实验四：第8章设备管理，第9章 文件管理[仅供4-5-6班参考]

- 一、实验目的
  - （1）理解设备是文件的概念。
  - （2）掌握Linux模块、驱动的概念和编程流程
  - （3）Windows /Linux下掌握文件读写基本操作
  - （4） Windows /Linux体会缓冲作用-提前读/延迟写
- 二、实验内容
  - （1）编写一个Linux内核模块，并完成安装/卸载等操作。
  - （2）编写Linux驱动程序并编程应用程序测试。功能：write几个整数进去，read出其和或差或最大值。
  - （3）编写Linux驱动程序并编程应用程序测试。功能：有序读写内核缓冲区，返回实际读写字节数。
  - （4）在Win/Linux下采用缓冲或非缓冲方式读写大文件，体会效率差异。
- 三、实验要求
  - 寝室提前做完，老师机房检查和答疑。1,2必做，3,4任选其一。

实验四：第8章设备管理，第9章 文件管理[仅供4-5-6班参考]

- 四、实验指南
  - 1）编写一个Linux内核模块，并完成安装/卸载等操作。
    - ◆提示1：安装时和退出时在内核缓冲区显示不同的字符串。
    - ◆提示2：相关函数：module_init( )、 module_exit( )
    - ◆提示3： MODULE_LICENSE( )、 MODULE_AUTHOR ( )等可选
    - ◆提示4：安装命令：insmod XXXX.ko
    - ◆提示5：扩展：编写带参数的模块程序

```
int mytest = 100;
module_param(mytest, int, 0644);
```

实验四：第8章设备管理，第9章 文件管理[仅供4-5-6班参考]

- 四、实验指南
  - 2）编写Linux驱动程序并编程应用程序测试。
    - ◆提示1：参考任务1
    - ◆提示2：至少实现xx_open,xx_write,xx_read等函数
    - ◆提示3：功能：
      - □xx_write( )写进去2个整数
      - □xx_read( )读回结果（和或差或最大值）
    - ◆提示4: [可选的设备注册方式，其余方式参考baidu]

```
struct miscdevice    mydemodrv_misc_device ;
ret = misc_register( &mydemodrv_misc_device );
```

实验四：第8章设备管理，第9章 文件管理[仅供4-5-6班参考]

```
static ssize_t
demodrv_read(struct file *file, char __user *buf,
             size_t lbuf, loff_t *ppos)
{
    printk("%s enter\n", __func__);
    return 0;
}

static ssize_t
demodrv_write(struct file *file, const char __user *buf,
              size_t count, loff_t *f_pos)
{
    printk("%s enter\n", __func__);
    return 0;

}
```

实验四：第8章设备管理，第9章 文件管理[仅供4-5-6班参考]

- 四、实验指南
  - 3）编写Linux驱动程序并编程应用程序测试。
    - ◆提示1：参考任务1
    - ◆提示2：至少实现xx_open,xx_write,xx_read等函数
    - ◆提示3：功能：
      - 内核分配一定长度的缓冲区，比如64字节。

```
static char *device_buffer;
#define MAX_DEVICE_BUFFER_SIZE 64
device_buffer = kmalloc(MAX_DEVICE_BUFFER_SIZE, GFP_KERNEL);
```

      - xx_write()写进去若干字符，注意维护写入位置。下次继续写的话，接着该位置往后写，直到缓冲区末尾。要返回实际写入字数。
      - xx_read()读出若干字符串，注意维护读出位置。下次继续读的话，接着该位置往后读，直到缓冲区末尾。要返回实际读回字数。
      - 扩展：
        - ▲缓冲区设置为循环缓冲区？
        - ▲如何避免写覆盖，避免读重复？

实验四：第8章设备管理，第9章 文件管理[仅供4-5-6班参考]

- 四、实验指南
  - （4）在Win/Linux下采用缓冲或非缓冲方式读写大文件，体会效率差异。
    - ◆提示1：下面两段代码分别是非缓冲方式和缓冲方式。

```
19   HandleSrcFile=CreateFile(SourceFile,
20       GENERIC_READ,    0,
21       NULL,   OPEN_ALWAYS,
22       FILE_FLAG_NO_BUFFERING,   NULL);
23
24   HandleDstFile=CreateFile(DestinationFile,
25       GENERIC_WRITE,   NULL,   NULL,
26       OPEN_ALWAYS,   NULL,   NULL);
```

```
57     HandleSrcFile = CreateFile( SourceFile,
58         GENERIC_READ|GENERIC_WRITE, 0,NULL,
59         CREATE_ALWAYS, FILE_FLAG_SEQUENTIAL_SCAN, NULL);
60
61     HandleDstFile = CreateFile( DestinationFile,
62         GENERIC_WRITE, NULL, NULL, CREATE_ALWAYS,
63         FILE_FLAG_SEQUENTIAL_SCAN, NULL);
```