一、 应用层和驱动层的数据交换

1、编写驱动层 write 函数

函数 unsigned long copy_from_user(void *to,const void __user *from,unsigned long n)

//读取用户空间传递来的数据;

参数：to： 存储用户空间的数据的内存块;

from： 用户空间的数据;

n： 大小;

返回值： 0 表示成功 非 0 代表失败

2、编写驱动层 read 函数

函数 unsigned long copy_to_user(void __user *to,const void *from,unsigned long n)

//将数据返回给应用层;

参数：to： 拷贝到应用层内存块;

form： 要拷贝给应用层的内存块;

n： 大小;

返回值： 0 表示成功 非 0 代表失败

二、 read 函数

应用层：ssize_t read(int fd, void *buf, size_t count) //对设备节点进行读写;

参数：fd:文件； buf：读取内容存储位置； count：读写的个数;

驱动层：ssize_t read(struct file *node,char __user*buff,size_t size,loff_t *offset)

//去让应用层可以读取到想要的数据;

参数：node：节点； buff：返回给应用层的数据； size：大小； offset：偏移量;

驱动层 read 函数获取应用层想要读取的数据，并将数据返回给应用层(copy_to_user)。

三、 write 函数

应用层： ssize_t write(int fd, const void *buf, size_t count);

驱动层： ssize_t write(struct file *file,const char __user *buff,size_t size,loff_t *offset)

驱动层接收应用层的数据并进行下一步动作。

应用层： write 一个 buf （数据传输到驱动层 buff 里面）， 驱动层接收到数据并进行下一步动作。

四、 lseek 函数

定位当前文件内部的读写位置 并返回给用户;

应用层：off_t lseek(int fildes, off_t offset, int whence);

参数:fildes:代表文件； offset:偏移量； whence:参考位置(SEEK_SET、SEEK_CUR、 SEEK_END);

驱动层： loff_t llseek(struct file *fp, loff_t offset, int whence);

参数：fp： 文件； offset：偏移量； whence： 参考位置;

fp->f_ops： 保存当前文件的读写位置

例:

```
#define size 4
int myllseek(struct file *fp, loff_t offset, int whence)
{
    int offset1=fp->f_ops;
```

```
    if(whence == SEEK_SET)
    {
        if(offset<0)
            return -1;
        if(offset+offset1>4)
            return -1;
        offset1=offset;
    }
    if(whence == SEEK_END)
    {
        if(offset>0)
            return -1;
        if(-offset>4)
            return -1;
        offset1 = 4+offset;
    }
    if(whence == SEEK_CUR)
    {
        if(-offset>offset1)
            return -1;
        if(offset+offset1>4)
            return -1;
        offset1+=offset;
    }
    fp->f_ops=offset1;
    return offset1;
}
```

应用层： lseek(fd,1,SEEK_SET); //参数 2 参数 3 传递给驱动层的参数 2 参数 3；

驱动层： 改变 fp->f_ops //保存当前文件的偏移量；

注：没有考虑文件大小， 文件大小自己去定义。

五、 ioctl 函数

应用层： int ioctl(int d, int request, unsigned long data);

驱动层： long unlocked_ioctl(struct file *fp, unsigned int cmd, unsigned long data)

做控制命令 request ------ cmd

data ------ data

例：应用层： ioctl(fd,1); ---- 灯亮

ioctl(fd,0); ---- 灯灭

驱动层函数接口的参数：

1、 open 和 close 里面的 struct inode *node 可以提取主次设备号；

2、 read 和 write 里面的 char __user *buff 是底层和用户之间交换数据的缓冲区；

3、 llseek：参数 fp->f_ops 保存当前文件读写位置 , loff_t offset 和 int whence 是应用层传递过来的；

4、unlocked_ioctl cmd 和 data 也是应用层传递过来的参数；