混杂设备驱动程序（MISC Device）

#include <linux/miscdevice.h>

一种特殊的字符驱动程序，它们拥有一些相同的特性。内核将这些共同行抽象至一个API中（具体实现代码见 drivers/char/misc）,这些简化了驱动程序的初始化的方式。所有的混杂设备都被分配一个主设备号10（主设备号固定），但每个设备可选择一个单独的次设备号。

普通的字符驱动程序完成初始化的顺序如下。

  1，通过alloc\_chrdev\_region()及相关函数分配主/次设备号。

  2，使用device\_create()创建/dev和/sys节点。

  3，使用cdev\_init()和cdev\_add()将自身注册为字符驱动程序。

混杂驱动只需要调用misc\_register()即可完成字符驱动的所有步骤：

       static struct misdevice mydrv\_dev={

            MYDRV\_MINOR,

             "mydrv",

            &mydrv\_fops

};

 misc\_register(&mydrv\_dev);

其中MYDRV\_MINOR是静态分配给混杂驱动程序的次设备号。也可以在mydrv\_dev机构中通过指定MISC\_DYNAMIC\_MINOR而不是MYDRV\_MINOR以要求动态分配的次设备号。每一个混杂驱动程序自动出现在/sys/class/misc 文件中。

下面以友善6410提供的buttons驱动来说明一下。

#define DEVICE\_NAME     "buttons"

static struct file\_operations dev\_fops = {

    .owner   =   THIS\_MODULE,

    .open    =   s3c64xx\_buttons\_open,

    .release =   s3c64xx\_buttons\_close,

    .read    =   s3c64xx\_buttons\_read,

    .poll    =   s3c64xx\_buttons\_poll,

};

static struct miscdevice misc = {

 .minor = MISC\_DYNAMIC\_MINOR,//动态分配次设备号

 .name = DEVICE\_NAME,//设备名

 .fops = &dev\_fops,//文件操作的方法

};

static int \_\_init dev\_init(void)

{

 int ret;

 ret = misc\_register(&misc);//设备初始化

 printk (DEVICE\_NAME"/tinitialized/n");

 return ret;

}

static void \_\_exit dev\_exit(void)

{

 misc\_deregister(&misc);

}

总的来看混杂设备驱动程序主要是简化了驱动程序初始化的方式。

#define DEVICE\_NAME "buzzer" //设备名

#define BUZZER\_MINOR MISC\_DYNAMIC\_MINOR //动态申请次设备号

//#define BUZZER\_MINOR 30;                //静态分配次设备号

static struct miscdevice misc = {

.minor = BUZZER\_MINOR,

.name = DEVICE\_NAME,

.fops = &buzzer\_fops,

};

static int \_\_init buzzer\_init(void) //设备驱动模块加载函数

{

int ret;

{

unsigned tmp;

tmp = readl(S3C64XX\_GPFCON);

tmp = (tmp & ~(0x3U<<28))|(0x1U<<28);

writel(tmp, S3C64XX\_GPFCON);

tmp = readl(S3C64XX\_GPFDAT);

tmp &= ~(0x1 << 14);

writel(tmp, S3C64XX\_GPFDAT);

}

ret = misc\_register(&misc);

printk (DEVICE\_NAME"\tinitialized\n");

sema\_init(&sem,1);

return ret;

}

static void \_\_exit buzzer\_exit(void) //设备驱动模块卸载函数

{

misc\_deregister(&misc);

}