# Linux驱动\_misc设备

Linux包含了许多的设备驱动类型，而不管分类有多细，总会有些漏网的，这就是我们经常说到的“其他的”等等。

在Linux里面，把无法归类的五花八门的设备定义为混杂设备（用miscdevice结构体来描述）。

miscdevice共享一个主设备号MISC\_MAJOR（10），但次设备号不同，所有的miscdevice设备形成一个链表，对设备访问时内核根据次设备号查找对应的 miscdevice设备，然后调用其中的file\_operations结构体中注册的文件操作接口进程操作。

## 1.简单的misc设备驱动

### 1.1 misc\_led驱动代码

#include <linux/module.h>

#include <linux/types.h>

#include <linux/mm.h>

#include <linux/kernel.h>

#include <linux/fs.h>

#include <linux/device.h>

#include <linux/io.h>

#include <linux/cdev.h> //cdev\_init等函数定义

#include <asm/uaccess.h>

#include <linux/miscdevice.h>

#define DEVICE\_NAME "misc\_led"

volatile unsigned long **\***gpfcon **=** **NULL;**

volatile unsigned long **\***gpfdat **=** **NULL;**

static int led\_ioctl**(**struct inode **\***inode**,** struct file **\***file**,** unsigned int cmd**,**unsigned long arg**)**

**{**

**if(**cmd **==** 1**)**

**{**

//点亮

**\***gpfdat **&=** **~((**1**<<**4**)** **|** **(**1**<<**5**)** **|** **(**1**<<**6**));**

**}**

**else**

**{**

//熄灭

**\***gpfdat **|=** **(**1**<<**4**)** **|** **(**1**<<**5**)** **|** **(**1**<<**6**);**

**}**

**}**

static struct file\_operations dev\_fops **=** **{**

**.**owner **=** THIS\_MODULE**,**

**.**ioctl **=** led\_ioctl**,**

**};**

static struct miscdevice misc **=** **{**

**.**minor **=** MISC\_DYNAMIC\_MINOR**,**

**.**name **=** DEVICE\_NAME**,**

**.**fops **=** **&**dev\_fops**,**

**};**

static void \_\_exit misc\_dev\_exit**()**

**{**

iounmap**(**gpfcon**);**

misc\_deregister**(&**misc**);**

**}**

static int \_\_init misc\_dev\_init**(**void**)**

**{**

int ret**;**

ret **=** misc\_register**(&**misc**);**

gpfcon **=** **(**volatile unsigned long **\*)**ioremap**(**0x56000050**,** 16**);**

gpfdat **=** gpfcon **+** 1**;**

//配置GPF4,5,6为输出

**\***gpfcon **&=** **~((**0x3**<<(**4**\***2**))** **|** **(**0x3**<<(**5**\***2**))** **|** **(**0x3**<<(**6**\***2**)));**

**\***gpfcon **|=** **((**0x1**<<(**4**\***2**))** **|** **(**0x1**<<(**5**\***2**))** **|** **(**0x1**<<(**6**\***2**)));**

**return** 0**;**

**}**

module\_init**(**misc\_dev\_init**);**

module\_exit**(**misc\_dev\_exit**);**

MODULE\_LICENSE**(**"GPL"**);**

代码很简单调用misc\_register注册一个miscdevice。miscdevice中的fops定义了相关的操作函数。

### 1.2 测试代码

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <stdio.h>

void main**()**

**{**

int fd**;**

fd **=** open**(**"/dev/misc\_led"**,**O\_RDWR**);**

ioctl**(**fd**,** 1 **,**0**);**

**}**

### 1.3 测试结果

当insmod上面的驱动后，会产生设备文件/dev/misc\_led。

# ls -l /sys/class/misc/

drwxr**-**xr**-**x 3 0 0 0 Jan 1 00**:**01 misc\_led

drwxr**-**xr**-**x 3 0 0 0 Jan 1 00**:**00 psaux

drwxr**-**xr**-**x 3 0 0 0 Jan 1 00**:**00 watchdog

# ls -l /dev/misc\_led

crw**-**rw**----** 1 0 0 10**,** 63 Jan 1 00**:**01 **/**dev**/**misc\_led

# cat /proc/misc

63 misc\_led

1 psaux

130 watchdog

操作该设备可以打开和关闭所有的led灯。

## 2. misc设备驱动架构

### 2.1 misc\_init

misc\_init完成了/proc /sys相关文件的创建，并注册了字符设备。

static int \_\_init misc\_init**(**void**)**

**{**

#ifdef CONFIG\_PROC\_FS

struct proc\_dir\_entry **\***ent**;**

//创建/proc/misc文件

//可以从该文件读取注册的misc设备信息

ent **=** create\_proc\_entry**(**"misc"**,** 0**,** **NULL);**

**if** **(**ent**)**

ent**->**proc\_fops **=** **&**misc\_proc\_fops**;**

#endif

//创建/sys/class/misc文件夹

misc\_class **=** class\_create**(**THIS\_MODULE**,** "misc"**);**

**if** **(**IS\_ERR**(**misc\_class**))**

**return** PTR\_ERR**(**misc\_class**);**

//注册字符设备，主设备号为10

**if** **(**register\_chrdev**(**MISC\_MAJOR**,**"misc"**,&**misc\_fops**))** **{**

printk**(**"unable to get major %d for misc devices\n"**,**

MISC\_MAJOR**);**

class\_destroy**(**misc\_class**);**

**return** **-**EIO**;**

**}**

**return** 0**;**

**}**

static const struct file\_operations misc\_fops **=** **{**

**.**owner **=** THIS\_MODULE**,**

**.**open **=** misc\_open**,**

**};**

### 2.2 misc\_register

misc\_register将miscdevice添加到misc设备链表中，并根据子设备号创建对应的/sys文件。

int misc\_register**(**struct miscdevice **\*** misc**)**

**{**

struct miscdevice **\***c**;**

dev\_t dev**;**

int err **=** 0**;**

//初始化miscdevice中list的指针

INIT\_LIST\_HEAD**(&**misc**->**list**);**

mutex\_lock**(&**misc\_mtx**);**

//轮询misc\_list，如果有相同的次设备号，则退出

list\_for\_each\_entry**(**c**,** **&**misc\_list**,** list**)** **{**

**if** **(**c**->**minor **==** misc**->**minor**)** **{**

mutex\_unlock**(&**misc\_mtx**);**

**return** **-**EBUSY**;**

**}**

**}**

//如果采用自动分配子设备号

**if** **(**misc**->**minor **==** MISC\_DYNAMIC\_MINOR**)** **{**

//i = 64

int i **=** DYNAMIC\_MINORS**;**

//从64往前找没有使用的子设备号

**while** **(--**i **>=** 0**)**

**if** **(** **(**misc\_minors**[**i**>>**3**]** **&** **(**1 **<<** **(**i**&**7**)))** **==** 0**)**

**break;**

**if** **(**i**<**0**)** **{**

mutex\_unlock**(&**misc\_mtx**);**

**return** **-**EBUSY**;**

**}**

misc**->**minor **=** i**;**

**}**

//将misc\_minors对应的位置1，标识该子设备号已经被使用

**if** **(**misc**->**minor **<** DYNAMIC\_MINORS**)**

misc\_minors**[**misc**->**minor **>>** 3**]** **|=** 1 **<<** **(**misc**->**minor **&** 7**);**

dev **=** MKDEV**(**MISC\_MAJOR**,** misc**->**minor**);**

//创建/sys/class/misc/misc->name目录

//mdev会创建/dev/misc->name设备

misc**->**this\_device **=** device\_create**(**misc\_class**,** misc**->**parent**,** dev**,**

"%s"**,** misc**->**name**);**

**if** **(**IS\_ERR**(**misc**->**this\_device**))** **{**

err **=** PTR\_ERR**(**misc**->**this\_device**);**

**goto** out**;**

**}**

//将该miscdevice添加到misc\_list链表中。

list\_add**(&**misc**->**list**,** **&**misc\_list**);**

out**:**

mutex\_unlock**(&**misc\_mtx**);**

**return** err**;**

**}**

### 2.3 misc\_open

当打开任何一个misc设备时，都会调用到misc\_open函数。并将file的f\_op指针设置为miscdevice的fops指针，以后进行read/write/ioctl等操作时，便会直接调用miscdevice对应的fops中的操作函数。

static int misc\_open**(**struct inode **\*** inode**,** struct file **\*** file**)**

**{**

//获得子设备号

int minor **=** iminor**(**inode**);**

struct miscdevice **\***c**;**

int err **=** **-**ENODEV**;**

const struct file\_operations **\***old\_fops**,** **\***new\_fops **=** **NULL;**

mutex\_lock**(&**misc\_mtx**);**

//轮询misc\_list链表，根据子设备号找到对应的fops

list\_for\_each\_entry**(**c**,** **&**misc\_list**,** list**)** **{**

**if** **(**c**->**minor **==** minor**)** **{**

new\_fops **=** fops\_get**(**c**->**fops**);**

**break;**

**}**

**}**

**if** **(!**new\_fops**)** **{**

mutex\_unlock**(&**misc\_mtx**);**

request\_module**(**"char-major-%d-%d"**,** MISC\_MAJOR**,** minor**);**

mutex\_lock**(&**misc\_mtx**);**

list\_for\_each\_entry**(**c**,** **&**misc\_list**,** list**)** **{**

**if** **(**c**->**minor **==** minor**)** **{**

new\_fops **=** fops\_get**(**c**->**fops**);**

**break;**

**}**

**}**

**if** **(!**new\_fops**)**

**goto** fail**;**

**}**

err **=** 0**;**

old\_fops **=** file**->**f\_op**;**

//设置file的f\_op为miscdevice的fops

file**->**f\_op **=** new\_fops**;**

//调用miscdevice的open函数

**if** **(**file**->**f\_op**->**open**)** **{**

err**=**file**->**f\_op**->**open**(**inode**,**file**);**

**if** **(**err**)** **{**

fops\_put**(**file**->**f\_op**);**

file**->**f\_op **=** fops\_get**(**old\_fops**);**

**}**

**}**

fops\_put**(**old\_fops**);**

fail**:**

mutex\_unlock**(&**misc\_mtx**);**

**return** err**;**

**}**