**实现在/proc目录下添加文件  
及中断的简单应用实验**

1. **实验目的**
2. 理解Linux中虚拟文件系统的内容
3. 学习编写内核模块的方法

3、在虚拟文件系统/proc中实现文件操作算法

4、了解文件超级块的结构

5、了解linux定时器机制

1. **实验环境**
2. macOS Mojave 10.14.1 (x86\_64-apple-darwin18.2.0)  
    Apple LLVM version 10.0.1 (clang-1001.0.46.4)
3. Ubuntu 18.04 (Linux 5.4.0-51-generic) x64  
    gcc version 7.5.0 (Ubuntu 7.5.0-3ubuntu1~18.04)   
    GNU Make 4.1
4. **实验内容**
5. 编写一个内核模块，在/proc文件系统中增加一个目录hello，并在这个目 录中增加一个文件world，文件的内容为hello world。
6. 编写内核模块，打印super\_blocks结构中一些域的值
7. 编写内核模块，计算两次中断的时间间隔
8. 创建和使用进程延时，说明定时器在内核中实际使用
9. **实验原理 实验中用到的系统调用函数（包括实验原理中介绍的和自己采用的），实验步骤，**

1、/proc 文件系统简介

/proc 文件系统是Linux上的一种虚拟文件系统，存储的是当前内核运行状态的一系列特殊文件，用户可以通过这些文件查看有关系统硬件及当前正在运行进程的信息，甚至可以更改其中某些文件来改变内核的运行状态。最初开发 /proc 文件系统是为了提供有关系统中进程的信息。但是由于这个文件系统非常有用，因此内核中的很多元素也开始使用它来报告信息或启用动态运行时配置。图 1 是对 /proc 中部分元素进行一次交互查询的结果。它显示的是 /proc 文件系统的根目录中的内容。注意，在左边是一系列数字编号的文件。每个实际上都是一个目录，表示系统中的一个进程。由于在 GNU/Linux 中创建的第一个进程是 init 进程，因此它的 process-id 为 1。然后对这个目录执行一个 ls 命令，这会显示很多文件。每个文件都提供了有关这个特殊进程的详细信息。/proc 中另外一些有趣的文件有：cpuinfo，它标识了处理器的类型和速度；pci，显示在 PCI 总线上找到的设备；modules，标识了当前加载到内核中的模块。

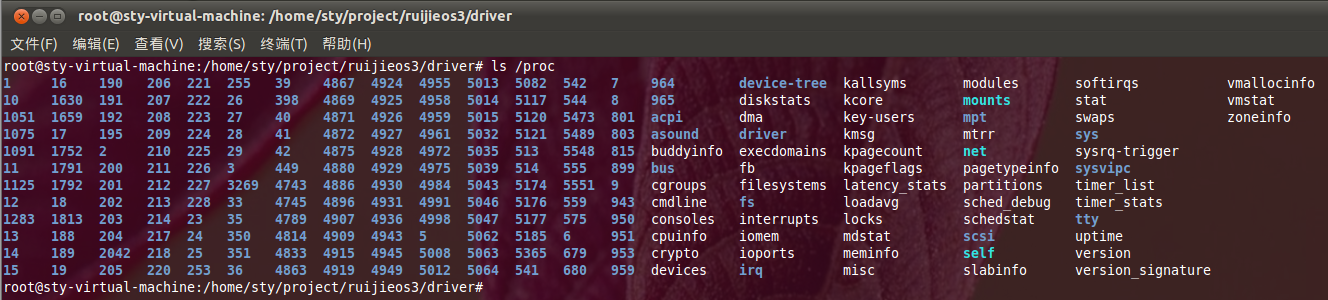
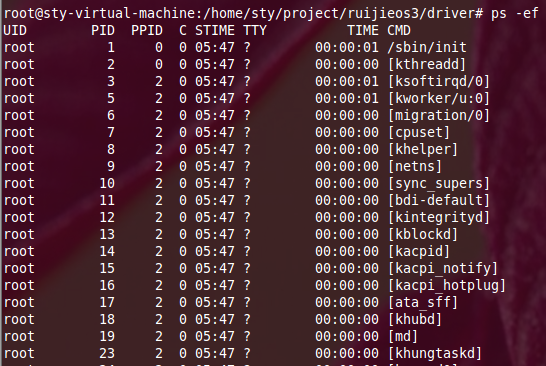
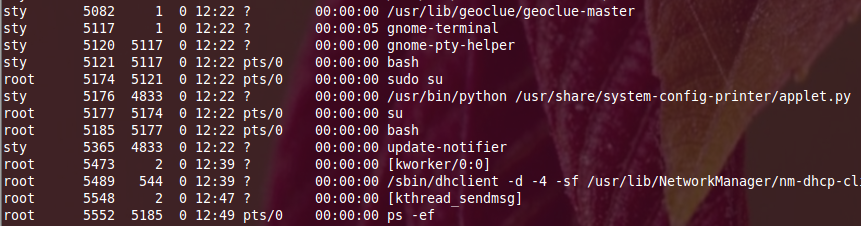


图1





这些文件的解释和意义如下：

cmdline：系统启动时输入给内核命令行参数

cpuinfo：CPU的硬件信息 (型号, 家族, 缓存大小等)

devices：主设备号及设备组的列表，当前加载的各种设备（块设备/字符设备）

dma：使用的DMA通道

filesystems：当前内核支持的文件系统，当没有给 mount(1) 指明哪个文件系统的时候， mount(1) 就依靠该文件遍历不同的文件系统

interrupts ：中断的使用及触发次数，调试中断时很有用

ioports I/O：当前在用的已注册 I/O 端口范围

kcore：该伪文件以 core 文件格式给出了系统的物理内存映象(比较有用)，可以用 GDB 查探当前内核的任意数据结构。该文件的总长度是物理内存 (RAM) 的大小再加上 4KB

kmsg：可以用该文件取代系统调用 syslog(2) 来记录内核日志信息，对应dmesg命令

kallsym：内核符号表，该文件保存了内核输出的符号定义, modules(X)使用该文件动态地连接和捆绑可装载的模块

loadavg：负载均衡，平均负载数给出了在过去的 1、 5,、15 分钟里在运行队列里的任务数、总作业数以及正在运行的作业总数。

locks：内核锁 。

meminfo物理内存、交换空间等的信息，系统内存占用情况，对应df命令。

misc：杂项 。

modules：已经加载的模块列表，对应lsmod命令 。

mounts：已加载的文件系统的列表，对应mount命令，无参数。

partitions：系统识别的分区表 。

slabinfo：sla池信息。

stat：全面统计状态表，CPU内存的利用率等都是从这里提取数据。对应ps命令。

swaps：对换空间的利用情况。

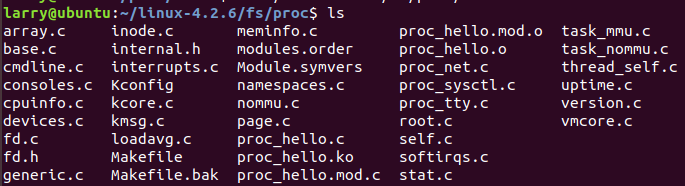
version：指明了当前正在运行的内核版本。

1. 编写内核模块，计算两次中断的时间间隔。
2. 编写内核模块来实现定时器及应用。

4、观察数据结构中的数据，s\_list把系统中已安装的文件系统的超级块通过双向链表链接起来，其中super\_blocks变量指向链表头，超级快就是对一个文件系统的描述。

**实验步骤**

（1）、实现proc系统的代码主要在linux/fs/proc目录下。该目录下的文件如下图所示。



1、创建目录

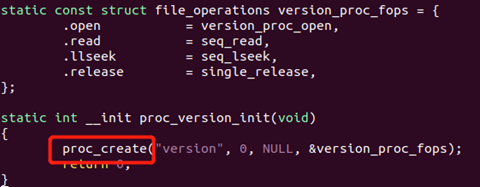
在Linux内核网址https://www.kernel.org/doc/html/latest/查阅相关文档可知，proc中创建目录的函数为proc\_mkdir，该函数原型为：

struct proc\_dir\_entry \*proc\_mkdir(const char \*name, struct proc\_dir\_entry \*parent)；

该函数接受两个参数，分别为要创建的目录的名称以及一个指向该目录的父目录的指针，并返回指向当前目录的指针结构体。

2、创建文件

考虑到在/proc目录下有version和softirqs等文件，且源码中有以上文件的实现，分别为version.c和softirqs.c，则可参考这两个文件的实现进一步实现创建world文件。以version.c为例，部分代码如下图所示。

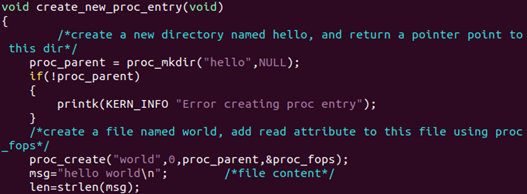


显然，创建version文件的函数为proc\_create，查阅文档可知，该函数原型为：

static inline struct proc\_dir\_entry \*proc\_create(const char \*name, umode\_t mode, struct proc\_dir\_entry \*parent, const struct file\_operations \*proc\_fops)；

该函数接收4个参数，第一个参数为文件名，第二个参数为文件的读写权限，第三个参数为其父目录的结构体指针，第四个参数为文件的读写操作结构体。

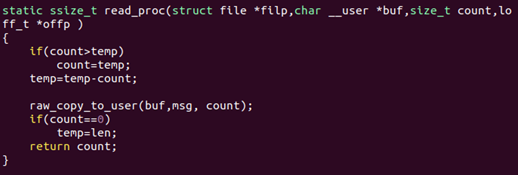
参照/proc/version文件的实现代码和本实验要求，可写出创建world文件的主要代码：



其中，world文件的父目录指针即为先前创建hello目录所返回的指针，proc\_fops定义了该文件所能执行的操作，由于本实验只要求读取文件内容，因此只实现了read属性：



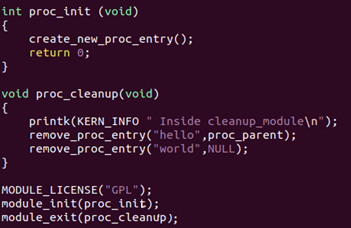
其中，read\_proc为实现读取文件内容的函数指针，其实现如下：



该函数将文件内容通过msg复制给buf，以此实现文件内容的读取。

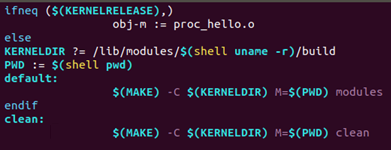
1. 定义模块的初始化和清理函数

其实现如下：



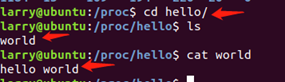
5、编写Makefile

为编译该模块，还需编写Makefile文件，该文件内容如图所示：



6、实验结果

结果如图所示。由图可见，/proc目录下生成了hello目录，且该目录下的world文件的内容为hello world。



7、运行与验证

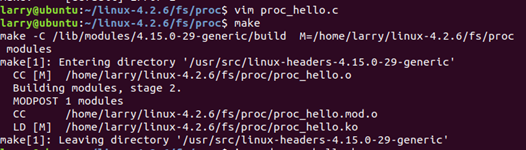
尽管本实验的内核版本为Linux-4.2.6，但是经测试可知，该模块在3.10及以上版本均可使用。

为使用该模块，首先需要下载3.10及以上Linux内核版本并解压，然后将proc\_hello.c文件拷贝至Linux/fs/proc目录下，由于该目录下原本就有Makefile文件，因此要对原Makefile文件备份后再拷贝本实验中的Makefile：

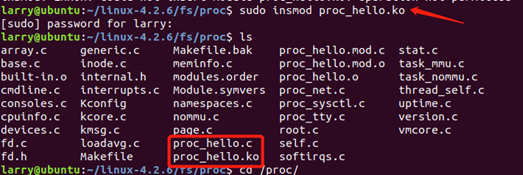
mv Makefile Makefile.bak

mv my\_makefile Makefile

然后，再使用make命令编译模块：

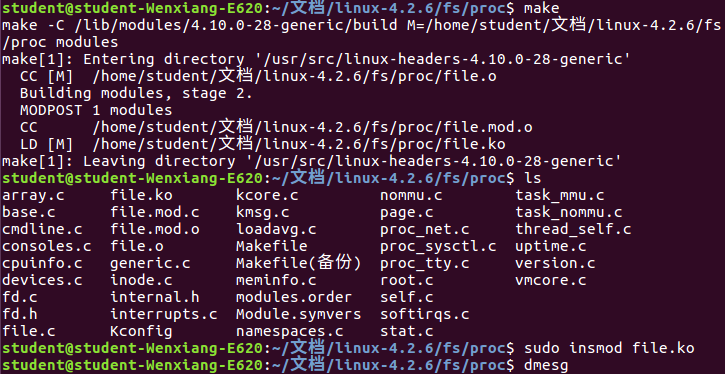


编译成功后，再使用insmod命令安装模块，可以看到该目录下多了一个ko文件，该文件即为编译成功的模块文件。

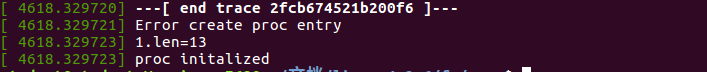


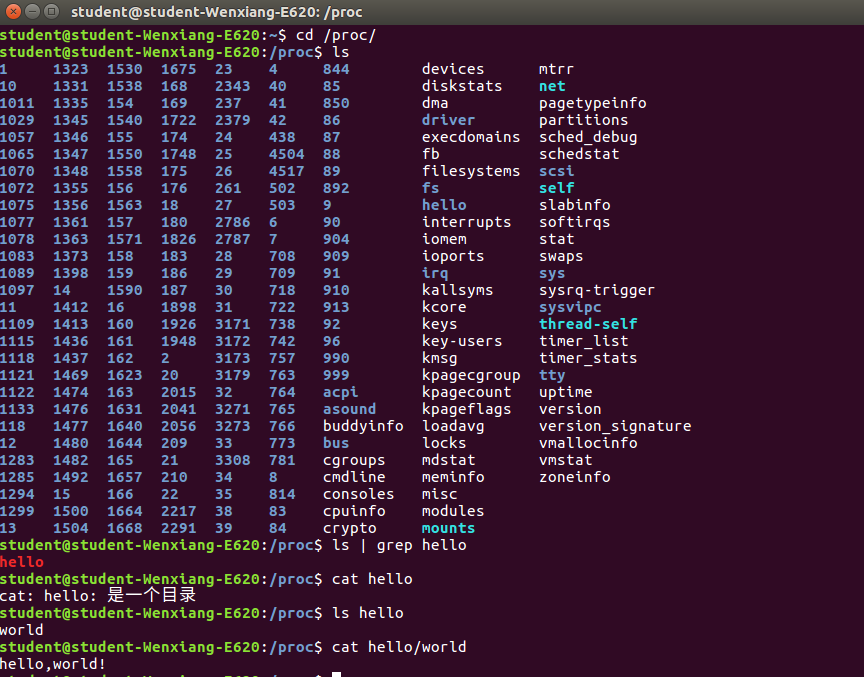
最后切换至/proc目录，即可看到上述结果。

1. 、在内核中，时间用无符号长整型数jiffies表示，其是一个全局变量，表示自系统启动以来的时钟节拍数，另外，通过给内核模块传递参数的形式，把设备名和对应的中断号irq传给模块。interface是设备名，irq是所要申请的中断号，可以从/proc/interrupts文件中查找得到，这里要申请的中断好必须是可共享的。
2. 、定时器是管理内核所花时间的基础，有时候也被称为动态定时器或内核定时器，内核经常需要推后执行某些代码，这时候定时器就可以开始发挥作用了。定时器的使用很简单，只需要执行一些初始化工作，设置一个到期时间，指定到时候执行的函数，然后激活定时器就可以了，指定的函数将在定时器到期时自动执行。
3. 、编写内核模块，打印super\_blocks结构中一些域的值，在程序中super\_blocks变量sb\_lock变量的地址在fs.h和spinlock.h头文件已经定义，但是，在linux内核中，并不是每个变量和函数都可以在其他子系统和模块中被引用，只有导出后才能被使用，但这两个变量并没有被导出，所有不能再模块中直接使用这些变量。为了解决这些问题，可以通过EXPORT\_SYMBOL宏可以将这些变量导出，然后重新编译内核，但这样太麻烦了，我们可以先从proc文件系统下的kallsyms文件中存放的内核所有符号信息中直接获得这些变量的具体值
4. **实验结果分析（截屏的实验结果，与实验结果对应的实验分析）**
5. 编写一个内核模块，在/proc文件系统中增加一个目录hello，并在这个目 录中增加一个文件world，文件的内容为hello world。

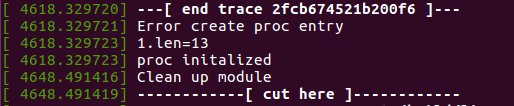
在linux/fs/proc文件中书写要创建的内核模块：file.c，编译内核模块后查看是否成功创建 file.ko ，如果成功创建则将其装入内核中执行  


使用dmesg重看内核输出到系统文件的信息为：



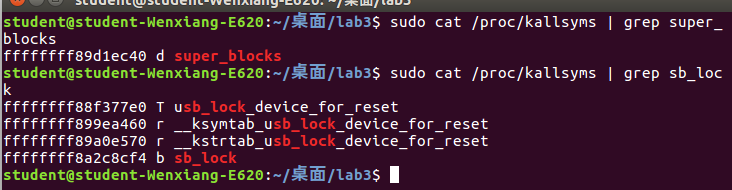
此时查看/proc目录下是否创建了hello/world 以及 world文件里有无成功写入信息：hello，world  


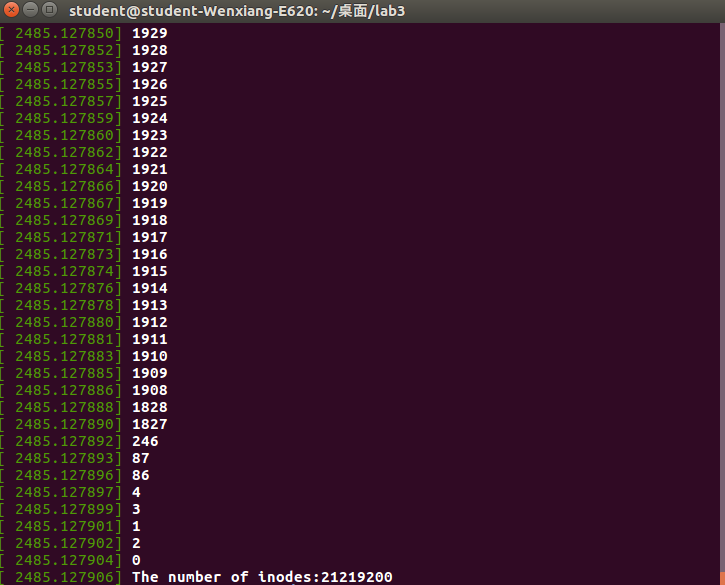
卸载模块后：



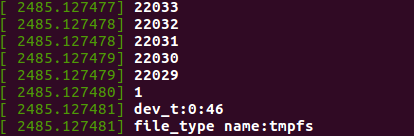
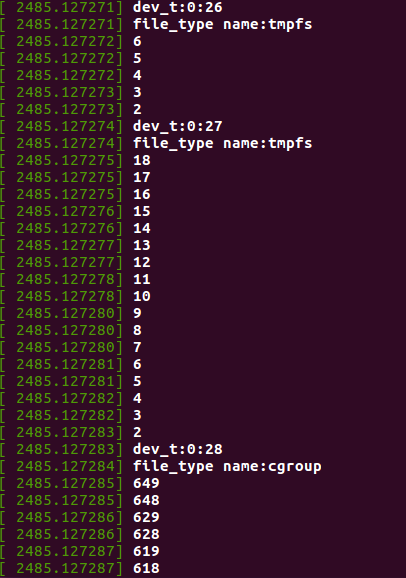
可见，我们利用内核模块成功创建了出了文件，并且该文件是可读的，这是通过proc\_fops结构来设置文件操作权限的，还可以设.open .llseek等权限，卸载模块时，在卸载函数中调用remove\_proc\_entry()函数就可以清除已创建的文件，注意要将清除文件的父目录指针也设置好。而且在上述实验过程中，我们还使用了grep管道命令来过滤出我们需要的信息，这样大大提高了信息搜索的效率。

1. 编写内核模块，打印super\_blocks结构中一些域的值

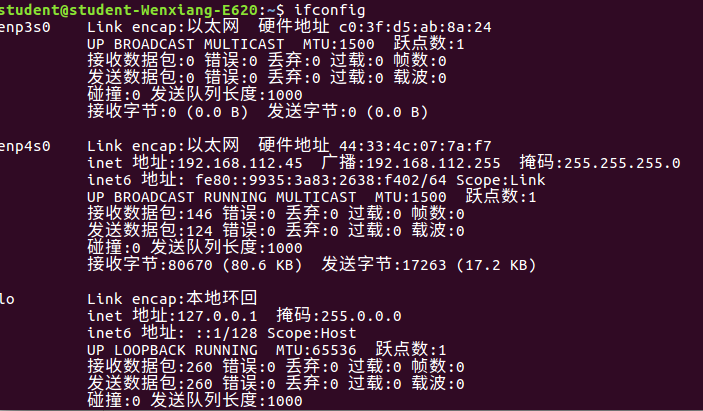
编译程序前我们先找到proc/kallsyms文件中存放的super\_blocks、sb\_lock变量的值：（如图为 0xffffffff89d1ec40 和 0xffffffff8a2c8cf4 这是在64位机器上的地址，不能错写成32位，否则模版装入的时候一定会报错）  


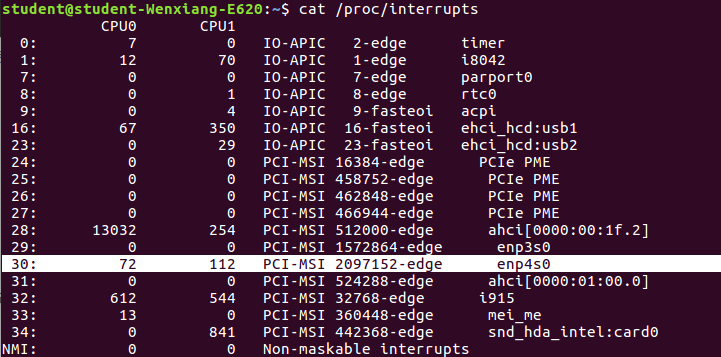
编译内核模块并装入后，先看一下系统文件中的最终输出结果：  


可以看到当前系统中总VFS索引结点数量为21219200，再仔细查看不同文件系统的超级块结构信息：

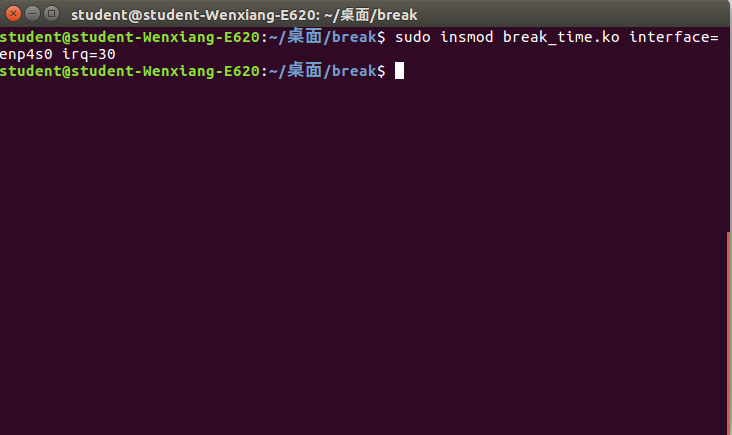
  
  
tmpfs称为临时文件系统，是一种基于内存的文件系统，最大的特点就是它的存储空间在虚存之中，因此其读写速度会特别快；另一个cgroup文件系统，其名称源自控制组群（control groups）的简写，是Linux内核的一个功能，用来限制、控制与分离一个进程组的资源（如CPU、内存、磁盘输入输出等）。

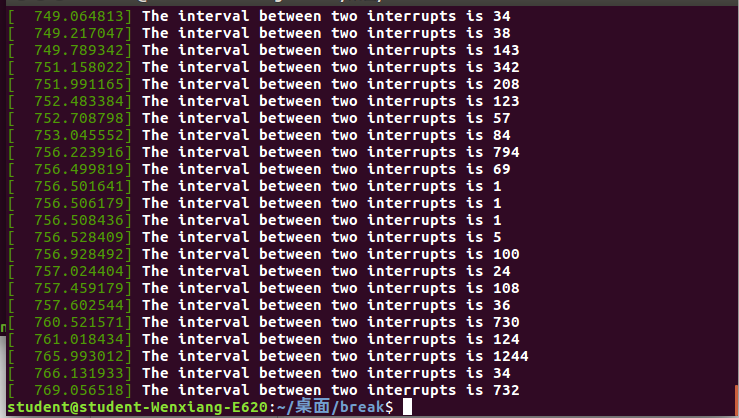
1. 编写内核模块，计算两次中断的时间间隔

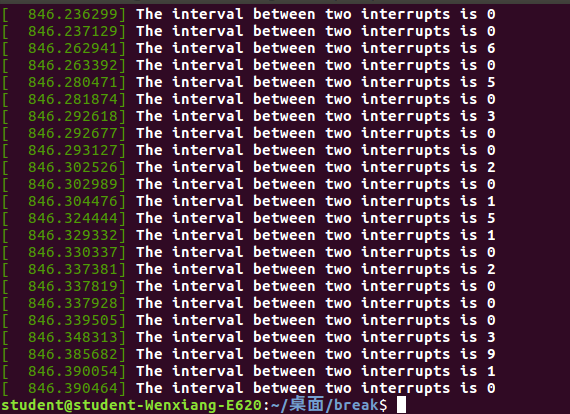
先选用网卡设备来进行本次实验，使用ifconfig命令来查看当前使用的网卡信息：（可见使用的网卡端口为enp4s0）  


再到proc/interrupts 文件中查看其对应的中断号：（30号中断-enp4s0）  


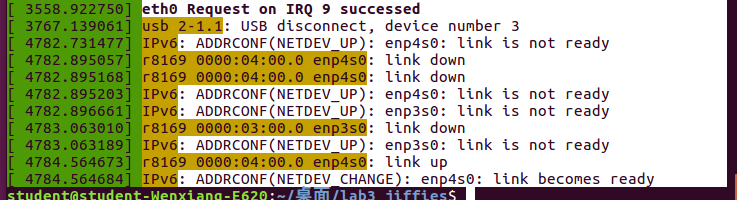
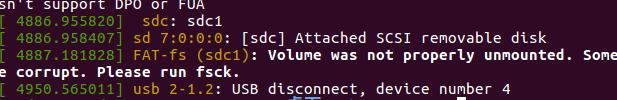
编译模块并写好相关参数将模块装入内核中，查看系统文件：

  
2020-10-28 16-51-28屏幕截图

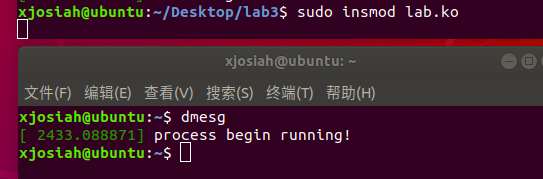
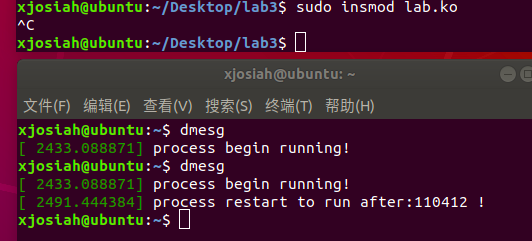
可以看到设备号和中断号对应成功，并开始在系统文件中输出两次中断的间隔时间，我们先将设备端口再观察断开后的系统信息：  


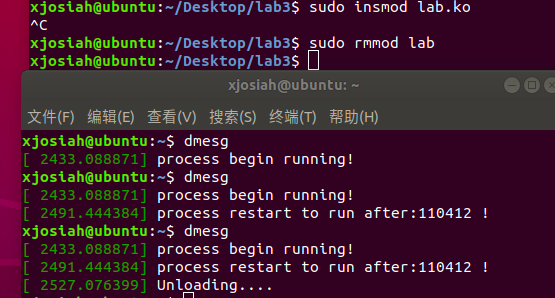
再将设备连通，并尝试建立网络连接，观察有网络连接时的系统信息：  


对比两次我们可以发现，当网络连接正常时，设备发起中断的频率明显变高了很多，这表明了不断的有数据从设备中流入到处理器中。

通过监控eth0设备，我们还可以观察到系统上很多扩展端口的信息：如网卡端口、usb接口等，尝试断开网络设备和插入拔出usb设备观察不同的现象：  
  
拔出u盘可以发现系统信息输出了usb退出的中断:  


1. 创建和使用进程延时，说明定时器在内核中实际使用

实验中使用了两个终端来查看内核模块装入后进程使用定时器睡眠的效果，首先将编译完的模块装入，然后查看系统信息可以看到，进程已经开始运行但是还没有被唤醒：  
  
中断程序的运行，并同时查看系统信息可以查看到进程再被中断唤醒前睡眠了多少个节拍数（1000个节拍/s），也就代表了定时器运行的时间：  


卸载模块：  


此次使用中断来操作定时器的原因是在程序编写完成后，我尝试使用了5000个节拍数（也就是5秒）和50000个节拍来设置定时数，程序运行基本在5~7秒（50~55秒）后会自动结束，但查看系统信息时却发现schedule\_timeout()的返回值一直都是0，无法正确的查看到进程真正睡眠的节拍数，不得以才换了另一种思路即使用了中断的方法来直接打断进程设置的定时器，具体原因是在schedule\_timeout()末尾的时候 timeout=当前程序节拍数-预测结束的节拍数（程序起始时节拍数+timeout），而返回的时候如果timeout为负值则输出0，否则才会输出timeout，也就是说只有程序提前结束的时候timeout才会是正值，而正常情况下程序总会在预定的节拍数之后结束，所以值总是0没办法看到睡眠的具体节拍数。

1. **实验总结**
2. 一个超级块对应一个文件系统，一种文件系统下面可能有多个超级块，不同的磁盘分区可以是不同的文件系统，但一个超级快始终都是对应着一个文件系统，保存着文件系统的类型、大小和状态等等信息，而索引结点是对一个文件物理属性的描述。
3. 在观察外设设备发起的中断时可以发现一个事情：u盘如果只是在系统中卸载是不算真正的退出的，操作系统只是将其驱动程序卸载了，只有在u盘真正的拔出时，系统才会响应一个usb退出的中断，此时USB才算真正的退出了。
4. Linux系统中grep命令是一种强大的文本搜索工具，它能使用正则表达式搜索文本，并把匹 配的行打印出来。grep全称是Global Regular Expression Print，表示全局正则表达式版本，是linux中三大常用的文本处理工具之一，好好掌握其使用可以帮助我们在众多输出消息中快速且准确地找到我们需要的信息。