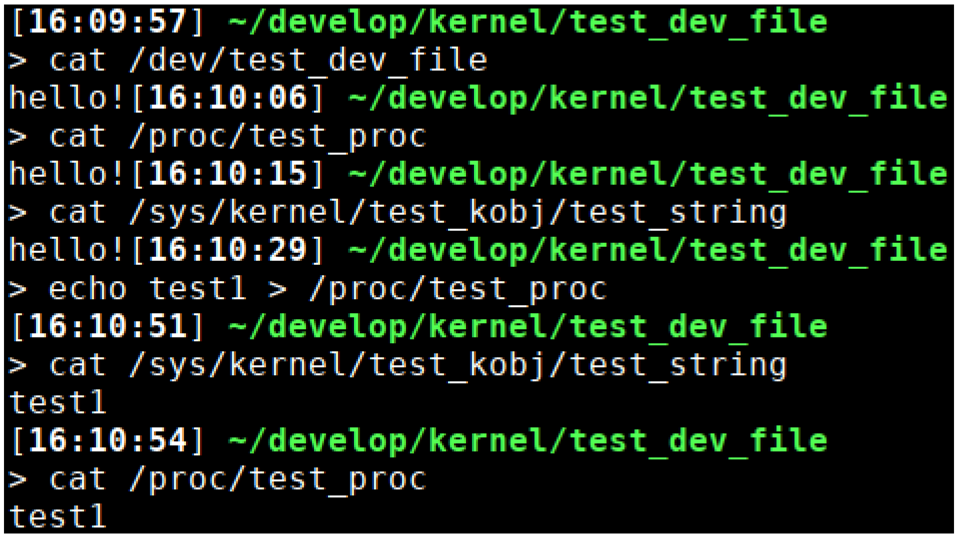
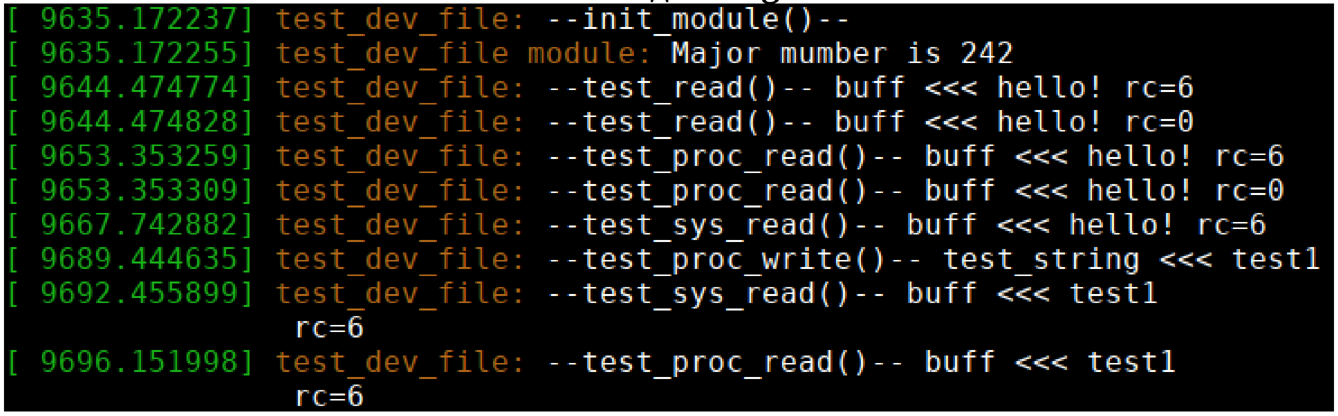
Модули ядра задание 3

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 mumber 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

```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>

#include <linux/proc_fs.h>
#include <linux/sysfs.h>
#include <linux/kobject.h>

#define BUFLEN 30

static int major = 0;

static struct proc_dir_entry *test = NULL;
static struct kobject *test_kobj = NULL;

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); <----- ошибка rc = -14. Неправильный адрес
        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
};


static ssize_t test_proc_read(struct file *fd, char __user *buff, size_t size, loff_t
*off)
{
        size_t rc = 0;
        //read_lock(&lock); <----- ошибка rc = -14. Неправильный адрес
        rc = simple_read_from_buffer(buff, size, off, test_string,
strlen(test_string));
        //read_unlock(&lock);
        pr_info("test_dev_file: --test_proc_read()-- buff <<< %s rc=%d", buff, rc);
        return rc;
}

static ssize_t test_proc_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_proc_write()-- test_string <<< %s",
test_string);
        return rc;
```

```c
}


static ssize_t test_sys_read(struct kobject *kobj, struct kobj_attribute *attr, char
*buff)
{
        read_lock(&lock);
        memcpy(buff, test_string, strlen(test_string));
        read_unlock(&lock);
        pr_info("test_dev_file: --test_sys_read()-- buff <<< %s rc=%d", buff,
strlen(buff));
        return strlen(buff);
}

static ssize_t test_sys_write(struct kobject *kobj, struct kobj_attribute *attr, const
char *buff, size_t count)
{
        if(count > BUFLEN)
                return -EINVAL;

        write_lock(&lock);
        memcpy(test_string, buff, count);
        write_unlock(&lock);
        pr_info("test_dev_file: --test_sys_write()-- test_string <<< %s",
test_string);
        return strlen(test_string);
}

static const struct proc_ops pops =
{
        .proc_read = test_proc_read,
        .proc_write = test_proc_write
};

static struct kobj_attribute string_attribute =
__ATTR(test_string, 0644, test_sys_read, test_sys_write);

static struct attribute *attrs[] =
{
        &string_attribute.attr,
        NULL
};

static struct attribute_group attr_group =
{
        .attrs = attrs
};

int init_module(void) {

        int retval = 0;
        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);

        test = proc_create("test_proc", 0666, NULL, &pops);
        test_kobj = kobject_create_and_add("test_kobj", kernel_kobj);
        if(!test_kobj)
                return -ENOMEM;

        retval = sysfs_create_group(test_kobj, &attr_group);

        if(retval)
                kobject_put(test_kobj);
        return retval;
}

void cleanup_module(void)
{
        unregister_chrdev(major, "test_dev_file");
        proc_remove(test);
        kobject_put(test_kobj);
        //pr_info("test_dev_file: --Cleanup_module()--\n");
        pr_info("test_dev_file: --CleanUp()-- test_string = %s", test_string);
}

MODULE_LICENSE("GPL");
```