每创建一个文件，都会对应唯一的一个**inode**结构体

```
mknod  /dev/hello  c  252  0
```

```
crw------- 1 root root 252, 0  2010-02-04 08:50  /dev/hello
```

用户空间

应用程序：

```
int main(int argc, char *argv[])
{
        ……
        fd = open(“/dev/hello”, O_RDWR);
        ……
}
```

系统调用：

```
sys_open(const char *, int  , int , )
```

VFS层

```
static int chrdev_open(struct inode *inode, struct file *filp)
```

内核空间

inode结构体用于描述文件的静态属性

```
include/linux/fs.h
struct inode {
        ……
        dev_t  i_rdev;     //包含真正的设备编号
        struct  cdev  *i_cdev;   //指向cdev结构的指针
        ……
};
```

通过设备编号找到驱动程序

将注册在驱动里的**file_operations**赋给**file**结构体里的**f_ops**

定义了针对文件的一系列操作方法（包含了面向对象的思想）：

```
include/linux/fs.h
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 *);
        unsigned int (*poll) (struct file *, struct poll_table_struct *);
        int (*ioctl) (struct inode *, struct file *, unsigned int, unsigned long);
        int (*mmap) (struct file *, struct vm_area_struct *);
        int (*open) (struct inode *, struct file *);
        int (*release) (struct inode *, struct file *);
        ……
};
```

系统中每个打开的文件在内核空间都有一个对应的**file**结构体

```
include/linux/fs
struct file {
        mode_t  f_mode;
        loff_t   f_ops;
        unsigned int  f_flags;
        struct  file_operations  *f_op;
        void  *private_data;
        struct  dentry   *f_dentry;
};
```

内核使用**cdev**结构体来表示字符设备：

```
include/linux/cdev.h
struct cdev {
        struct kobject kobj;
        struct module *owner;
        const struct file_operations *ops;
        struct list_head list;
        dev_t dev;
        unsigned int count;
};
```

令**i_cdev**指向找到的**cdev**

驱动程序：

描述设备的结构体，封装了**struct cdev**

```
struct hello_dev
{
        wait_queue_head_t  head;
        char *buffer;
        struct semaphore sem;
        struct cdev *cdev;
};
```

定义在设备文件之上的操作方法

```
struct file_operations  hello_fops
{
        .open = hello_open,
        .read = hello_read,
        .write = hello_write,
        .ioctl = hello_ioctl,
        .release = hello_close,
};
```

Kernel

将一个**file_operations**和驱动程序绑定：

```
void cdev_init(struct cdev *cdev, const struct file_operations *fops)
```

```
int cdev_add(struct cdev *p, dev_t dev, unsigned count)
```

将字符驱动注册至内核