

linux misc设备

转载

冰瓜瓜 于 2019-07-23 14:55:24 发布 685 收藏 1
分类专栏: [linux](#)



[linux](#) 专栏收录该内容

17 篇文章 1 订阅
订阅专栏

Linux驱动程序--misc设备驱动

(2017-07-17 10:54:28)

转载 ▼

分类: [Linux驱动](#)

1. 原由:

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。

2. 解决方案:

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,不需要像cdev设备那样,需要自己手动创建,所以使用起来更为方便。

3. 底层支持:

(1) 两个数据结构:

```
struct file_operations dev_fops = {  
.open = test_open,  
.release = test_close,  
.write = test_write,  
.read = test_read,  
};
```

文件操作结构,这一步可cdev是一样的,都是底层文件操作。

```
#define DEVICE_NAME "misc_dev"  
struct miscdevice misc_dev={  
.minor = MISC_DYNAMIC_MINOR,  
.name = DEVICE_NAME,  
.fops = &dev_fops,  
};
```

misc设备结构，主要包含次设备号，设备名和文件操作结构指针。设备名最终会出现在/dev下，生成设备文件misc_dev。在/sys/devices/virtual/misc/下生成设备目录，里面包含设备属性文件。在/sys/class/misc/下生成符号链接文件misc_dev指向/sys/devices/virtual/misc/misc_dev目录。

(2) 函数接口：

int misc_register(struct miscdevice * misc) 完成misc设备注册。

int misc_deregister(struct miscdevice *misc) 完成misc设备注销。

4. 说明：

(1) misc设备完成的是用户层到底层文件操作函数的映射，底层操作函数和具体设备的映射需要驱动自己完成，可以使用设备地址或者结合其他总线接口，例如i2c的ifly设备，可以在底层函数中，直接调用i2c相关接口函数实现。

(2) misc设备被实现为一个子系统，在子系统初始化阶段调用misc_init，设置了主设备号以及misc设备的文件操作。此时的misc设备文件操作和具体设备文件操作不同，这里的open函数需要根据不同的此设备号选择相应具体设备的底层文件操作。

(3) 在misc设备注册时，系统自动创建设备文件/dev/misc_dev，这需要Linux设备模型以及udev的支持。