```c
/* ANRC RHKI */
/* Lab16: Kernel Devices Lab */
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>

#include <linux/kobject.h>
#include <linux/string.h>
#include <linux/sysfs.h>
#include <linux/fs.h>

#include <asm/current.h>
#include <asm/segment.h>
#include <asm/uaccess.h>

#define DRIVER_AUTHOR "ANRC"
#define DRIVER_DESC   "Lab16: Char Device"

MODULE_LICENSE("GPL");           // Get rid of taint message by declaring code as GPL.

/*  Or with defines, like this: */
MODULE_AUTHOR(DRIVER_AUTHOR);    // Who wrote this module?
MODULE_DESCRIPTION(DRIVER_DESC); // What does this module do?


char my_data[80]="chardev LKM says HI!"; /* our device */

int my_open(struct inode *inode,struct file *filep);
int my_release(struct inode *inode,struct file *filep);
ssize_t my_read(struct file *filep,char *buff,size_t count,loff_t *offp );
ssize_t my_write(struct file *filep,const char *buff,size_t count,loff_t *offp );

struct file_operations my_fops={
        open: my_open,
        read: my_read,
        write: my_write,
        release:my_release,
};

int my_open(struct inode *inode,struct file *filep)
{
        return 0;
}

int my_release(struct inode *inode,struct file *filep)
{
        return 0;
}
ssize_t my_read(struct file *filep,char *buff,size_t count,loff_t *offp )
{
        /* function to copy kernel space buffer to user space*/
        if ( copy_to_user(buff,my_data,strlen(my_data)) != 0 )
                printk( "chardev: Kernel -> userspace copy failed!\n" );
        return strlen(my_data);

}
ssize_t my_write(struct file *filep,const char *buff,size_t count,loff_t *offp )
{
        /* function to copy user space buffer to kernel space*/
        if ( copy_from_user(my_data,buff,count) != 0 )
                printk( "chardev: Userspace -> kernel copy failed!\n" );
        return 0;
}

int init(void)
{
        printk(KERN_INFO "init_module() called\n");
```

```
        if(register_chrdev(222,"my_device",&my_fops)){
                printk("chardev: failed to register");
        }

        return 0;
}

void cleanup(void)
{
   printk(KERN_ALERT "Unloading chardev ...\n");
   unregister_chrdev(222,"my_device");
}

module_init(init);
module_exit(cleanup);
```