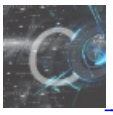
字符设备驱动程序之misc_dev方式注册字符设备

原创

xyz-x 于 2017-08-04 18:16:19 发布 981 收藏 1

分类专栏： 驱动开发 文章标签： 字符设备-misc



注册字符设备有三种方法：chardev、cdev、misc注册，本文介绍用misc_dev注册方法注册设备，编写简单字符设备驱动程序，实现字符设备驱动程序的基本框架。

编写字符设备驱动的基本步骤为：

1、编写对该设备的各种操作函数（open、write、ioctl）

2、定义一个file_operation结构体，该结构体用来存储驱动内核模块提供的对设备进行各种操作的函数的指针即open()、write()、ioctl()

3、编写入口函数，该函数中进行设备注册

4、编写出口函数，该函数进行设备的注销

随着字符设备种类和数量的增加，设备号越来越紧张，为此Linux系统提出misc设备模型以解决此问题。所有misc设备其主设备号都是10，不同设备使用不同的次设备号区分。另外misc设备驱动会为设备自动创建设备文件，不需要想cdev设备那样，需要自己手动创建，所以使用起来更为方便。

以下是一个简单的字符设备驱动程序：

```
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/fs.h>
#include <linux/init.h>
#include <linux/delay.h>
#include <asm/uaccess.h>
#include <asm/irq.h>
#include <asm/io.h>
#include <linux/major.h>
#include <linux/fs.h>
#include <linux/init.h>
#include <linux/backing-dev.h>
#include <linux/raw.h>
#include <linux/capability.h>
#include <linux/uio.h>
#include <linux/cdev.h>
#include <linux/device.h>
#include <linux/mutex.h>
#include <linux/gfp.h>
#include <linux/compat.h>
#include <linux/vmalloc.h>
#include <linux/miscdevice.h>
#define NXP_ERR(fmt, args...)    pr_err("[TFA98XX] %s %d : "fmt, __func__, __LINE__, ##args)

#define RED 1
#define GREEN 0
```

```
#define GREEN 0

static int Hello_Misc_open(struct inode *inode, struct file *fp)
{
    printk("neo: Hello_Misc_open \n");
    return 0;
}


static ssize_t Hello_Misc_write(struct file *fp, const char __user *data, size_t count, loff_t *offset)
{
    printk("neo: Hello_Misc_write \n");
    return 0;
}
static ssize_t Hello_Misc_read(struct file *fp, char __user *data, size_t count, loff_t *offset)
{
    printk("neo: Hello_Misc_read\n");
    return 0;
}

static long Hello_Misc_ioctl(struct file *fp,unsigned int cmd, unsigned long arg) //
{
    printk("neo: Hello_Misc_ioctl\n");
    switch(cmd)
    {
        case RED:
            printk("neo: turn on the red light\n");
            break;
        case GREEN:
            printk("neo: turn on the green light\n");
            break;
        default:
            return -1;
    }
    return 0;
}
//手机是64位的，Android.mk 编译的是32 位的，底层需要使用 compat_ioctl ,不是使用unlocked_ioctl
static long Misc_compat_ioctl(struct file *file, unsigned int cmd, unsigned long arg)
{
    printk("neo: Misc_compat_ioctl\n");
    switch(cmd)
    {
        case RED:
            printk("neo: turn on the red light\n");
            break;
        case GREEN:
            printk("neo: turn on the green light\n");
            break;
        default:
            return -1;
    }
    return 0;
}
//用来存储驱动内核模块提供的对设备进行各种操作的函数的指针
static struct file_operations Hello_Misc_fops =
{
    .owner    = THIS_MODULE,
    .open     = Hello_Misc_open,
    .read     = Hello_Misc_read,
    .unlocked_ioctl    = Hello_Misc_ioctl,
```

```
    .write   = Hello_Misc_write,
    .compat_ioctl = Misc_compat_ioctl,
};
//misc设备结构体
static struct miscdevice Hello_Misc =
{
    .minor = MISC_DYNAMIC_MINOR,
    .name = "hello_misc",
    .fops = &Hello_Misc_fops,
};
//入口函数
static int hello_init(void)
{
    dev_t devid;
    int ret = 0;
    printk("hello_init \n");
    /* register MISC device */
    printk("neo: misc_register\n");
    if ((ret = misc_register(&Hello_Misc)))
    {
        NXP_ERR("AudDrv_nxpspk_mod_init misc_register Fail:%d\n", ret);
        return ret;
    }
    return 0;
}

//出口函数
static void hello_exit(void)
{
    // unregister misc 设备
    misc_deregister(&Hello_Misc);
}

module_init(hello_init);
module_exit(hello_exit);
```