《操作系统原理》实验报告

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 姓名 | 李欣宇 | 学号 | U201911658 | 专业班级 | 信安1901 | 时间 | 2022.1.5 |

**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

**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

**三、实验过程**

**3.1 编写linux内核模块**

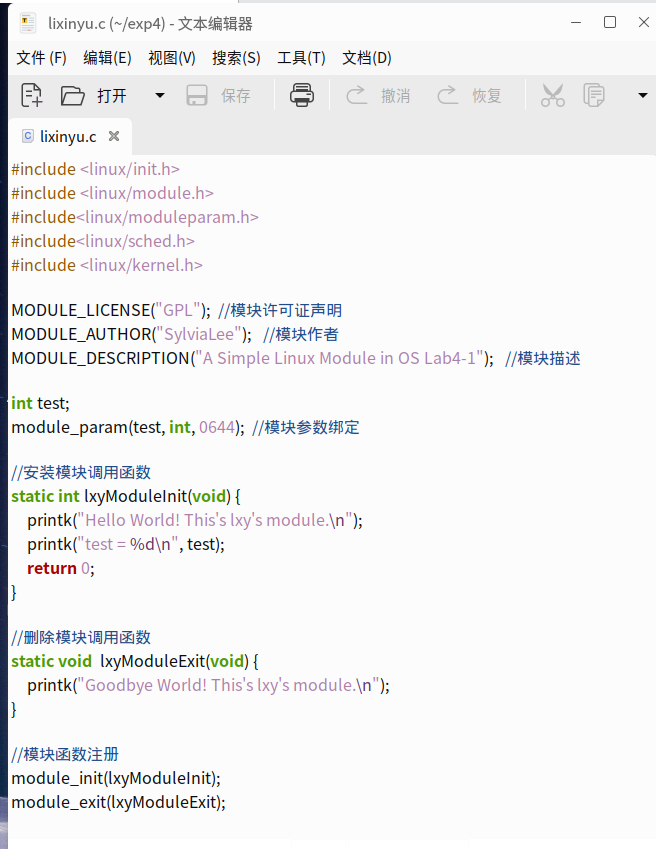
（1）lixinyu.c 内核模块编写

该部分实验编写了一个简单的模块 lixinyu。

首先编写模块的 C 文件 lixinyu.c。 MODULE\_LICENSE 声明模块许可证，用于模块的验证，MODULE\_AUTHOR、 MODULE\_DESCRIPTION 声明了模块的作者和描述。

函数 lxyModuleInit()为安装模块时调用的函数，lxyModuleExit()为删除模块时调用的函数，两者都使用了 printk()函数向内核缓冲区打印一段字符。此外，全局变量 test 使用 module\_param()函数进行参数绑定，作为 lxyModuleInit()的参数，可在安装模块时进行参数传递。

最后使用module\_init()函数和 module\_exit()函数对上述两个函数进行注册。

****

（2）Makefile编写

****

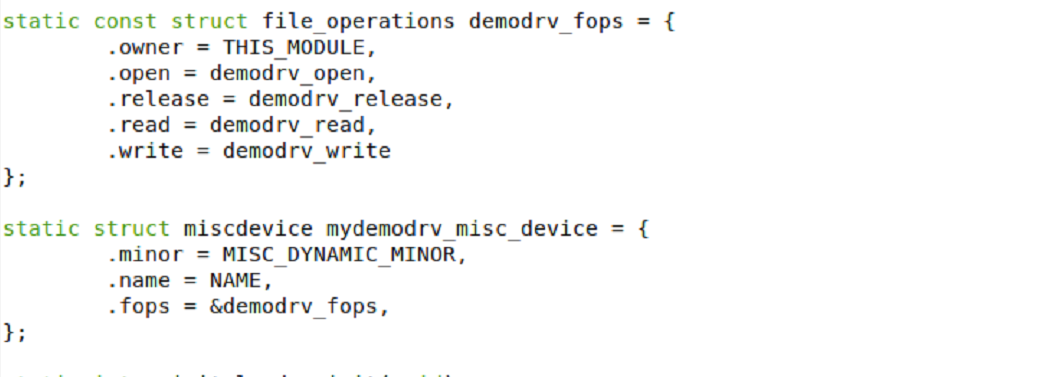
**3.2 编写Linux中求和驱动程序1**

**（1）驱动程序的编写**

一部分代码与上述模块文件一致，不再赘述。

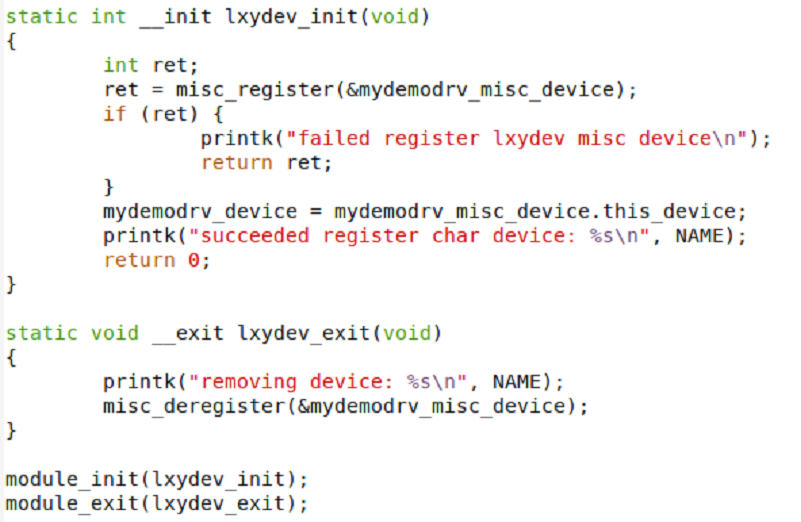
代码中定义了一个 file\_operations 类型的结构体 demodrv\_fops，该函数的成员使用键值对进行赋值，表明了该驱动定义的函数与 Linux 的标准文件读取函数的对应关系，此处仅列出了 read 、 write 、open、release、owner与自定义函数 demodrv\_read 和 demodrv\_write等的映射关系。

接下来的一个miscdevice类型的结构体，标识主设备名、设备名和定义的驱动函数映射的结构体。



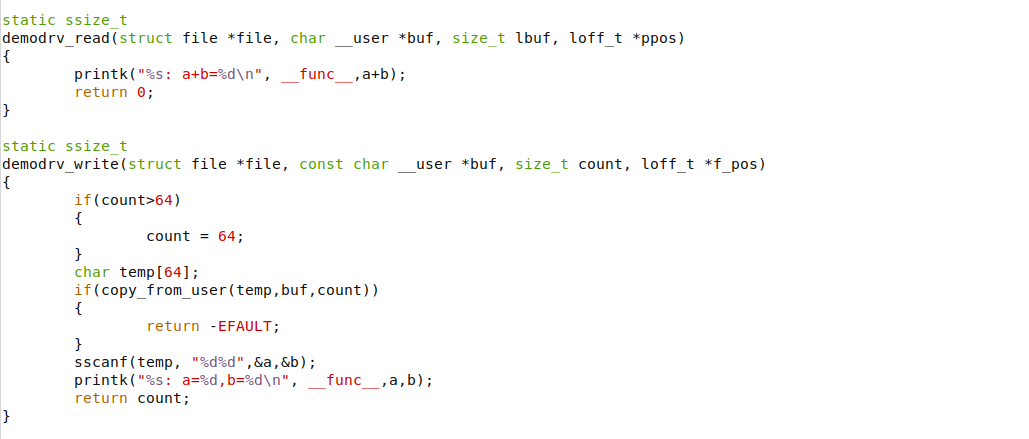
与任务一中的模块相同，驱动程序也需要安装和删除驱动的函数，lxydev\_init()和lxydev\_exit()，并使用 module\_init()和 module\_exit()进行注册和卸载。

在安装驱动调用的函数 lxydev\_init()中，使用了 misc\_register()函数来注册一个字符型的设备，需要传入参数主设备号、设备名 、 以及定义的驱动函数映射的结构体，这些都包括在了mydemodrv\_misc\_device结构体中。在删除驱动调用的函数 lxydev\_exit()中，使用了 misc\_deregister()对注册的设备进行删除。



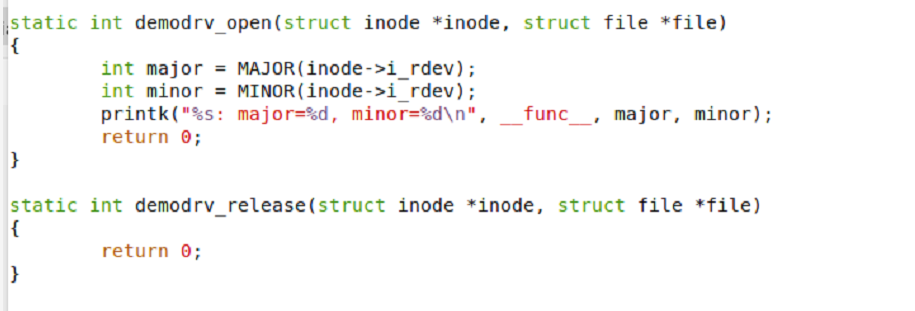
对于demodrv\_read()读功能函数直接加为和。

对于demodrv\_write()写功能，count记录了从用户缓冲区需要写入驱动缓冲区的数据字节数，当count大于驱动缓冲区大小（64byte）时要更新count，然后使用 Linux 提供的函数 copy\_from\_user()将用户空间的数据复制到内核空间的驱动缓冲区，最后返回实际写入数据字节数。

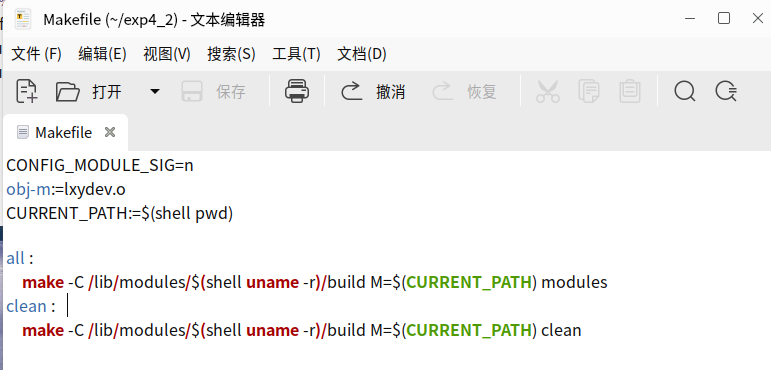


对于demdrv\_open()打开设备函数，第一个参数是设备文件名，第二个参数用来指定打开文件的属性，使用MAJOR和MINOR获取主设备号，次设备号。

对于demodrv\_release()关闭设备函数，直接return 0



（2）Makefile编写

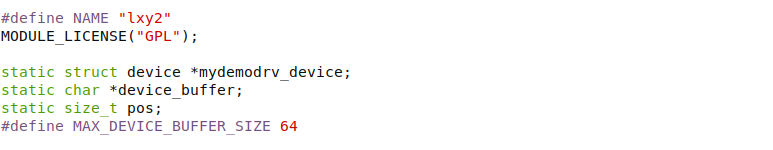


**3.3 编写Linux中求和驱动程序2**

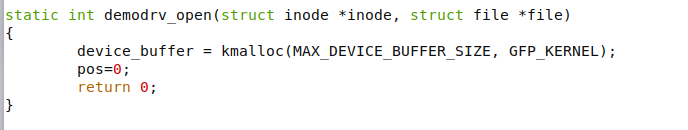
（1）编写驱动程序

大部分与任务二编写的相同，仅把不同之处进行说明

驱动缓冲区最大设置为64byte



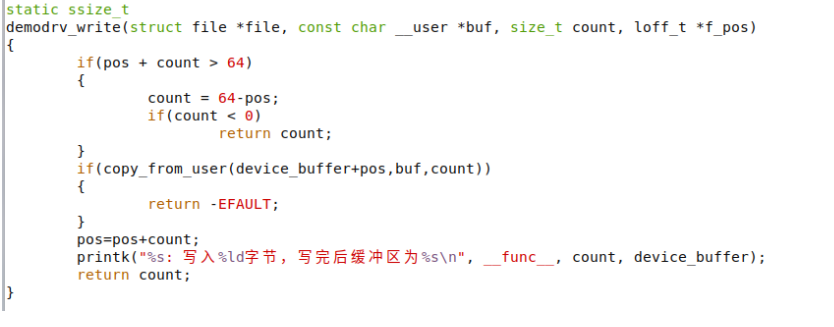
demodrv\_open函数中设置驱动缓冲区，并且初始时驱动缓冲区为空，即pos为0



demodrv\_read()读函数中，如果pos小于要读取数据的长度，修改lbuf，最长仅可读缓冲区内小于等于pos的数据，使用copy\_to\_user()函数将位于驱动缓冲区的整数读出到用户空间，并修改pos大小，返回实际读到的数据长度。



demodrv\_write()写函数，要先验证读入这些数据是否会溢出驱动缓冲区，如果会则修改要读入数据的长度，使用copy\_from\_user()函数从用户空间中把数据写入驱动缓冲区中，pos进行相应修改，返回实际写入的数据长度。



**四、实验结果**

**4.1 编写linux内核模块**

使用sudo make进行模块编译

dmesg -c清空内核缓冲区

sudo insmod lixinyu.ko test=11 安装模块并传递参数test为11

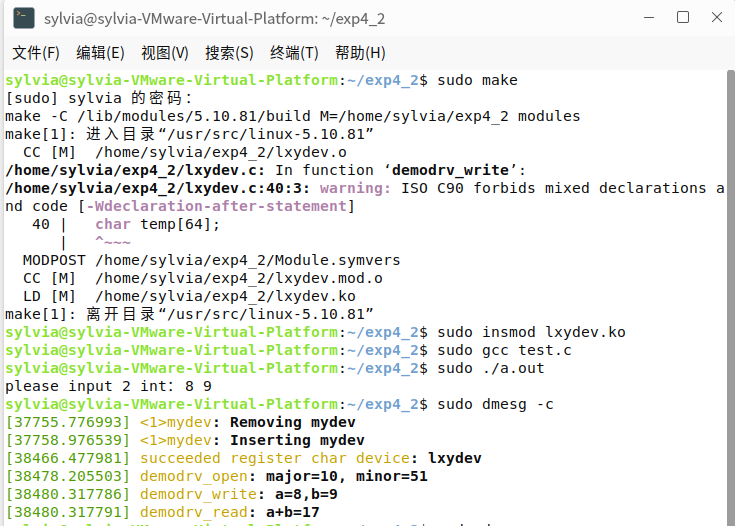
dmesg查看内核缓冲区可以看到输出信息，并且传递的参数也是正确的

sudo rmmod lixinyu.ko 卸载模块

dmesg 查看内核缓冲区，看到了卸载的提示信息

****

**4.2 编写Linux中求和驱动程序1**

****

**4.3 编写Linux中驱动程序2**

****



**五、实验错误排查和解决方法**

**5.1 编写linux内核模块**

1.问题：在Makefile编写时，使用$PWD 在make时路径会显示空，然后默认重新编译所有内核模块，时间非常长

解决方法：Makefile中使用绝对路径，正确情况下一秒出结果，后来查阅到也可以使用$ shell pwd

**5.2 编写Linux中求和驱动程序1**

1.问题：安装好编写的驱动后并不能打开驱动对应的设备文件。

解决方法：在安装模块后 Linux 并不会创建对应的设备文件，需要我们自己创建，使用命令“sudo mknod /dev/<设备名> <设备类型> <主设备号> <次设备号>”进行设备文件的创 建，之后就可以正常读取设备文件了

2.问题：安装ko文件时报错[module](https://so.csdn.net/so/search?q=module" \t "https://blog.csdn.net/zhlw_199008/article/details/_blank) verification failed : signature and/or required key missing - tainting kernel

解决方法：在Makafile文件前面加上CONFIG\_MODULE\_SIG=n

3.问题：mknod 时提示文件已存在，但是无法使用

解决方法：问题根源没有找到，模块卸载还是在，最终直接sudo rm /dev/lixinyu 然后重新make

**5.3 编写Linux中驱动程序2**

1.问题：if(pos - lbuf < 0) {}有问题

解决方法：使用if(pos < lbuf)，这里的原因没有找到，但是猜测与整数类型有关系，可能是不能与0比较。

**六、实验参考资料和网址**

（1）教学课件

（2）https://blog.csdn.net/zhlw\_199008/article/details/81386875

（3）https://www.cnblogs.com/deepsprings/p/4075181.html

（4）https://blog.csdn.net/SoaringLee\_fighting/article/details/89968526

（5）https://blog.csdn.net/iamjjh/article/details/91664251

（6）https://blog.csdn.net/xiao\_xiao\_lan/article/details/104945163

（7）https://blog.csdn.net/zhoulaowu/article/details/14005679

（8）https://xknote.com/blog/130126.html