misc设备驱动

原创

eurphan_y 于 2020-02-11 21:00:39 发布 336 收藏 1

分类专栏： Linux内核驱动 文章标签： linux

本文链接：https://blog.csdn.net/eurphan_y/article/details/104269806

misc设备驱动

misc 的意思是混合、杂项的,因此 MISC 驱动也叫做杂项驱动,也就是当我们板子上的某些外设无法进行分类的时候就可以使用 MISC 驱动。MISC 驱动其实就是最简单的字符设备驱动,通常嵌套在 platform 总线驱动中,实现复杂的驱动,本章我们就来学习一下 MISC 驱动的编写。

一、MISC 设备驱动简介

所有的 MISC 设备驱动的主设备号都为 10,不同的设备使用不同的从设备号。随着 Linux字符设备驱动的不断增加,设备号变得越来越紧张,尤其是主设备号, MISC 设备驱动就用于解决此问题。MISC 设备会自动创建 cdev,不需要像我们以前那样手动创建,因此采用 MISC 设备驱动可以简化字符设备驱动的编写。我们需要向 Linux 注册一个 miscdevice 设备,miscdevice是一个结构体,定义在文件 include/linux/miscdevice.h 中,内容如下:

```
struct miscdevice
{
 int minor; /* 子设备号 */
 const char *name; /* 设备名字 */
 const struct file_operations *fops;/* 设备操作函数集 */
 struct list_head list;
 struct device *parent;
    struct device *this_device;
    const struct attribute_group **groups;
    const char *nodename;
    umode_t mode;
};
```

定义一个 MISC 设备(miscdevice 类型)以后我们需要设置 minor、name 和 fops 这三个成员变量。minor 表示子设备号,MISC 设备的主设备号为 10,这个是固定的,需要用户指定子设备号;name 就是此 MISC 设备名字,当此设备注册成功以后就会在/dev 目录下生成一个名为 name的设备文件。fops 就是字符设备的操作集合,MISC 设备驱动最终是需要使用用户提供的 fops操作集合。当设置好 miscdevice 以后就需要使用 misc_register 函数向系统中注册一个 MISC 设备,此函数原型如下:

```
int misc_register(struct miscdevice * misc)
```

函数参数和返回值含义如下:

misc: 要注册的 MISC 设备。

返回值: 负数,失败;0,成功。

以前我们需要自己调用一堆的函数去创建设备,比如在以前的字符设备驱动中我们会使用如下几个函数完成设备创建过程:
示例代码, 传统的创建设备过程:

```
alloc_chrdev_region(); /* 申请设备号 */
cdev_init(); /* 初始化 cdev */
cdev_add(); /* 添加 cdev */
class_create(); /* 创建类 */
device_create(); /* 创建设备 */
```

现在我们可以直接使用 **misc_register** 一个函数来完成示例代码 中的这些步骤。当我们卸载设备驱动模块的时候需要调用 **misc_deregister** 函数来注销掉 MISC 设备,函数原型如下:

```
int misc_deregister(struct miscdevice *misc)
```

函数参数和返回值含义如下:
misc:要注销的 MISC 设备。
返回值:负数,失败;0,成功。

以前注销设备驱动的时候,我们需要调用一堆的函数去删除此前创建的 cdev、设备等等内
容,如下所示:
示例代码, 传统的删除设备的过程

```
cdev_del(); /* 删除 cdev */
unregister_chrdev_region(); /* 注销设备号 */
device_destroy(); /* 删除设备 */
class_destroy(); /* 删除类 */
```

现在我们只需要一个 **misc_deregister** 函数即可完成示例代码 中的这些工作。关于MISC 设备驱动就讲解到这里。

二、misc驱动示例

```
#include <linux/module.h>
#include <linux/slab.h>
#include <linux/errno.h>
#include <linux/init.h>
#include <linux/cdev.h>
#include <linux/kernel.h>
#include <linux/types.h>
#include <linux/ide.h>
#include <linux/device.h>
#include <asm/mach/map.h>
#include <asm/uaccess.h>
#include <asm/io.h>
#include <linux/miscdevice.h>

#define MISC_DEVICE_NAME "misc_test"
#define MISC_DEVICE_MINOR 144

static int misc_open(struct inode *inode, struct file *filep)
{
 return 0;
}

static ssize_t misc_read(struct file *filp, char __user *buff, size_t cnt, loff_t *offt)
{
 return 0;
}

static ssize_t misc_write(struct file *filp, char __user *buff, size_t cnt, loff_t *offt)
```

```
static ssize_t misc_write(struct file *filp, char __user *buff, size_t cnt, loff_t *offt)
{
 return 0;
}

static int misc_release(struct inode *inode, struct file *filep)
{
 return 0;
}

/* 操作函数结构体 */
static struct file_operations misc_fops = {
 .owner = THIS_MODULE,
 .open = misc_open,
 .read = misc_read,
 .write = misc_write,
 .release = misc_release,
};

/* misc设备结构体 */
static struct miscdevice miscdevice_test = {
 .minor = MISC_DEVICE_MINOR,
 .name = MISC_DEVICE_NAME,
 .fops = &misc_fops,
};

static __init int misc_init(void)
{
 int ret = misc_register(&miscdevice_test);
 if (ret < 0)
  return -EFAULT;
 return 0;
}

static __exit void misc_exit(void)
{
 misc_deregister(&miscdevice_test);
}

module_init(misc_init);
module_exit(misc_exit);

MODULE_AUTHOR("eurphan<eurphan@163.com>");
MODULE_DESCRIPTION("Misc Driver");
MODULE_LICENSE("GPL");
```