第十六章 储存媒介

In previous chapters we’ve looked at manipulating data at the file level. In this chapter, we will consider data at the device level. Linux has amazing capabilities for handling storage devices, whether physical storage, such as hard disks, or network storage, or virtual storage devices like RAID (Redundant Array of Independent Disks) and LVM (Logical Volume Manager).

在前面章节中，我们已经从文件级别看了操作数据。在这章里，我们将从设备级别来考虑数据。 Linux 有着令人惊奇的能力来处理存储设备，不管是物理设备，比如说硬盘，还是网络设备，或者是 虚拟存储设备，像 RAID（独立磁盘冗余阵列)和 LVM（逻辑卷管理器）。

However, since this is not a book about system administration, we will not try to cover this entire topic in depth. What we will try to do is introduce some of the concepts and key commands that are used to manage storage devices.

然而，这不是一本关于系统管理的书籍，我们不会试图深入地覆盖整个主题。我们将努力做的就是 介绍一些概念和用来管理存储设备的重要命令。

To carry out the exercises in this chapter, we will use a USB flash drive, a CD-RW disk (for systems equipped with a CD-ROM burner) and a floppy disk (again, if the system is so equipped.)

我们将会使用 USB 闪存，CD-RW 光盘（因为系统配备了 CD-ROM 烧写器）和一张软盘（若系统这样配备）， 来做这章的练习题。

We will look at the following commands:

我们将看看以下命令：

* mount – Mount a file system
* mount – 挂载一个文件系统
* umount – Unmount a file system
* umount – 卸载一个文件系统
* fsck – Check and repair a file system
* fsck – 检查和修复一个文件系统
* fdisk – Partition table manipulator
* fdisk – 分区表控制器
* mkfs – Create a file system
* mkfs – 创建文件系统
* fdformat – Format a floppy disk
* fdformat – 格式化一张软盘
* dd – Write block oriented data directly to a device
* dd — 把面向块的数据直接写入设备
* genisoimage (mkisofs) – Create an ISO 9660 image file
* genisoimage (mkisofs) – 创建一个 ISO 9660的映像文件
* wodim (cdrecord) – Write data to optical storage media
* wodim (cdrecord) – 把数据写入光存储媒介
* md5sum – Calculate an MD5 checksum
* md5sum – 计算 MD5检验码

**挂载和卸载存储设备**

Recent advances in the Linux desktop have made storage device management extremely easy for desktop users. For the most part, we attach a device to our system and it “just works.” Back in the old days (say, 2004), this stuff had to be done manually. On non- desktop systems (i.e., servers) this is still a largely manual procedure since servers often have extreme storage needs and complex configuration requirements.

Linux 桌面系统的最新进展已经使存储设备管理对于桌面用户来说极其容易。大多数情况下，我们 只要把设备连接到系统中，它就能工作。在过去（比如说，2004年），这个工作必须手动完成。 在非桌面系统中（例如，服务器中），这仍然是一个主要地手动过程，因为服务器经常有极端的存储需求 和复杂的配置要求。

The first step in managing a storage device is attaching the device to the file system tree. This process, called mounting, allows the device to participate with the operating system. As we recall from Chapter 3, Unix-like operating systems, like Linux, maintain a single file system tree with devices attached at various points. This contrasts with other operating systems such as MS-DOS and Windows that maintain separate trees for each device (for example C:\, D:\, etc.).

管理存储设备的第一步是把设备连接到文件系统树中。这个过程叫做挂载，允许设备参与到操作系统中。 回想一下第三章，类 Unix 的操作系统，像 Linux，维护单一文件系统树，设备连接到各个结点上。 这与其它操作系统形成对照，比如说 MS-DOS 和 Windows 系统中，每个设备（例如 C:\，D:\，等） 保持着单独的文件系统树。

There is a file named /etc/fstab that lists the devices (typically hard disk partitions) that are to be mounted at boot time. Here is an example /etc/fstab file from a Fedora 7 system:

有一个叫做/etc/fstab 的文件可以列出系统启动时要挂载的设备（典型地，硬盘分区）。下面是 来自于 Fedora 7系统的/etc/fstab 文件实例：

LABEL=/12 / ext3 defaults 1 1

LABEL=/home /home ext3 defaults 1 2

LABEL=/boot /boot ext3 defaults 1 2

tmpfs /dev/shm tmpfs defaults 0 0

devpts /dev/pts devpts gid=5,mode=620 0 0

sysfs /sys sysfs defaults 0 0

proc /proc proc defaults 0 0

LABEL=SWAP-sda3 /swap swap defaults 0 0

Most of the file systems listed in this example file are virtual and are not applicable to our discussion. For our purposes, the interesting ones are the first three:

在这个实例中所列出的大多数文件系统是虚拟的，并不适用于我们的讨论。就我们的目的而言， 前三个是我们感兴趣的：

LABEL=/12 / ext3 defaults 1 1

LABEL=/home /home ext3 defaults 1 2

LABEL=/boot /boot ext3 defaults 1 2

These are the hard disk partitions. Each line of the file consists of six fields, as follows:

这些是硬盘分区。每行由六个字段组成，如下所示：

|  |  |  |
| --- | --- | --- |
| *Table 16-1: /etc/fstab Fields* | | |
| Field | Contents | Description |
| 1 | Device | Traditionally, this field contains the actual name of a device file associated with the physical device, such as /dev/hda1 (the first partition of the master device on the first IDE channel). But with today's computers, which have many devices that are hot pluggable (like USB drives), many modern Linux distributions associate a device with a text label instead. This label (which is added to the storage media when it is formatted) is read by the operating system when the device is attached to the system. That way, no matter which device file is assigned to the actual physical device, it can still be correctly identified. |
| 2 | Mount Point | The directory where the device is attached to the file system tree. |
| 3 | File System Type | Linux allows many file system types to be mounted. Most native Linux file systems are ext3, but many others are supported, such as FAT16 (msdos), FAT32 (vfat), NTFS (ntfs), CD-ROM (iso9660), etc. |
| 4 | Options | File systems can be mounted with various options. It is possible, for example, to mount file systems as read-only, or prevent any programs from being executed from them (a useful security feature for removable media.) |
| 5 | Frequency | A single number that specifies if and when a file system is to be backed up with the dump command. |
| 6 | Order | A single number that specifies in what order file systems should be checked with the fsck command. |

|  |  |  |
| --- | --- | --- |
| *表16-1: /etc/fstab 字段* | | |
| 字段 | 内容 | 说明 |
| 1 | 设备名 | 传统上，这个字段包含与物理设备相关联的设备文件的实际名字，比如说/dev/hda1（第一个 IDE 通道上第一个主设备分区）。然而今天的计算机，有很多热插拔设备（像 USB 驱动设备），许多 现代的 Linux 发行版用一个文本标签和设备相关联。当这个设备连接到系统中时， 这个标签（当储存媒介格式化时，这个标签会被添加到存储媒介中）会被操作系统读取。 那样的话，不管赋给实际物理设备哪个设备文件，这个设备仍然能被系统正确地识别。 |
| 2 | 挂载点 | 设备所连接到的文件系统树的目录。 |
| 3 | 文件系统类型 | Linux 允许挂载许多文件系统类型。大多数本地的 Linux 文件系统是 ext3， 但是也支持很多其它的，比方说 FAT16 (msdos), FAT32 (vfat)，NTFS (ntfs)，CD-ROM (iso9660)，等等。 |
| 4 | 选项 | 文件系统可以通过各种各样的选项来挂载。有可能，例如，挂载只读的文件系统， 或者挂载阻止执行任何程序的文件系统（一个有用的安全特性，避免删除媒介。） |
| 5 | 频率 | 一位数字，指定是否和在什么时间用 dump 命令来备份一个文件系统。 |
| 6 | 次序 | 一位数字，指定 fsck 命令按照什么次序来检查文件系统。 |

**查看挂载的文件系统列表**

The mount command is used to mount file systems. Entering the command without arguments will display a list of the file systems currently mounted:

这个 mount 命令被用来挂载文件系统。执行这个不带参数的命令，将会显示 一系列当前挂载的文件系统：

[me@linuxbox ~]$ mount

/dev/sda2 on / type ext3 (rw)

proc on /proc type proc (rw)

sysfs on /sys type sysfs (rw)

devpts on /dev/pts type devpts (rw,gid=5,mode=620)

/dev/sda5 on /home type ext3 (rw)

/dev/sda1 on /boot type ext3 (rw)

tmpfs on /dev/shm type tmpfs (rw)

none on /proc/sys/fs/binfmt\_misc type binfmt\_misc (rw)

sunrpc on /var/lib/nfs/rpc\_pipefs type rpc\_pipefs (rw)

fusectl on /sys/fs/fuse/connections type fusectl (rw)

/dev/sdd1 on /media/disk type vfat (rw,nosuid,nodev,noatime,

uhelper=hal,uid=500,utf8,shortname=lower)

twin4:/musicbox on /misc/musicbox type nfs4 (rw,addr=192.168.1.4)

The format of the listing is: device on mount\_point type file\_system\_type (options). For example, the first line shows that device /dev/sda2 is mounted as the root file system and it is of type ext3 and is both readable and writable (the option “rw”). This listing also has two interesting entries at the bottom of the list. The next to last entry shows a 2 gigabyte SD memory card in a card reader mounted at /media/disk, and the last entry is a network drive mounted at /misc/musicbox.

这个列表的格式是：设备 on 挂载点 type 文件系统类型（可选的）。例如，第一行所示设备/dev/sda2 作为根文件系统被挂载，文件系统类型是 ext3，并且可读可写（这个“rw”选项）。在这个列表的底部有 两个有趣的条目。倒数第二行显示了在读卡器中的一张2G 的 SD 内存卡，挂载到了/media/disk 上。最后一行 是一个网络设备，挂载到了/misc/musicbox 上。

For our first experiment, we will work with a CD-ROM. First, let’s look at a system before a CD-ROM is inserted:

第一次实验，我们将使用一张 CD-ROM。首先，在插入 CD-ROW 之前，我们将看一下系统：

[me@linuxbox ~]$ mount

/dev/mapper/VolGroup00-LogVol00 on / type ext3 (rw)

proc on /proc type proc (rw)

sysfs on /sys type sysfs (rw)

devpts on /dev/pts type devpts (rw,gid=5,mode=620)

/dev/hda1 on /boot type ext3 (rw)

tmpfs on /dev/shm type tmpfs (rw)

none on /proc/sys/fs/binfmt\_misc type binfmt\_misc (rw)

sunrpc on /var/lib/nfs/rpc\_pipefs type rpc\_pipefs (rw)

This listing is from a CentOS 5 system, which is using LVM (Logical Volume Manager) to create its root file system. Like many modern Linux distributions, this system will attempt to automatically mount the CD-ROM after insertion. After we insert the disk, we see the following:

这个列表来自于 CentOS 5系统，使用 LVM（逻辑卷管理器）来创建它的根文件系统。正如许多现在的 Linux 发行版一样，这个 系统试图自动挂载插入的 CD-ROM。当我们插入光盘后，我们看看下面的输出：

[me@linuxbox ~]$ mount

/dev/mapper/VolGroup00-LogVol00 on / type ext3 (rw)

proc on /proc type proc (rw)

sysfs on /sys type sysfs (rw)

devpts on /dev/pts type devpts (rw,gid=5,mode=620)

/dev/hda1 on /boot type ext3 (rw)

tmpfs on /dev/shm type tmpfs (rw)

none on /proc/sys/fs/binfmt\_misc type binfmt\_misc (rw)

sunrpc on /var/lib/nfs/rpc\_pipefs type rpc\_pipefs (rw)

/dev/hdc on /media/live-1.0.10-8 type iso9660 (ro,noexec,nosuid,

nodev,uid=500)

After we insert the disk, we see the same listing as before with one additional entry. At the end of the listing we see that the CD-ROM (which is device /dev/hdc on this system) has been mounted on /media/live-1.0.10-8, and is type iso9660 (a CD- ROM). For purposes of our experiment, we’re interested in the name of the device. When you conduct this experiment yourself, the device name will most likely be different.

当我们插入光盘后，除了额外的一行之外，我们看到和原来一样的列表。在列表的末尾，我们 看到 CD-ROW 已经挂载到了/media/live-1.0.10-8上，它的文件类型是 iso9660（CD-ROW）。 就我们的实验目的而言，我们对这个设备的名字感兴趣。当你自己进行这个实验时，这个 设备名字是最有可能不同的。

Warning: In the examples that follow, it is vitally important that you pay close attention to the actual device names in use on your system and do not use the names used in this text!

警告：在随后的实例中，至关重要的是你要密切注意用在你系统中的实际设备名，并且 不要使用此文本中使用的名字！

Also note that audio CDs are not the same as CD-ROMs. Audio CDs do not contain file systems and thus cannot be mounted in the usual sense.

还要注意音频 CD 和 CD-ROW 不一样。音频 CD 不包含文件系统，这样在通常意义上，它就不能被挂载了。

Now that we have the device name of the CD-ROM drive, let’s unmount the disk and remount it another location in the file system tree. To do this, we become the superuser (using the command appropriate for our system) and unmount the disk with the umount (notice the spelling) command:

现在我们拥有 CD-ROW 光盘的设备名字，让我们卸载这张光盘，并把它重新挂载到文件系统树 的另一个位置。我们需要超级用户身份（使用系统相应的命令）来进行操作，并且用 umount（注意这个命令的拼写）来卸载光盘：

[me@linuxbox ~]$ su -

Password:

[root@linuxbox ~]# umount /dev/hdc

The next step is to create a new mount point for the disk. A mount point is simply a directory somewhere on the file system tree. Nothing special about it. It doesn’t even have to be an empty directory, though if you mount a device on a non-empty directory, you will not be able to see the directory’s previous contents until you unmount the device. For our purposes, we will create a new directory:

下一步是创建一个新的光盘挂载点。简单地说，一个挂载点就是文件系统树中的一个目录。它没有 什么特殊的。它甚至不必是一个空目录，即使你把设备挂载到了一个非空目录上，你也不能看到 这个目录中原来的内容，直到你卸载这个设备。就我们的目的而言，我们将创建一个新目录：

[root@linuxbox ~]# mkdir /mnt/cdrom

Finally, we mount the CD-ROM at the new mount point. The -t option is used to specify the file system type:

最后，我们把这个 CD-ROW 挂载到一个新的挂载点上。这个-t 选项用来指定文件系统类型：

[root@linuxbox ~]# mount -t iso9660 /dev/hdc /mnt/cdrom

Afterward, we can examine the contents of the CD-ROM via the new mount point:

之后，我们可以通过这个新挂载点来查看 CD-ROW 的内容：

[root@linuxbox ~]# cd /mnt/cdrom

[root@linuxbox cdrom]# ls

Notice what happens when we try to unmount the CD-ROM:

注意当我们试图卸载这个 CD-ROW 时，发生了什么事情。

[root@linuxbox cdrom]# umount /dev/hdc

umount: /mnt/cdrom: device is busy

Why is this? The reason is that we cannot unmount a device if the device is being used by someone or some process. In this case, we changed our working directory to the mount point for the CD-ROM, which causes the device to be busy. We can easily remedy the issue by changing the working directory to something other than the mount point:

这是怎么回事呢？原因是我们不能卸载一个设备，如果某个用户或进程正在使用这个设备的话。在这种 情况下，我们把工作目录更改到了 CD-ROW 的挂载点，这个挂载点导致设备忙碌。我们可以很容易地修复这个问题 通过把工作目录改到其它目录而不是这个挂载点。

[root@linuxbox cdrom]# cd

[root@linuxbox ~]# umount /dev/hdc

Now the device unmounts successfully.

现在这个设备成功卸载了。

Why Unmounting Is Important

为什么卸载重要

If you look at the output of the free command, which displays statistics about memory usage, you will see a statistic called “buffers.” Computer systems are designed to go as fast as possible. One of the impediments to system speed is slow devices. Printers are a good example. Even the fastest printer is extremely slow by computer standards. A computer would be very slow indeed if it had to stop and wait for a printer to finish printing a page. In the early days of PCs (before multi-tasking), this was a real problem. If you were working on a spreadsheet or text document, the computer would stop and become unavailable every time you printed. The computer would send the data to the printer as fast as the printer could accept it, but it was very slow since printers don’t print very fast. This problem was solved by the advent of the printer buffer, a device containing some RAM memory that would sit between the computer and the printer. With the printer buffer in place, the computer would send the printer output to the buffer and it would quickly be stored in the fast RAM so the computer could go back to work without waiting. Meanwhile, the printer buffer would slowly spool the data to the printer from the buffer’s memory at the speed at which the printer could accept it.

如果你看一下 free 命令的输出结果，这个命令用来显示关于内存使用情况的统计信息，你 会看到一个统计值叫做”buffers“。计算机系统旨在尽可能快地运行。系统运行速度的 一个阻碍是缓慢的设备。打印机是一个很好的例子。即使最快速的打印机相比于计算机标准也 极其地缓慢。一台计算机确实会运行地非常慢，如果它要停下来等待一台打印机打印完一页。 在早期的个人电脑时代（多任务之前），这真是个问题。如果你正在编辑电子表格 或者是文本文档，每次你要打印文件时，计算机都会停下来而且变得不能使用。 计算机能以打印机可接受的最快速度把数据发送给打印机，但由于打印机不能快速地打印， 这个发送速度会非常慢。这个问题被解决了，由于打印机缓存的出现，一个包含一些 RAM 内存 的设备，位于计算机和打印机之间。通过打印机缓存，计算机把要打印的结果发送到这个缓存区， 数据会迅速地存储到这个 RAM 中，这样计算机就能回去工作，而不用等待。与此同时，打印机缓存将会 以打印机可接受的速度把缓存中的数据缓慢地输出给打印机。

This idea of buffering is used extensively in computers to make them faster. Don’t let the need to occasionally read or write data to/from slow devices impede the speed of the system. Operating systems store data read from, and to be written to storage devices in memory for as long as possible before actually having to interact with the slower device. On a Linux system for example, you will notice that the system seems to fill up memory the longer it is used. This does not mean Linux is “using“ all the memory, it means that Linux is taking advantage of all the available memory to do as much buffering as it can.

缓存被广泛地应用于计算机中，使其运行地更快。别让偶尔地需要读取或写入慢设备阻碍了 系统的运行速度。在实际与慢设备交互之前，操作系统会尽可能多的读取或写入数据到内存中的 存储设备里。以 Linux 操作系统为例，你会注意到系统看似填充了多于它所需要的内存。 这不意味着 Linux 正在使用所有的内存，它意味着 Linux 正在利用所有可用的内存，来作为缓存区。

This buffering allows writing to storage devices to be done very quickly, because the writing to the physical device is being deferred to a future time. In the meantime, the data destined for the device is piling up in memory. From time to time, the operating system will write this data to the physical device.

这个缓存区允许非常快速地写入存储设备，因为写入物理设备的操作被延迟到后面进行。同时， 这些注定要传送到设备中的数据正在内存中堆积起来。时不时地，操作系统会把这些数据 写入物理设备。

Unmounting a device entails writing all the remaining data to the device so that it can be safely removed. If the device is removed without unmounting it first, the possibility exists that not all the data destined for the device has been transferred. In some cases, this data may include vital directory updates, which will lead to file system corruption, one of the worst things that can happen on a computer.

卸载一个设备需要把所有剩余的数据写入这个设备，所以设备可以被安全地移除。如果 没有卸载设备，就移除了它，就有可能没有把注定要发送到设备中的数据输送完毕。在某些情况下， 这些数据可能包含重要的目录更新信息，这将导致文件系统损坏，这是发生在计算机中的最坏的事情之一。

**确定设备名称**

It’s sometimes difficult to determine the name of a device. Back in the old days, it wasn’t very hard. A device was always in the same place and it didn’t change. Unix-like systems like it that way. Back when Unix was developed, “changing a disk drive” involved using a forklift to remove a washing machine-sized device from the computer room. In recent years, the typical desktop hardware configuration has become quite dynamic and Linux has evolved to become more flexible than its ancestors. In the examples above we took advantage of the modern Linux desktop’s ability to “automagically” mount the device and then determine the name after the fact. But what if we are managing a server or some other environment where this does not occur? How can we figure it out?

有时很难来确定设备名称。在以前，这并不是很难。一台设备总是在某个固定的位置，也不会 挪动它。类 Unix 的系统喜欢设备那样安排。之前在开发 Unix 系统的时候，“更改一个磁盘驱动器”要用一辆 叉车从机房中移除一台如洗衣机大小的设备。最近几年，典型的桌面硬件配置已经变得相当动态，并且 Linux 已经发展地比其祖先更加灵活。在以上事例中，我们利用现代 Linux 桌面系统的功能来“自动地”挂载 设备，然后再确定设备名称。但是如果我们正在管理一台服务器或者是其它一些（这种自动挂载功能）不会 发生的环境，我们又如何能查清设备名呢？

First, let’s look at how the system names devices. If we list the contents of the /dev directory (where all devices live), we can see that there are lots and lots of devices:

首先，让我们看一下系统怎样来命名设备。如果我们列出目录/dev（所有设备的住所）的内容，我们 会看到许许多多的设备：

[me@linuxbox ~]$ ls /dev

The contents of this listing reveal some patterns of device naming. Here are a few:

这个列表的内容揭示了一些设备命名的模式。这里有几个：

|  |  |
| --- | --- |
| *Table 16-2: Linux Storage Device Names* | |
| Pattern | Device |
| /dev/fd\* | Floppy disk drives |
| /dev/hd\* | IDE (PATA) disks on older systems. Typical motherboards contain two IDE connectors or channels, each with a cable with two attachment points for drives. The first drive on the cable is called the master device and the second is called the slave device. The device names are ordered such that /dev/hda refers to the master device on the first channel, /dev/hdb is the slave device on the first channel; /dev/hdc, the master device on the second channel, and so on. A trailing digit indicates the partition number on the device. For example, /dev/hda1 refers to the first partition on the first hard drive on the system while / dev/hda refers to the entire drive. |
| /dev/lp\* | Printers |
| /dev/sd\* | SCSI disks. On recent Linux systems, the kernel treats all disk- like devices (including PATA/SATA hard disks, flash drives, and USB mass storage devices, such as portable music players and digital cameras) as SCSI disks. The rest of the naming system is similar to the older /dev/hd\* naming scheme described above. |
| /dev/sr\* | Optical drives (CD/DVD readers and burners) |

|  |  |
| --- | --- |
| *表16-2: Linux 存储设备名称* | |
| 模式 | 设备 |
| /dev/fd\* | 软盘驱动器 |
| /dev/hd\* | 老系统中的 IDE(PATA)磁盘。典型的主板包含两个 IDE 连接器或者是通道，每个连接器 带有一根缆线，每根缆线上有两个硬盘驱动器连接点。缆线上的第一个驱动器叫做主设备， 第二个叫做从设备。设备名称这样安排，/dev/hdb 是指第一通道上的主设备名；/dev/hdb 是第一通道上的从设备名；/dev/hdc 是第二通道上的主设备名，等等。末尾的数字表示 硬盘驱动器上的分区。例如，/dev/hda1是指系统中第一硬盘驱动器上的第一个分区，而 /dev/hda 则是指整个硬盘驱动器。 |
| /dev/lp\* | 打印机 |
| /dev/sd\* | SCSI 磁盘。在最近的 Linux 系统中，内核把所有类似于磁盘的设备（包括 PATA/SATA 硬盘， 闪存，和 USB 存储设备，比如说可移动的音乐播放器和数码相机）看作 SCSI 磁盘。 剩下的命名系统类似于上述所描述的旧的/dev/hd\*命名方案。 |
| /dev/sr\* | 光盘（CD/DVD 读取器和烧写器） |

In addition, we often see symbolic links such as /dev/cdrom, /dev/dvd and /dev/ floppy, which point to the actual device files, provided as a convenience. If you are working on a system that does not automatically mount removable devices, you can use the following technique to determine how the removable device is named when it is attached. First, start a real-time view of the /var/log/messages file (you may require superuser privileges for this):

另外，我们经常看到符号链接比如说/dev/cdrom，/dev/dvd 和/dev/floppy，它们指向实际的 设备文件，提供这些链接是为了方便使用。如果你工作的系统不能自动挂载可移动的设备，你可以使用 下面的技巧来决定当可移动设备连接后，它是怎样被命名的。首先，启动一个实时查看文件/var/log/messages （你可能需要超级用户权限）：

[me@linuxbox ~]$ sudo tail -f /var/log/messages

The last few lines of the file will be displayed and then pause. Next, plug in the removable device. In this example, we will use a 16 MB flash drive. Almost immediately, the kernel will notice the device and probe it:

这个文件的最后几行会被显示，然后停止。下一步，插入这个可移动的设备。在 这个例子里，我们将使用一个16MB 闪存。瞬间，内核就会发现这个设备， 并且探测它：

Jul 23 10:07:53 linuxbox kernel: usb 3-2: new full speed USB device

using uhci\_hcd and address 2

Jul 23 10:07:53 linuxbox kernel: usb 3-2: configuration #1 chosen

from 1 choice

Jul 23 10:07:53 linuxbox kernel: scsi3 : SCSI emulation for USB Mass

Storage devices

Jul 23 10:07:58 linuxbox kernel: scsi scan: INQUIRY result too short

(5), using 36

Jul 23 10:07:58 linuxbox kernel: scsi 3:0:0:0: Direct-Access Easy

Disk 1.00 PQ: 0 ANSI: 2

Jul 23 10:07:59 linuxbox kernel: sd 3:0:0:0: [sdb] 31263 512-byte

hardware sectors (16 MB)

Jul 23 10:07:59 linuxbox kernel: sd 3:0:0:0: [sdb] Write Protect is

off

Jul 23 10:07:59 linuxbox kernel: sd 3:0:0:0: [sdb] Assuming drive

cache: write through

Jul 23 10:07:59 linuxbox kernel: sd 3:0:0:0: [sdb] 31263 512-byte

hardware sectors (16 MB)

Jul 23 10:07:59 linuxbox kernel: sd 3:0:0:0: [sdb] Write Protect is

off

Jul 23 10:07:59 linuxbox kernel: sd 3:0:0:0: [sdb] Assuming drive

cache: write through

Jul 23 10:07:59 linuxbox kernel: sdb: sdb1

Jul 23 10:07:59 linuxbox kernel: sd 3:0:0:0: [sdb] Attached SCSI

removable disk

Jul 23 10:07:59 linuxbox kernel: sd 3:0:0:0: Attached scsi generic

sg3 type 0

After the display pauses again, type Ctrl-c to get the prompt back. The interesting parts of the output are the repeated references to “[sdb]” which matches our expectation of a SCSI disk device name. Knowing this, two lines become particularly illuminating:

显示再次停止之后，输入 Ctrl-c，重新得到提示符。输出结果的有趣部分是一再提及“[sdb]”， 这正好符和我们期望的 SCSI 磁盘设备名称。知道这一点后，有两行输出变得颇具启发性：

Jul 23 10:07:59 linuxbox kernel: sdb: sdb1

Jul 23 10:07:59 linuxbox kernel: sd 3:0:0:0: [sdb] Attached SCSI

removable disk

This tells us the device name is /dev/sdb for the entire device and /dev/sdb1 for the first partition on the device. As we have seen, working with Linux is full of interesting detective work!

这告诉我们这个设备名称是/dev/sdb 指整个设备，/dev/sdb1是这个设备的第一分区。 正如我们所看到的，使用 Linux 系统充满了有趣的监测工作。

Tip: Using the tail -f /var/log/messages technique is a great way to watch what the system is doing in near real-time.

小贴士：使用这个 tail -f /var/log/messages 技巧是一个很不错的方法，可以实时 观察系统的一举一动。

With our device name in hand, we can now mount the flash drive:

既然知道了设备名称，我们就可以挂载这个闪存驱动器了：

[me@linuxbox ~]$ sudo mkdir /mnt/flash

[me@linuxbox ~]$ sudo mount /dev/sdb1 /mnt/flash

[me@linuxbox ~]$ df

Filesystem 1K-blocks Used Available Use% Mounted on

/dev/sda2 15115452 5186944 9775164 35% /

/dev/sda5 59631908 31777376 24776480 57% /home

/dev/sda1 147764 17277 122858 13% /boot

tmpfs 776808 0 776808 0% /dev/shm

/dev/sdb1 15560 0 15560 0% /mnt/flash

The device name will remain the same as long as it remains physically attached to the computer and the computer is not rebooted.

这个设备名称会保持不变只要设备与计算机保持连接并且计算机不会重新启动。

**创建新的文件系统**

Let’s say that we want to reformat the flash drive with a Linux native file system, rather than the FAT32 system it has now. This involves two steps: 1. (optional) create a new partition layout if the existing one is not to our liking, and 2. create a new, empty file system on the drive.

假若我们想要用 Linux 本地文件系统来重新格式化这个闪存驱动器，而不是它现用的 FAT32系统。 这涉及到两个步骤：1.（可选的）创建一个新的分区布局若已存在的分区不是我们喜欢的。2. 在这个闪存上创建一个新的空的文件系统。

Warning! In the following exercise, we are going to format a flash drive. Use a drive that contains nothing you care about because it will be erased! Again, make absolutely sure you are specifying the correct device name for your system, not the one shown in the text. Failure to heed this warning could result in you formatting (i.e., erasing) the wrong drive!

注意！在下面的练习中，我们将要格式化一个闪存驱动器。拿一个不包含有用数据的驱动器 作为实验品，因为它将会被擦除！再次，请确定你指定了正确的系统设备名称。未能注意此 警告可能导致你格式化（即擦除）错误的驱动器！

**用 fdisk 命令操作分区**

The fdisk program allows us to interact directly with disk-like devices (such as hard disk drives and flash drives) at a very low level. With this tool we can edit, delete, and create partitions on the device. To work with our flash drive, we must first unmount it (if needed) and then invoke the fdisk program as follows:

这个 fdisk 程序允许我们直接在底层与类似磁盘的设备（比如说硬盘驱动器和闪存驱动器）进行交互。 使用这个工具可以在设备上编辑，删除，和创建分区。以我们的闪存驱动器为例， 首先我们必须卸载它（如果需要的话），然后调用 fdisk 程序，如下所示：

[me@linuxbox ~]$ sudo umount /dev/sdb1

[me@linuxbox ~]$ sudo fdisk /dev/sdb

Notice that we must specify the device in terms of the entire device, not by partition number. After the program starts up, we will see the following prompt:

注意我们必须指定设备名称，就整个设备而言，而不是通过分区号。这个程序启动后，我们 将看到以下提示：

Command (m for help):

Entering an “m” will display the program menu:

输入”m”会显示程序菜单：

Command action

a toggle a bootable flag

....

The first thing we want to do is examine the existing partition layout. We do this by entering “p” to print the partition table for the device:

我们想要做的第一件事情是检查已存在的分区布局。输入”p”会打印出这个设备的分区表：

Command (m for help): p

Disk /dev/sdb: 16 MB, 16006656 bytes

1 heads, 31 sectors/track, 1008 cylinders

Units = cylinders of 31 \* 512 = 15872 bytes

Device Boot Start End Blocks Id System

/dev/sdb1 2 1008 15608+ b w95 FAT32

In this example, we see a 16 MB device with a single partition (1) that uses 1006 of the available 1008 cylinders on the device. The partition is identified as Windows 95 FAT32 partition. Some programs will use this identifier to limit the kinds of operation that can be done to the disk, but most of the time it is not critical to change it. However, in the interest of demonstration, we will change it to indicate a Linux partition. To do this, we must first find out what ID is used to identify a Linux partition. In the listing above, we see that the ID “b” is used to specify the exiting partition. To see a list of the available partition types, we refer back to the program menu. There we can see the following choice:

在此例中，我们看到一个16MB 的设备只有一个分区(1)，此分区占用了可用的1008个柱面中的1006个, 并被标识为 Windows 95 FAT32分区。有些程序会使用这个标志符来限制一些可以对磁盘所做的操作， 但大多数情况下更改这个标志符没有危害。然而，为了叙述方便，我们将会更改它， 以此来表明是个 Linux 分区。在更改之前，首先我们必须找到被用来识别一个 Linux 分区的 ID 号码。 在上面列表中，我们看到 ID 号码“b”被用来指定这个已存在的分区。要查看可用的分区类型列表， 参考之前的程序菜单。我们会看到以下选项：

l list known partition types

If we enter “l” at the prompt, a large list of possible types is displayed. Among them we see “b” for our existing partition type and “83” for Linux.

如果我们在提示符下输入“l”，就会显示一个很长的可能类型列表。在它们之中会看到“b”为已存在分区 类型的 ID 号，而“83”是针对 Linux 系统的 ID 号。

Going back to the menu, we see this choice to change a partition ID:

回到之前的菜单，看到这个选项来更改分区 ID 号：

t change a partition's system id

We enter “t” at the prompt enter the new ID:

我们先输入“t”，再输入新的 ID 号：

Command (m for help): t

Selected partition 1

Hex code (type L to list codes): 83

Changed system type of partition 1 to 83 (Linux)

This completes all the changes that we need to make. Up to this point, the device has been untouched (all the changes have been stored in memory, not on the physical device), so we will write the modified partition table to the device and exit. To do this, we enter “w” at the prompt:

这就完成了我们需要做得所有修改。到目前为止，还没有接触这个设备（所有修改都存储在内存中， 而不是在此物理设备中），所以我们将会把修改过的分区表写入此设备，再退出。为此，我们输入 在提示符下输入”w”:

Command (m for help): w

The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: If you have created or modified any DOS 6.x

partitions, please see the fdisk manual page for additional

information.

Syncing disks.

[me@linuxbox ~]$

If we had decided to leave the device unaltered, we could have entered “q” at the prompt, which would have exited the program without writing the changes. We can safely ignore the ominous sounding warning message.

如果我们已经决定保持设备不变，可在提示符下输入”q”，这将退出程序而没有写更改。我们 可以安全地忽略这些不祥的警告信息。

**用 mkfs 命令创建一个新的文件系统**

With our partition editing done (lightweight though it might have been) it’s time to create a new file system on our flash drive. To do this, we will use mkfs (short for “make file system”), which can create file systems in a variety of formats. To create an ext3 file system on the device, we use the “-t” option to specify the “ext3” system type, followed by the name of device containing the partition we wish to format:

完成了分区编辑工作（它或许是轻量级的），是时候在我们的闪存驱动器上创建一个新的文件系统了。 为此，我们会使用 mkfs（”make file system”的简写），它能创建各种格式的文件系统。 在此设备上创建一个 ext3文件系统，我们使用”-t” 选项来指定这个”ext3”系统类型，随后是我们要格式化的设备分区名称：

[me@linuxbox ~]$ sudo mkfs -t ext3 /dev/sdb1

mke2fs 1.40.2 (12-Jul-2007)

Filesystem label=

OS type: Linux

Block size=1024 (log=0)

Fragment size=1024 (log=0)

3904 inodes, 15608 blocks

780 blocks (5.00%) reserved for the super user

First data block=1

Maximum filesystem blocks=15990784

2 block groups

8192 blocks per group, 8192 fragments per group

1952 inodes per group

Superblock backups stored on blocks:

8193

Writing inode tables: done

Creating journal (1024 blocks): done

Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 34 mounts or

180 days, whichever comes first. Use tune2fs -c or -i to override.

[me@linuxbox ~]$

The program will display a lot of information when ext3 is the chosen file system type. To re-format the device to its original FAT32 file system, specify “vfat” as the file system type:

当 ext3被选为文件系统类型时，这个程序会显示许多信息。若把这个设备重新格式化为它最初的 FAT32文件 系统，指定”vfat”作为文件系统类型：

[me@linuxbox ~]$ sudo mkfs -t vfat /dev/sdb1

This process of partitioning and formatting can be used anytime additional storage devices are added to the system. While we worked with a tiny flash drive, the same process can be applied to internal hard disks and other removable storage devices like USB hard drives.

任何时候添加额外的存储设备到系统中时，都可以使用这个分区和格式化的过程。虽然我们 只以一个小小的闪存驱动器为例，同样的操作可以被应用到内部硬盘和其它可移动的存储设备上 像 USB 硬盘驱动器。

**测试和修复文件系统**

In our earlier discussion of the /etc/fstab file, we saw some mysterious digits at the end of each line. Each time the system boots, it routinely checks the integrity of the file systems before mounting them. This is done by the fsck program (short for “file system check”). The last number in each fstab entry specifies the order the devices are to be checked. In our example above, we see that the root file system is checked first, followed by the home and boot file systems. Devices with a zero as the last digit are not routinely checked.

在之前讨论文件/etc/fstab 时，我们会在每行的末尾看到一些神秘的数字。每次系统启动时， 在挂载系统之前，都会按照惯例检查文件系统的完整性。这个任务由 fsck 程序（是”file system check”的简写）完成。每个 fstab 项中的最后一个数字指定了设备的检查顺序。 在上面的实例中，我们看到首先检查根文件系统，然后是 home 和 boot 文件系统。若最后一个数字 是零则相应设备不会被检查。

In addition to checking the integrity of file systems, fsck can also repair corrupt file systems with varying degrees of success, depending on the amount of damage. On Unix- like file systems, recovered portions of files are placed in the lost+found directory, located in the root of each file system.

除了检查文件系统的完整性之外，fsck 还能修复受损的文件系统，其成功度依赖于损坏的数量。 在类 Unix 的文件系统中，文件恢复的部分被放置于 lost+found 目录里面，位于每个文件 系统的根目录下面。

To check our flash drive (which should be unmounted first), we could do the following:

检查我们的闪存驱动器（首先应该卸载），我们能执行下面的操作：

[me@linuxbox ~]$ sudo fsck /dev/sdb1

fsck 1.40.8 (13-Mar-2008)

e2fsck 1.40.8 (13-Mar-2008)

/dev/sdb1: clean, 11/3904 files, 1661/15608 blocks

In my experience, file system corruption is quite rare unless there is a hardware problem, such as a failing disk drive. On most systems, file system corruption detected at boot time will cause the system to stop and direct you to run fsck before continuing.

以我的经验，文件系统损坏情况相当罕见，除非硬件存在问题，如磁盘驱动器故障。 在大多数系统中，系统启动阶段若探测到文件系统已经损坏了，则会导致系统停止下来， 在系统继续执行之前，会指导你运行 fsck 程序。

What The fsck?

什么是 fsck?

In Unix culture, the word “fsck” is often used in place of a popular word with which it shares three letters. This is especially appropriate, given that you will probably be uttering the aforementioned word if you find yourself in a situation where you are forced to run fsck.

在 Unix 文化中，”fsck”这个单词往往会被用来代替一个流行的词，“fsck”和这个词共享了三个 字母。这个尤其适用，因为你可能会说出上文提到的词，若你发现自己处于这种境况下， 被强制来运行 fsck 命令时。

**格式化软盘**

For those of us still using computers old enough to be equipped with floppy diskette drives, we can manage those devices, too. Preparing a blank floppy for use is a two step process. First, we perform a low-format on the diskette, then create a file system. To accomplish the formatting, we use the fdformat program specifying the name of the floppy device (usually /dev/fd0):

对于那些还在使用配备了软盘驱动器的计算机的用户，我们也能管理这些设备。准备一 张可用的空白软盘要分两个步骤。首先，对这张软盘执行低级格式化，然后创建一个文件系统。 为了完成格式化，我们使用 fdformat 程序，同时指定软盘设备名称（通常为/dev/fd0）：

[me@linuxbox ~]$ sudo fdformat /dev/fd0

Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.

Formatting ... done

Verifying ... done

Next, we apply a FAT file system to the diskette with mkfs:

接下来，通过 mkfs 命令，给这个软盘创建一个 FAT 文件系统：

[me@linuxbox ~]$ sudo mkfs -t msdos /dev/fd0

Notice that we use the “msdos” file system type to get the older (and smaller) style file allocation tables. After a diskette is prepared, it may be mounted like other devices.

注意我们使用这个“msdos”文件系统类型来得到旧（小的）风格的文件分配表。当一个软磁盘 被准备好之后，则可能像其它设备一样挂载它。

**直接把数据移入/出设备**

While we usually think of data on our computers as being organized into files, it is also possible to think of the data in “raw” form. If we look at a disk drive, for example, we see that it consists of a large number of “blocks” of data that the operating system sees as directories and files. However, if we could treat a disk drive as simply a large collection of data blocks, we could perform useful tasks, such as cloning devices.

虽然我们通常认为计算机中的数据以文件形式来组织数据，也可以“原始的”形式来考虑数据。 如果我们看一下磁盘驱动器，例如， 我们看到它由大量的数据“块”组成，而操作系统却把这些数据块看作目录和文件。然而，如果 把磁盘驱动器简单地看成一个数据块大集合，我们就能执行有用的任务，如克隆设备。

The dd program performs this task. It copies blocks of data from one place to another. It uses a unique syntax (for historical reasons) and is usually used this way:

这个 dd 程序能执行此任务。它可以把数据块从一个地方复制到另一个地方。它使用独特的语法（由于历史原因） ，经常它被这样使用：

dd if=input\_file of=output\_file [bs=block\_size [count=blocks]]

Let’s say we had two USB flash drives of the same size and we wanted to exactly copy the first drive to the second. If we attached both drives to the computer and they are assigned to devices /dev/sdb and /dev/sdc respectively, we could copy everything on the first drive to the second drive with the following:

比方说我们有两个相同容量的 USB 闪存驱动器，并且要精确地把第一个驱动器（中的内容） 复制给第二个。如果连接两个设备到计算机上，它们各自被分配到设备/dev/sdb 和 /dev/sdc 上，这样我们就能通过下面的命令把第一个驱动器中的所有数据复制到第二个 驱动器中。

dd if=/dev/sdb of=/dev/sdc

Alternately, if only the first device were attached to the computer, we could copy its contents to an ordinary file for later restoration or copying:

或者，如果只有第一个驱动器被连接到计算机上，我们可以把它的内容复制到一个普通文件中供 以后恢复或复制数据：

dd if=/dev/sdb of=flash\_drive.img

Warning! The dd command is very powerful. Though its name derives from “data definition,” it is sometimes called “destroy disk” because users often mistype either the if or of specifications. **Always double check your input and output specifications before pressing enter!**

警告！这个 dd 命令非常强大。虽然它的名字来自于“数据定义”，有时候也把它叫做“清除磁盘” 因为用户经常会误输入 if 或 of 的规范。**在按下回车键之前，要再三检查输入与输出规范！**

**创建 CD-ROM 映像**

Writing a recordable CD-ROM (either a CD-R or CD-RW) consists of two steps; first, constructing an iso image file that is the exact file system image of the CD-ROM and second, writing the image file onto the CD-ROM media.

写入一个可记录的 CD-ROM（一个 CD-R 或者是 CD-RW）由两步组成；首先，构建一个 iso 映像文件， 这就是一个 CD-ROM 的文件系统映像，第二步，把这个映像文件写入到 CD-ROM 媒介中。

**创建一个 CD-ROM 的映像拷贝**

If we want to make an iso image of an existing CD-ROM, we can use dd to read all the data blocks off the CD-ROM and copy them to a local file. Say we had an Ubuntu CD and we wanted to make an iso file that we could later use to make more copies. After inserting the CD and determining its device name (we’ll assume /dev/cdrom), we can make the iso file like so:

如果想要制作一张现有 CD-ROM 的 iso 映像，我们可以使用 dd 命令来读取 CD-ROW 中的所有数据块， 并把它们复制到本地文件中。比如说我们有一张 Ubuntu CD，用它来制作一个 iso 文件，以后我们可以用它来制作更多的拷贝。插入这张 CD 之后，确定 它的设备名称（假定是/dev/cdrom），然后像这样来制作 iso 文件：

dd if=/dev/cdrom of=ubuntu.iso

This technique works for data DVDs as well, but will not work for audio CDs, as they do not use a file system for storage. For audio CDs, look at the cdrdao command.

这项技术也适用于 DVD 光盘，但是不能用于音频 CD，因为它们不使用文件系统来存储数据。 对于音频 CD，看一下 cdrdao 命令。

**从文件集合中创建一个映像**

To create an iso image file containing the contents of a directory, we use the genisoimage program. To do this, we first create a directory containing all the files we wish to include in the image and then execute the genisoimage command to create the image file. For example, if we had created a directory called ~/cd-rom-files and filled it with files for our CD-ROM, we could create an image file named cd- rom.iso with the following command:

创建一个包含目录内容的 iso 映像文件，我们使用 genisoimage 程序。为此，我们首先创建 一个目录，这个目录中包含了要包括到此映像中的所有文件，然后执行这个 genisoimage 命令 来创建映像文件。例如，如果我们已经创建一个叫做~/cd-rom-files 的目录，然后用文件 填充此目录，再通过下面的命令来创建一个叫做 cd-rom.iso 映像文件：

genisoimage -o cd-rom.iso -R -J ~/cd-rom-files

The “-R” option adds metadata for the Rock Ridge extensions, which allows the use of long filenames and POSIX style file permissions. Likewise, the “-J” option enables the Joliet extensions, which permit long filenames for Windows.

“-R”选项添加元数据为 Rock Ridge 扩展，这允许使用长文件名和 POSIX 风格的文件权限。 同样地，这个”-J”选项使 Joliet 扩展生效，这样 Windows 中就支持长文件名了。

A Program By Any Other Name…

一个有着其它名字的程序。。。

If you look at on-line tutorials for creating and burning optical media like CD- ROMs and DVDs, you will frequently encounter two programs called mkisofs and cdrecord. These programs were part of a popular package called “cdrtools” authored by Jorg Schilling. In the summer of 2006, Mr. Schilling made a license change to a portion of the cdrtools package which, in the opinion of many in the Linux community, created a license incompatibility with the GNU GPL. As a result, a fork of the cdrtools project was started that now includes replacement programs for cdrecord and mkisofs named wodim and genisoimage, respectively.

如果你看一下关于创建和烧写光介质如 CD-ROMs 和 DVD 的在线文档，你会经常碰到两个程序 叫做 mkisofs 和 cdrecord。这些程序是流行软件包”cdrtools”的一部分，”cdrtools”由 Jorg Schilling 编写成。在2006年春天，Schilling 先生更改了部分 cdrtools 软件包的协议，Linux 社区许多人的看法是， 这创建了一个与 GNU GPL 不相兼容的协议。结果，就 fork 了这个 cdrtools 项目， 目前新项目里面包含了 cdrecord 和 mkisofs 的替代程序，分别是 wodim 和 genisoimage。

**写入 CD-ROM 镜像**

After we have an image file, we can burn it onto our optical media. Most of the commands we will discuss below can be applied to both recordable CD-ROM and DVD media.

有了一个映像文件之后，我们可以把它烧写到光盘中。下面讨论的大多数命令对可 记录的 CD-ROW 和 DVD 媒介都适用。

**直接挂载一个 ISO 镜像**

There is a trick that we can use to mount an iso image while it is still on our hard disk and treat it as though it was already on optical media. By adding the “-o loop” option to mount (along with the required “-t iso9660” file system type), we can mount the image file as though it were a device and attach it to the file system tree:

有一个诀窍，我们可以用它来挂载 iso 映像文件，虽然此文件仍然在我们的硬盘中，但我们 当作它已经在光盘中了。添加 “-o loop” 选项来挂载（同时带有必需的 “-t iso9660” 文件系统类型）， 挂载这个映像文件就好像它是一台设备，把它连接到文件系统树上：

mkdir /mnt/iso\_image

mount -t iso9660 -o loop image.iso /mnt/iso\_image

In the example above, we created a mount point named /mnt/iso\_image and then mounted the image file image.iso at that mount point. After the image is mounted, it can be treated just as though it were a real CD-ROM or DVD. Remember to unmount the image when it is no longer needed.

上面的示例中，我们创建了一个挂载点叫做/mnt/iso\_image，然后把此映像文件 image.iso 挂载到挂载点上。映像文件被挂载之后，可以把它当作，就好像它是一张 真正的 CD-ROM 或者 DVD。当不再需要此映像文件后，记得卸载它。

**清除一张可重写入的 CD-ROM**

Rewritable CD-RW media needs to be erased or blanked before it can be reused. To do this, we can use wodim, specifying the device name for the CD writer and the type of blanking to be performed. The wodim program offers several types. The most minimal (and fastest) is the “fast” type:

可重写入的 CD-RW 媒介在被重使用之前需要擦除或清空。为此，我们可以用 wodim 命令，指定 设备名称和清空的类型。此 wodim 程序提供了几种清空类型。最小（且最快）的是 “fast” 类型：

wodim dev=/dev/cdrw blank=fast

**写入镜像**

To write an image, we again use wodim, specifying the name of the optical media writer device and the name of the image file:

写入一个映像文件，我们再次使用 wodim 命令，指定光盘设备名称和映像文件名：

wodim dev=/dev/cdrw image.iso

In addition to the device name and image file, wodim supports a very large set of options. Two common ones are “-v” for verbose output, and “-dao” which writes the disk in disk-at-once mode. This mode should be used if you are preparing a disk for commercial reproduction. The default mode for wodim is track-at-once, which is useful for recording music tracks.

除了设备名称和映像文件之外，wodim 命令还支持非常多的选项。常见的两个选项是，”-v” 可详细输出， 和 “－dao” 以 disk-at-once 模式写入光盘。如果你正在准备一张光盘为的是商业复制，那么应该使用这种模式。 wodim 命令的默认模式是 track-at-once，这对于录制音乐很有用。

**拓展阅读**

We have just touched on the many ways that the command line can be used to manage storage media. Take a look at the man pages of the commands we have covered. Some of them support huge numbers of options and operations. Also, look for on-line tutorials for adding hard drives to your Linux system (there are many) and working with optical media.

我们刚才谈到了很多方法，可以使用命令行管理存储介质。看看我们所讲过命令的手册页。 一些命令支持大量的选项和操作。此外，寻找一些如何添加硬盘驱动器到 Linux 系统（有许多）的在线教程， 这些教程也要适用于光介质存储设备。

**友情提示**

It’s often useful to verify the integrity of an iso image that we have downloaded. In most cases, a distributor of an iso image will also supply a checksum file. A checksum is the result of an exotic mathematical calculation resulting in a number that represents the content of the target file. If the contents of the file change by even one bit, the resulting checksum will be much different. The most common method of checksum generation uses the md5sum program. When you use md5sum, it produces a unique hexadecimal number:

通常验证一下我们已经下载的 iso 映像文件的完整性很有用处。在大多数情况下，iso 映像文件的贡献者也会提供 一个 checksum 文件。一个 checksum 是一个神奇的数学运算的计算结果，这个数学计算会产生一个能表示目标文件内容的数字。 如果目标文件的内容即使更改一个二进制位，checksum 的结果将会非常不一样。 生成 checksum 数字的最常见方法是使用 md5sum 程序。当你使用 md5sum 程序的时候， 它会产生一个独一无二的十六进制数字：

md5sum image.iso

34e354760f9bb7fbf85c96f6a3f94ece image.iso

After you download an image, you should run md5sum against it and compare the results with the md5sum value supplied by the publisher.

当你下载完映像文件之后，你应该对映像文件执行 md5sum 命令，然后把运行结果与发行商提供的 md5sum 数值作比较。

In addition to checking the integrity of a downloaded file, we can use md5sum to verify newly written optical media. To do this, we first calculate the checksum of the image file and then calculate a checksum for the media. The trick to verifying the media is to limit the calculation to only the portion of the optical media that contains the image. We do this by determining the number of 2048 byte blocks the image contains (optical media is always written in 2048 byte blocks) and reading that many blocks from the media. On some types of media, this is not required. A CD-R written in disk-at-once mode can be checked this way:

除了检查下载文件的完整性之外，我们也可以使用 md5sum 程序验证新写入的光学存储介质。 为此，首先我们计算映像文件的 checksum 数值，然后计算此光学存储介质的 checksum 数值。 这种验证光学介质的技巧是限定只对 光学存储介质中包含映像文件的部分计算 checksum 数值。 通过确定映像文件所包含的 2048 个字节块的数目（光学存储介质总是以 2048 个字节块的方式写入） 并从存储介质中读取那么多的字节块，我们就可以完成操作。 某些类型的存储介质，并不需要这样做。一个以 disk-at-once 模式写入的 CD-R，可以用下面的方式检验：

md5sum /dev/cdrom

34e354760f9bb7fbf85c96f6a3f94ece /dev/cdrom

Many types of media, such as DVDs require a precise calculation of the number of blocks. In the example below, we check the integrity of the image file dvd-image.iso and the disk in the DVD reader /dev/dvd. Can you figure out how this works?

许多存储介质类型，如 DVD 需要精确地计算字节块的数目。在下面的例子中，我们检验了映像文件 dvd-image.iso 以及 DVD 光驱中磁盘 /dev/dvd 文件的完整性。你能弄明白这是怎么回事吗？

md5sum dvd-image.iso; dd if=/dev/dvd bs=2048 count=$(( $(stat -c "%s" dvd-image.iso) / 2048 )) | md5sum