http://blog.csdn.net/bonnshore/article/details/7860997

Linux中I/O设备分为两类：块设备和字符设备。两种设备本身没有严格限制，但是，基于不同的功能进行了分类。

(1) 字符设备：提供连续的数据流，应用程序可以顺序读取，通常不支持随机存取。相反，此类设备支持按字节/字符来读写数据。举例来说，调制解调器是典型的字符设备。

(2) 块设备：应用程序可以随机访问设备数据，程序可自行确定读取数据的位置。硬盘是典型的块设备，应用程序可以寻址磁盘上的任何位置，并由此读取数据。此外，数据的读写只能以块(通常是512B)的倍数进行。与字符设备不同，块设备并不支持基于字符的寻址。

两种设备本身并没用严格的区分，主要是字符设备和块设备驱动程序提供的访问接口（file I/O API）是不一样的。本文主要就数据接口、访问接口和设备注册方法对两种设备进行比较。

1、[**数据结构**](http://lib.csdn.net/base/31)

**1.1字符设备数据结构**

struct file;   
        struct inode;

file定义于 <linux/fs.h>, 是设备驱动中第二个最重要的数据结构. 文件结构代表一个打开的文件. 它由内核在 open 时创建, 并传递给在文件上操作的任何函数, 直到最后的关闭. 在文件的所有实例都关闭后, 内核释放这个数据结构。

inode 结构由内核在内部用来表示文件.inode 结构包含大量关于文件的信息其中dev\_t i\_rdev成员包含实际的设备编号.struct cdev \*i\_cdev中struct cdev 是内核的内部结构, 代表字符设备。

**1.2块设备数据结构**

struct gendisk (定义于 <linux/genhd.h>) 是单独一个磁盘驱动器的内核表示. 事实上, 内核还使用 gendisk 来表示分区。

2、设备访问接口

**2.1字符设备访问接口**

struct file\_operations 其中file\_operation 结构中的每个成员必须指向驱动中的函数, 这些函数实现一个特别的操作, 或者对于不支持的操作留置为 NULL. 当指定为 NULL 指针时内核的确切的行为是每个函数不同的，该结构中主要函数如下：

ssize\_t read(struct file \*filp, char \_\_user \*buff, size\_t count, loff\_t \*offp);  
        ssize\_t write(struct file \*filp, const char \_\_user \*buff, size\_t count, loff\_t \*offp);

filp 是文件指针, count 是请求的传输数据大小. buff 参数指向持有被写入数据的缓存, 或者放入新数据的空缓存. 最后, offp 是一个指针指向一个"long offset type"对象, 它指出用户正在存取的文件位置. 返回值是一个"signed size type"。

**2.2块设备访问接口**

字符设备通过 file\_ 操作结构使它们的操作对系统可用. 一个类似的结构用在块设备上; 它是 struct block\_device\_operations, 定义在 <linux/fs.h>，其主要操作方法如下：

int (\*open)(struct inode \*inode, struct file \*filp);   
        int (\*release)(struct inode \*inode, struct file \*filp);

就像它们的字符驱动对等体一样工作的函数; 无论何时设备被打开和关闭都调用它们. 一个字符驱动可能通过启动设备或者锁住门(为可移出的介质)来响应一个 open 调用. 如果你将介质锁入设备, 你当然应当在 release 方法中解锁。

int (\*ioctl)(struct inode \*inode, struct file \*filp, unsigned int cmd, unsigned long arg);

实现 ioctl 系统调用的方法. 但是, 块层首先解释大量的标准请求; 因此大部分的块驱动 ioctl 方法相当短。

3、设备注册

**3.1字符设备注册**

int register\_chrdev\_region(dev\_t first, unsigned int count, char \*name)   
        int alloc\_chrdev\_region(dev\_t \*dev, unsigned int firstminor, unsigned int count, char \*name)   
        void unregister\_chrdev\_region(dev\_t first, unsigned int count);

允许驱动分配和释放设备编号的范围的函数. register\_chrdev\_region 应当用在事先知道需要的主编号时; 对于动态分配, 使用 alloc\_chrdev\_region 代替.

**3.2块设备注册**

int register\_blkdev(unsigned int major, const char \*name);   
        int unregister\_blkdev(unsigned int major, const char \*name);

register\_blkdev 注册一个块驱动到内核, 并且, 可选地, 获得一个主编号. 一个驱动可被注销, 使用 unregister\_blkdev。