядра fortune

```

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 <linux/sched.h>
9. #include <linux/init_task.h>

10.     MODULE_LICENSE("GPL");
11.     MODULE_DESCRIPTION("Fortune Cookie Kernel Module");

12.     #define COOKIE_BUF_SIZE PAGE_SIZE
13.     #define TEMP_BUF_SIZE 256

14.     ssize_t fortune_read(struct file *file, char *buf, size_t count,
15.         loff_t *f_pos);
16.     ssize_t fortune_write(struct file *file, const char *buf, size_t
17.         count, loff_t *f_pos);

18.     int fortune_init(void);
19.     void fortune_exit(void);

20.     struct file_operations fops = {
21.         .owner = THIS_MODULE,
22.         .read = fortune_read,
23.         .write = fortune_write,

```

```

22.     };

23.     static char *cookie_buf;
24.     static struct proc_dir_entry *proc_entry;
25.     static unsigned read_index;
26.     static unsigned write_index;

27.     char temp[TEMP_BUF_SIZE];

28.     struct task_struct *task = &init_task;

29.     int len;

30.     ssize_t fortune_read(struct file *file, char *buf, size_t count,
lofff_t *f_pos)
31.     {
32.         if (*f_pos > 0)
33.             return 0;

34.         if (read_index >= write_index)
35.             read_index = 0;

36.         len = 0;
37.         if (write_index > 0)
38.         {
39.             len = sprintf(temp, "%s\n", &cookie_buf[read_index]);

40.             copy_to_user(buf, temp, len);
41.             buf += len;
42.             read_index += len;
43.         }

44.         *f_pos += len;

45.         return len;
46.     }

47.     ssize_t fortune_write(struct file *file, const char *buf, size_t
count, lofff_t *f_pos)
48.     {
49.         int space_available = (COOKIE_BUF_SIZE - write_index) + 1;

50.         if (count > space_available)
51.         {
52.             printk(KERN_INFO "+_+ cookie pot is full\n");
53.             return -ENOSPC;
54.         }

55.         if (copy_from_user(&cookie_buf[write_index], buf, count))
56.             return -EFAULT;

57.         write_index += count;
58.         cookie_buf[write_index - 1] = 0;

59.         return count;
60.     }

```

```

61.     int fortune_init(void)
62.     {
63.         cookie_buf = (char *) vmalloc(COOKIE_BUF_SIZE);

64.         if (!cookie_buf)
65.         {
66.             printk(KERN_INFO "+_+ not enough memory for the cookie pot\
n");
67.             return -ENOMEM;
68.         }

69.         memset(cookie_buf, 0, COOKIE_BUF_SIZE);
70.         proc_entry = proc_create("fortune", 0666, NULL, &fops);

71.         if (!proc_entry)
72.         {
73.             vfree(cookie_buf);
74.             printk(KERN_INFO "+_+ Couldn't create proc entry\n");
75.             return -ENOMEM;
76.         }

77.         read_index = 0;
78.         write_index = 0;

79.         proc_mkdir("my_dir_fortune", NULL);
80.         proc_symlink("my_symbolic_fortune", NULL, "/proc/fortune");

81.         printk(KERN_INFO "+_+ fortune module loaded.\n");
82.         return 0;
83.     }

84.     void fortune_exit(void)
85.     {
86.         remove_proc_entry("fortune", NULL);

87.         if (cookie_buf)
88.             vfree(cookie_buf);

89.         printk(KERN_INFO "+_+ fortune module unloaded.\n");
90.     }

91.     module_init(fortune_init);
92.     module_exit(fortune_exit);

```

3 Результаты работы программы

```
STAT

      pid:          7227
  filename:        (main.exe)
      state:        R
      ppid:         4578
      gid:          7227
  session:         4578
  tty_nr:          34817
  tp_gid:          7227
  flags:           4194304
  minflt:          72
  cminflt:          0
  majflt:           0
  cmajflt:          0
  utime:            0
  stime:            0
  cutime:           0
  cstime:           0
  priority:         20
    nice:           0
  num_threads:      1
  itrealvalue:      0
  start_ttime:      366073
    vsize:          4620288
      rss:           191
  rsslim:           18446744073709551615
  startcode:         94101579444224
  endcode:           94101579449544
  startstack:        140730375982352
  kstkesp:           0
  kstkeip:           0
  signal:            0
  blocked:           0
  sigignore:         0
  sigcatch:          0
  wchan:             0
  nswap:             0
  cnsnap:            0
  exit_signal:       17
  processor:         5
  rt_priority:       0
  policy:            0
  delayacct_blkio_tics: 0
    quest_time:      0
  cquest_time:       0
  start_data:         94101581548920
  end_data:           94101581550016
  start_brk:          94101582249984
  arg_start:          140730375987715
  arg_end:            140730375987726
  env_start:          140730375987726
  env_end:            140730375991277
```

Рис 1. Вывод программы из части 1. Содержание файла STAT.

```

ENVIRON

CLUTTER_IM_MODULE=xim
NVM_DIR=/home/nastya/.nvm
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=3
7;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:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tztst=0
1;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.taz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sa
r=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:*.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:*.
pg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vo
b=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:*.o
gx=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;
6:*.spx=00;36:*.xspf=00;36:
LESSCLOSE=/usr/bin/lesspipe %s %s
ANDROID_HOME=/home/nastya/Android/Sdk
XDG_MENU_PREFIX=gnome-
LANG=ru_RU.UTF-8
DISPLAY=:1
OLDPWD=/home/nastya/lu7/sem6/os/lab4
GNOME_SHELL_SESSION_MODE=ubuntu
GTK2_MODULES=overlay-scrollbar
COLORTERM=truecolor
NVM_CD_FLAGS=
USERNAME=nastya
XDG_VTNR=2
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
XDG_SESSION_ID=3
USER=nastya
DESKTOP_SESSION=ubuntu
QT4_IM_MODULE=xim
TEXTDOMAINDIR=/usr/share/locale/
G
OME_TERMINAL_SCREEN=/org/gnome/terminal/screen/d6e90db6_6c2b_493f_ac51_e5ac60503f1c
DEFAULTS_PATH=/usr/share/gconf/ubuntu.default.path
QT_QPA_PLATFORMTHEME=appmenu-qt5
PWD=/home/nastya/lu7/sem6/os/lab4/part1
HOME=/home/nastya
TEXTDOMAIN=im-config
SSH_AGENT_PID=1657
QT_ACCESSIBILITY=1

```

Рис 2. Вывод программы из части 1. Содержимое файла ENVIRON

Рис 3.
Вывод

```

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
GJS_DEBUG_OUTPUT=stderr
GTK_MODULES=gail:atk-bridge
WINDOWPATH=2
TERM=xterm-256color
SHELL=/bin/bash
VTE_VERSION=5202
QT_IM_MODULE=ibus
XMODIFIERS=@im=ibus
IM_CONFIG_PHASE=2
NVM_BIN=/home/nastya/.nvm/versions/node/v12.16.1/bin
XDG_CURRENT_DESKTOP=ubuntu:GNOME
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
GNOME_TERMINAL_SERVICE=:1.145
XDG_SEAT=seat0
SHLVL=1
GDMSESSION=ubuntu
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
LOGNAME=nastya
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:/
tc/xdg
PATH=/home/nastya/.nvm/versions/node/v12.16.1/bin:/home/nastya/bin:/home/nastya/.local/bin:/usr/local/sbin:/usr/
local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/home/nastya/Android/Sdk/tools:/ho
me/nastya/Android/Sdk/platform-tools:/home/nastya/android-studio/bin
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
SESSION_MANAGER=local/Nastya:@/tmp/.ICE-unix/1571,unix/Nastya:/tmp/.ICE-unix/1571
LESSOPEN=| /usr/bin/lesspipe %s
GTK_IM_MODULE=ibus
_=./main.exe

CMDLINE

./main.exe

```

программы из части 1. Содержимое файла Environ, cmdline.


```

FD
итого 0
lrwx----- 1 nastya nastya 64 map 30 17:54 0 -> /dev/pts/1
lrwx----- 1 nastya nastya 64 map 30 17:54 1 -> /dev/pts/1
lrwx----- 1 nastya nastya 64 map 30 17:54 2 -> /dev/pts/1
lr-x----- 1 nastya nastya 64 map 30 17:54 3 -> /proc/9853/fd
nastya@Nastya:~/iu7/sen6/os/lab4/part1$

```

Рис 4. Вывод программы из части 1. Содержимое директории fd.

```

nastya@Nastya:~/iu7/sen6/os/lab4/part2$ sudo insmod fortune.ko
nastya@Nastya:~/iu7/sen6/os/lab4/part2$ lsmod | grep fortune
fortune                16384 0
nastya@Nastya:~/iu7/sen6/os/lab4/part2$ echo "Success is an individual proposition. Thomas Watson" > /proc/fortune
nastya@Nastya:~/iu7/sen6/os/lab4/part2$ echo "If a man does his best, what else is there? Gen. Patton" > /proc/fortune
nastya@Nastya:~/iu7/sen6/os/lab4/part2$ echo "Cats: All your base are belong to us. Zero Wing" > /proc/fortune
nastya@Nastya:~/iu7/sen6/os/lab4/part2$ cat /proc/fortune
Success is an individual proposition. Thomas Watson
nastya@Nastya:~/iu7/sen6/os/lab4/part2$ cat /proc/fortune
If a man does his best, what else is there? Gen. Patton
nastya@Nastya:~/iu7/sen6/os/lab4/part2$ cat /proc/fortune
Cats: All your base are belong to us. Zero Wing

```

Рис 5. Демонстрация работы загружаемого модуля ядра Fortune

```

nastya@Nastya:~$ ls -l /proc | grep fortune
-rw-rw-rw- 1 root      root      0 anp 12 16:28 fortune
dr-xr-xr-x 2 root      root      0 anp 12 16:28 my_dir_fortune
lrwxrwxrwx 1 root      root     13 anp 12 16:28 my_symbolic_fortune -> /proc/fortune

```

Рис 6. Созданные файлы в директории /proc

На рисунке 1 представлено содержимое файла stat. В этом файле содержится вся информация о процессе. В данном случае это процесс main.exe (написанная мной программа, которая выводит информацию из файла).

На рисунке 2 и 3 представлено содержимое файла environ. Данный файл содержит исходное окружение, которое было установлено при запуске текущего процесса. Переменные окружения разделены символами конца строки (для доступа к поддиректории текущего '\0').

Также на рисунке 3 представлен вывод файла cmdline. Данный файл содержит полную командную строку процесса, если процесс не находится в состоянии зомби, иначе файл пуст. В данном случае выводит ./main.exe, так как этой командой был запущен текущий процесс.

На рисунке 4 представлено содержимое директории fd. Данная поддиректория содержит одну запись для каждого файла, который открыт процессом. Имя каждой такой записи соответствует номеру файлового дескриптора и является символьной ссылкой на реальный файл. Программа вывела 4 файла, являющиеся символическими ссылками на файлы, которые были открыты процессом.

На рисунке 5 представлен пример работы загружаемого модуля ядра fortune.

Из листинга 2 видно, что помимо файла /proc/fortune (строка 70), создается также поддиректория «my_dir_fortune» (строка 79) и символическая ссылка «my_symbolic_fortune» (строка 80). С помощью символической ссылки можно также обращаться к файлу fortune.

На рисунке 6 видно, что в файловой системе /proc были созданы три файла: файл /proc/fortune, директория «my_dir_fortune» и символическая ссылка «my_symbolic_fortune».