

## Модули ядра задание 3

### Тест

```
[16:09:57] ~/develop/kernel/test_dev_file
> cat /dev/test_dev_file
hello! [16:10:06] ~/develop/kernel/test_dev_file
> cat /proc/test_proc
hello! [16:10:15] ~/develop/kernel/test_dev_file
> cat /sys/kernel/test_kobj/test_string
hello! [16:10:29] ~/develop/kernel/test_dev_file
> echo test1 > /proc/test_proc
[16:10:51] ~/develop/kernel/test_dev_file
> cat /sys/kernel/test_kobj/test_string
test1
[16:10:54] ~/develop/kernel/test_dev_file
> cat /proc/test_proc
test1
```

### Вывод dmesg

```
[ 9635.172237] test_dev_file: --init_module()--
[ 9635.172255] test_dev_file module: Major number is 242
[ 9644.474774] test_dev_file: --test_read()-- buff <<< hello! rc=6
[ 9644.474828] test_dev_file: --test_read()-- buff <<< hello! rc=0
[ 9653.353259] test_dev_file: --test_proc_read()-- buff <<< hello! rc=6
[ 9653.353309] test_dev_file: --test_proc_read()-- buff <<< hello! rc=0
[ 9667.742882] test_dev_file: --test_sys_read()-- buff <<< hello! rc=6
[ 9689.444635] test_dev_file: --test_proc_write()-- test_string <<< test1
[ 9692.455899] test_dev_file: --test_sys_read()-- buff <<< test1
rc=6
[ 9696.151998] test_dev_file: --test_proc_read()-- buff <<< test1
rc=6
```

### Код test\_dev\_file.c

```
1  #include <linux/module.h>
2  #include <linux/printk.h>
3  #include <linux/kernel.h>
4  #include <linux/fs.h>
5  #include <linux/rwlock.h>
6  #include <linux/string.h>
7
8  #include <linux/proc_fs.h>
9  #include <linux/sysfs.h>
10 #include <linux/kobject.h>
11
12 #define BUFLen 30
13
14 static int major = 0;
15
16 static struct proc_dir_entry *test = NULL;
17 static struct kobject *test_kobj = NULL;
18
```

```

19 static rwlock_t lock;
20 static char test_string[BUFLEN] = "hello!";
21
22
23
24 ssize_t test_read(struct file *fd, char __user *buff, size_t size, loff_t *off)
25 {
26     size_t rc = 0;
27
28     //read_lock(&lock); <----- ошибка rc = -14. Неправильный адрес
29     rc = simple_read_from_buffer(buff, size, off, test_string,
30 strlen(test_string));
31     //read_unlock(&lock);
32     pr_info("test_dev_file: --test_read()-- buff <<< %s rc=%d", buff, rc);
33
34     return rc;
35 }
36
37 ssize_t test_write(struct file *fd, const char __user *buff, size_t size, loff_t *off)
38 {
39     size_t rc = 0;
40     if(size > BUFLen)
41         return -EINVAL;
42
43     write_lock(&lock);
44     rc = simple_write_to_buffer(test_string, BUFLen, off, buff, size);
45     write_unlock(&lock);
46     pr_info("test_dev_file: --test_write()-- test_string <<< %s", test_string);
47     return rc;
48 }
49
50
51 static struct file_operations fops =
52 {
53     .owner = THIS_MODULE,
54     .read = test_read,
55     .write = test_write
56 };
57
58
59 static ssize_t test_proc_read(struct file *fd, char __user *buff, size_t size, loff_t
60 *off)
61 {
62     size_t rc = 0;
63     //read_lock(&lock); <----- ошибка rc = -14. Неправильный адрес
64     rc = simple_read_from_buffer(buff, size, off, test_string,
65 strlen(test_string));
66     //read_unlock(&lock);
67     pr_info("test_dev_file: --test_proc_read()-- buff <<< %s rc=%d", buff, rc);
68     return rc;
69 }
70
71 static ssize_t test_proc_write(struct file *fd, const char __user *buff, size_t size,
72 loff_t *off)
73 {
74     size_t rc = 0;
75     if(size > BUFLen)
76         return -EINVAL;
77
78     write_lock(&lock);
79     rc = simple_write_to_buffer(test_string, BUFLen, off, buff, size);
80     write_unlock(&lock);
81     pr_info("test_dev_file: --test_proc_write()-- test_string <<< %s",
82 test_string);
83     return rc;

```

```

84 }
85
86
87 static ssize_t test_sys_read(struct kobject *kobj, struct kobj_attribute *attr, char
88 *buff)
89 {
90     read_lock(&lock);
91     memcpy(buff, test_string, strlen(test_string));
92     read_unlock(&lock);
93     pr_info("test_dev_file: --test_sys_read()-- buff <<< %s rc=%d", buff,
94 strlen(buff));
95     return strlen(buff);
96 }
97
98 static ssize_t test_sys_write(struct kobject *kobj, struct kobj_attribute *attr, const
99 char *buff, size_t count)
100 {
101     if(count > BUFLen)
102         return -EINVAL;
103
104     write_lock(&lock);
105     memcpy(test_string, buff, count);
106     write_unlock(&lock);
107     pr_info("test_dev_file: --test_sys_write()-- test_string <<< %s",
108 test_string);
109     return strlen(test_string);
110 }
111
112 static const struct proc_ops pops =
113 {
114     .proc_read = test_proc_read,
115     .proc_write = test_proc_write
116 };
117
118 static struct kobj_attribute string_attribute =
119 __ATTR(test_string, 0644, test_sys_read, test_sys_write);
120
121 static struct attribute *attrs[] =
122 {
123     &string_attribute.attr,
124     NULL
125 };
126
127 static struct attribute_group attr_group =
128 {
129     .attrs = attrs
130 };
131
132 int init_module(void) {
133
134     int retval = 0;
135     pr_info("test_dev_file: --init_module()--");
136     rwlock_init(&lock);
137     major= register_chrdev(major, "test_dev_file", &fops);
138
139     if(major < 0)
140         return major;
141     pr_info("test_dev_file module: Major number is %d", major);
142
143     test = proc_create("test_proc", 0666, NULL, &pops);
144     test_kobj = kobject_create_and_add("test_kobj", kernel_kobj);
145     if(!test_kobj)
146         return -ENOMEM;
147
148     retval = sysfs_create_group(test_kobj, &attr_group);

```

```
149
150     if(retval)
151         kobject_put(test_kobj);
152     return retval;
153 }
154
155 void cleanup_module(void)
156 {
157     unregister_chrdev(major, "test_dev_file");
158     proc_remove(test);
159     kobject_put(test_kobj);
160     //pr_info("test_dev_file: --Cleanup_module()--\n");
161     pr_info("test_dev_file: --CleanUp()-- test_string = %s", test_string);
162 }
163
164 MODULE_LICENSE("GPL");
```