北京邮电大学 操作系统 实验报告

实验日期：2010-01-04
实验名称：串行通信与模块设计

目录

一、实验目的

了解 LINUX 下文件系统和 I/O 的实现，了解其中模块的设计。

二、实验内容

1. 实现一个 LINUX 下串口的非阻塞读写程序。
2. 了解 LINUX 下模块的实现方式，及相应的编译和安装及卸载过程。实现一个模块，在模块安装时可打印信息（Hello）和卸载时打印信息（Goodbye）。
3. 实现一个 procfs 文件系统的模块。

三、项目要求及分析

1.串口操作需要的头文件

```
#include      <stdio.h>          /*标准输入输出定义*/
#include      <sys/types.h>    /*数据类型*/
#include      <sys/stat.h>     /*定义了一些返回值的结构*/
#include      <fcntl.h>          /*文件控制定义*/
#include      <termios.h>       /*PPSIX 终端控制定义*/
```

2. 打开串口

在 Linux 下串口文件是位于 /dev 下的

串口一 为 /dev/ttyS0

串口二 为 /dev/ttyS1

打开串口是通过使用标准的文件打开函数操作：

```
int fd;
/*以读写方式打开串口*/
fd = open( "/dev/ttyS0", O_RDWR);
if (-1 == fd){ /* 不能打开串口一*/
perror(" 提示错误！");
}
```

3．模块分析

linux 提供一个模块机制，是因为它本身是一个单内核。内核模块是 linux 内核向外部提供的一个接口，模块机制是为了弥补单内核的可扩展性和可维护性差等缺点。

模块是具有独立功能的程序，它可以被单独编译，但不能独立运行。它在运行时被链接到 内核作为内核的一部分在内核空间运行，这与运行在用户空间的进程是不同的。模块通常由一组函数和数据结构组成，用来实现一种文件系统、一个驱动程序或其他内核上层的功能。

模块程序必须通过 module_init()和 module_exit()函数来告诉内核“我来了”和“我走了”。

4．模块安装于卸载命令

1. Insmod 命令

调用 insmod 程序把需要插入的模块以目标代码的形式插入到内核中。在插入的时候，ins

mod 自动调用 init_module()函数运行。注意，只有超级用户才能使用这个命令，其命令

格式为：

```
# insmod [path] modulename.ko
```

2. rmmod 命令

调用 rmmod 程序将已经插入内核的模块从内核中移出，rmmod 会自动运行 cleanup_m

odule()函数，其命令格式为：

```
#rmmod [path] modulename.ko
```

5. procfs 分析

procfs 是比较老的一种用户态与内核态的数据交换方式,procfs 一般用于向用户出口少量的数据信息,或用户通过它设置内核变量从而控制内核行为。

如果一个驱动程序被直接编译到了内核中,那么即使这个驱动程序没有运行,它的代码和静态数据也会占据一部分空 间。但是如果这个驱动程序被编译成一个模块,就只有在需要内存并将其加载到内核时才会真正占用内存空间。

四、具体实现及运行结果

1. 串口编程

如果没有接串口，会收到一些乱码。详见附录中有程序(serial.c)

2. 模块步骤

第一步：准备源代码

首先我们还是要来编写一个符合 linux 格式的模块文件,这样我们才能开始我们的模块编译。假设我们有一个源文件 mymodules.c

第二步：编写 Makefile 文件

```
obj-m := modules.o                          #要生成的模块名
modules-objs:= mymodules.o                  #生成这个模块名所需要的目标文件
KDIR := /lib/modules/2.6.36huihui010/build
PWD := $(shell pwd)
default:
	make -C $(KDIR) M=$(PWD) modules
clean:
	rm -rf *.o .* .cmd *.ko *.mod.c .tmp_versions
```

现在我们已经准备好了我们所需要的源文件和相应的 Makefile。我们现在就可以编译了。在终端进入源文件目录输入 make：

```
root@hui-laptop:/home/hui/桌面# sudo make
make -C /lib/modules/2.6.36huihui010/build M=/home/hui/桌面 modules
make[1]: 正在进入目录 `/usr/src/linux-2.6.36'
  CC [M]  /home/hui/桌面/mymodules.o
  LD [M]  /home/hui/桌面/modules.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/hui/桌面/modules.mod.o
  LD [M]  /home/hui/桌面/modules.ko
make[1]:正在离开目录 `/usr/src/linux-2.6.36'
root@hui-laptop:/home/hui/桌面#
```

有一个 modules.ko 生成了，这就是我们的模块了，现在我们就可以来加载了。

首先在终端输入：sudo insmod modules.ko

现在我们来看看我们的模块加载成功没有呢？

在终端输入：dmesg | tail -12

tail -12 显示最后 12 条：

```
root@hui-laptop:/home/hui/桌面# sudo insmod modules.ko
root@hui-laptop:/home/hui/桌面# dmesg | tail -12
[   18.325456] b43-phy0 ERROR: You must go to http://wireless.kernel.org/en/user
s/Drivers/b43#devicefirmware and download the correct firmware for this driver v
ersion. Please carefully read all instructions on this website.
[   18.392230] input: HDA Intel Mic at Ext Front Jack as /devices/pci0000:00/000
0:00:1b.0/sound/card0/input11
[   18.392342] input: HDA Intel HP Out at Ext Front Jack as /devices/pci0000:00/
0000:00:1b.0/sound/card0/input12
[   18.392415] input: HDA Intel HP Out at Ext Front Jack as /devices/pci0000:00/
0000:00:1b.0/sound/card0/input13
[   19.519862] ppdev: user-space parallel port driver
[   19.787615] tg3 0000:09:00.0: eth0: Link is up at 100 Mbps, full duplex
[   19.787619] tg3 0000:09:00.0: eth0: Flow control is on for TX and on for RX
[   19.787790] ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[  450.196162] CE: hpet increased min_delta_ns to 7500 nsec
[  498.439497] usb 2-1: USB disconnect, address 2
[ 3086.208073] CE: hpet increased min_delta_ns to 11250 nsec
[ 3704.204737] Hello
root@hui-laptop:/home/hui/桌面# dmesg | tail -1
[ 3704.204737] Hello
root@hui-laptop:/home/hui/桌面# dmesg | tail -2
[ 3086.208073] CE: hpet increased min_delta_ns to 11250 nsec
[ 3704.204737] Hello
```

我们的模块的初始化函数 yuer_init();已经成功运行了。说明我们的模块已经加载成功；

现在我们再来卸载模块试试看。

在终端输入：sudo rmmod modules

在终端输入：dmesg | tail -3

```
root@hui-laptop:/home/hui/桌面# sudo insmod modules.ko
insmod: error inserting 'modules.ko': -1 File exists
root@hui-laptop:/home/hui/桌面# sudo rmmod modules
root@hui-laptop:/home/hui/桌面# dmesg | tail -4
[  498.439497] usb 2-1: USB disconnect, address 2
[ 3086.208073] CE: hpet increased min_delta_ns to 11250 nsec
[ 3704.204737] Hello
[ 3861.945180] Goodbye
root@hui-laptop:/home/hui/桌面#
```

3．实现一个 procfs 文件系统的模块。

Procfs 提供了如下 API：

Struct proc_dir_entry *create_proc_entry(const char *name,mode_t mode,struct pro_dir_entry *parent)

该函数用于创建一个正常的 proc 条目，参数 name 给出要建立的 proc 条目的名称，参数 mode 给出了建立的该 proc 条目的访问权限，参数 parent 指定建立的 proc 条目所在的目录。如果要在/proc 下建立 proc 条目，parent 应当为 NULL。否则它应当为 proc_mkdir 返回的 struct proc_dir_entry 结构的指针。

Extern void remove_proc_entry(const char *name,struct proc_dir_entry *parent)

该函数用于删除上面函数创建的 proc 条目，参数 name 给出要删除的 proc 条目的名称，参数 parent 指定建立的 proc 条目所在的目录。

通过/proc 文件系统实现财富分发，在加载这个模块之后，用户就可以使用echo 命令向其中导入文本财富，然后再使用 cat 命令逐一读出。详见附录三程序。

五、所遇问题及解决办法

在模块编译的时候出现错误，后来发现 Makefile 文件需要用 tab，改后就能编译通过。

六、实验心得

通过实验，了解了操作系统 linux 下的 I/O 的串口通信基本原理，学习了模块的安装和卸载。练习了一下 Makefile 文件的编写与编译，对操作系统有了更深一步的了解。

附：程序代码

1．串口程序(serial.c)：

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <termios.h>
#include <stdio.h>
#define BAUDRATE B1200
#define MODEMDEVICE "/dev/ttyS0"
#define _POSIX_SOURCE 1
main()
{
int fd,c,res;
struct termios oldios,newios;
char buf[255];

fd = open(MODEMDEVICE, O_RDWR | O_NOCTTY);
if (fd < 0)
```

```
{
perror(MODEMDEVICE);
exit(0);
}

tcgetattr(fd,&oldios);
bzero(&newios,sizeof(newios));

newios.c_cflag = BAUDRATE | CS7 | CLOCAL | CREAD;
newios.c_iflag = IGNPAR;
newios.c_oflag = 0;
newios.c_lflag = 0;
newios.c_cc[VTIME] = 0;
newios.c_cc[VMIN] = 1;

tcflush(fd,TCIFLUSH);
tcsetattr(fd,TCSANOW,&newios);

while (1) {
write(fd,'1',1);
res = read(fd,buf,1);
buf[res] = 0;
printf("%s",buf);
}
tcsetattr(fd,TCSANOW,&oldios);
}
```

2．模块程序(mymodules.c)

```
#include <linux/module.h>
#include <linux/init.h>
#include<linux/kernel.h>
#include <linux/moduleparam.h>
MODULE_LICENSE("GPL");
MODULE_AUTHOR("Yang huahui");

static int nbr = 10;
module_param(nbr, int, S_IRUGO);
static int __init hui_init(void)
{
        printk(KERN_ALERT "Hello\n");
        return 0;
}
```

```
static void __exit hui_cleanup(void)
{
        printk(KERN_ALERT"Goodbye\n");
}

module_init(hui_init);
module_exit(hui_cleanup);
```

3． Procfs

```
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/proc_fs.h>
#include <linux/string.h>
#include <linux/vmalloc.h>
#include <asm/uaccess.h>
MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Fortune Cookie Kernel Module");
MODULE_AUTHOR("M. Tim Jones");
#define MAX_COOKIE_LENGTH PAGE_SIZE
static struct proc_dir_entry *proc_entry;
static char *cookie_pot; // Space for fortune strings
static int cookie_index; // Index to write next fortune
static int next_fortune; // Index to read next fortune
int init_fortune_module( void )
{
int ret = 0;
cookie_pot = (char *)vmalloc( MAX_COOKIE_LENGTH );
if (!cookie_pot) {
ret = -ENOMEM;
} else {
memset( cookie_pot, 0, MAX_COOKIE_LENGTH );
proc_entry = create_proc_entry( "fortune", 0644, NULL );
if (proc_entry == NULL) {
ret = -ENOMEM;
vfree(cookie_pot);
printk(KERN_INFO "fortune: Couldn't create proc entry\n");
} else {
cookie_index = 0;
next_fortune = 0;
proc_entry->read_proc = fortune_read;
proc_entry->write_proc = fortune_write;
proc_entry->owner = THIS_MODULE;
```

```
printk(KERN_INFO "fortune: Module loaded.\n");
}
}
return ret;
}
void cleanup_fortune_module( void )
{
remove_proc_entry("fortune", &proc_root);
vfree(cookie_pot);
printk(KERN_INFO "fortune: Module unloaded.\n");
}
module_init( init_fortune_module );
module_exit( cleanup_fortune_module );
```