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 Операционые системы.
Тема Виртуальная файловая система /proc

Студент Степанов А. О.
Группа ИУ7-63Б
Оценка (баллы)
Преподаватель Рязанова Н.Ю.

Москва, 2020 г.

ЧАСТЬ 1

Задание

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

Листинги

Листинг 1: Вывод содержимого файла

```
1  int print_file(const char *filename)
2  {
3      char buf[BUF_SIZE] = { 0 };
4      int len, i;
5      FILE *f;
6
7      f = fopen(filename, "r");
8
9      if (f == NULL)
10     {
11         perror("Can't␣open␣file\n");
12         return -1;
13     }
14
15     while ((len = fread(buf, 1, BUF_SIZE, f)) > 0)
16     {
17         for (i = 0; i < len; ++i)
18             if (buf[i]  == 0)
19                 buf[i] = 10;
20
21         buf[len - 1] = 0;
22         printf("%s", buf);
23     }
24
25     printf("\n");
26
27     fclose(f);
28     return 0;
29 }
```

Листинг 2: Вывод содержимого файла `stat`

```
1  int print_stat()
2  {
3      char buf[BUF_SIZE] = { 0 };
4      FILE *f = fopen("/proc/self/stat", "r");
5
6      if (f == NULL)
7      {
8          perror("Can't␣open␣stat\n");
9          return -1;
10     }
11
12     fread(buf, 1, BUF_SIZE, f);
13     char *pch = strtok(buf, "␣");
14
15     while(pch != NULL)
16     {
17         printf("%s\n", pch);
18         pch = strtok(NULL, "␣");
19     }
20
21     fclose(f);
22     return 0;
23 }
```

Листинг 3: Вывод содержимого директории

```
1  int print_directory(const char *dirname)
2  {
3      struct dirent *dirp;
4      DIR *dp;
5      char str[BUF_SIZE] = { 0 };
6      char path[BUF_SIZE] = { 0 };
7
8      dp = opendir(dirname);
9
10     if (dp == NULL)
11     {
12         perror("Can't␣open␣dir");
13         return -1;
14     }
15
16     while ((dirp = readdir(dp)) != NULL)
```

```
17      {
18          if ((strcmp(dirp->d_name, ".") != 0) &&
19              (strcmp(dirp->d_name, "..") != 0))
20          {
21              sprintf(path, "%s%s", dirname, dirp->d_name);
22              readlink(path, str, BUF_SIZE);
23              printf("%s -> %s\n", dirp->d_name, str);
24          }
25      }
26
27      closedir(dp);
28      return 0;
29  }
```

Листинг 4: Функция `main`

```
1  int main(int argc, char *argv[])
2  {
3      printf("ENVIRON:\n");
4      if (print_file("/proc/self/environ") < 0)
5          return -1;
6
7      printf("CMDLINE:\n");
8      if (print_file("/proc/self/cmdline") < 0)
9          return -1;
10
11     printf("STAT:\n");
12     if (print_stat() < 0)
13         return -1;
14
15     printf("FD:\n");
16     if (print_directory("/proc/self/fd/") < 0)
17         return -1;
18
19     return 0;
20 }
```

Результат работы

Рис. 1: Вывод информации об окружении процесса

Рис. 2: Вывод директории процесса (файл `cmdline`)

Рис. 3: Вывод содержания директории `fd`

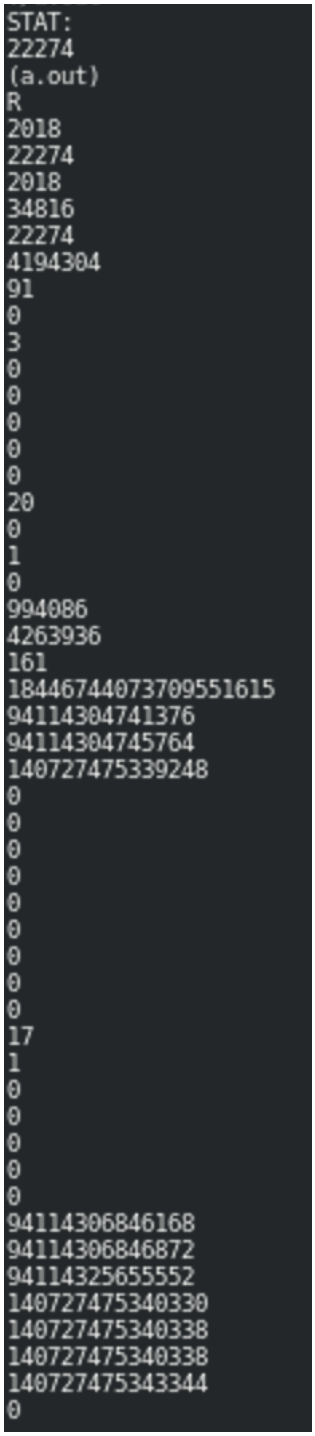

Рис. 4: Вывод состояния процесса (файл `stat`)

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.

Листинги

Листинг 5: Код модуля

```
1  #include <linux/module.h>
2  #include <linux/moduleparam.h>
3  #include <linux/init.h>
4  #include <linux/kernel.h>
5  #include <linux/proc_fs.h>
6  #include <asm/uaccess.h>
7  #include <linux/vmalloc.h>
8
9  #define MAX_COOKIE_LENGTH PAGE_SIZE
10
11 MODULE_LICENSE("Dual␣BSD/GPL");
12 MODULE_AUTHOR("Alexander␣Stepanov");
13
14 static struct proc_dir_entry *proc_entry;
15 static struct proc_dir_entry *proc_directory;
16 static struct proc_dir_entry *proc_link;
17 static char *cookie_pot;
18 static int cookie_index;
19 static int next_fortune;
20
21 static ssize_t fortune_write(struct file *f, const char __user *buff,
22     size_t len, loff_t *data)
23 {
24     int space_available = (MAX_COOKIE_LENGTH - cookie_index) + 1;
25
26     if (len > space_available)
27     {
28         printk(KERN_INFO "fortune:␣cookie␣pot␣is␣full!\n");
```

```
29          return -ENOSPC;
30      }
31
32      if (copy_from_user(&cookie_pot[cookie_index], buff, len))
33      {
34          return -EFAULT;
35      }
36
37      cookie_index += len;
38      cookie_pot[cookie_index-1] = 0;
39
40      return len;
41  }
42
43  static ssize_t fortune_read(struct file *f, char __user *buff,
44      size_t len, loff_t *data)
45  {
46      if (*data > 0) return 0;
47
48      if (next_fortune >= cookie_index)
49          next_fortune = 0;
50
51      len = copy_to_user(buff, &cookie_pot[next_fortune], len);
52      next_fortune += len;
53
54      *data = 1;
55
56      return len;
57  }
58
59  static struct file_operations ops =
60  {
61      .owner = THIS_MODULE,
62      .read = fortune_read,
63      .write = fortune_write,
64  };
65
66  static int md_init(void)
67  {
68      int ret = 0;
69      cookie_pot = (char *)vmalloc( MAX_COOKIE_LENGTH);
70
```

```
71      if (!cookie_pot)
72      {
73          ret = -ENOMEM;
74      }
75      else
76      {
77          memset(cookie_pot, 0, MAX_COOKIE_LENGTH );
78          proc_entry = proc_create("fortune", 0644, NULL, &ops);
79
80          if (proc_entry == NULL)
81          {
82              ret = -ENOMEM;
83              vfree(cookie_pot);
84              printk(KERN_INFO "fortune:␣Couldn't␣create␣proc␣entry\n");
85          }
86          else
87          {
88              cookie_index = 0;
89              next_fortune = 0;
90              printk(KERN_INFO "fortune:␣Module␣loaded.\n");
91
92              proc_directory = proc_mkdir("fortune_dir", NULL);
93
94              if (proc_directory == NULL)
95              {
96                  ret = -ENOMEM;
97                  printk(KERN_ERR "fortune:␣Couldn't␣create␣dir");
98              }
99
100             proc_link = proc_symlink("fortune_link", NULL, "fortune");
101
102             if (proc_link == NULL)
103             {
104                 ret = -ENOMEM;
105                 printk(KERN_ERR "fortune:␣Couldn't␣create␣symlink");
106             }
107         }
108     }
109
110     return ret;
111 }
112
```

```
113 static void md_exit(void)
114 {
115     proc_remove(proc_entry);
116     proc_remove(proc_directory);
117     proc_remove(proc_link);
118     vfree(cookie_pot);
119     printk(KERN_INFO "fortune:␣Module␣unloaded.\n");
120 }
121
122 module_init(md_init);
123 module_exit(md_exit);
```

Результат работы

```
debian-gnu-linux-vm# make
make -C /lib/modules/4.9.0-12-amd64/build M=/media/psf/Home/Repositories/operating-systems/sem_02/lab_04/part_02 modules
make[1]: Entering directory '/usr/src/linux-headers-4.9.0-12-amd64'
  CC [M]  /media/psf/Home/Repositories/operating-systems/sem_02/lab_04/part_02/md.o
  Building modules, stage 2.
  MODPOST 1 modules
  LD [M]  /media/psf/Home/Repositories/operating-systems/sem_02/lab_04/part_02/md.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.9.0-12-amd64'
debian-gnu-linux-vm# insmod md.ko
debian-gnu-linux-vm#
```

Рис. 5: Сборка и загрузка модуля

```
debian-gnu-linux-vm# lsmod
Module                  Size  Used by
md                     16384  0
fuse                   98304  3
```

Рис. 6: Модуль загружен

```
debian-gnu-linux-vm# ls -l /proc | grep fortune
-rw-r--r--  1 root             root                          0 Mar 21 21:28 fortune
dr-xr-xr-x  2 root             root                          0 Mar 21 21:28 fortune_dir
lrwxrwxrwx  1 root             root                          7 Mar 21 21:28 fortune_link -> fortune
```

Рис. 7: Модуль создал файл, символьную ссылку и директорию

```
debian-gnu-linux-vm# echo "test1\n" > /proc/fortune
debian-gnu-linux-vm# echo "test2\n" > /proc/fortune
debian-gnu-linux-vm# cat /proc/fortune
test1
test2
```

Рис. 8: Проверка работы файла

Рис. 9: Проверка работы ссылки