《操作系统》课第12次实验报告

学院:	软件学院
姓名:	张怡桢
学号:	2013747
邮箱:	2662765987@qq.com
时间:	12/08/2022

1. 开篇感言

“你长大后想成为什么人？”

“什么意思？长大后我就不能成为我自己了吗？”

-- 《阿甘正传》

2. 实验题目

Linux kernel Module Development

3. 实验要求

To implement a Linux kernel module to read and write one specified file

- 选择一个具体文件(xxxfile)，利用该内核机制，实现对该文件的读操作
    - cat xxxfile
- 进一步实现对该文件的写操作
    - echo "hello 学号姓名日期..." > xxxfile
    - 或
    - echo "hello 学号姓名日期..." >> xxxfile

4. 实验步骤

4.1 Char device based on Linux Kernel Module

实现字符设备的读写其实就是实现驱动中 file_operation 结构体里的 read 和 write 成员.

```c
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/fs.h>
#include <linux/uaccess.h>

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Robert W. Oliver II");
MODULE_DESCRIPTION("A simple example Linux module.");
MODULE_VERSION("0.01");

#define DEVICE_NAME "lkm_example"
#define EXAMPLE_MSG "Hello, 2013747 zhangyizhen!\n"
#define MSG_BUFFER_LEN 128

/* Prototypes for device functions */
static int device_open(struct inode *, struct file *);
static int device_release(struct inode *, struct file *);
static ssize_t device_read(struct file *, char *, size_t, loff_t *);
static ssize_t device_write(struct file *, const char *, size_t, loff_t *);

static int major_num;
static char *name = "lkm_example";
static struct file *filp;
static int device_open_count = 0;
static char msg_buffer[MSG_BUFFER_LEN];

module_param(name, charp, S_IRUGO);

/* This structure points to all of the device functions */
static struct file_operations file_ops = {
    .read = device_read,
    .write = device_write,
    .open = device_open,
    .release = device_release
    };

/* When a process reads from our device, this gets called. */
static ssize_t device_read(struct file *s, char *buffer, size_t len, loff_t *off
    int i = 0;
    int res = 0;
    int j = 0;
    filp->f_pos = 0;
    while (1) {
        i = kernel_read(filp, msg_buffer, 128, &filp->f_pos);
        if (i == 0) {
            break;
        }
        j = 0;
        while (i > 0) {
            i--;
            put_user(msg_buffer[res++], buffer + (j++));
        }
    }
    return res;
}

/* Called when a process tries to write to our device */
static ssize_t device_write(struct file *s, const char *buffer, size_t len,
                            loff_t *offset) {
    bool flag;
    flag=copy_from_user(msg_buffer, buffer, len);
    printk("write to file: %s\n", name);
    printk(KERN_INFO "%s\n", msg_buffer);
    filp->f_pos = file_inode(filp)->i_size;
    printk("file size: %lld\n", filp->f_pos);
    kernel_write(filp, msg_buffer, len, &filp->f_pos);
    return len;
}

/* Called when a process opens our device */
static int device_open(struct inode *inode, struct file *file) {
    /* If device is open, return busy */
    if (device_open_count) {
        return -EBUSY;
    }
    device_open_count++;
    try_module_get(THIS_MODULE);
    return 0;
}

/* Called when a process closes our device */
static int device_release(struct inode *inode, struct file *file) {
    /* Decrement the open counter and usage count. Without this, the module woul
    device_open_count--;
    module_put(THIS_MODULE);
    return 0;
}

static int __init lkm_example_init(void) {
    // 打开文件
```

```c
    filp = filp_open(name, O_RDWR, 0);
    printk("open file: %s", name);
    /* Try to register character device */
    major_num = register_chrdev(0, "lkm_example", &file_ops);
    if (major_num < 0) {
        printk(KERN_ALERT "Could not register device: %d\n", major_num);
        return major_num;
    } else {
        printk(KERN_INFO "lkm_example module loaded with device major number %d\
               major_num);
        return 0;
    }
}
static void __exit lkm_example_exit(void) {
    /* Remember — we have to clean up after ourselves. Unregister the character
    device. */
    unregister_chrdev(major_num, DEVICE_NAME);
    filp_close(filp, NULL);
    printk(KERN_INFO "Goodbye, 2013747 zhangyizhen!\n");
}
/* Register module functions */
module_init(lkm_example_init);
module_exit(lkm_example_exit);

```

4.2 Compile the above Linux kernel module

4.2.1 创建Makefile文件:

```makefile
ModuleName=lkm_example
obj-m +=${ModuleName}.o
all:${ModuleName}.ko
${ModuleName}.ko:${ModuleName}.c
        make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
test:${ModuleName}.ko
        echo make test_ins
        echo make test_mk
        echo make test_test
        echo make test_rm
test_ins:${ModuleName}.ko
        sudo dmesg -C
        sudo insmod ${ModuleName}.ko name=/home/parallels/Documents/lab12/zyz
        sudo dmesg
test_mk:${ModuleName}.ko
        sudo mknod /dev/${ModuleName} c 509 0
```

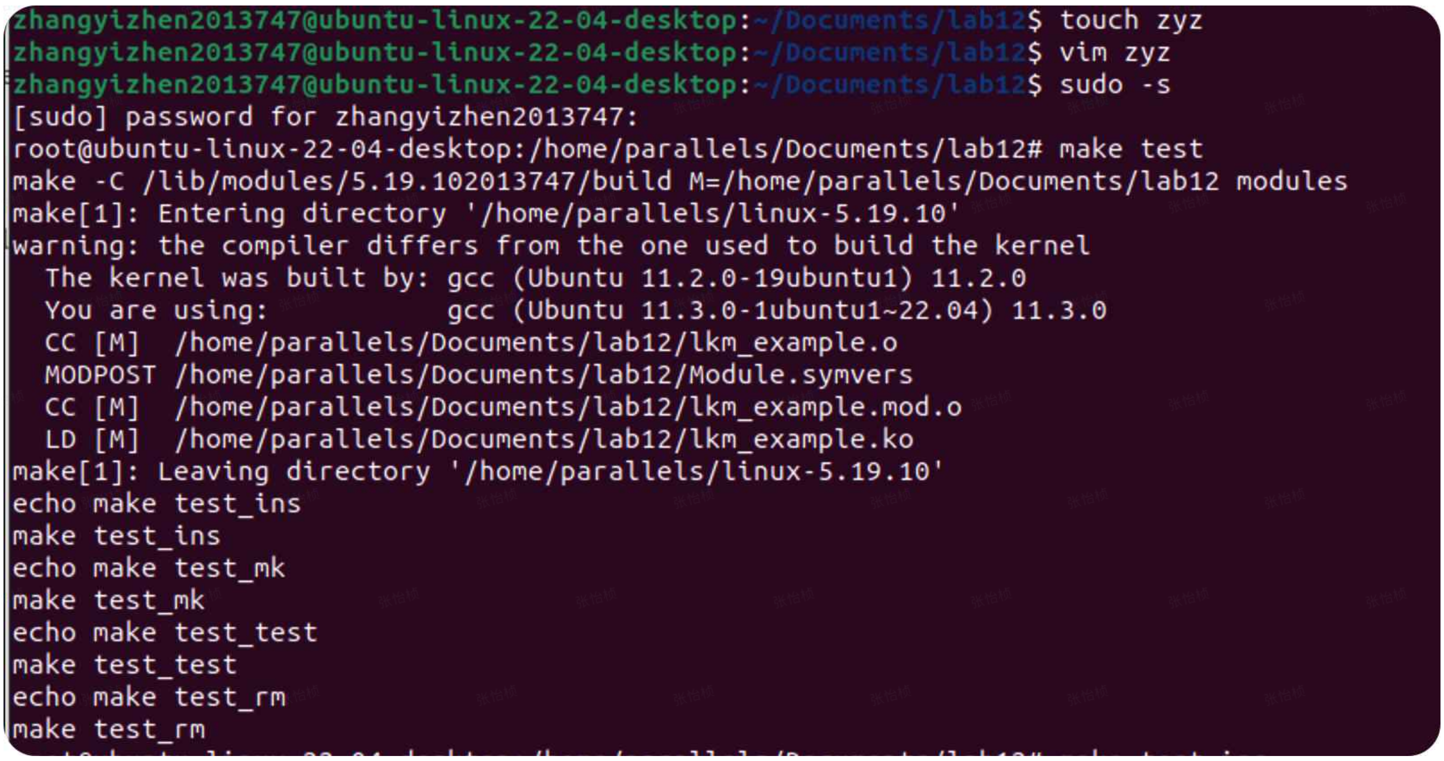
```
test_read:${ModuleName}.ko
        sudo dmesg -C
        cat /dev/${ModuleName}
        sudo dmesg
test_write:${ModuleName}.ko
        echo "hello 2013747 zhangyizhen 2022/12/8" >> /dev/${ModuleName}
test_rm:${ModuleName}.ko
        sudo rmmod ${ModuleName}.ko
        sudo dmesg
.PHONY:clean
clean:
        make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
        sudo rm /dev/lkm_example

```

4.2.2 编译

```
make test
```

```
test:${ModuleName}.ko
        echo make test_ins
        echo make test_mk
        echo make test_test
        echo make test_rm
```



```
zhangyizhen2013747@ubuntu-linux-22-04-desktop:~/Documents/lab12$ touch zyz
zhangyizhen2013747@ubuntu-linux-22-04-desktop:~/Documents/lab12$ vim zyz
zhangyizhen2013747@ubuntu-linux-22-04-desktop:~/Documents/lab12$ sudo -s
[sudo] password for zhangyizhen2013747:
root@ubuntu-linux-22-04-desktop:/home/parallels/Documents/lab12# make test
make -C /lib/modules/5.19.102013747/build M=/home/parallels/Documents/lab12 modules
make[1]: Entering directory '/home/parallels/linux-5.19.10'
warning: the compiler differs from the one used to build the kernel
  The kernel was built by: gcc (Ubuntu 11.2.0-19ubuntu1) 11.2.0
  You are using:           gcc (Ubuntu 11.3.0-1ubuntu1~22.04) 11.3.0
  CC [M]  /home/parallels/Documents/lab12/lkm_example.o
  MODPOST /home/parallels/Documents/lab12/Module.symvers
  CC [M]  /home/parallels/Documents/lab12/lkm_example.mod.o
  LD [M]  /home/parallels/Documents/lab12/lkm_example.ko
make[1]: Leaving directory '/home/parallels/linux-5.19.10'
echo make test_ins
make test_ins
echo make test_mk
make test_mk
echo make test_test
make test_test
echo make test_rm
make test_rm
```

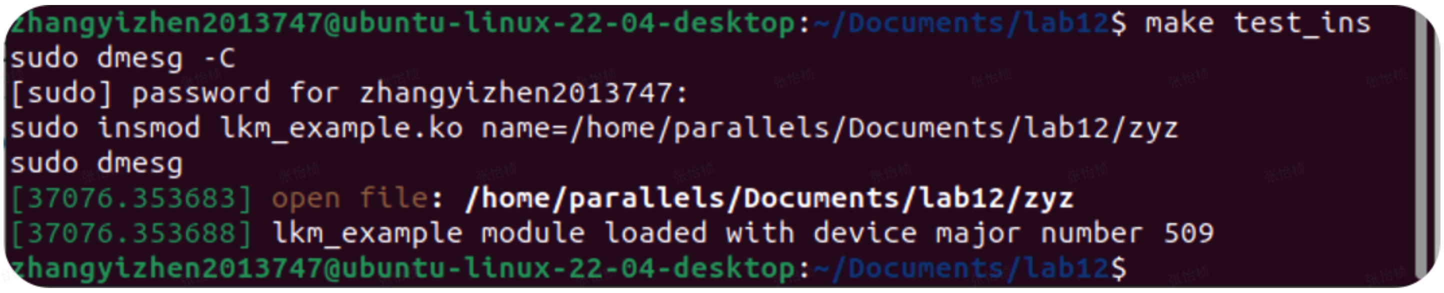
4.2.3 执行

```
make test_ins
make test_mk
make test_read
make test_write
make test_rm
```

1. make test_ins ----内核自动分配的设备主号码

```
test_ins:${ModuleName}.ko
	sudo dmesg -C
	sudo insmod ${ModuleName}.ko name=/home/parallels/Documents/lab12/zyz
	sudo dmesg
```

具体操作的文件为/home/parallels/Documents/lab12/zyz



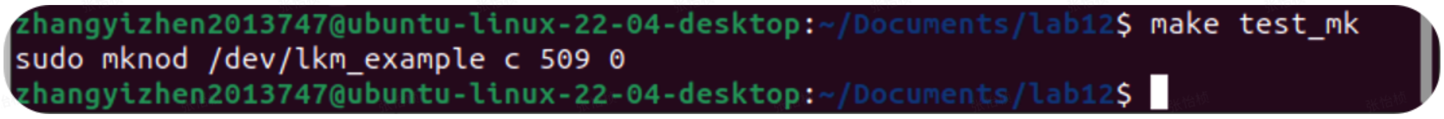
```
zhangyizhen2013747@ubuntu-linux-22-04-desktop:~/Documents/lab12$ make test_ins
sudo dmesg -C
[sudo] password for zhangyizhen2013747:
sudo insmod lkm_example.ko name=/home/parallels/Documents/lab12/zyz
sudo dmesg
[37076.353683] open file: /home/parallels/Documents/lab12/zyz
[37076.353688] lkm_example module loaded with device major number 509
zhangyizhen2013747@ubuntu-linux-22-04-desktop:~/Documents/lab12$
```

由内核自动分配的设备主号码的输出,可以看到是509号。

2. make test_mk ----创建一个字符设备文件

将 test_mk 中的 MajorNum 替换为运行 make test_ins 或 dmesg 后得到的值（在本实验中为509）。Mknod 命令中的“c”告诉 mknod 我们需要创建一个字符设备文件。

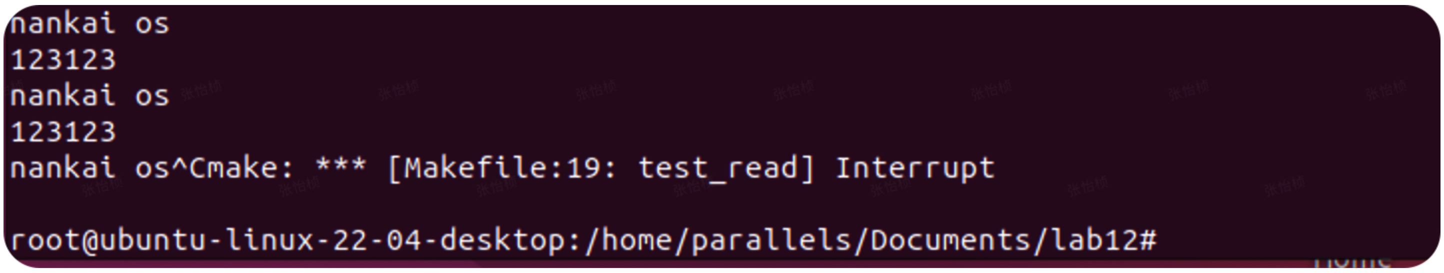
```
test_mk:${ModuleName}.ko
        sudo mknod /dev/${ModuleName} c 509 0
```



```
zhangyizhen2013747@ubuntu-linux-22-04-desktop:~/Documents/lab12$ make test_mk
sudo mknod /dev/lkm_example c 509 0
zhangyizhen2013747@ubuntu-linux-22-04-desktop:~/Documents/lab12$ 
```

3. make test_read ----读取zyz文件的具体内容到终端

```
test_read:${ModuleName}.ko
        sudo dmesg -C
        cat /dev/${ModuleName}
        sudo dmesg
```

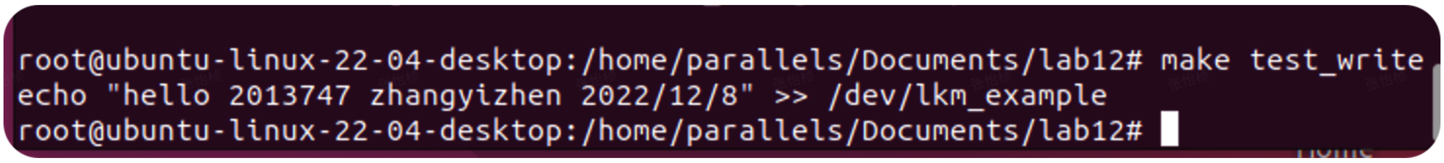


```
nankai os
123123
nankai os
123123
nankai os^Cmake: *** [Makefile:19: test_read] Interrupt

root@ubuntu-linux-22-04-desktop:/home/parallels/Documents/lab12#
```

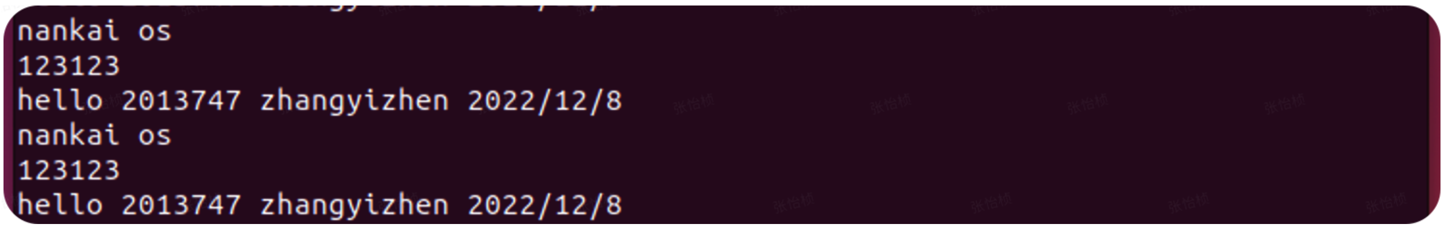
4. make test_write ----写入文件设置的内容

```
test_write:${ModuleName}.ko
        echo "hello 2013747 zhangyizhen 2022/12/8" >> /dev/${ModuleName}
```



```
root@ubuntu-linux-22-04-desktop:/home/parallels/Documents/lab12# make test_write
echo "hello 2013747 zhangyizhen 2022/12/8" >> /dev/lkm_example
root@ubuntu-linux-22-04-desktop:/home/parallels/Documents/lab12# 
```

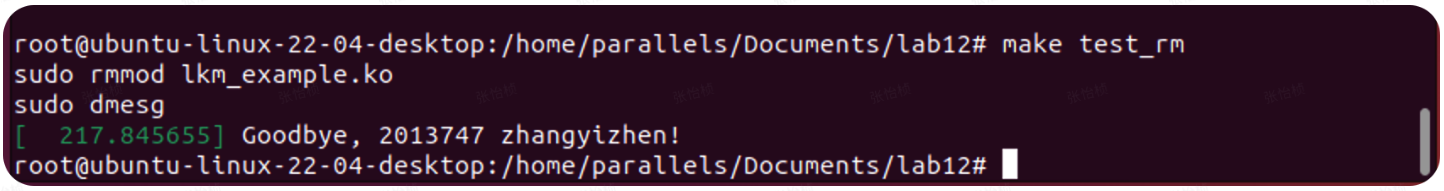
使用make test_read 读取文件内容，可以看到写入成功:



```
nankai os
123123
hello 2013747 zhangyizhen 2022/12/8
nankai os
123123
hello 2013747 zhangyizhen 2022/12/8
```

5. make test_rm

```
test_rm:${ModuleName}.ko
	sudo rmmod ${ModuleName}.ko
	sudo dmesg
```

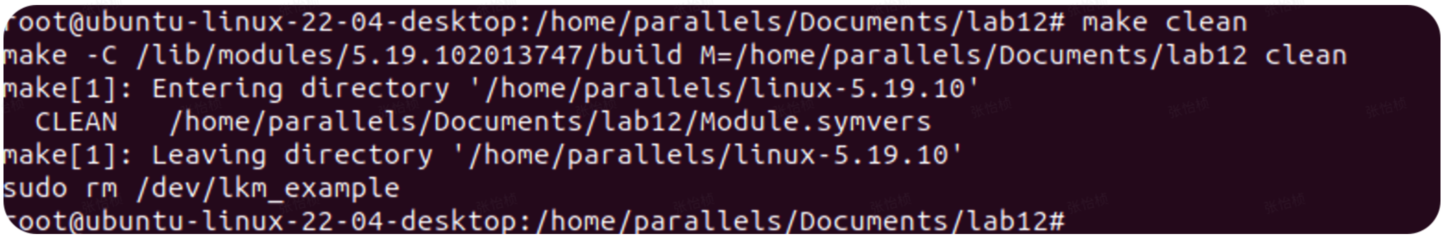


```
root@ubuntu-linux-22-04-desktop:/home/parallels/Documents/lab12# make test_rm
sudo rmmod lkm_example.ko
sudo dmesg
[  217.845655] Goodbye, 2013747 zhangyizhen!
root@ubuntu-linux-22-04-desktop:/home/parallels/Documents/lab12# 
```

4.2.4 清除

make clean

```
.PHONY:clean
clean:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
	sudo rm /dev/lkm_example
```



```
oot@ubuntu-linux-22-04-desktop:/home/parallels/Documents/lab12# make clean
ake -C /lib/modules/5.19.102013747/build M=/home/parallels/Documents/lab12 clean
ake[1]: Entering directory '/home/parallels/linux-5.19.10'
  CLEAN   /home/parallels/Documents/lab12/Module.symvers
ake[1]: Leaving directory '/home/parallels/linux-5.19.10'
sudo rm /dev/lkm_example
oot@ubuntu-linux-22-04-desktop:/home/parallels/Documents/lab12#
```

5. 资料

老师的github实验文档

6. 附件

代码在压缩包中