杭州应用工程技术学院学报,第12卷第4期,2000年12月
Journal of Hangzhou Institute of Applied Engineering
Vol.12 No.4, Dec. 2000

Linux字符设备驱动程序的设计

潘俊强　刘　莉
(杭州应用工程技术学院 计算机系　杭州 310012)

摘　要　介绍了Linux字符设备驱动程序中建立设备,初始化设备、设备的资源分配和如何访问设备的方法及相关函数的实现.

关键词　Linux　字符设备　设备驱动程序
中图分类号　TP316.81

设备驱动程序实质上是一组完成不同任务的函数的集合,通过这些函数所提供的功能可以使得从设备接受输入和将输出送到设备就象读写文件一样,因此,Linux中的每一个设备都具有文件的外在特征,都能使用open(),close(),read(),write()等系统调用.

Linux设备驱动程序的主要功能有:

a初始化设备;b提供各类设备服务;c负责内核和设备之间的数据交换;d检测和处理设备工作过程中出现的错误.

1　命名规则[1]

每个设备的驱动程序都有一组实质上相同的函数,并且都需添加至内核原码中以重新生成内核,因此为了防止不同驱动程序之间函数名的冲突,必须确保名称的唯一性,最好的方法是在各驱动程序的函数前加一以设备名为字符串的前缀.

本文约定要开发驱动程序的设备名为"mydev".

2　设备文件的建立

为了使对设备的读写操作象文件的存取一样处理,Linux所有的设备在目录树中的适当位置都有相对应的文件名称,这样才能对它们进行open()、close()等系统调用;这些文件称为字符设备特殊文件或块设备特殊文件,一般存放在/dev目录中,如主硬盘第一分区的块设备特殊文件为/dev/hda0,第一虚拟控制终端的字符设备特殊文件为/dev/tty1.

设备特殊文件的建立使用mknod命令或mknod()系统调用(只有root帐号才有权建立),命令格

收稿日期:1999-12-27

式为:mknod filename type major minor,其中 filename 为要建立的设备特殊文件名(含路径),type 说明设备类型(c 为字符设备,b 为块设备),major 和 minor 说明与该文件结合的主设备号和次设备号.

如:# mknod/dev/mydev c 40 1 命令,用主设备号 40 和次设备号 1 建立了字符设备特殊文件/dev/mydev.

3 init 函数

init 函数用来完成对所控设备的初始化工作,以及调用 register_chrdev()函数来注册字符设备.根据命名规则和本文约定,设备"mydev"的 init 函数为:

```
void mydev_init(void)
{
  if(register_chrdev(40,"mydev",& mydev_fops))
      TRACE_TXT("Device(40) driver registered error")
  else
      TRACE_TXT("Device(40) driver registered successfully")
//以下为对设备进行初始化的代码,略
.....
}
```

其中,register_chrdev 函数中的参数 40 为主设备号,"mydev"为设备名,mydev_fops 为以下所要介绍的包含基本入口点的结构,类型为 file_operations;当执行 myder_init 时,它将调用内核函数 register_chrdev,把驱动程序的基本入口点指针存放在内核的字符设备地址表中,在用户进程对该设备执行系统调用时提供入口地址.

4 基本入口点

每个设备驱动程序都有一个称为 file_operation 的数据结构,其中包含指向驱动程序内部大多数函数的入口(函数指针),完整结构如下:

```
struct file_operations {
  int (* lseek) ();
  int (* read) ();
  int (* write) ();
  int (* readdir) ();
  int (* select) ();
  int (* ioctl) ();
  int (* mmap) ();
  int (* open) ();
  void (* release) ();
  int (* fsync) ();
  int (* fasync) ();
  int (* check_media_change) ();
  void (* revalidate) ();
```

};

大多数的驱动程序只是利用了其中的一部分(对于驱动程序中不提供的功能,把相应位置的值设为NULL),对于字符设备来说,要提供的主要入口有:open()、release()、read()、write()、ioctl(),现分别说明如下:

open()函数　对设备特殊文件进行open()系统调用时,将调用驱动程序的open()函数:int open(struct inode * inode,struct file * file),其中参数inode为设备特殊文件的inode(索引结点)结构的指针,参数file是指向这一设备的文件结构的指针.open()的主要任务是确定硬件处在就绪状态、验证次设备号的合法性(次设备号可以用MINOR(inode->i_rdev)取得)、控制使用设备的进程数、根据执行情况返回状态码(0表示成功,负数表示存在错误)等.

release()函数　当最后一个打开设备的用户进程执行close()系统调用时,内核将调用驱动程序的release()函数:void release(struct inode * inode,struct file * file),参数说明同上.release函数的主要任务是清理未结束的输入/输出操作、释放资源、用户自定义排他标志的复位等.

read()函数　当对设备特殊文件进行read()系统调用时,将调用驱动程序read()函数:void read(struct inode * inode,struct file * file,char * buf,int count);inode和file参数定义同上,参数buf是指向用户空间缓冲区的指针,由用户进程给出,count为用户进程要求读取的字节数,也由用户给出.read()函数的功能就是从硬设备或内核内存中读取或复制count个字节到buf指定的缓冲区中;在复制数据时要注意,驱动程序运行在内核中,而buf指定的缓冲区在用户内存区中,是不能直接在内核中访问使用的,因此,必须使用特殊的复制函数来完成复制工作,这些函数在〈asm/segment.h〉中定义:

```
void put_user_byte(char data_byte,char * u_addr);
void put_user_word(short data_word,short * u_addr);
void put_user_long(long data_long,long * u_addr);
void memcpy_tofs(void * u_addr,void * k_addr,unsigned long cnt);
```

参数u_addr为用户空间地址,k_addr为内核空间地址,cnt为字节数.

另外,在复制数据前,必须检验用户对指定内存空间的权限以确定是否予以完成操作,可以用内核函数verify_area(定义在linux/mm.h中)来完成;void verify_area(int access,void * u_addr,unsigned long size),其中,参数access为访问类型,VERIFY_WRITE为写访问,VERIFY_READ为读访问,参数u_addr为用户空间的起始地址,参数size是内存块的字节数.如果允许按指定方式访问指定内存空间,函数verify_area返回0,否则-EFAULT.

write()函数　当设备特殊文件进行write()系统调用时,将调用驱动程序的write()函数:void write(struct inode * inode,struct file * file,char * buf,int count);参数说明同上.write()的功能是将参数buf指定的缓冲区中的count个字节内容复制到硬件或内核内存中;和read()一样,复制工作也需要由特殊函数来完成:

```
unsigned char_get_user_byte(char * u_addr);
unsigned char_get_user_word(short * u_addr);
unsigned char_get_user_long(long * u_addr);
unsigned memcpy--fromfs(void * k_addr,void * u_addr,unsigned long cnt);
```

ioctl()函数　该函数是特殊的控制函数,可以通过它向设备传递控制信息或从设备取得状态信息,函数原型为:int ioctl(struct inode * inode,struct file * file,unsigned int cmd,unsigned long arg),参数inode和file的含义同上,参数cmd为设备驱动程序要执行的命令的代码,由用户自定义(0x5421,0x5450,0x5451,0x5452系统保留),参数arg为相应的命令提供参数,类型可以是整型、指针等.

同样,在驱动程序中,这些函数的定义也必须符合命名规则,按照本文约定,设备"mydev"的驱动程序的这些函数应分别命名为 mydev_open,mydev_release,mydev_read,mydev_write,mydev_ioctl,因此设备"mydev"的基本入口点结构变量 mydev_fops 赋值如下:

```
struct file_operations mydev_fops {
  NULL,
  mydev_read,
  mydev_write,
  NULL,
  NULL,
  mydev_ioctl,
  NULL,
  mydev_open,
  mydev_release,
  NULL,
  NULL,
  NULL,
  NULL
};
```

5 内存分配

和其他应用程序一样,设备驱动程序在运行时也需要一定的内存空间来存放临时数据,但由于驱动程序是在内核运行,所以它的内存管理有它的特殊性,在 Linux 中有几种分配内存的方法:

(1) 采用固定内存分配法;即在源程序中就定义一固定大小的数据缓冲区,如:static char my_buffer[1000];

(2) 使用内核的动态内存分配,由函数对 kmalloc()和 kfree()来实现.kmalloc()函数的原型为:vold kmalloc(size_t size,int priority),参数 size 为要分配内存容量的字节数,返回时指向已分配内存的起始地址,出错时,返回 0;参数 priority 说明若 kmalloc()不能马上分配内存时用户进程要采用的动作,GFP_KERNEL 表示等待,等 kmalloc()函数将一些内存安排到交换区来满足你的内存需要,GFP_ATOMIC 表示不等待,如不能立即分配到内存则返回 0 值.用 kmalloc()分配的内存块可以用 kfree()函数来释放,在〈linux/malloc.h〉中 kfree()被定义为:# define kfree(n) kfree_s((n),0),其中 kfree_s()函数原型为:void kfree_s(void * ptr,int size),参数 ptr 为 kmalloc()返回的已分配内存的指针,size 是要释放的以字节计数的内存容量,若为 0 时,由内核自动确定内存块的大小.

6 中断

由于大多数设备驱动程序是用来监视和控制硬件设备工作的,因此需要对硬件产生的中断请求提供中断服务子程序;与登记基本入口点一样,驱动程序也要请求内核将特定的中断请求和中断服务子程序联系在一起.在 Linux 中,用 request_irq()函数(在 Linux/signal.h 中定义)来实现请求:int request_irq(unsigned int irq,void(* handler)int,unsigned long type,char * name),参数 irq 为要截取的中断请求号,参数 handler 为指向中断服务子程序的指针,参数 type 用来确定是正常中断还是快速

中断，所谓正常中断就是中断服务子程序返回后，内核可以执行调度程序来确定将运行哪一个进程，而快速中断是指中断服务子程序返回后，立即执行被中断程序，正常中断 type 取值为 0，快速中断 type 取值为 SA_INTERRUPT，参数 name 是设备驱动程序的名称.

7　驱动程序的安装

当你编写完了驱动程序后，就要对它进行编译和装入内核，步骤如下：

(1) 将任何与设备驱动程序有关的.c 和.h 文件复制到包含字符设备驱动程序源码的目录中，一般在 Linux 源程序目录中的 drivers/char 子目录中.

(2) 在 chr_dev_init()函数的最后加上调用驱动程序 init()函数的源码，如本文约定，需加上 mydev_init()，chrdev_init()函数在 drivers/char/mem.c 中.

(3) 编辑 drivers/char 目录中的 makefile，将驱动程序的 OBJ 文件名称(.o)放在 OBJS 定义的后面，并将源文件名(.c)放在 SRCS 定义的后面.

(4) 编译、连接并重新安装内核.

请注意，在改变内核代码时，请备份你的旧系统，以避免由于新内核的工作混乱造成永久性的破坏.

8　结束语

设备驱动程序是操作系统内核和机器硬件之间的接口，通过它可以使得设备文件化.本文只是简单介绍了编写一个字符设备驱动程序的基本规则和方法，如读者欲了解更多、更详细的内容，请参阅 Linux 的 HowTo 文档.

参　考　文　献

1　Comes Phil. The Linux A－Z. 童寿彬等译．北京：电子工业出版社，1999.296

Design of character device driver in Linux

Pan Junqiang　Liu Li

(Hangzhou Institute of Applied Engineering, Hangzhou 310012)

Abstract　This paper introduces the ways to creat and initiate device, distribute resource for device and access device in Linux character device driver.

Key words　Linux　character device　device driver