Linux中MISC驱动简介及其简单使用

原创

一只青木呀 于 2021-07-10 17:22:36 发布 293 收藏 1

分类专栏： Linux 驱动 文章标签： linux MISC 驱动

本文链接：https://blog.csdn.net/weixin_45309916/article/details/118636702


Linux 同时被 2 个专栏收录

132 篇文章 13 订阅

订阅专栏


驱动

26 篇文章 6 订阅

订阅专栏

Linux中MISC驱动简介及其简单使用

Linux里面的misc杂项设备是主设备号为10的驱动设备，它的注册跟使用比较的简单，所以比较适用于功能简单的设备。正因为简单，所以它通常嵌套在platform 总线驱动中，配合总线驱动达到更复杂，多功能的效果。

1、MISC 设备驱动简介

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。他可以夹杂到其它结构体当中，以丰富驱动的血肉。一般情况下，我们是将它嵌套在其它结构当中的。

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;    /* 设备操作集 */
 struct list_head list;
 struct device *parent;
 struct device *this_device;
 const struct attribute_group **groups;
 const char *nodename;
 umode_t mode;
};
```

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。

minor：次设备号

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。

parent:这个指针决定了在/sys文件系统里面，它是创建在哪个目录下。如果为空就在/sys/class根目录下创建，如果不为空都是在/sys/class/misc 文件下面创建的一些属性文件。

this_device：这个就代表当前设备的设备结构体，这个在查找扩充数据结构时，非常有用。

2、一般操作流程

1、定义以及填充结构体

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， Linux 系统已经预定义了一些 MISC 设备的子设备号，这些预定义的子设备号定义在include/linux/miscdevice.h 文件中，如下所示：

```
#define PSMOUSE_MINOR  1
#define MS_BUSMOUSE_MINOR 2 /* unused */
#define ATIXL_BUSMOUSE_MINOR 3 /* unused */
/*#define AMIGAMOUSE_MINOR 4 FIXME OBSOLETE */
#define ATARIMOUSE_MINOR 5 /* unused */
#define SUN_MOUSE_MINOR  6 /* unused */
#define APOLLO_MOUSE_MINOR 7 /* unused */
#define PC110PAD_MINOR  9 /* unused */
/*#define ADB_MOUSE_MINOR 10 FIXME OBSOLETE */
#define WATCHDOG_MINOR  130 /* Watchdog timer     */
#define TEMP_MINOR  131 /* Temperature Sensor */
#define RTC_MINOR  135
#define EFI_RTC_MINOR  136 /* EFI Time services */
#define VHCI_MINOR  137
#define SUN_OPENPROM_MINOR 139
#define DMAPI_MINOR  140 /* unused */
#define NVRAM_MINOR  144
#define SGI_MMTIMER  153
#define STORE_QUEUE_MINOR 155 /* unused */
#define I2O_MINOR  166
#define MICROCODE_MINOR  184
#define VFIO_MINOR  196
#define TUN_MINOR  200
#define CUSE_MINOR  203
#define MWAVE_MINOR  219 /* ACP/Mwave Modem */
#define MPT_MINOR  220
#define MPT2SAS_MINOR  221
#define MPT3SAS_MINOR  222
#define UINPUT_MINOR  223
#define MISC_MCELOG_MINOR 227
#define HPET_MINOR  228
#define FUSE_MINOR  229
#define KVM_MINOR  232
#define BTRFS_MINOR  234
#define AUTOFS_MINOR  235
#define MAPPER_CTRL_MINOR 236
#define LOOP_CTRL_MINOR  237
#define VHOST_NET_MINOR  238
#define UHID_MINOR  239
#define MISC_DYNAMIC_MINOR 255
```

我们在使用的时候可以从这些预定义的子设备号中挑选一个，当然也可以自己定义，只要这个子设备号没有被其他设备使用接口。

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。

2、注册MISC 设备

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。

此函数可以省去一堆的字符设备注册的过程，如下：

```
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 */
```

3、卸载MISC 设备

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

此函数可以省去一堆的字符设备注销的过程，如下：

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