外，如果每次只请求读写一块，那么它的I/O性能很难提高。一个SCATTERED_IO请求允许文件系统连续读写多个块。对于 READ 操作来说，要求多读的块可能并不是由调用它的进程提出的，而是由系统预测的将来的数据请求；当写数据时，则不存在上述问题。但系统可能会暂存许多写请求随后一次写出，这显然比每次处理一个请求效率高。在SCATTERED_IO请求中，申请读写块的请求被排序，这样比随机处理效率高。而且调用一次驱动程序传输很多块，减少了调用时传递的参数数量的同时也减少了调用中断处理程序处理外设中断的次数。

3.1.2 公用块设备驱动程序软件

我们将所有块设备都需要的定义Device.h中，其中最重要就是下面所列出来的数据结构——设备驱动程序接口。它保存了各种驱动程序执行具体I/O操作函数的地址。

```
struct driver {
    char *(*dr_name)(void);
    int (*dr_open)(struct driver *dp);
    int (*dr_close)(struct driver *dp);
    int (*dr_ioctl)(struct driver *dp);
    struct device *(*dr_prepare)(int device);
    int (*dr_schedule)(struct driver *dp);
    void (*dr_rdwt)(void);
    void (*dr_geometry)(struct partition *entry);
};
```

以上的数据结构有如下几点需要说明：

（1） 该设备驱动程序接口是所有块设备驱动程序公用的。但不是所有设备都要提供全部函数，比如 RAM Disk 就不需要 dr_ioctl，它们和§3.1.1中提到的设备的 6 种操作也没有必然的一一对应的关系。

（2） 调用具体设备的驱动程序要在设备安装之后，采用间接调用方式，如：

```
code = （harddisk->dr_finish）();
```

有的设备操作非常类似，如 RAM Disk 的 CLOSE 和空设备的 CLOSE，可以共享同一段函数代码，基于此种目的，我们为每个函数设计了缺省实现，供具体的设备调用。

（3） 各个函数的作用：

- dr_name：返回设备的名称，缺省实现中返回一个空字符串。
- dr_open：视设备的不同，dr_open 执行的功能可以是安装设备、初始化设备、打开设备或者验证设备是否可用，当以上过程发生错误时返回一条错误信息。
- dr_close：释放设备。
- dr_ioctl：检测和改变设备的分区、切换工作模式等工作。
- dr_schedule：该函数入口为不同的设备提供不同的扩展功能，例如：硬盘驱动程序中，用于计算柱面号、扇区号和磁头号等参数。部分设备没有该函数的实现，但是这是一个公用的设备驱动程序入口，要提供所有支持的外设操作的并集。
- dr_prepare：初始化硬件 I/O 参数。
- dr_rdwt：执行硬件 I/O。
- dr_geometry：返回驱动器的物理特性。如硬盘的磁头数、扇区数等。对于有的设备没有这些物理特性，如 RAM Disk，为了接口的统一性以及其他程序处理方便，我们返回一个伪物理参数。

3.2 典型块设备驱动程序的设计与实现

3.2.1 磁盘

传统意义上的磁盘，包含硬盘和软盘两种。两者在很多方面都非常相似，但是由于软盘具有比硬盘更简单的控制器和移动介质两大特点，使得软盘驱动程序的实现比硬盘复杂。为了突出我们的设计与实现的重点，避免纠缠过多的硬件特征，在本文以后的部分，特将硬盘作为描述对象，除非特殊说明，所提到的磁盘均是指硬盘。

对于每一类总线的系统板都有不同类型的 I/O 适配器，系统板为这些适配器