Модули ядра 2

Тест /dev/test_dev_file

```
[13:25:53] ~/develop/kernel/test_dev_file
> cat /dev/test_dev_file
hello![13:27:03] ~/develop/kernel/test_dev_file
> echo test > /dev/test_dev_file
[13:32:25] ~/develop/kernel/test_dev_file
> cat /dev/test_dev_file
test
![13:32:27] ~/develop/kernel/test_dev_file
>
```

Вывод dmesg

```
[ 6295.254468] test_dev_file: --init_module()--
[ 6295.254504] test_dev_file module: Major number is 242
[ 6365.422220] test_dev_file: --test_read()-- buff <<< hello! rc=6
[ 6365.422302] test_dev_file: --test_read()-- buff <<< hello! rc=0
[ 6687.569276] test_dev_file: --test_write()-- test_string <<< test
               !
[ 6689.217945] test_dev_file: --test_read()-- buff <<< test
               ! rc=6
```

Создание файла устройства

```
> sudo mknod /dev/test_dev_file c 242 1
```

Код test_dev_file.c

```c
#include <linux/module.h>
#include <linux/printk.h>
#include <linux/kernel.h>
#include <linux/fs.h>
#include <linux/rwlock.h>
#include <linux/string.h>
#define BUFLEN 30

static int major = 0;
static rwlock_t lock;
static char test_string[BUFLEN] = "hello!";

ssize_t test_read(struct file *fd, char __user *buff, size_t size, loff_t *off)
{
        size_t rc = 0;
        // read_lock(&lock); << вызывает ошибку rc = -14
        rc = simple_read_from_buffer(buff, size, off, test_string,
strlen(test_string));
```

```
        // read_unlock(&lock);
        pr_info("test_dev_file: --test_read()-- buff <<< %s rc=%d", buff, rc);
        return rc;
}

ssize_t test_write(struct file *fd, const char __user *buff, size_t size, loff_t *off)
{
        size_t rc = 0;
        if(size > BUFLEN)
                return -EINVAL;

        write_lock(&lock);
        rc = simple_write_to_buffer(test_string, BUFLEN, off, buff, size);
        write_unlock(&lock);
        pr_info("test_dev_file: --test_write()-- test_string <<< %s",     test_string);
        return rc;
}

static struct file_operations fops = {
        .owner = THIS_MODULE,
        .read = test_read,
        .write = test_write
};

int init_module(void) {
        pr_info("test_dev_file: --init_module()--");
        rwlock_init(&lock);
        major= register_chrdev(major, "test_dev_file", &fops);

        if(major < 0)
                return major;
        pr_info("test_dev_file module: Major mumber is %d", major);

        return 0;
}

void cleanup_module(void) {
        unregister_chrdev(major, "test_dev_file");
        pr_info("test_dev_file: --CleanUp()--");
}

MODULE_LICENSE("GPL");
```