驱动框架分析：

1. 2.4 内核中的理解：

①，确定主设备号。

②，file\_operations 结构

③，register\_chrdev(主设备号，名字，file\_operations 结构）；

④，入口函数。

⑤，出口函数。

chrdevs 数组，以“主设备号”为下标的“file\_operations”结构数组。

2. 2.6 内核这样理解：

①，chrdevs数组只用255，则一个内核只能支持255 个字符设备驱动。以前2.4 内核确实是有这样的缺点。

但2.6内核中很多书上建议不再用“register\_chrdev()”了。

②，以前open一个字符设备时，虚拟文件系统(VFS)层，有sys\_open,以前是以主设备号为下标，在chrdevs数中找到以前注册的file\_oprerations结构体，现在变化了。是以“主设备号”和“次设备号”两个作为一个整体来找到“file\_operations”结构体。

分析“drivers/rtc/rtc-s3c.c”：

int register\_chrdev(unsigned int major, const char \*name,const struct file\_operations \*fops)

-->cdev = cdev\_alloc();

搜索cdev\_alloc()，可以见到别人的展开，在“scx200\_gpio.c”

int \_\_init scx200\_gpio\_init(void)

-->rc = register\_chrdev\_region(devid, MAX\_PINS, "scx200\_gpio");

参1，参2的意思：从哪里开始，共有多少个。

-->cd = \_\_register\_chrdev\_region(MAJOR(n), MINOR(n),next - n, name);

-->rc = alloc\_chrdev\_region(&devid, 0, MAX\_PINS, "scx200\_gpio");

-->cdev\_init(&scx200\_gpio\_cdev, &scx200\_gpio\_fileops);

-->cdev\_add(&scx200\_gpio\_cdev, devid, MAX\_PINS);

③，2.6内核中对注册字符设备的扩展：对"register\_chrdev"的拆分。

❶，若确定了“主设备号”时用“register\_chrdev\_region()”.没有确定主设备号时用“alloc\_chrdev\_region()”。区域是指从（某个主设备号、某个次设备号）~（某主设备号，某次设备号+n）都对应于这个file\_operations结构体。而“register\_chrdev()”是从“主设备号0”到“主设备号255”都对应“file\_operations"结构体。

❷，cdev\_init（）;

❸，cdev\_add（）;

以上❶~❸步，在"register\_chrdev()"中可以见到此过程：

int register\_chrdev(unsigned int major, const char \*name, const struct file\_operations \*fops)

-->cd = \_\_register\_chrdev\_region(major, 0, 256, name);

-->if (major == 0)

-->cdev = cdev\_alloc();

-->开始设置

cdev->owner = fops->owner;

cdev->ops = fops;

kobject\_set\_name(&cdev->kobj, "%s", name);

-->err = cdev\_add(cdev, MKDEV(cd->major, 0), 256);

int register\_chrdev\_region(dev\_t from, unsigned count, const char \*name)

有主设备号时，有多少个次设备号的范围的字符设备都对些相同“file\_operations”结构。

参1，从哪里开始。

参2，有多少个。

参3，名字。

int alloc\_chrdev\_region(dev\_t \*dev, unsigned baseminor, unsigned count,const char \*name)

若没有分配主设备号时，先分配一个主设备号放到“&dev”中，

参1，存放主设备号。

参2，次设备号的基地址。

参3，个数。

参4，名字。

若“devid = MKDEV (major, 0);”时，register\_chrdev\_region (devid, 2, "hello");是指“(major,0~1)对应'hello\_fops'，(major,2~255)不对应helle\_fops”。“register\_chrdev\_region()”参2是指在有主设备号的情况下，这里有2个次设备号（major,0~1）对应于这个“file\_operations结构体--hello\_fops”。

若“devid = MKDEV (major, 1);”时，register\_chrdev\_region (devid, 2, "hello");是指“(major,1~2)对应'hello\_fops'，(major,3~255)不对应helle\_fops”。

编译测试：



这是缺少头文件：#include <linux/cdev.h>

测试如下：

insmod new\_chrdriver.ko

# ls /dev/hello\* -l

crw-rw---- 1 0 0 252, 0 Jan 1 00:04 /dev/hello0

crw-rw---- 1 0 0 252, 1 Jan 1 00:04 /dev/hello1

crw-rw---- 1 0 0 252, 2 Jan 1 00:04 /dev/hello2

# ./newchar hello0

can't open hello0

# ./newchar /dev/hello0

hello word

can open /dev/hello0

# ./newchar /dev/hello1

hello word

can open /dev/hello1

# ./newchar /dev/hello2

can't open /dev/hello2

我们发现hello2就打不开了。

继续实验加上测试代码：

测试：

cat /proc/devices

发现主设备号是一样

252 hello

252 hello2

因为我们注册了两个

register\_chrdev\_region(devid, HELLO\_CNT, "hello");

register\_chrdev\_region(devid, 1, "hello2");

# ./newchar /dev/hello2

hello2\_open

can open /dev/hello2

# ./newchar /dev/hello3

can't open /dev/hello3

#