迅为电子Linux驱动教程
——生成设备节点

北京迅为电子有限公司

本节目标

- 申明

- 杂项设备（生成设备节点就可以在/dev/ 下看到相应misc_register生成的名字）

- 生成杂项设备的设备节点

申明

- Linux到2.6版本的时候，改动巨大，网上有些资料是针对以前的版本的，大家看到老版本相关的资料，直接跳过即可。

- 现在2.6版本以前的基本都废弃了，不用管了，学了也没有。学习要“以始为终”，学了之后是为了在实际工作中应用的，那么工作中已经用不到的知识就不要浪费时间了。

杂项设备
——为什么用杂项设备

- 杂项设备可以说是对一部分字符设备的封装，还有一部分不好归类驱动也归到杂项设备。
- **为什么会引入杂项设备？**
- 第一、节省主设备号
  - 如果所有的驱动都是用字符设备，那么所有的设备号很快就用完了，总共就255个主设备号。
- 第二、驱动写起来相对简单
  - 如果直接使用封装好的杂项设备，那么就可以减少一步注册主设备号的过程

杂项设备
——杂项设备初始化文件

- 杂项设备初始化部分源文件“drivers/char/ misc.c”，这一部分通过Makefile可知，是强制编译的。而且是Linux官方（不是三星官方）出来的时候就带的，为了一些简单的驱动更容易实现。

- 这部分了解即可，里面的内容也比较简单，就是给字符驱动做一个简单的封装

杂项设备
——注册文件

- 杂项设备注册头文件
  - include/linux/miscdevice.h

- 结构体miscdevice以及注册函数如下所示

```
struct miscdevice  {
        int minor;
        const char *name;
        const struct file_operations *fops;
        struct list_head list;
        struct device *parent;
        struct device *this_device;
        const char *nodename;
        mode_t mode;
};

extern int misc_register(struct miscdevice * misc);
extern int misc_deregister(struct miscdevice *misc);
```

杂项设备

——注册文件

- 常用的参数
  - .minor设备号
  - .name生成设备节点的名称（设备节点名称可以不跟device和driver的相同，但是device和driver的名字相同才能匹配上）
  - .fops指向一个设备节点文件

杂项设备
——内核文件的结构体

- Linux中一切皆文件，上层调用底层也是通过读取文件的方式
  - 注册设备节点，本质也是新建一个特殊的文件，包含文件名，打开、关闭、操作等函数
- 包含文件结构体的头文件是“include/linux/fs.h ”
- 文件的结构体file_operations如下所示

```
*/
struct file_operations {
        struct module *owner;
        loff_t (*llseek) (struct file *, loff_t, int);
        ssize_t (*read) (struct file *, char __user *, size_t, loff_t *);
        ssize_t (*write) (struct file *, const char __user *, size_t, loff_t *);
        ssize_t (*aio_read) (struct kiocb *, const struct iovec *, unsigned long
, loff_t);
        ssize_t (*aio_write) (struct kiocb *, const struct iovec *, unsigned lon
g, loff_t);
        int (*readdir) (struct file *, void *, filldir_t);
        unsigned int (*poll) (struct file *, struct poll_table_struct *);
```

杂项设备
——内核文件的结构体

- 文件的结构体file_operations参数很多，根据需求选择。
- 必选的是参数是
  - .owner一般是THIS_MODULE,
  - .open打开文件函数
  - .release关闭文件函数
- 这里在必选之外使用参数（为了介绍接下来的GPIO的操作）
  - .unlocked_ioctl对GPIO的操作，应用向底层驱动传值

杂项设备

- 驱动代码，在probe_linux_module基础上写devicenode_linux_module驱动
  - 写代码的时候，注意一下函数调用顺序
- 编译，在开发板上加载驱动生成设备节点
  - 在/dev中查看是否生成了设备节点

谢谢！