第十九章 归档和备份

One of the primary tasks of a computer system’s administrator is keeping the system’s data secure. One way this is done is by performing timely backups of the system’s files. Even if you’re not system administrators, it is often useful to make copies of things and to move large collections of files from place to place and from device to device. In this chapter, we will look at several common programs that are used to manage collections of files. There are the file compression programs:

计算机系统管理员的一个主要任务就是保护系统的数据安全，其中一种方法是通过时时备份系统文件，来保护 数据。即使你不是一名系统管理员，像做做拷贝或者在各个位置和设备之间移动大量的文件，通常也是很有帮助的。 在这一章中，我们将会看看几个经常用来管理文件集合的程序。它们就是文件压缩程序：

* gzip – Compress or expand files
* gzip – 压缩或者展开文件
* bzip2 – A block sorting file compressor
* bzip2 – 块排序文件压缩器

The archiving programs:

归档程序：

* tar – Tape archiving utility
* tar – 磁带打包工具
* zip – Package and compress files
* zip – 打包和压缩文件

And the file synchronization program:

还有文件同步程序：

* rsync – Remote file and directory synchronization
* rsync – 同步远端文件和目录

**压缩文件**

Throughout the history of computing, there has been a struggle to get the most data into the smallest available space, whether that space be memory, storage devices or network bandwidth. Many of the data services that we take for granted today, such as portable music players, high definition television, or broadband Internet, owe their existence to effective data compression techniques.

纵观计算领域的发展历史，人们努力想把最多的数据存放到到最小的可用空间中，不管是内存，存储设备 还是网络带宽。今天我们把许多数据服务都看作是理所当然的事情，但是诸如便携式音乐播放器， 高清电视，或宽带网络之类的存在都应归功于高效的数据压缩技术。

Data compression is the process of removing redundancy from data. Let’s consider an imaginary example. Say we had an entirely black picture file with the dimensions of one hundred pixels by one hundred pixels. In terms of data storage (assuming twenty-four bits, or three bytes per pixel), the image will occupy thirty thousand bytes of storage:

数据压缩就是一个删除冗余数据的过程。让我们考虑一个假想的例子，比方说我们有一张100\*100像素的 纯黑的图片文件。根据数据存储方案（假定每个像素占24位，或者3个字节），那么这张图像将会占用 30,000个字节的存储空间：

100 \* 100 \* 3 = 30,000

An image that is all one color contains entirely redundant data. If we were clever, we could encode the data in such a way that we simply describe the fact that we have a block of thirty thousand black pixels. So, instead of storing a block of data containing thirty thousand zeros (black is usually represented in image files as zero), we could compress the data into the number 30,000, followed by a zero to represent our data. Such a data compression scheme is called run-length encoding and is one of the most rudimentary compression techniques. Today’s techniques are much more advanced and complex but the basic goal remains the same—get rid of redundant data.

一张单色图像包含的数据全是多余的。我们要是聪明的话，可以用这种方法来编码这些数据， 我们只要简单地描述这个事实，我们有3万个黑色的像素数据块。所以，我们不存储包含3万个0 （通常在图像文件中，黑色由0来表示）的数据块，取而代之，我们把这些数据压缩为数字30,000， 后跟一个0，来表示我们的数据。这种数据压缩方案被称为游程编码，是一种最基本的压缩技术。今天的技术更加先进和复杂，但是基本目标依然不变——避免多余数据。

Compression algorithms (the mathematical techniques used to carry out the compression) fall into two general categories, lossless and lossy. Lossless compression preserves all the data contained in the original. This means that when a file is restored from a compressed version, the restored file is exactly the same as the original, uncompressed version. Lossy compression, on the other hand, removes data as the compression is performed, to allow more compression to be applied. When a lossy file is restored, it does not match the original version; rather, it is a close approximation. Examples of lossy compression are JPEG (for images) and MP3 (for music.) In our discussion, we will look exclusively at lossless compression, since most data on computers cannot tolerate any data loss.

压缩算法（数学技巧被用来执行压缩任务）分为两大类，无损压缩和有损压缩。无损压缩保留了 原始文件的所有数据。这意味着，当还原一个压缩文件的时候，还原的文件与原文件一模一样。 而另一方面，有损压缩，执行压缩操作时会删除数据，允许更大的压缩。当一个有损文件被还原的时候， 它与原文件不相匹配; 相反，它是一个近似值。有损压缩的例子有 JPEG（图像）文件和 MP3（音频）文件。 在我们的讨论中，我们将看看完全无损压缩，因为计算机中的大多数数据是不能容忍丢失任何数据的。

**gzip**

The gzip program is used to compress one or more files. When executed, it replaces the original file with a compressed version of the original. The corresponding gunzip program is used to restore compressed files to their original, uncompressed form. Here is an example:

这个 gzip 程序被用来压缩一个或多个文件。当执行 gzip 命令时，则原始文件的压缩版会替代原始文件。 相对应的 gunzip 程序被用来把压缩文件复原为没有被压缩的版本。这里有个例子：

[me@linuxbox ~]$ ls -l /etc > foo.txt

[me@linuxbox ~]$ ls -l foo.\*

-rw-r--r-- 1 me me 15738 2008-10-14 07:15 foo.txt

[me@linuxbox ~]$ gzip foo.txt

[me@linuxbox ~]$ ls -l foo.\*

-rw-r--r-- 1 me me 3230 2008-10-14 07:15 foo.txt.gz

[me@linuxbox ~]$ gunzip foo.txt.gz

[me@linuxbox ~]$ ls -l foo.\*

-rw-r--r-- 1 me me 15738 2008-10-14 07:15 foo.txt

In this example, we create a text file named foo.txt from a directory listing. Next, we run gzip, which replaces the original file with a compressed version named foo.txt.gz. In the directory listing of foo.\*, we see that the original file has been replaced with the compressed version, and that the compressed version about one-fifth the size of the original. We can also see that the compressed file has the same permissions and time stamp as the original.

在这个例子里，我们创建了一个名为 foo.txt 的文本文件，其内容包含一个目录的列表清单。 接下来，我们运行 gzip 命令，它会把原始文件替换为一个叫做 foo.txt.gz 的压缩文件。在 foo.\* 文件列表中，我们看到原始文件已经被压缩文件替代了，并将这个压缩文件大约是原始 文件的五分之一。我们也能看到压缩文件与原始文件有着相同的权限和时间戳。

Next, we run the gunzip program to uncompress the file. Afterward, we can see that the compressed version of the file has been replaced with the original, again with the permissions and time stamp preserved.

接下来，我们运行 gunzip 程序来解压缩文件。随后，我们能见到压缩文件已经被原始文件替代了， 同样地保留了相同的权限和时间戳。

gzip has many options. Here are a few:

gzip 命令有许多选项。这里列出了一些：

|  |  |
| --- | --- |
| *Table 19-1: gzip Options* | |
| Option | Description |
| -c | Write output to standard output and keep original files. May also be specified with --stdout and --to-stdout. |
| -d | Decompress. This causes gzip to act like gunzip. May also be specified with --decompress or --uncompress. |
| -f | Force compression even if compressed version of the original file already exists. May also be specified with --force. |
| -h | Display usage information. May also be specified with --help. |
| -l | List compression statistics for each file compressed. May also be specified with --list. |
| -r | If one or more arguments on the command line are directories, recursively compress files contained within them. May also be specified with --recursive. |
| -t | Test the integrity of a compressed file. May also be specified with --test. |
| -v | Display verbose messages while compressing. May also be specified with --verbose. |
| -number | Set amount of compression. number is an integer in the range of 1 (fastest, least compression) to 9 (slowest, most compression). The values 1 and 9 may also be expressed as --fast and --best, respectively. The default value is 6. |

|  |  |
| --- | --- |
| *表19-1: gzip 选项* | |
| 选项 | 说明 |
| -c | 把输出写入到标准输出，并且保留原始文件。也有可能用--stdout 和--to-stdout 选项来指定。 |
| -d | 解压缩。正如 gunzip 命令一样。也可以用--decompress 或者--uncompress 选项来指定. |
| -f | 强制压缩，即使原始文件的压缩文件已经存在了，也要执行。也可以用--force 选项来指定。 |
| -h | 显示用法信息。也可用--help 选项来指定。 |
| -l | 列出每个被压缩文件的压缩数据。也可用--list 选项。 |
| -r | 若命令的一个或多个参数是目录，则递归地压缩目录中的文件。也可用--recursive 选项来指定。 |
| -t | 测试压缩文件的完整性。也可用--test 选项来指定。 |
| -v | 显示压缩过程中的信息。也可用--verbose 选项来指定。 |
| -number | 设置压缩指数。number 是一个在1（最快，最小压缩）到9（最慢，最大压缩）之间的整数。 数值1和9也可以各自用--fast 和--best 选项来表示。默认值是整数6。 |

Going back to our earlier example:

返回到我们之前的例子中：

[me@linuxbox ~]$ gzip foo.txt

[me@linuxbox ~]$ gzip -tv foo.txt.gz

foo.txt.gz: OK

[me@linuxbox ~]$ gzip -d foo.txt.gz

Here, we replaced the file foo.txt with a compressed version, named foo.txt.gz. Next, we tested the integrity of the compressed version, using the -t and -v options. Finally, we decompressed the file back to its original form. gzip can also be used in interesting ways via standard input and output:

这里，我们用压缩文件来替代文件 foo.txt，压缩文件名为 foo.txt.gz。下一步，我们测试了压缩文件 的完整性，使用了-t 和-v 选项。

[me@linuxbox ~]$ ls -l /etc | gzip > foo.txt.gz

This command creates a compressed version of a directory listing.

这个命令创建了一个目录列表的压缩文件。

The gunzip program, which uncompresses gzip files, assumes that filenames end in the extension .gz, so it’s not necessary to specify it, as long as the specified name is not in conflict with an existing uncompressed file:

这个 gunzip 程序，会解压缩 gzip 文件，假定那些文件名的扩展名是.gz，所以没有必要指定它， 只要指定的名字与现有的未压缩文件不冲突就可以：

[me@linuxbox ~]$ gunzip foo.txt

If our goal were only to view the contents of a compressed text file, we can do this:

如果我们的目标只是为了浏览一下压缩文本文件的内容，我们可以这样做：

[me@linuxbox ~]$ gunzip -c foo.txt | less

Alternately, there is a program supplied with gzip, called zcat, that is equivalent to gunzip with the -c option. It can be used like the cat command on gzip compressed files:

另外，对应于 gzip 还有一个程序，叫做 zcat，它等同于带有-c 选项的 gunzip 命令。 它可以被用来如 cat 命令作用于 gzip 压缩文件：

[me@linuxbox ~]$ zcat foo.txt.gz | less

**Tip:** There is a zless program, too. It performs the same function as the pipeline above.

**小贴士:** 还有一个 zless 程序。它与上面的管道线有相同的功能。

**bzip2**

The bzip2 program, by Julian Seward, is similar to gzip, but uses a different compression algorithm that achieves higher levels of compression at the cost of compression speed. In most regards, it works in the same fashion as gzip. A file compressed with bzip2 is denoted with the extension .bz2:

这个 bzip2 程序，由 Julian Seward 开发，与 gzip 程序相似，但是使用了不同的压缩算法， 舍弃了压缩速度，而实现了更高的压缩级别。在大多数情况下，它的工作模式等同于 gzip。 由 bzip2 压缩的文件，用扩展名 .bz2 来表示：

[me@linuxbox ~]$ ls -l /etc > foo.txt

[me@linuxbox ~]$ ls -l foo.txt

-rw-r--r-- 1 me me 15738 2008-10-17 13:51 foo.txt

[me@linuxbox ~]$ bzip2 foo.txt

[me@linuxbox ~]$ ls -l foo.txt.bz2

-rw-r--r-- 1 me me 2792 2008-10-17 13:51 foo.txt.bz2

[me@linuxbox ~]$ bunzip2 foo.txt.bz2

As we can see, bzip2 can be used the same way as gzip. All the options (except for -r) that we discussed for gzip are also supported in bzip2. Note, however, that the compression level option (-number) has a somewhat different meaning to bzip2. bzip2 comes with bunzip2 and bzcat for decompressing files. bzip2 also comes with the bzip2recover program, which will try to recover damaged .bz2 files.

正如我们所看到的，bzip2 程序使用起来和 gzip 程序一样。我们之前讨论的 gzip 程序的所有选项（除了-r） ，bzip2 程序同样也支持。注意，然而，压缩级别选项（-number）对于 bzip2 程序来说，有少许不同的含义。 伴随着 bzip2 程序，有 bunzip2 和 bzcat 程序来解压缩文件。bzip2 文件也带有 bzip2recover 程序，其会 试图恢复受损的 .bz2 文件。

Don’t Be Compressive Compulsive

不要强迫性压缩

I occasionally see people attempting to compress a file, which has been already compressed with an effective compression algorithm, by doing something like this:

我偶然见到人们试图用高效的压缩算法，来压缩一个已经被压缩过的文件，通过这样做：

**$ gzip picture.jpg**

Don’t do it. You’re probably just wasting time and space! If you apply compression to a file that is already compressed, you will actually end up a larger file. This is because all compression techniques involve some overhead that is added to the file to describe the compression. If you try to compress a file that already contains no redundant information, the compression will not result in any savings to offset the additional overhead.

不要这样。你可能只是在浪费时间和空间！如果你再次压缩已经压缩过的文件，实际上你 会得到一个更大的文件。这是因为所有的压缩技术都会涉及一些开销，文件中会被添加描述 此次压缩过程的信息。如果你试图压缩一个已经不包含多余信息的文件，那么再次压缩不会节省 空间，以抵消额外的花费。

**归档文件**

A common file management task used in conjunction with compression is archiving. Archiving is the process of gathering up many files and bundling them together into a single large file. Archiving is often done as a part of system backups. It is also used when old data is moved from a system to some type of long-term storage.

一个常见的，与文件压缩结合一块使用的文件管理任务是归档。归档就是收集许多文件，并把它们 捆绑成一个大文件的过程。归档经常作为系统备份的一部分来使用。当把旧数据从一个系统移到某 种类型的长期存储设备中时，也会用到归档程序。

**tar**

In the Unix-like world of software, the tar program is the classic tool for archiving files. Its name, short for tape archive, reveals its roots as a tool for making backup tapes. While it is still used for that traditional task, it is equally adept on other storage devices as well. We often see filenames that end with the extension .tar or .tgz which indicate a “plain” tar archive and a gzipped archive, respectively. A tar archive can consist of a group of separate files, one or more directory hierarchies, or a mixture of both. The command syntax works like this:

在类 Unix 的软件世界中，这个 tar 程序是用来归档文件的经典工具。它的名字，是 tape archive 的简称，揭示了它的根源，它是一款制作磁带备份的工具。而它仍然被用来完成传统任务， 它也同样适用于其它的存储设备。我们经常看到扩展名为 .tar 或者 .tgz 的文件，它们各自表示“普通” 的 tar 包和被 gzip 程序压缩过的 tar 包。一个 tar 包可以由一组独立的文件，一个或者多个目录，或者 两者混合体组成。命令语法如下：

tar mode[options] pathname...

where mode is one of the following operating modes (only a partial list is shown here; see the tar man page for a complete list):

这里的 mode 是指以下操作模式（这里只展示了一部分，查看 tar 的手册来得到完整列表）之一：

|  |  |
| --- | --- |
| *Table 19-2: tar Modes* | |
| Mode | Description |
| c | Create an archive from a list of files and/or directories. |
| x | Extract an archive. |
| r | Append specified pathnames to the end of an archive. |
| t | List the contents of an archive. |

|  |  |
| --- | --- |
| *表19-2: tar 模式* | |
| 模式 | 说明 |
| c | 为文件和／或目录列表创建归档文件。 |
| x | 抽取归档文件。 |
| r | 追加具体的路径到归档文件的末尾。 |
| t | 列出归档文件的内容。 |

tar uses a slightly odd way of expressing options, so we’ll need some examples to show how it works. First, let’s re-create our playground from the previous chapter:

tar 命令使用了稍微有点奇怪的方式来表达它的选项，所以我们需要一些例子来展示它是 怎样工作的。首先，让我们重新创建之前我们用过的操练场:

[me@linuxbox ~]$ mkdir -p playground/dir-{00{1..9},0{10..99},100}

[me@linuxbox ~]$ touch playground/dir-{00{1..9},0{10..99},100}/file-{A..Z}

Next, let’s create a tar archive of the entire playground:

下一步，让我们创建整个操练场的 tar 包：

[me@linuxbox ~]$ tar cf playground.tar playground

This command creates a tar archive named playground.tar that contains the entire playground directory hierarchy. We can see that the mode and the f option, which is used to specify the name of the tar archive, may be joined together, and do not require a leading dash. Note, however, that the mode must always be specified first, before any other option.

这个命令创建了一个名为 playground.tar 的 tar 包，其包含整个 playground 目录层次结果。我们 可以看到模式 c 和选项 f，其被用来指定这个 tar 包的名字，模式和选项可以写在一起，而且不 需要开头的短横线。注意，然而，必须首先指定模式，然后才是其它的选项。

To list the contents of the archive, we can do this:

要想列出归档文件的内容，我们可以这样做：

[me@linuxbox ~]$ tar tf playground.tar

For a more detailed listing, we can add the v (verbose) option:

为了得到更详细的列表信息，我们可以添加选项 v：

[me@linuxbox ~]$ tar tvf playground.tar

Now, let’s extract the playground in a new location. We will do this by creating a new directory named foo, and changing the directory and extracting the tar archive:

现在，抽取 tar 包 playground 到一个新位置。我们先创建一个名为 foo 的新目录，更改目录， 然后抽取 tar 包中的文件：

[me@linuxbox ~]$ mkdir foo

[me@linuxbox ~]$ cd foo

[me@linuxbox ~]$ tar xf ../playground.tar

[me@linuxbox ~]$ ls

playground

If we examine the contents of ~/foo/playground, we see that the archive was successfully installed, creating a precise reproduction of the original files. There is one caveat, however: unless you are operating as the superuser, files and directories extracted from archives take on the ownership of the user performing the restoration, rather than the original owner.

如果我们检查 ~/foo/playground 目录中的内容，会看到这个归档文件已经被成功地安装了，就是创建了 一个精确的原始文件的副本。有一个警告，然而：除非你是超级用户，要不然从归档文件中抽取的文件 和目录的所有权由执行此复原操作的用户所拥有，而不属于原始所有者。

Another interesting behavior of tar is the way it handles pathnames in archives. The default for pathnames is relative, rather than absolute. tar does this by simply removing any leading slash from the pathname when creating the archive. To demonstrate, we will recreate our archive, this time specifying an absolute pathname:

tar 命令另一个有趣的行为是它处理归档文件路径名的方式。默认情况下，路径名是相对的，而不是绝对 路径。当创建归档文件的时候，tar 命令会简单地删除路径名开头的斜杠。为了说明问题，我们将会 重新创建我们的归档文件，这次指定一个绝对路径：

[me@linuxbox foo]$ cd

[me@linuxbox ~]$ tar cf playground2.tar ~/playground

Remember, ~/playground will expand into /home/me/playground when we press the enter key, so we will get an absolute pathname for our demonstration. Next, we will extract the archive as before and watch what happens:

记住，当按下回车键后，~/playground 会展开成 /home/me/playground，所以我们将会得到一个 绝对路径名。接下来，和之前一样我们会抽取归档文件，观察发生什么事情：

[me@linuxbox ~]$ cd foo

[me@linuxbox foo]$ tar xf ../playground2.tar

[me@linuxbox foo]$ ls

home playground

[me@linuxbox foo]$ ls home

me

[me@linuxbox foo]$ ls home/me

playground

Here we can see that when we extracted our second archive, it recreated the directory home/me/playground relative to our current working directory, ~/foo, not relative to the root directory, as would have been the case with an absolute pathname. This may seem like an odd way for it to work, but it’s actually more useful this way, as it allows us to extract archives to any location rather than being forced to extract them to their original locations. Repeating the exercise with the inclusion of the verbose option (v) will give a clearer picture of what’s going on.

这里我们看到当我们抽取第二个归档文件时，它重新创建了 home/me/playground 目录， 相对于我们当前的工作目录，~/foo，而不是相对于 root 目录，作为带有绝对路径名的案例。 这看起来似乎是一种奇怪的工作方式，但事实上这种方式很有用，因为这样就允许我们抽取文件 到任意位置，而不是强制地把抽取的文件放置到原始目录下。加上 verbose（v）选项，重做 这个练习，将会展现更加详细的信息。

Let’s consider a hypothetical, yet practical example, of tar in action. Imagine we want to copy the home directory and its contents from one system to another and we have a large USB hard drive that we can use for the transfer. On our modern Linux system, the drive is “automagically” mounted in the /media directory. Let’s also imagine that the disk has a volume name of BigDisk when we attach it. To make the tar archive, we can do the following:

让我们考虑一个假设，tar 命令的实际应用。假定我们想要复制家目录及其内容到另一个系统中， 并且有一个大容量的 USB 硬盘，可以把它作为传输工具。在现代 Linux 系统中， 这个硬盘会被“自动地”挂载到 /media 目录下。我们也假定硬盘中有一个名为 BigDisk 的逻辑卷。 为了制作 tar 包，我们可以这样做：

[me@linuxbox ~]$ sudo tar cf /media/BigDisk/home.tar /home

After the tar file is written, we unmount the drive and attach it to the second computer. Again, it is mounted at /media/BigDisk. To extract the archive, we do this:

tar 包制作完成之后，我们卸载硬盘，然后把它连接到第二个计算机上。再一次，此硬盘被 挂载到 /media/BigDisk 目录下。为了抽取归档文件，我们这样做：

[me@linuxbox2 ~]$ cd /

[me@linuxbox2 /]$ sudo tar xf /media/BigDisk/home.tar

What’s important to see here is that we must first change directory to /, so that the extraction is relative to the root directory, since all pathnames within the archive are relative.

值得注意的一点是，因为归档文件中的所有路径名都是相对的，所以首先我们必须更改目录到根目录下， 这样抽取的文件路径就相对于根目录了。

When extracting an archive, it’s possible to limit what is extracted from the archive. For example, if we wanted to extract a single file from an archive, it could be done like this:

当抽取一个归档文件时，有可能限制从归档文件中抽取什么内容。例如，如果我们想要抽取单个文件， 可以这样实现：

tar xf archive.tar pathname

By adding the trailing pathname to the command, tar will only restore the specified file. Multiple pathnames may be specified. Note that the pathname must be the full, exact relative pathname as stored in the archive. When specifying pathnames, wildcards are not normally supported; however, the GNU version of tar (which is the version most often found in Linux distributions) supports them with the –wildcards option. Here is an example using our previous playground.tar file:

通过给命令添加末尾的路径名，tar 命令就只会恢复指定的文件。可以指定多个路径名。注意 路径名必须是完全的，精准的相对路径名，就如存储在归档文件中的一样。当指定路径名的时候， 通常不支持通配符；然而，GNU 版本的 tar 命令（在 Linux 发行版中最常出现）通过 --wildcards 选项来 支持通配符。这个例子使用了之前 playground.tar 文件：

[me@linuxbox ~]$ cd foo

[me@linuxbox foo]$ tar xf ../playground2.tar --wildcards 'home/me/playground/dir-\\*/file-A'

This command will extract only files matching the specified pathname including the wildcard dir-\*.

这个命令将只会抽取匹配特定路径名的文件，路径名中包含了通配符 dir-\*。

tar is often used in conjunction with find to produce archives. In this example, we will use find to produce a set of files to include in an archive:

tar 命令经常结合 find 命令一起来制作归档文件。在这个例子里，我们将会使用 find 命令来 产生一个文件集合，然后这些文件被包含到归档文件中。

[me@linuxbox ~]$ find playground -name 'file-A' -exec tar rf playground.tar '{}' '+'

Here we use find to match all the files in playground named file-A and then, using the -exec action, we invoke tar in the append mode (r) to add the matching files to the archive playground.tar.

这里我们使用 find 命令来匹配 playground 目录中所有名为 file-A 的文件，然后使用-exec 行为，来 唤醒带有追加模式（r）的 tar 命令，把匹配的文件添加到归档文件 playground.tar 里面。

Using tar with find is a good way of creating incremental backups of a directory tree or an entire system. By using find to match files newer than a timestamp file, we could create an archive that only contains files newer than the last archive, assuming that the timestamp file is updated right after each archive is created.

使用 tar 和 find 命令，来创建逐渐增加的目录树或者整个系统的备份，是个不错的方法。通过 find 命令匹配新于某个时间戳的文件，我们就能够创建一个归档文件，其只包含新于上一个 tar 包的文件， 假定这个时间戳文件恰好在每个归档文件创建之后被更新了。

tar can also make use of both standard input and output. Here is a comprehensive example:

tar 命令也可以利用标准输出和输入。这里是一个完整的例子:

[me@linuxbox foo]$ cd

[me@linuxbox ~]$ find playground -name 'file-A' | tar cf - --files-from=-

| gzip > playground.tgz

In this example, we used the find program to produce a list of matching files and piped them into tar. If the filename “-” is specified, it is taken to mean standard input or output, as needed (by the way, this convention of using “-” to represent standard input/output is used by a number of other programs, too.) The –files-from option (which may be also be specified as -T) causes tar to read its list of pathnames from a file rather than the command line. Lastly, the archive produced by tar is piped into gzip to create the compressed archive playground.tgz. The .tgz extension is the conventional extension given to gzip-compressed tar files. The extension .tar.gz is also used sometimes.

在这个例子里面，我们使用 find 程序产生了一个匹配文件列表，然后把它们管道到 tar 命令中。 如果指定了文件名“-”，则其被看作是标准输入或输出，正是所需（顺便说一下，使用“-”来表示 标准输入／输出的惯例，也被大量的其它程序使用）。这个 --file-from 选项（也可以用 -T 来指定） 导致 tar 命令从一个文件而不是命令行来读入它的路径名列表。最后，这个由 tar 命令产生的归档 文件被管道到 gzip 命令中，然后创建了压缩归档文件 playground.tgz。此 .tgz 扩展名是命名 由 gzip 压缩的 tar 文件的常规扩展名。有时候也会使用 .tar.gz 这个扩展名。

While we used the gzip program externally to produced our compressed archive, modern versions of GNU tar support both gzip and bzip2 compression directly, with the use of the z and j options, respectively. Using our previous example as a base, we can simplify it this way:

虽然我们使用 gzip 程序来制作我们的压缩归档文件，但是现在的 GUN 版本的 tar 命令 ，gzip 和 bzip2 压缩两者都直接支持，各自使用 z 和 j 选项。以我们之前的例子为基础， 我们可以这样简化它：

[me@linuxbox ~]$ find playground -name 'file-A' | tar czf playground.tgz -T -

If we had wanted to create a bzip2 compressed archive instead, we could have done this:

如果我们本要创建一个由 bzip2 压缩的归档文件，我们可以这样做：

[me@linuxbox ~]$ find playground -name 'file-A' | tar cjf playground.tbz -T -

By simply changing the compression option from z to j (and changing the output file’s extension to .tbz to indicate a bzip2 compressed file) we enabled bzip2 compression. Another interesting use of standard input and output with the tar command involves transferring files between systems over a network. Imagine that we had two machines running a Unix-like system equipped with tar and ssh. In such a scenario, we could transfer a directory from a remote system (named remote-sys for this example) to our local system:

通过简单地修改压缩选项，把 z 改为 j（并且把输出文件的扩展名改为 .tbz，来指示一个 bzip2 压缩文件）， 就使 bzip2 命令压缩生效了。另一个 tar 命令与标准输入和输出的有趣使用，涉及到在系统之间经过 网络传输文件。假定我们有两台机器，每台都运行着类 Unix，且装备着 tar 和 ssh 工具的操作系统。 在这种情景下，我们可以把一个目录从远端系统（名为 remote-sys）传输到我们的本地系统中：

[me@linuxbox ~]$ mkdir remote-stuff

[me@linuxbox ~]$ cd remote-stuff

[me@linuxbox remote-stuff]$ ssh remote-sys 'tar cf - Documents' | tar xf -

me@remote-sys’s password:

[me@linuxbox remote-stuff]$ ls

Documents

Here we were able to copy a directory named Documents from the remote system remote-sys to a directory within the directory named remote-stuff on the local system. How did we do this? First, we launched the tar program on the remote system using ssh. You will recall that ssh allows us to execute a program remotely on a networked computer and “see” the results on the local system—the standard output produced on the remote system is sent to the local system for viewing. We can take advantage of this by having tar create an archive (the c mode) and send it to standard output, rather than a file (the f option with the dash argument), thereby transporting the archive over the encrypted tunnel provided by ssh to the local system. On the local system, we execute tar and have it expand an archive (the x mode) supplied from standard input (again, the f option with the dash argument).

这里我们能够从远端系统 remote-sys 中复制目录 Documents 到本地系统名为 remote-stuff 目录中。 我们怎样做的呢？首先，通过使用 ssh 命令在远端系统中启动 tar 程序。你可记得 ssh 允许我们 在远程联网的计算机上执行程序，并且在本地系统中看到执行结果——远端系统中产生的输出结果 被发送到本地系统中查看。我们可以利用。在本地系统中，我们执行 tar 命令，

**zip**

The zip program is both a compression tool and an archiver. The file format used by the program is familiar to Windows users, as it reads and writes .zip files. In Linux, however, gzip is the predominant compression program with bzip2 being a close second.

这个 zip 程序既是压缩工具，也是一个打包工具。这程序使用的文件格式，Windows 用户比较熟悉， 因为它读取和写入.zip 文件。然而，在 Linux 中 gzip 是主要的压缩程序，而 bzip2则位居第二。

In its most basic usage, zip is invoked like this:

在 zip 命令最基本的使用中，可以这样唤醒 zip 命令：

zip options zipfile file...

For example, to make a zip archive of our playground, we would do this:

例如，制作一个 playground 的 zip 版本的文件包，这样做：

[me@linuxbox ~]$ zip -r playground.zip playground

Unless we include the -r option for recursion, only the playground directory (but none of its contents) is stored. Although the addition of the extension .zip is automatic a, we will include the file extension for clarity.

除非我们包含-r 选项，要不然只有 playground 目录（没有任何它的内容）被存储。虽然会自动添加 .zip 扩展名，但为了清晰起见，我们还是包含文件扩展名。

During the creation of the zip archive, zip will normally display a series of messages like this:

在创建 zip 版本的文件包时，zip 命令通常会显示一系列的信息：

adding: playground/dir-020/file-Z (stored 0%)

adding: playground/dir-020/file-Y (stored 0%)

adding: playground/dir-020/file-X (stored 0%)

adding: playground/dir-087/ (stored 0%)

adding: playground/dir-087/file-S (stored 0%)

These messages show the status of each file added to the archive. zip will add files to the archive using one of two storage methods: either it will “store” a file without compression, as shown here, or it will “deflate” the file which performs compression. The numeric value displayed after the storage method indicates the amount of compression achieved. Since our playground only contains empty files, no compression is performed on its contents.

这些信息显示了添加到文件包中每个文件的状态。zip 命令会使用两种存储方法之一，来添加 文件到文件包中：要不它会“store”没有压缩的文件，正如这里所示，或者它会“deflate”文件， 执行压缩操作。在存储方法之后显示的数值表明了压缩量。因为我们的 playground 目录 只是包含空文件，没有对它的内容执行压缩操作。

Extracting the contents of a zip file is straightforward when using the unzip program:

使用 unzip 程序，来直接抽取一个 zip 文件的内容。

[me@linuxbox ~]$ cd foo

[me@linuxbox foo]$ unzip ../playground.zip

One thing to note about zip (as opposed to tar) is that if an existing archive is specified, it is updated rather than replaced. This means that the existing archive is preserved, but new files are added and matching files are replaced. Files may be listed and extracted selectively from a zip archive by specifying them to unzip:

对于 zip 命令（与 tar 命令相反）要注意一点，就是如果指定了一个已经存在的文件包，其被更新 而不是被替代。这意味着会保留此文件包，但是会添加新文件，同时替换匹配的文件。可以列出 文件或者有选择地从一个 zip 文件包中抽取文件，只要给 unzip 命令指定文件名：

[me@linuxbox ~]$ unzip -l playground.zip playground/dir-87/file-Z

Archive: ../playground.zip

Length Date Time Name

0 10-05-08 09:25 playground/dir-87/file-Z

0 1 file

[me@linuxbox ~]$ cd foo

[me@linuxbox foo]$ unzip ./playground.zip playground/dir-87/file-Z

Archive: ../playground.zip

replace playground/dir-87/file-Z? [y]es, [n]o, [A]ll, [N]one,

[r]ename: y

extracting: playground/dir-87/file-Z

Using the -l option causes unzip to merely list the contents of the archive without extracting the file. If no file(s) are specified, unzip will list all files in the archive. The -v option can be added to increase the verbosity of the listing. Note that when the archive extraction conflicts with an existing file, the user is prompted before the file is replaced.

使用-l 选项，导致 unzip 命令只是列出文件包中的内容而没有抽取文件。如果没有指定文件， unzip 程序将会列出文件包中的所有文件。添加这个-v 选项会增加列表的冗余信息。注意当抽取的 文件与已经存在的文件冲突时，会在替代此文件之前提醒用户。

Like tar, zip can make use of standard input and output, though its implementation is somewhat less useful. It is possible to pipe a list of filenames to zip via the -@ option:

像 tar 命令一样，zip 命令能够利用标准输入和输出，虽然它的实施不大有用。通过-@选项，有可能把一系列的 文件名管道到 zip 命令。

[me@linuxbox foo]$ cd

[me@linuxbox ~]$ find playground -name "file-A" | zip -@ file-A.zip

Here we use find to generate a list of files matching the test -name “file-A”, and pipe the list into zip, which creates the archive file-A.zip containing the selected files.

这里我们使用 find 命令产生一系列与“file-A”相匹配的文件列表，并且把此列表管道到 zip 命令， 然后创建包含所选文件的文件包 file-A.zip。

zip also supports writing its output to standard output, but its use is limited because very few programs can make use of the output. Unfortunately, the unzip program, does not accept standard input. This prevents zip and unzip from being used together to perform network file copying like tar.

zip 命令也支持把它的输出写入到标准输出，但是它的使用是有限的，因为很少的程序能利用输出。 不幸地是，这个 unzip 程序，不接受标准输入。这就阻止了 zip 和 unzip 一块使用，像 tar 命令那样， 来复制网络上的文件。

zip can, however, accept standard input, so it can be used to compress the output of other programs:

然而，zip 命令可以接受标准输入，所以它可以被用来压缩其它程序的输出：

[me@linuxbox ~]$ ls -l /etc/ | zip ls-etc.zip -

adding: - (deflated 80%)

In this example we pipe the output of ls into zip. Like tar, zip interprets the trailing dash as “use standard input for the input file.”

在这个例子里，我们把 ls 命令的输出管道到 zip 命令。像 tar 命令，zip 命令把末尾的横杠解释为 “使用标准输入作为输入文件。”

The unzip program allows its output to be sent to standard output when the -p (for pipe) option is specified:

这个 unzip 程序允许它的输出发送到标准输出，当指定了-p 选项之后：

[me@linuxbox ~]$ unzip -p ls-etc.zip | less

We touched on some of the basic things that zip/unzip can do. They both have a lot of options that add to their flexibility, though some are platform specific to other systems. The man pages for both zip and unzip are pretty good and contain useful examples. However, the main use of these programs is for exchanging files with Windows systems, rather than performing compression and archiving on Linux, where tar and gzip are greatly preferred.

我们讨论了一些 zip/unzip 可以完成的基本操作。它们两个都有许多选项，其增加了 命令的灵活性，虽然一些选项只针对于特定的平台。zip 和 unzip 命令的说明手册都相当不错， 并且包含了有用的实例。然而，这些程序的主要用途是为了和 Windows 系统交换文件， 而不是在 Linux 系统中执行压缩和打包操作，tar 和 gzip 程序在 Linux 系统中更受欢迎。

**同步文件和目录**

A common strategy for maintaining a backup copy of a system involves keeping one or more directories synchronized with another directory (or directories) located on either the local system (usually a removable storage device of some kind) or with a remote system. We might, for example, have a local copy of a web site under development and synchronize it from time to time with the “live” copy on a remote web server. In the Unix-like world, the preferred tool for this task is rsync. This program can synchronize both local and remote directories by using the rsync remote-update protocol, which allows rsync to quickly detect the differences between two directories and perform the minimum amount of copying required to bring them into sync. This makes rsync very fast and economical to use, compared to other kinds of copy programs.

维护系统备份的常见策略是保持一个或多个目录与另一个本地系统（通常是某种可移动的存储设备） 或者远端系统中的目录（或多个目录）同步。我们可能，例如有一个正在开发的网站的本地备份， 需要时不时的与远端网络服务器中的文件备份保持同步。在类 Unix 系统的世界里，能完成此任务且 备受人们喜爱的工具是 rsync。这个程序能同步本地与远端的目录，通过使用 rsync 远端更新协议，此协议 允许 rsync 快速地检测两个目录的差异，执行最小量的复制来达到目录间的同步。比起其它种类的复制程序， 这就使 rsync 命令非常快速和高效。

rsync is invoked like this:

rsync 被这样唤醒：

rsync options source destination

where source and destination are one of the following:

这里 source 和 destination 是下列选项之一：

* A local file or directory
* A remote file or directory in the form of [user@]host:path
* A remote rsync server specified with a URI of rsync://[user@]host[:port]/path
* 一个本地文件或目录
* 一个远端文件或目录，以[user@]host:path 的形式存在
* 一个远端 rsync 服务器，由 rsync://[user@]host[:port]/path 指定

Note that either the source or destination must be a local file. Remote to remote copying is not supported.

注意 source 和 destination 两者之一必须是本地文件。rsync 不支持远端到远端的复制

Let’s try rsync out on some local files. First, let’s clean out our foo directory:

让我们试着对一些本地文件使用 rsync 命令。首先，清空我们的 foo 目录：

[me@linuxbox ~]$ rm -rf foo/\*

Next, we’ll synchronize the playground directory with a corresponding copy in foo:

下一步，我们将同步 playground 目录和它在 foo 目录中相对应的副本

[me@linuxbox ~]$ rsync -av playground foo

We’ve included both the -a option (for archiving—causes recursion and preservation of file attributes) and the -v option (verbose output) to make a mirror of the playground directory within foo. While the command runs, we will see a list of the files and directories being copied. At the end, we will see a summary message like this:

我们包括了-a 选项（递归和保护文件属性）和-v 选项（冗余输出）， 来在 foo 目录中制作一个 playground 目录的镜像。当这个命令执行的时候， 我们将会看到一系列的文件和目录被复制。在最后，我们将看到一条像这样的总结信息：

sent 135759 bytes received 57870 bytes 387258.00 bytes/sec

total size is 3230 speedup is 0.02

indicating the amount of copying performed. If we run the command again, we will see a different result:

说明复制的数量。如果我们再次运行这个命令，我们将会看到不同的结果：

[me@linuxbox ~]$ rsync -av playgound foo

building file list ... done

sent 22635 bytes received 20 bytes

total size is 3230 speedup is 0.14

45310.00 bytes/sec

Notice that there was no listing of files. This is because rsync detected that there were no differences between ~/playground and ~/foo/playground, and therefore it didn’t need to copy anything. If we modify a file in playground and run rsync again:

注意到没有文件列表。这是因为 rsync 程序检测到在目录~/playground 和 ~/foo/playground 之间 不存在差异，因此它不需要复制任何数据。如果我们在 playground 目录中修改一个文件，然后 再次运行 rsync 命令：

[me@linuxbox ~]$ touch playground/dir-099/file-Z

[me@linuxbox ~]$ rsync -av playground foo

building file list ... done

playground/dir-099/file-Z

sent 22685 bytes received 42 bytes 45454.00 bytes/sec

total size is 3230 speedup is 0.14

we see that rsync detected the change and copied only the updated file. As a practical example, let’s consider the imaginary external hard drive that we used earlier with tar. If we attach the drive to our system and, once again, it is mounted at / media/BigDisk, we can perform a useful system backup by first creating a directory, named /backup on the external drive and then using rsync to copy the most important stuff from our system to the external drive:

我们看到 rsync 命令检测到更改，并且只是复制了更新的文件。作为一个实际的例子， 让我们考虑一个假想的外部硬盘，之前我们在 tar 命令中用到过的。如果我们再次把此 硬盘连接到我们的系统中，它被挂载到/media/BigDisk 目录下，我们可以执行一个有 用的系统备份了，首先在外部硬盘上创建一个目录，名为/backup，然后使用 rsync 程序 从我们的系统中复制最重要的数据到此外部硬盘上：

[me@linuxbox ~]$ mkdir /media/BigDisk/backup

[me@linuxbox ~]$ sudo rsync -av --delete /etc /home /usr/local /media/BigDisk/backup

In this example, we copied the /etc, /home, and /usr/local directories from our system to our imaginary storage device. We included the –delete option to remove files that may have existed on the backup device that no longer existed on the source device (this is irrelevant the first time we make a backup, but will be useful on subsequent copies.) Repeating the procedure of attaching the external drive and running this rsync command would be a useful (though not ideal) way of keeping a small system backed up. Of course, an alias would be helpful here, too. We could create an alias and add it to our .bashrc file to provide this feature:

在这个例子里，我们把/etc，/home，和/usr/local 目录从我们的系统中复制到假想的存储设备中。 我们包含了–delete 这个选项，来删除可能在备份设备中已经存在但却不再存在于源设备中的文件， （这与我们第一次创建备份无关，但是会在随后的复制操作中有用途）。挂载外部驱动器，运行 rsync 命令，不断重复这个过程，是一个不错的（虽然不理想）方式来保存少量的系统备份文件。 当然，别名会对这个操作更有帮助些。我们将会创建一个别名，并把它添加到.bashrc 文件中， 来提供这个特性：

alias backup='sudo rsync -av --delete /etc /home /usr/local /media/BigDisk/backup'

Now all we have to do is attach our external drive and run the backup command to do the job.

现在我们所做的事情就是连接外部驱动器，然后运行 backup 命令来完成工作。

**在网络间使用 rsync 命令**

One of the real beauties of rsync is that it can be used to copy files over a network. After all, the “r” in rsync stands for “remote.” Remote copying can be done in one of two ways. The first way is with another system that has rsync installed, along with a remote shell program such as ssh. Let’s say we had another system on our local network with a lot of available hard drive space and we wanted to perform our backup operation using the remote system instead of an external drive. Assuming that it already had a directory named /backup where we could deliver our files, we could do this:

rsync 程序的真正好处之一，是它可以被用来在网络间复制文件。毕竟，rsync 中的“r”象征着“remote”。 远程复制可以通过两种方法完成。第一个方法要求另一个系统已经安装了 rsync 程序，还安装了 远程 shell 程序，比如 ssh。比方说我们本地网络中的一个系统有大量可用的硬盘空间，我们想要 用远程系统来代替一个外部驱动器，来执行文件备份操作。假定远程系统中有一个名为/backup 的目录， 其用来存放我们传送的文件，我们这样做：

[me@linuxbox ~]$ sudo rsync -av --delete --rsh=ssh /etc /home /usr/local remote-sys:/backup

We made two changes to our command to facilitate the network copy. First, we added the –rsh=ssh option, which instructs rsync to use the ssh program as its remote shell. In this way, we were able to use an ssh encrypted tunnel to securely transfer the data from the local system to the remote host. Second, we specified the remote host by prefixing its name (in this case the remote host is named remote-sys) to the destination path name.

我们对命令做了两处修改，来方便网络间文件复制。首先，我们添加了--rsh=ssh 选项，其指示 rsync 使用 ssh 程序作为它的远程 shell。以这种方式，我们就能够使用一个 ssh 加密通道，把数据 安全地传送到远程主机中。其次，通过在目标路径名前加上远端主机的名字（在这种情况下， 远端主机名为 remote-sys），来指定远端主机。

The second way that rsync can be used to synchronize files over a network is by using an rysnc server. rsync can be configured to run as a daemon and listen to incoming requests for synchronization. This is often done to allow mirroring of a remote system. For example, Red Hat Software maintains a large repository of software packages under development for its Fedora distribution. It is useful for software testers to mirror this collection during the testing phase of the distribution release cycle. Since files in the repository change frequently (often more than once a day), it is desirable to maintain a local mirror by periodic synchronization, rather than by bulk copying of the repository. One of these repositories is kept at Georgia Tech; we could mirror it using our local copy of rsync and their rsync server like this:

rsync 可以被用来在网络间同步文件的第二种方式是通过使用 rsync 服务器。rsync 可以被配置为一个 守护进程，监听即将到来的同步请求。这样做经常是为了允许一个远程系统的镜像。例如，Red Hat 软件中心为它的 Fedora 发行版，维护着一个巨大的正在开发中的软件包的仓库。对于软件测试人员， 在发行周期的测试阶段，镜像这些软件集合是非常有帮助的。因为仓库中的这些文件会频繁地 （通常每天不止一次）改动，定期同步本地镜像，这是可取的，而不是大量地拷贝软件仓库。 这些软件库之一被维护在 Georgia Tech；我们可以使用本地 rsync 程序和它们的 rsync 服务器来镜像它。

[me@linuxbox ~]$ mkdir fedora-devel

[me@linuxbox ~]$ rsync -av -delete rsync://rsync.gtlib.gatech.edu/fedora-linux-

core/development/i386/os fedora-devel

In this example, we use the URI of the remote rsync server, which consists of a protocol (rsync://), followed by the remote host name (rsync.gtlib.gatech.edu), followed by the pathname of the repository.

在这个例子里，我们使用了远端 rsync 服务器的 URI，其由协议（rsync://），远端主机名 （rsync.gtlib.gatech.edu），和软件仓库的路径名组成。

**拓展阅读**

* The man pages for all of the commands discussed here are pretty clear and contain useful examples. In addition, the GNU Project has a good online manual for its version of tar. It can be found here:
* 在这里讨论的所有命令的手册文档都相当清楚明白，并且包含了有用的例子。另外， GNU 版本的 tar 命令有一个不错的在线文档。可以在下面链接处找到：

<http://www.gnu.org/software/tar/manual/index.html>