## 1. 实验目的：

通过编写字符设备驱动，理解linux下的设备驱动程序，学习相关设备驱动知识，加强对内核编程的理解。

## 2. 实验目标：

编写字符设备驱动，实现open、write、read、icotl、release操作，并与前面的内核模块编程相结合，向系统中注册一个字符设备，并在用户态对驱动进行测试

## 3. 实验原理：

### 3.1设备驱动

设备驱动程序在内核中的角色：他们是一个个独立的“黑盒子”，使某个特定的硬件响应一个定义良好的内部编程接口，这些接口完全隐藏了设备的工作细节。（驱动程序除了对外提供特定的接口外，任何实现细节对应用程序都是不可见的。）用户的操作通过一组标准化的调用执行，而这些调用独立于特定的驱动程序。驱动程序任务是把这些标准化调用映射到实际硬件的设备特有操作。

驱动程序就是应用程序与实际硬件之间的一个软件层，相同的硬件，不同的驱动程序可能提供不同的功能。实际的驱动程序设计要在许多要考虑的因素之间做出平衡。总的来说，驱动程序设计主要还是综合考虑下面三个方面的因素：提供给用户尽量多的选项、编写驱动程序要占用的时间以及尽量保持程序简单而不至于错误丛生。

其中linux的设备驱动程序的分类为：字符设备，块设备，网络设备。

### 3.2 字符设备

字符设备是能够像字节流(类似文件)一样被访问的设备，有字符设备驱动程序来实现这种特性。字符设备驱动程序通常至少要实现open、close、read、write系统调用。字符设备可以通过文件系统节点来访问，这些设备文件和普通文件之间的唯一差别在于对普通文件的访问可以前后移动访问位置，而大多数字符设备是一个只能顺序访问的数据通道。一个字符设备是一种字节流设备，对设备的存取只能按顺序按字节的存取而不能随机访问，字符设备没有请求缓冲区，所有的访问请求都是按顺序执行的。但事实上现在一些高级字符设备也可以从指定位置一次读取一块数据。

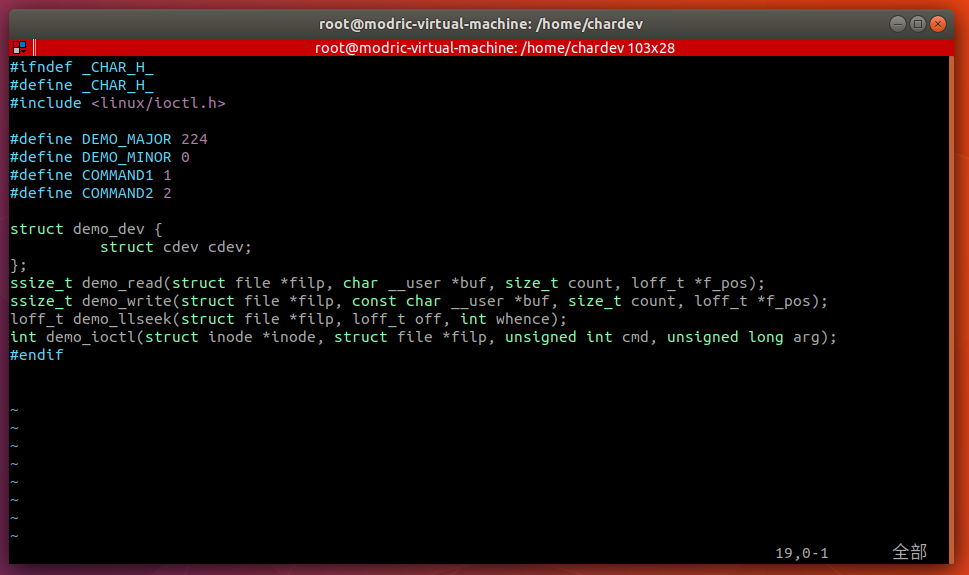
## 4. 实验方案：

### 4.1 编写字符驱动程序

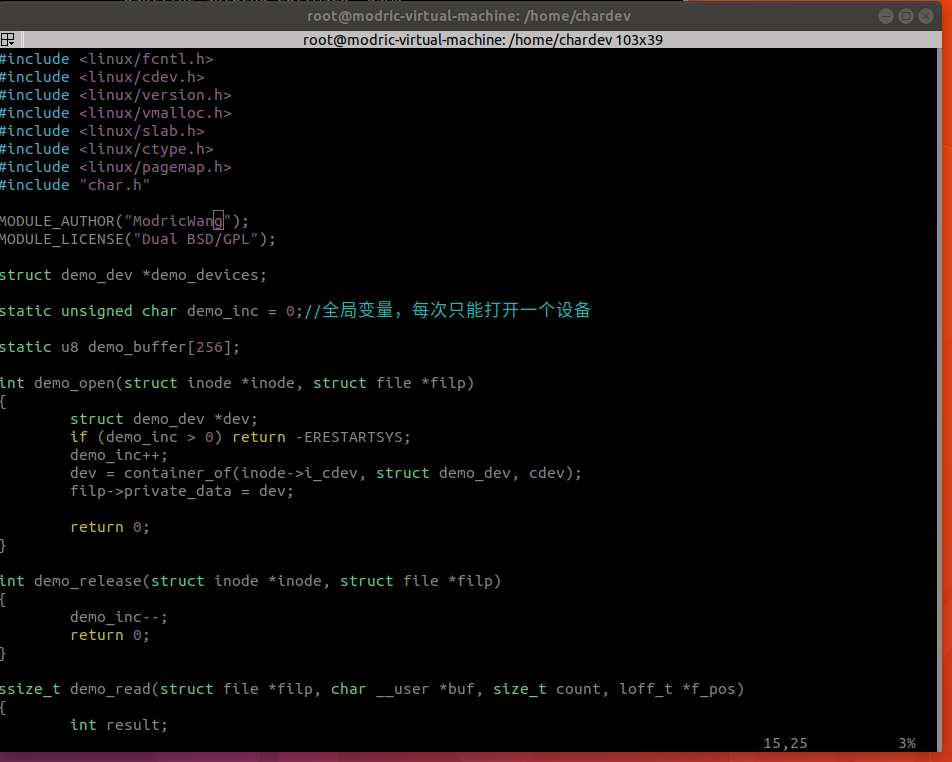
#### 步骤1：使用cd&&mkdir命令创建程序所在目录



#### 步骤2：进入该目录创建字符设备驱动程序头文件char.h, 编写该头文件



#### 步骤3：在同目录下编写源文件



#### 步骤4：编写makefile文件进行编译, 编译字符驱动程序并加载

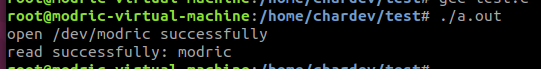


#### 步骤5：创建设备节点

命令：# mknod /dev/modric c 224 0

此处的节点设备号要与驱动程序中的注册的设备号相同

#### 步骤6：创建测试代码，编译测试代码并运行



## 5. 实验结论：

经过本次实验，对linux操作系统的内核模块有了新的理解与认识。在学习字符设备驱动编程中，对上次实验做了巩固。对用户态和内核态的关系和联系有了更深的了解。对linux下的驱动程序有了直观的认识。