第十五章 软件包管理

If we spend any time in the Linux community, we hear many opinions as to which of the many Linux distributions is “best.” Often, these discussions get really silly, focusing on such things as the prettiness of the desktop background (some people won’t use Ubuntu because its default color scheme is brown!) and other trivial matters.

如果我们花些时间在 Linux 社区里，我们会得知很多针对, 类如在众多 Linux 发行版中哪个是最好的(等问题的)看法。 这些集中在像这些事情上的讨论，比方说最漂亮的桌面背景（一些人不使用 Ubuntu， 只是因为 Ubuntu 默认主题颜色是棕色的！）和其它的琐碎东西，经常变得非常无聊。

The most important determinant of distribution quality is the packaging system and the vitality of the distribution’s support community. As we spend more time with Linux, we see that its software landscape is extremely dynamic. Things are constantly changing. Most of the top-tier Linux distributions release new versions every six months and many individual program updates every day. To keep up with this blizzard of software, we need good tools for package management.

Linux 发行版本质量最重要的决定因素是软件包管理系统和其支持社区的持久性。随着我们 花更多的时间在 Linux 上，我们会发现它的软件园地是非常动态的。软件不断变化。大多数一线 Linux 发行版每隔六个月发布一个新版本，并且许多独立的程序每天都会更新。为了能和这些 如暴风雪一般多的软件保持联系，我们需要一些好工具来进行软件包管理。

Package management is a method of installing and maintaining software on the system. Today, most people can satisfy all of their software needs by installing packages from their Linux distributor. This contrasts with the early days of Linux, when one had to download and compile source code in order to install software. Not that there is anything wrong with compiling source code; in fact, having access to source code is the great wonder of Linux. It gives us (and everybody else) the ability to examine and improve the system. It’s just that having a pre-compiled package is faster and easier to deal with. In this chapter, we will look at some of the command line tools used for package management. While all of the major distributions provide powerful and sophisticated graphical programs for maintaining the system, it is important to learn about the command line programs, too. They can perform many tasks that are difficult (or impossible) to do with their graphical counterparts.

软件包管理是指系统中一种安装和维护软件的方法。今天，通过从 Linux 发行版中安装的软件包， 已能满足许多人所有需要的软件。这不同于早期的 Linux，人们需要下载和编辑源码来安装软件。 编辑源码没有任何问题，事实上，拥有对源码的访问权限是 Linux 的伟大奇迹。它赋予我们（ 其它每个人）才干来检测和提高系统性能。只是若有一个预先编译好的软件包处理起来要相对 容易快速些。这章中，我们将查看一些用于包管理的命令行工具。虽然所有主流 Linux 发行版都 提供了强大且精致的图形管理程序来维护系统，但是学习命令行程序也非常重要。因为它们 可以完成许多让图形化管理程序处理起来困难（或者不可能）的任务。

**打包系统**

Different distributions use different packaging systems and as a general rule, a package intended for one distribution is not compatible with another distribution. Most distributions fall into one of two camps of packaging technologies: the Debian “.deb” camp and the Red Hat “.rpm” camp. There are some important exceptions such as Gentoo, Slackware, and Foresight, but most others use one of these two basic systems.

不同的 Linux 发行版使用不同的打包系统，一般而言，大多数发行版分别属于两大包管理技术阵营： Debian 的”.deb”，和红帽的”.rpm”。也有一些重要的例外，比方说 Gentoo， Slackware，和 Foresight，但大多数会使用这两个基本系统中的一个。

|  |  |
| --- | --- |
| *Table 15-1: Major Packaging System Families* | |
| Packaging System | Distributions (Partial Listing) |
| Debian Style (.deb) | Debian, Ubuntu, Xandros, Linspire |
| Red Hat Style (.rpm) | Fedora, CentOS, Red Hat Enterprise Linux, OpenSUSE, Mandriva, PCLinuxOS |

|  |  |
| --- | --- |
| *表15-1: 主要的包管理系统家族* | |
| 包管理系统 | 发行版 (部分列表) |
| Debian Style (.deb) | Debian, Ubuntu, Xandros, Linspire |
| Red Hat Style (.rpm) | Fedora, CentOS, Red Hat Enterprise Linux, OpenSUSE, Mandriva, PCLinuxOS |

**软件包管理系统是怎样工作的**

The method of software distribution found in the proprietary software industry usually entails buying a piece of installation media such as an “install disk” and then running an “installation wizard” to install a new application on the system.

在专有软件产业中找到的软件发布方法通常需要买一张安装媒介，比方说”安装盘”，然后运行 “安装向导”，来在系统中安装新的应用程序。

Linux doesn’t work that way. Virtually all software for a Linux system will be found on the Internet. Most of it will be provided by the distribution vendor in the form of package files and the rest will be available in source code form that can be installed manually. We’ll talk a little about how to install software by compiling source code in a later chapter.

Linux 不是这样。Linux 系统中几乎所有的软件都可以在互联网上找到。其中大多数软件由发行商以 包文件的形式提供，剩下的则以源码形式存在，可以手动安装。在后面章节里，我们将会谈谈怎样 通过编译源码来安装软件。

**包文件**

The basic unit of software in a packaging system is the package file. A package file is a compressed collection of files that comprise the software package. A package may consist of numerous programs and data files that support the programs. In addition to the files to be installed, the package file also includes metadata about the package, such as a text description of the package and its contents. Additionally, many packages contain pre- and post-installation scripts that perform configuration tasks before and after the package installation.

在包管理系统中软件的基本单元是包文件。包文件是一个构成软件包的文件压缩集合。一个软件包 可能由大量程序以及支持这些程序的数据文件组成。除了安装文件之外，软件包文件也包括 关于这个包的元数据，如软件包及其内容的文本说明。另外，许多软件包还包括预安装和安装后脚本， 这些脚本用来在软件安装之前和之后执行配置任务。

Package files are created by a person known as a package maintainer, often (but not always) an employee of the distribution vendor. The package maintainer gets the software in source code form from the upstream provider (the author of the program), compiles it, and creates the package metadata and any necessary installation scripts. Often, the package maintainer will apply modifications to the original source code to improve the program’s integration with the other parts of the Linux distribution.

软件包文件是由软件包维护者创建的，他通常是（但不总是）一名软件发行商的雇员。软件维护者 从上游提供商（程序作者）那里得到软件源码，然后编辑源码，创建软件包元数据以及所需要的 安装脚本。通常，软件包维护者要把所做的修改应用到最初的源码当中，来提高此软件与 Linux 发行版其它部分的融合性。

**资源库**

While some software projects choose to perform their own packaging and distribution, most packages today are created by the distribution vendors and interested third parties. Packages are made available to the users of a distribution in central repositories that may contain many thousands of packages, each specially built and maintained for the distribution.

虽然某些软件项目选择执行他们自己的打包和发布策略，但是现在大多数软件包是由发行商和感兴趣 的第三方创建的。系统发行版的用户可以在一个中心资源库中得到这些软件包，这个资源库可能 包含了成千上万个软件包，每一个软件包都是专门为这个系统发行版建立和维护的。

A distribution may maintain several different repositories for different stages of the software development life cycle. For example, there will usually be a “testing” repository that contains packages that have just been built and are intended for use by brave souls who are looking for bugs before they are released for general distribution. A distribution will often have a “development” repository where work-in-progress packages destined for inclusion in the distribution’s next major release are kept.

因软件开发生命周期不同阶段的需要，一个系统发行版可能维护着几个不同的资源库。例如，通常会 有一个”测试”资源库，其中包含刚刚建立的软件包，它们想要勇敢的用户来使用， 在这些软件包正式发布之前，让用户查找错误。系统发行版经常会有一个”开发”资源库， 这个资源库中保存着注定要包含到下一个主要版本中的半成品软件包。

A distribution may also have related third-party repositories. These are often needed to supply software that, for legal reasons such as patents or DRM anti-circumvention issues, cannot be included with the distribution. Perhaps the best known case is that of encrypted DVD support, which is not legal in the United States. The third-party repositories operate in countries where software patents and anti-circumvention laws do not apply. These repositories are usually wholly independent of the distribution they support and to use them, one must know about them and manually include them in the configuration files for the package management system.

一个系统发行版可能也会拥有相关第三方的资源库。这些资源库需要支持一些因法律原因， 比如说专利或者是 DRM 反规避问题，而不能被包含到发行版中的软件。可能最著名的案例就是 那个加密的 DVD 支持，在美国这是不合法的。第三方资源库在这些软件专利和反规避法案不 生效的国家中起作用。这些资源库通常完全地独立于它们所支持的资源库，要想使用它们， 你必须了解它们，手动地把它们包含到软件包管理系统的配置文件中。

**依赖性**

Programs seldom “standalone;” rather they rely on the presence of other software components to get their work done. Common activities, such as input/output for example, are handled by routines shared by many programs. These routines are stored in what are called shared libraries, which provide essential services to more than one program. If a package requires a shared resource such as a shared library, it is said to have a dependency. Modern package management systems all provide some method of dependency resolution to ensure that when a package is installed, all of its dependencies are installed, too.

程序很少是”孤立的”，而是依赖于其它软件组件来完成它们的工作。常见活动，以 输入/输出为例，就是由共享程序例程来处理的。这些程序例程存储在共享库中，共享库不只 为一个程序提供基本服务。如果一个软件包需要共享资源，比如说共享库，据说就有一个依赖。 现代的软件包管理系统都提供了一些依赖项解析方法，以此来确保当安装软件包时，也安装了 其所有的依赖程序。

**上层和底层软件包工具**

Package management systems usually consist of two types of tools: low-level tools which handle tasks such as installing and removing package files, and high-level tools that perform metadata searching and dependency resolution. In this chapter, we will look at the tools supplied with Debian-style systems (such as Ubuntu and many others) and those used by recent Red Hat products. While all Red Hat-style distributions rely on the same low-level program (rpm), they use different high-level tools. For our discussion, we will cover the high-level program yum, used by Fedora, Red Hat Enterprise Linux, and CentOS. Other Red Hat-style distributions provide high-level tools with comparable features.

软件包管理系统通常由两种工具类型组成：底层工具用来处理这些任务，比方说安装和删除软件包文件， 和上层工具，完成元数据搜索和依赖解析。在这一章中，我们将看一下由 Debian 风格的系统 （比如说 Ubuntu，还有许多其它系统）提供的工具，还有那些由 Red Hat 产品使用的工具。虽然所有基于 Red Hat 风格的发行版都依赖于相同的底层程序（rpm）, 但是它们却使用不同的上层工具。我们将研究上层程序 yum 供我们讨论，Fedora, Red Hat 企业版，和 CentOs 都是使用 yum。其它基于 Red Hat 风格的发行版提供了带有可比较特性的上层工具。

|  |  |  |
| --- | --- | --- |
| *Table15- 2: Packaging System Tools* | | |
| Distributions | Low-Level Tools | High-Level Tools |
| Debian-Style | dpkg | apt-get, aptitude |
| Fedora, Red Hat Enterprise Linux, CentOS | rpm | yum |

|  |  |  |
| --- | --- | --- |
| *表15-2: 包管理工具* | | |
| 发行版 | 底层工具 | 上层工具 |
| Debian-Style | dpkg | apt-get, aptitude |
| Fedora, Red Hat Enterprise Linux, CentOS | rpm | yum |

**常见软件包管理任务**

There are many operations that can be performed with the command line package management tools. We will look at the most common. Be aware that the low-level tools also support creation of package files, an activity outside the scope of this book. In the discussion below, the term “package\_name” refers to the actual name of a package rather than the term “package\_file,” which is the name of the file that contains the package.

通过命令行软件包管理工具可以完成许多操作。我们将会看一下最常用的工具。注意底层工具也 支持软件包文件的创建，这个话题超出了本书叙述的范围。在以下的讨论中，”package\_name” 这个术语是指软件包实际名称，而不是指”package\_file”，它是包含在软件包中的文件名。

**查找资源库中的软件包**

Using the high-level tools to search repository metadata, a package can be located based on its name or description.

使用上层工具来搜索资源库元数据，可以根据软件包的名字和说明来定位它。

|  |  |
| --- | --- |
| *Table 15-3: Package Search Commands* | |
| Style | Command(s) |
| Debian | apt-get update; apt-cache search search\_string |
| Red Hat | yum search search\_string |

|  |  |
| --- | --- |
| *表15-3: 软件包查找工具* | |
| 风格 | 命令 |
| Debian | apt-get update; apt-cache search search\_string |
| Red Hat | yum search search\_string |

Example: To search a yum repository for the emacs text editor, this command could be used:

例如：搜索一个 yum 资源库来查找 emacs 文本编辑器，使用以下命令：

yum search emacs

**从资源库中安装一个软件包**

High-level tools permit a package to be downloaded from a repository and installed with full dependency resolution.

上层工具允许从一个资源库中下载一个软件包，并经过完全依赖解析来安装它。

|  |  |
| --- | --- |
| *Table 15-4: Package Installation Commands* | |
| Style | Command(s) |
| Debian | apt-get update; apt-get install package\_name |
| Red Hat | yum install package\_name |

|  |  |
| --- | --- |
| *表15-4: 软件包安装命令* | |
| 风格 | 命令 |
| Debian | apt-get update; apt-get install package\_name |
| Red Hat | yum install package\_name |

Example: To install the emacs text editor from an apt repository:

例如：从一个 apt 资源库来安装 emacs 文本编辑器：

apt-get update; apt-get install emacs

**通过软件包文件来安装软件**

If a package file has been downloaded from a source other than a repository, it can be installed directly (though without dependency resolution) using a low-level tool.

如果从某处而不是从资源库中下载了一个软件包文件，可以使用底层工具来直接（没有经过依赖解析）安装它。

|  |  |
| --- | --- |
| *Table 15-5: Low-Level Package Installation Commands* | |
| Style | Command(s) |
| Debian | dpkg --install package\_file |
| Red Hat | rpm -i package\_file |

|  |  |
| --- | --- |
| *表15-5: 底层软件包安装命令* | |
| 风格 | 命令 |
| Debian | dpkg --install package\_file |
| Red Hat | rpm -i package\_file |

Example: If the emacs-22.1-7.fc7-i386.rpm package file had been downloaded from a non-repository site, it would be installed this way:

例如：如果已经从一个并非资源库的网站下载了软件包文件 emacs-22.1-7.fc7-i386.rpm， 则可以通过这种方法来安装它：

rpm -i emacs-22.1-7.fc7-i386.rpm

Note: Since this technique uses the low-level rpm program to perform the installation, no dependency resolution is performed. If rpm discovers a missing dependency, rpm will exit with an error.

注意：因为这项技术使用底层的 rpm 程序来执行安装任务，所以没有运行依赖解析。 如果 rpm 程序发现缺少了一个依赖，则会报错并退出。

**卸载软件**

Packages can be uninstalled using either the high-level or low-tools. The high-level tools are shown below.

可以使用上层或者底层工具来卸载软件。下面是可用的上层工具。

|  |  |
| --- | --- |
| *Table15-6: Package Removal Commands* | |
| Style | Command(s) |
| Debian | apt-get remove package\_name |
| Red Hat | yum erase package\_name |

|  |  |
| --- | --- |
| *表15-6: 软件包删除命令* | |
| 风格 | 命令 |
| Debian | apt-get remove package\_name |
| Red Hat | yum erase package\_name |

Example: To uninstall the emacs package from a Debian-style system:

例如：从 Debian 风格的系统中卸载 emacs 软件包：

apt-get remove emacs

**经过资源库来更新软件包**

The most common package management task is keeping the system up-to-date with the latest packages. The high-level tools can perform this vital task in one single step.

最常见的软件包管理任务是保持系统中的软件包都是最新的。上层工具仅需一步就能完成 这个至关重要的任务。

|  |  |
| --- | --- |
| *Table 15-7: Package Update Commands* | |
| Style | Command(s) |
| Debian | apt-get update; apt-get upgrade |
| Red Hat | yum update |

|  |  |
| --- | --- |
| *表15-7: 软件包更新命令* | |
| 风格 | 命令 |
| Debian | apt-get update; apt-get upgrade |
| Red Hat | yum update |

Example: To apply any available updates to the installed packages on a Debian-style system:

例如：更新安装在 Debian 风格系统中的软件包：

apt-get update; apt-get upgrade

**经过软件包文件来升级软件**

If an updated version of a package has been downloaded from a non-repository source, it can be installed, replacing the previous version:

如果已经从一个非资源库网站下载了一个软件包的最新版本，可以安装这个版本，用它来 替代先前的版本：

|  |  |
| --- | --- |
| *Table 15-8: Low-Level Package Upgrade Commands* | |
| Style | Command(s) |
| Debian | dpkg --install package\_file |
| Red Hat | rpm -U package\_file |

|  |  |
| --- | --- |
| *表15-8: 底层软件包升级命令* | |
| 风格 | 命令 |
| Debian | dpkg --install package\_file |
| Red Hat | rpm -U package\_file |

Example: Updating an existing installation of emacs to the version contained in the package file emacs-22.1-7.fc7-i386.rpm on a Red Hat system:

例如：把 Red Hat 系统中所安装的 emacs 的版本更新到软件包文件 emacs-22.1-7.fc7-i386.rpmz 所包含的 emacs 版本。

rpm -U emacs-22.1-7.fc7-i386.rpm

Note: dpkg does not have a specific option for upgrading a package versus installing one as rpm does.

注意：rpm 程序安装一个软件包和升级一个软件包所用的选项是不同的，而 dpkg 程序所用的选项是相同的。

**列出所安装的软件包**

These commands can be used to display a list of all the packages installed on the system:

下表中的命令可以用来显示安装到系统中的所有软件包列表：

|  |  |
| --- | --- |
| *Table 15-9: Package Listing Commands* | |
| Style | Command(s) |
| Debian | dpkg --list |
| Red Hat | rpm -qa |

|  |  |
| --- | --- |
| *表15-9: 列出所安装的软件包命令* | |
| 风格 | 命令 |
| Debian | dpkg --list |
| Red Hat | rpm -qa |

**确定是否安装了一个软件包**

These low-level tools can be used to display whether a specified package is installed:

这些底端工具可以用来显示是否安装了一个指定的软件包：

|  |  |
| --- | --- |
| *Table 15-10: Package Status Commands* | |
| Style | Command(s) |
| Debian | dpkg --status package\_name |
| Red Hat | rpm -q package\_name |

|  |  |
| --- | --- |
| *表15-10: 软件包状态命令* | |
| 风格 | 命令 |
| Debian | dpkg --status package\_name |
| Red Hat | rpm -q package\_name |

Example: To determine if the emacs package is installed on a Debian style system:

例如：确定是否 Debian 风格的系统中安装了这个 emacs 软件包：

dpkg --status emacs

**显示所安装软件包的信息**

If the name of an installed package is known, the following commands can be used to display a description of the package:

如果知道了所安装软件包的名字，使用以下命令可以显示这个软件包的说明信息：

|  |  |
| --- | --- |
| *Table 15-11: Package Information Commands* | |
| Style | Command(s) |
| Debian | apt-cache show package\_name |
| Red Hat | yum info package\_name |

|  |  |
| --- | --- |
| *表15-11: 查看软件包信息命令* | |
| 风格 | 命令 |
| Debian | apt-cache show package\_name |
| Red Hat | yum info package\_name |

Example: To see a description of the emacs package on a Debian-style system:

例如：查看 Debian 风格的系统中 emacs 软件包的说明信息：

apt-cache show emacs

**查找安装了某个文件的软件包**

To determine what package is responsible for the installation of a particular file, the following commands can be used:

确定哪个软件包对所安装的某个特殊文件负责，使用下表中的命令：

|  |  |
| --- | --- |
| *Table 15-12: Package File Identification Commands* | |
| Style | Command(s) |
| Debian | dpkg --search file\_name |
| Red Hat | rpm -qf file\_name |

|  |  |
| --- | --- |
| *表15-12: 包文件识别命令* | |
| 风格 | 命令 |
| Debian | dpkg --search file\_name |
| Red Hat | rpm -qf file\_name |

Example: To see what package installed the /usr/bin/vim file on a Red Hat system:

例如：在 Red Hat 系统中，查看哪个软件包安装了/usr/bin/vim 这个文件

rpm -qf /usr/bin/vim

**总结归纳**

In the chapters that follow, we will explore many different programs covering a wide range of application areas. While most of these programs are commonly installed by default, we may need to install additional packages if necessary programs are not already installed on our system. With our newfound knowledge (and appreciation) of package management, we should have no problem installing and managing the programs we need.

在随后的章节里面，我们将探讨许多不同的程序，这些程序涵盖了广泛的应用程序领域。虽然 大多数程序一般是默认安装的，但是若所需程序没有安装在系统中，那么我们可能需要安装额外的软件包。 通过我们新学到的（和了解的）软件包管理知识，我们应该没有问题来安装和管理所需的程序。

The Linux Software Installation Myth

Linux 软件安装谣言

People migrating from other platforms sometimes fall victim to the myth that software is somehow difficult to install under Linux and that the variety of packaging schemes used by different distributions is a hindrance. Well, it is a hindrance, but only to proprietary software vendors who wish to distribute binary- only versions of their secret software.

从其它平台迁移过来的用户有时会成为谣言的受害者，说是在 Linux 系统中，安装软件有些 困难，并且不同系统发行版所使用的各种各样的打包方案是一个障碍。唉，它是一个障碍， 但只是针对于那些希望把他们的秘密软件只以二进制版本发行的专有软件供应商。

The Linux software ecosystem is based on the idea of open source code. If a program developer releases source code for a product, it is likely that a person associated with a distribution will package the product and include it in their repository. This method ensures that the product is well integrated into the distribution and the user is given the convenience of “one-stop shopping” for software, rather than having to search for each product’s web site.

Linux 软件生态系统是基于开放源代码理念。如果一个程序开发人员发布了一款产品的 源码，那么与系统发行版相关联的开发人员可能就会把这款产品打包，并把它包含在 他们的资源库中。这种方法保证了这款产品能很好地与系统发行版整合在一起，同时为用户 “一站式采购”软件提供了方便，从而用户不必去搜索每个产品的网站。

Device drivers are are handled in much the same way, except that instead of being separate items in a distribution’s repository, they become part of the Linux kernel itself. Generally speaking, there is no such thing as a “driver disk” in Linux. Either the kernel supports a device or it doesn’t, and the Linux kernel supports a lot of devices. Many more, in fact, than Windows does. Of course, this is of no consolation if the particular device you need is not supported. When that happens, you need to look at the cause. A lack of driver support is usually caused by one of three things:

设备驱动差不多也以同样的方式来处理，但它们不是系统发行版资源库中单独的项目， 它们本身是 Linux 系统内核的一部分。一般来说，在 Linux 当中没有一个类似于“驱动盘”的东西。 要不内核支持一个设备，要不不支持，反正 Linux 内核支持很多设备，事实上，多于 Windows 所支持的设备数目。当然，如果你需要的特定设备不被支持，这里也没有安慰。当那种情况 发生时，你需要查找一下原因。缺少驱动程序支持通常是由以下三种情况之一导致：

1. **The device is too new.** Since many hardware vendors don’t actively support Linux development, it falls upon a member of the Linux community to write the kernel driver code. This takes time.
2. **The device is too exotic.** Not all distributions include every possible device driver. Each distribution builds their own kernels, and since kernels are very configurable (which is what makes it possible to run Linux on everything from wristwatches to mainframes) they may have overlooked a particular device. By locating and downloading the source code for the driver, it is possible for you (yes, you) to compile and install the driver yourself. This process is not overly difficult, but it is rather involved. We’ll talk about compiling software in a later chapter.
3. **The hardware vendor is hiding something.** They have neither released source code for a Linux driver, nor have they released the technical documentation for somebody to create one for them. This means that the hardware vendor is trying to keep the programming interfaces to the device a secret. Since we don’t want secret devices in our computers, I suggest that you remove the offending hardware and pitch it into the trash, with your other useless items.
4. **设备太新。** 因为许多硬件供应商没有积极地支持 Linux 的发展，那么编写内核 驱动代码的任务就由一些 Linux 社区来承担，而这需要花费时间。
5. **设备太奇异。** 不是所有的发行版都包含每个可能的设备驱动。每个发行版会建立 它们自己的内核，因为内核是可以配置的（这使得从手表到主机的每台设备上运行 Linux 成为可能）， 这样它们可能会忽略某个特殊设备。通过定位和下载驱动程序的源码，可能需要你自己（是的，由你） 来编译和安装驱动。这个过程不是很难，而是参与。我们将在随后的章节里来讨论编译软件。
6. **硬件供应商隐藏信息。** 他们既不发布应用于 Linux 系统的驱动程序代码， 也不发布技术文档来让某人创建它。这意味着硬件供应商试图保密此设备的程序接口。因为我们 不想在计算机中使用保密的设备，所以我建议删除这令人厌恶的软件， 把它和其它无用的项目都仍到垃圾桶里。