第三章 块设备驱动程序

3.1 块设备驱动程序接口

3.1.1 块设备驱动程序概述

在这一章中，我们将要从块设备的共性开始讨论块设备驱动程序。块设备驱动程序接口是所有块设备共用的，它的作用是屏蔽硬件的具体细节向它的上一层——高速缓存管理程序提供统一的设备驱动接口。

所有的设备都可能有6种操作：

（1） OPEN

（2） CLOSE

（3） READ

（4） WRITE

（5） IOCTL

（6） SCATTERED_IO

其中，READ是从设备读进一个数据块放到内存缓冲区；WRITE正好相反。调用READ的进程将一直阻塞直到数据传输完成；而对于WRITE而言，我们将它放在数据缓冲区中，随后再将其真正传送到设备（系统的关键数据，如FAT表，超级块等不采用这种延迟写策略），这时WRITE系统调用就能很快返回，这样不仅能提高系统I/O的效率，还能在任务反复对同一个缓冲区I/O时提高任务的执行效率。

OPEN用来验证设备是否可用，当不可用时返回一条错误信息。CLOSE将确保延迟写的数据真正被写到设备上。

许多设备都有一些I/O参数，这些参数经常会被检查，还有可能被修改。IOCTL就是完成这一系列工作的。以硬盘为例，IOCTL用于完成检测和改变设备的分区、切换工作模式等工作。

SCATTERED_IO操作是最特殊的。除了个别非常快的块设备（如RAMDisk）

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）();
```