APT HOWTO (Obsolete Documentation)

Gustavo Noronha Silva <kov@debian.org>

刘浩 <iamlyoo@163.com>

1.8.10.4 - 2005年3月

摘要

这篇文档试图让用户对于Debian包管理工具APT的工作方式有一个很好的理解。它的目标是让新的Debian用户更容易上手，也让那些想帮助那些想更深入理解如何管理他们的系统的人。它为Debian项目而编写，目的是提高此发行版对它的用户的支持水平。

版权声明

Contents

Chapter 1

导言

最初只有.tar.gz的打包文件，用户必须编译每个他想在GNU/Linux上运行的软件。用户们普遍认为系统很有必要提供一种方法来管理这些安装在机器上的软件包，当Debian诞生时，这样一个管理工具也就应运而生，它被命名为dpkg。从而著名的“package”概念第一次出现在GNU/Linux系统中，稍后Red Hat才决定开发自己的“rpm”包管理系统。

很快一个新的问题难倒了GNU/Linux制作者，他们需要一个快速、实用、高效的方法来安装软件包，当软件包更新时，这个工具应该能自动管理关联文件和维护已有配置文件。Debian在次率先解决了这个问题，APT(Advanced Packaging Tool)诞生了。APT后来还被Conectiva改造用来管理rpm，并被其它Linux发行版本采用为它们的软件包管理工具。

本文档不打算讲解apt-rpm相关知识，因为Conectiva移植的APT已很有名了，不过提供有关这部分的补充文档还是欢迎的。

本文档是基于Debian下一个发行版Sarge的。

Chapter 2

基础设置

2.1 /etc/apt/sources.list文件

作为操作的一部分，APT使用一个文件列出可获得软件包的镜像站点地址，这个文件就是/etc/apt/sources.list。

文件中的各项信息通常按如下格式列出：

```
deb http://host/debian distribution section1 section2 section3
deb-src http://host/debian distribution section1 section2 section3
```

当然，上面所列的地址项都是假设的且不应该使用它们。每行的第一个单词deb或deb-src描述了文件类型：目录中包含的是二进制软件包(deb)，即我们通常使用的已编译好的软件包；或包含的是源码包(deb-src)，源码包包含源程序编码、Debian控制文件(.dsc)和“Debian化”该程序所做更改的记录文件diff.gz。

在Debian缺省的sources.list中通常是如下内容：

```
# See sources.list(5) for more information, especially
# Remember that you can only use http, ftp or file URIs
# CDROMs are managed through the apt-cdrom tool.
deb http://http.us.debian.org/debian stable main contrib non-free
deb http://non-us.debian.org/debian-non-US stable/non-US main contrib non-fre
deb http://security.debian.org stable/updates main contrib non-free

# Uncomment if you want the apt-get source function to work
#deb-src http://http.us.debian.org/debian stable main contrib non-free
#deb-src http://non-us.debian.org/debian-non-US stable/non-US main contrib no
```

这些是Debian基本安装所需的软件包来源地址。第一个deb行指向官方正式软件包来源，第二个行指向non-US软件包来源，第三行指向Debian安全补丁更新包来源。

最后两行被注释掉了(在句首加“#”)，所以apt-get将忽略它们。这些是deb-src行指向Debian源码包来源，如果你常下载程序源码来测试或重编译，可取消对它们的注释。

/etc/apt/sources.list文件可包含多种类型的地址，APT知道如何处理这些不同的地址类型：http、ftp、file(本地文件，例如：一个加载了ISO9600文件系统的目录)和ssh。

别忘了在修改完/etc/apt/sources.list文件后运行apt-get使更改生效。你必须完成这个步骤，以便让APT从你指定的地方获得新的软件包列表。

2.2 如何在本地使用APT

有时你硬盘上有许多.deb软件包，你会希望通过APT来安装它们，以便让它去处理软件包间复杂的依赖关系。

想这么做，就建一个目录，将所有你想要安装的.deb文件放入其中。例如：

```
# mkdir /root/debs
```

你可以使用一个override文件直接去修改软件包中控制文件中的定义，使之符合你的软件储藏库管理规则。在这个覆盖文件中，你可能希望定义一些选项来覆盖那些软件包的定义，如下所示：

```
package priority section
```

package是软件包的名称，priority有三个级别low、medium或high，section是软件包所属的section，重载文件可任意命名，文件名将在接下来的步骤中做为参数传递给dpkg-scanpackages。如果你不想写重载文件，只需在调用dpkg-scanpackages时使用/dev/null就行了。

仍是在/root目录下执行：

```
# dpkg-scanpackages debs file | gzip > debs/Packages.gz
```

在上述的命令行中，*file*为override文件，命令生成一个Packages.gz文件，它包含了APT所需的各种软件包信息。最后，如果要使用这些软件包，加上：

```
deb file:/root debs/
```

完成了上面的工作，就可以通常那样使用APT命令操作这些软件包了。你可以使用同样的方法生成一个源码库，但请记住你需要将.orig.tar.gz文件、.dsc文件和.diff.gz文件包含在目录中，同时必须生成Source.gz文件而不是Packages.gz文件。所使用的命令也不相同，要使用dpkg-scansources，命令如下所示：

```
# dpkg-scansources debs | gzip > debs/Sources.gz
```

Notice that dpkg-scansources doesn't need an override file. The sources.list's line is:

```
deb-src file:/root debs/
```

2.3 选择最佳镜像发布站点加入source.list文件：netselect，netselect-apt

一个新用户经常问到的问题："该将哪个Debian镜像发布站点加入source.list文件？"。有很多方法来选择镜像发布站点，专家们可能会写一个脚本去测试不同站点的ping时间。不过其实有一个程序可以帮你：**netselect**。

要安装netselect，通常使用：

```
# apt-get install netselect
```

不带参数运行它时会显示它的帮助信息。运行它时加上以空格分隔的镜像主机列表，它会返回一个分值和列表中的一个主机名。这个分值通过评估ping时间和hops数(一个网络请求报文到达目标主机所经过的转发主机的个数)得出，它与镜像站点预计下载速度成反比(数值越小越好)。返回的主机名是主机列表中得分最低的那个(查看列表中所以主机的得分情况可使用-vv选项）。看出下的例子：

```
# netselect ftp.debian.org http.us.debian.org ftp.at.debian.org download.unes
  365 ftp.debian.org.br
#
```

它表示，在netselect后列出的所有主机中，ftp.debian.org.br是下载速度最快的主机，其得分为365。(注意！！这是在我电脑上的测试结果，不同的网络节点网速会大不相同，所以这个分值不一定适用于其它电脑)

现在将netselect找到的连接速度最快的镜像站点加入/etc/apt/sources.list文件(参考'/etc/apt/sources.list文件' on page 3)并按照'软件包管理' on page 7中的技巧来做。

注意：镜像站点列表通常包含在文件 http://www.debian.org/mirror/mirrors_full中。

从0.3.ds1版开始，netselect源码包中包含了**netselect-apt**二进制包，它使上述操作自动完成。只需将发布目录树做为参数(默认为stable)输入，sources.list文件就会生成速度最快的main和non-US镜像站点列表，并保存在当前目录下。下面的例子生成一个包含stable发布镜像站点列表的sources.list：

```
# ls sources.list
ls: sources.list: File or directory not found
# netselect-apt stable
(...)
# ls -l sources.list
sources.list
#
```

记住：sources.list生成在当前目录下，必须将其移至/etc/apt目录。

接着，按照'软件包管理' on page 7中的技巧来做。

2.4 将CD-ROM加入source.list文件

如果你用APT从CD-ROM上安装及升级软件包，你可以将它加入到sources.list文件中。完成该操作，可使用apt-cdrom程序：

```
# apt-cdrom add
```

将Debian光盘放入光驱，它将加载光盘目录，并在光盘上查找软件包信息。如果你的光驱需要额外设置，可使用以下选项：

```
-h            - program help
-d directory - CD-ROM mount point
-r            - Rename a recognized CD-ROM
-m            - No mounting
-f            - Fast mode, don't check package files
-a            - Thorough scan mode
```

例如：

```
# apt-cdrom -d /home/kov/mycdrom add
```

你还可以扫描一张光盘，但不将其加入列表：

```
# apt-cdrom ident
```

注意，只有当你在系统的/etc/fstab中正确设置了光驱后，该程序才会工作。

Chapter 3

软件包管理

3.1 更新可用软件包列表

软件包管理系统使用一个私有数据库跟踪列表中软件包的当前状态：已安装、未安装或可安装。apt-get通过该数据库来确定如何安装用户想用的软件包以及正常运行该软件包所必须的其它关联包。

你可以使用apt-get update来更新数据库列表。这个命令将扫描/etc/apt/sources.list文件中所指路径中的软件包列表文件。有关该列表文件的更多信息请参考'/etc/apt/sources.list文件' on page 3。

定时运行这个程序是个好主意，它将使你和你的系统获得最新的软件包更新和安全更新等信息。

3.2 安装软件包

现在，终于到了你一直期待的阶段！准备好了sources.list和最新版的的可用软件包，你所需做的就是运行apt-get来安装你渴望已久的软件了。例如，你可以这样：

```
# apt-get install xchat
```

APT会扫描它的数据库找到最新的版本的软件包，并将它从sources.list中所指的地方下载到本地。如果该软件包需要其它软件包才能正常运行——如本例一样——APT会做关联性检查并自动安装所关联软件包。如下所示：

```
# apt-get install nautilus
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  bonobo libmedusa0 libnautilus0
```

```
The following NEW packages will be installed:
  bonobo libmedusa0 libnautilus0 nautilus
0 packages upgraded, 4 newly installed, 0 to remove and 1  not upgraded.
Need to get 8329kB of archives. After unpacking 17.2MB will be used.
Do you want to continue? [Y/n]
```

nautilus软件包需要引用共享函数库，因此APT会从镜像源处下载相关共 享函数库，如果你在apt-get命令行中手动指定了这些共享函数库的名称， APT不会询问你是否要继续；它会自动认为你希望安装所有这些软件包。

也就是说APT只会在安装那些没有在命令行中指定的软件包时提示确认。

下列apt-get选项也许对你有用：

```
-h  这个帮助信息
-d  只下载——不安装或解压档案
-f  即便完整性检查失败了仍然继续
-s  不做什么。只是按顺序模拟
-y  对于所有问题都假定为Yes，不询问
-u  显示一系列已经将要更新的包
```

可以用一条命令安装多个软件包。包文件从网络上下载到本地/var/cache/apt/archives目录，稍后再安装。

你可以用同样的命令行删除指定软件包，只需在软件包名称后紧跟一个“-”，如下所示：

```
# apt-get install nautilus gnome-panel-
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  bonobo libmedusa0 libnautilus0
The following packages will be REMOVED:
  gnome-applets gnome-panel gnome-panel-data gnome-session
The following NEW packages will be installed:
  bonobo libmedusa0 libnautilus0 nautilus
0 packages upgraded, 4 newly installed, 4 to remove and 1  not upgraded.
Need to get 8329kB of archives. After unpacking 2594kB will be used.
Do you want to continue? [Y/n]
```

参考‘移除软件包’ on the facing page一节以获得更多的关于删除软件包的信息。

假如你不小心损坏了已安装的软件包而想修复它，或者仅仅想重新安装软件包中某 些文件的最新版本，这是可以做到的，你可以用如下的--reinstall选项：

```
# apt-get --reinstall install gdm
Reading Package Lists... Done
```

```
Building Dependency Tree... Done
0 packages upgraded, 0 newly installed, 1 reinstalled, 0 to remove and 1  not
Need to get 0B/182kB of archives. After unpacking 0B will be used.
Do you want to continue? [Y/n]
```

3.3 移除软件包

如果你不再使用某些软件包，你可以用APT将其从系统中删除。要删除软件包只需 输入：apt-get remove package。如下所示：

```
# apt-get remove gnome-panel
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages will be REMOVED:
  gnome-applets gnome-panel gnome-panel-data gnome-session
0 packages upgraded, 0 newly installed, 4 to remove and 1  not upgraded.
Need to get 0B of archives. After unpacking 14.6MB will be freed.
Do you want to continue? [Y/n]
```

由上例可知，APT会关注那些与被删除的软件包有依赖关系的软件包。使用APT删除 一个软件包将会连带删除那些与该软件包有依赖关系的软件包。

上例中运行apt-get会删除指定软件包以及与之有依赖关系的软件 包，但它们的配置文件，如果有的话，会完好无损地保留在系统里。如果想彻底删 除这些包及其配置文件，运行：

```
# apt-get --purge remove gnome-panel
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages will be REMOVED:
  gnome-applets* gnome-panel* gnome-panel-data* gnome-session*
0 packages upgraded, 0 newly installed, 4 to remove and 1  not upgraded.
Need to get 0B of archives. After unpacking 14.6MB will be freed.
Do you want to continue? [Y/n]
```

注意：软件包名字后面的*表示该软件包所有的配置文件也将被删除。

就象install时一样，你可以在remove命令中用一个符号来指 定安装某个软件包。在删除软件包时，如果你在软件包名字后面紧跟一个 “+”，那么该软件包就会被安装而不是删除。

```
# apt-get --purge remove gnome-panel nautilus+
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  bonobo libmedusa0 libnautilus0 nautilus
```

```
The following packages will be REMOVED:
  gnome-applets* gnome-panel* gnome-panel-data* gnome-session*
The following NEW packages will be installed:
  bonobo libmedusa0 libnautilus0 nautilus
0 packages upgraded, 4 newly installed, 4 to remove and 1  not upgraded.
Need to get 8329kB of archives. After unpacking 2594kB will be used.
Do you want to continue? [Y/n]
```

注意，apt-get列出了那些将要被安装的额外软件包(即保证该软件 包正常运行的其它软件包)和将要被删除关联软件包，然后，再次列出了将要被安 装的软件包(包括了额外的包)。

3.4 更新软件包

软件包更新是APT最成功的特点。只需一条命令即可完成更新： apt-get upgrade。你可以使用这条命令从相同版本号的发布版中更新 软件包，也可以从新版本号的发布版中更新软件包，尽管实现后一种更新的推荐 命令为apt-get dist-upgrade；详情请参考'升级到新版本' on the next page。

在运行该命令时加上-u选项很有用。这个选项让APT显示完整的可更新 软件包列表。不加这个选项，你就只能盲目地更新。APT会下载每个软件包的最新 更新版本，然后以合理的次序安装它们。注意在运行该命令前应先运行 apt-get update更新数据库。详情请参考'更新可用软件包列表' on page 7。请看下 面的例子：

```
# apt-get -u upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages have been kept back
  cpp gcc lilo
The following packages will be upgraded
  adduser ae apt autoconf debhelper dpkg-dev esound esound-common ftp indent
  ipchains isapnptools libaudiofile-dev libaudiofile0 libesd0 libesd0-dev
  libgtk1.2 libgtk1.2-dev liblockfile1 libnewt0 liborbit-dev liborbit0
  libstdc++2.10-glibc2.2 libtiff3g libtiff3g-dev modconf orbit procps psmisc
29 packages upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
Need to get 5055B/5055kB of archives. After unpacking 1161kB will be used.
Do you want to continue? [Y/n]
```

整个更新过程非常简单。注意在本例中头几行，apt-get报告有些软件包 的更新被kept back，这表明这些软件包的更新版本因故无法安装，可能 的原因有关联不同步(当前没有供下载的新版本关联包)或关联扩展(需要安装新的 关联包以配合新版软件包)。

对于第一种原因没有很好的解决方法，对于第二次原因，运行apt-get intall安装所需的新关联包就可以。另一个更好的解决方法就是使用dist-upgrade。详情请参考'升级到新版本' on the next page。

3.5 升级到新版本

APT的绝活就是让你一次就完成整个系统的更新，不论是通过Internet还是通过光盘文件(购买的碟片或下载的ISO镜像文件)。

它也可以用来更新那些关联关系发生改变的软件包。即如前所述的那些使用apt-get upgrade时被不被更新(kept back)的软件包。

例如，假设你目前使用的Debian为stable revision 0，而你购买了revision 3的新版Debian，你可以使用APT从新光盘上升级你的系统。使用apt-cdrom(参考'将CD-ROM加入source.list文件' on page 6)将光盘加载到/etc/apt/sources.list中，然后运行apt-get dist-upgrade。

请注意，APT总是搜索最新版本的软件包，因此，如果一个软件包在你的/etc/apt/sources.list中所列的版本比光盘上所列的版本要新，那么APT会下载其中的软件包而不是使用光盘上的软件包。

在'更新软件包' on the preceding page节的例子中，我们看到有些包被kept back了，现在我们就用dist-upgrade方法来解决这个问题：

```
# apt-get -u dist-upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
Calculating Upgrade... Done
The following NEW packages will be installed:
  cpp-2.95 cron exim gcc-2.95 libident libopenldap-runtime libopenldap1
  libpcre2 logrotate mailx
The following packages have been kept back
  lilo
The following packages will be upgraded
  adduser ae apt autoconf cpp debhelper dpkg-dev esound esound-common ftp gcc
  indent ipchains isapnptools libaudiofile-dev libaudiofile0 libesd0
  libesd0-dev libgtk1.2 libgtk1.2-dev liblockfile1 libnewt0 liborbit-dev
  liborbit0 libstdc++2.10-glibc2.2 libtiff3g libtiff3g-dev modconf orbit
  procps psmisc
31 packages upgraded, 10 newly installed, 0 to remove and 1 not upgraded.
Need to get 0B/7098kB of archives. After unpacking 3118kB will be used.
Do you want to continue? [Y/n]
```

注意现在那些软件包将会被更新，那些新的关联软件包也会被安装。但是lilo仍被 kept back，可能还存在一些比建立新关联更棘手的问题，我们通过如下 方法确定问题所在：

```
# apt-get -u install lilo
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  cron debconf exim libident libopenldap-runtime libopenldap1 libpcre2
```

```
  logrotate mailx
The following packages will be REMOVED:
  debconf-tiny
The following NEW packages will be installed:
  cron debconf exim libident libopenldap-runtime libopenldap1 libpcre2
  logrotate mailx
The following packages will be upgraded
  lilo
1 packages upgraded, 9 newly installed, 1 to remove and 31 not upgraded.
Need to get 225kB/1179kB of archives. After unpacking 2659kB will be used.
Do you want to continue? [Y/n]
```

查看上述提示信息可知，lilo与debconf-tiny包产生了一个新冲突，这表明除非删除debconf-tiny，否则将无法安装(或更新)lilo。

想知道该保留或删除哪些软件包，你可以使用：

```
# apt-get -o Debug::pkgProblemResolver=yes dist-upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
Calculating Upgrade... Starting
Starting 2
Investigating python1.5
Package python1.5 has broken dep on python1.5-base
  Considering python1.5-base 0 as a solution to python1.5 0
  Holding Back python1.5 rather than change python1.5-base
Investigating python1.5-dev
Package python1.5-dev has broken dep on python1.5
  Considering python1.5 0 as a solution to python1.5-dev 0
  Holding Back python1.5-dev rather than change python1.5
 Try to Re-Instate python1.5-dev
Done
Done
The following packages have been kept back
  gs python1.5-dev
0 packages upgraded, 0 newly installed, 0 to remove and 2  not upgraded.
```

现在，你很容易就知道不能安装python1.5-dev软件包是因为无法满足另一个软件包python1.5的关联要求。

3.6 移除无用软件包文件：apt-get clean and autoclean

当你需要安装某个软件包时，APT从/etc/apt/sources.list中所列的主机下载所需的文件，将它们保存到本机软件库(/var/cache/apt/archives/)，然后开始安装，参考'安装软件包' on page 7。

本地软件库会不断膨胀占用大量硬盘空间，幸运的是，APT提供了工具来管理本地软件库：apt-get的clean方法和autoclean方法。

apt-get clean将删除/var/cache/apt/archives目录和/var/cache/apt/archives/partial目录下锁文件以外的所有文件。这样以来，当你需要再次安装某个软件包时，APT将重新下载它。

apt-get autoclean仅删除那些不需要再次下载的文件。

下面这个例子显示了apt-get autoclean如何工作：

```
# ls /var/cache/apt/archives/logrotate* /var/cache/apt/archives/gpm*
logrotate_3.5.9-7_i386.deb
logrotate_3.5.9-8_i386.deb
gpm_1.19.6-11_i386.deb
```

在/var/cache/apt/archives目录下有两个不同版本的logrotate软件包文件以及一个gpm软件包文件。

```
# apt-show-versions -p logrotate
logrotate/stable uptodate 3.5.9-8
# apt-show-versions -p gpm
gpm/stable upgradeable from 1.19.6-11 to 1.19.6-12
```

apt-show-versions显示logrotate_3.5.9-8_i386.deb提供了logrotate的升级版本，所以logrotate_3.5.9-7_i386.deb没用了。同样gpm_1.19.6-11_i386.deb也没有用了，因为可以下载该软件包的更新版本。

```
# apt-get autoclean
Reading Package Lists... Done
Building Dependency Tree... Done
Del gpm 1.19.6-11 [145kB]
Del logrotate 3.5.9-7 [26.5kB]
```

总之，apt-get autoclean仅删除那些过时的文件。参考 '如何从Debian的专用版本下升级软件包' on page 15以了解apt-show-versions的更多详情。

3.7 在dselect中操作APT

dselect工具帮助用户选取想要安装的Debian软件包。它有点复杂甚至令人望而生厌，但经过实践你就能掌握它恐怖的终端界面。

dselect高级功能之一就是它知道利用Debian软件包的“推荐”和“建议”能力。(Debian软件包有一种能力：推荐或建议系统在安装自己的同时，安装别的软件包以配合自身的工作，当然

这些推荐的软件包不一定是必须的；而dselect工具可以识别和利用这个能力，使用dselect时你就能体会到。译者注)以root身份运行dselect，进入程序后选择apt作为连接方式(access)。该步骤不是必须的，但如果你没有光驱而且想通过Internet下载安装软件包，这是使用dselect的最好方法。

想深入学习dselect的用法，请到Debian网站查阅dselect文档页面http://www.debian.org/doc/ddp.

在dselect中选好了软件包后，运行：

```
# apt-get -u dselect-upgrade
```

如下例所示：

```
# apt-get -u dselect-upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages will be REMOVED:
  lbxproxy
The following NEW packages will be installed:
  bonobo console-tools-libs cpp-3.0 enscript expat fingerd gcc-3.0
  gcc-3.0-base icepref klogd libdigest-md5-perl libfnlib0 libft-perl
  libgc5-dev libgcc300 libhtml-clean-perl libltdl0-dev libsasl-modules
  libstdc++3.0 metamail nethack proftpd-doc psfontmgr python-newt talk tidy
  util-linux-locales vacation xbill xplanet-images
The following packages will be upgraded
  debian-policy
1 packages upgraded, 30 newly installed, 1 to remove and 0  not upgraded.
Need to get 7140kB of archives. After unpacking 16.3MB will be used.
Do you want to continue? [Y/n]
```

比较一下我们在相同系统上运行apt-get dist-upgrade时的情形：

```
# apt-get -u dist-upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
Calculating Upgrade... Done
The following packages will be upgraded
  debian-policy
1 packages upgraded, 0 newly installed, 0 to remove and 0  not upgraded.
Need to get 421kB of archives. After unpacking 25.6kB will be freed.
Do you want to continue? [Y/n]
```

我们看到在前例中许多软件包被安装是其它软件包“推荐”或“建议”的结果。另外一些软件包被安装或删除(例如lbxproxy软件包)是我们通过dselect工具作出的决定。由此可见dselect与APT结合起来将是一个功能强大的工具。

3.8 如何保持一个混合系统

人们有时会对这种情况有兴趣——使用一个版本的Debian作为其主发行版，但从另一个分支上安装一个或多个包。

要设定你的Debian主版本，应当修改/etc/apt/apt.conf文件，并加入：

```
APT::Default-Release "version";
```

其中*version*是你希望作为主发行版使用的Debian版本。你可以使用的版本有stable、testing和unstable。要从另外一个版本中安装软件包，你必须按照如下方式执行APT：

```
# apt-get -t distribution install package
```

为了使其可以个工作，在你的/etc/apt/sources.list中至少有一行是关于你要使用的那个版本的，而且要使用的软件包也必须存在于该版本中。

你也可以要求使用某个特定把本的软件包，如下所示：

```
# apt-get install package=version
```

例如，下面的命令将会安装2.2.4-1版的nautilus软件包：

```
# apt-get install nautilus=2.2.4-1
```

重要信息：最新版的Debian软件包首先会上传到“unstable”发布版中，这个发布版包含了软件包所有更改阶段，无论是小修小补还是影响到众多软件包乃至整个系统的重大修改。所以，新手和那些强调系统稳定性的用户不会使用这个发布版。

“testing”发布版比起“unstable”发布版，多注重了些系统稳定性，但正式运行的系统应当使用“stable”发布版。

3.9 如何从Debian的专用版本下升级软件包

apt-show-versions提供了一个安全的途径，让那些使用混合系统的用户放心升级他们的系统，不必担心升级会将原来属于stable的包升级成了unstable包。例如，在安装了apt-show-versions软件包之后，使用这条命令将只升级你的unstable软件包：

```
# apt-get install `apt-show-versions -u -b | grep unstable | cut -d ' ' -f 1`
```

3.10 如何维护已安装软件包的多个版本（复杂） How to keep specific versions of packages installed (complex)

你可能会遇到这种情况，更改了某个软件包中的一些文件，但你没有时间或根本就 不想将这些更改引入到新版本中。或者，你将系统升级到3.0，但仍想继续使用Debian 2.2下的某个软件包。你可以“钉住”这个版本，这样它就不会被更新了。

操作起来十分简单，你只需编辑/etc/apt/preferences文件。/etc/apt/preferences.

文件格式很简单：

```
Package: <package>
Pin: <pin definition>
Pin-Priority: <pin's priority>
```

每个条目都要以空白行与其它条目分割开。例如，我对sylpheed 软件包做了某些修改以使用“reply-to-list”功能，其版本为0.4.99。我想保留这 些修改不被更新，可加上：

```
Package: sylpheed
Pin: version 0.4.99*
```

注意我用了一个*(星号)。这是一个“通配符”；它表明我希望“钉住”所有 以0.4.99打头的版本(以防它们被下载并安装到我机器上。pin控制的是服务器端的 更新软件包而非本地的已安装软件包。译者注)。因为Debian使用“Debian版本号” 为其软件包定版本，我不想进行所有这些版本的升级，如果不用通配符，那么0.4.99-1版或0.4.99-10版只要一出炉系统就会安装它们。如果你修改了软件包，你一定不希望这么做。

Pin的优先级帮助我们检查一个与“Packages:”和“Pin:”相符合的软件包是否应该被 安装。当优先级比较高时，符合的软件包将会被安装。你可以查阅apt_preferences(7)，其中有关于优先级的详细讨论，但通过 一些简单的例子也可以了解基本的概念。下面就说明了在上面的sylpheed例子中设 定优先级域的效果。

1001 Sylpheed 0.4.99永远不会被apt替换。如果可能，apt甚至会用0.4.99版替换已经 安装的更高的版本呢。只有那些优先级比1000大的软件包才会降级。

1000 除了不会将高版本降级以外，与1001的效果相同。

990 版本0.4.99只会被首选发布系列中高版本的的软件包替换，首选发行版由变量“APT::Default-Release”定义(参考'如何保持一个混合系统' on the preceding page。

500 任何发布系列中比0.4.99版本高的sylpheed都会被安装，但相对于低版本而言，仍然 建议使用0.4.99。

100 任何发布系列中高版本的sylpheed都会被安装；因此只有没有其它版本可以安装时 才会安装0.4.99。已安装包的优先级。

-1 负的优先级也是允许的，它会组织0.4.99版被安装。

钉子也可以用来指定软件包的version、release或 origin。

我们已经看到，钉在一个version上，可以使用具体的版本号，也可以使 用通配符一次指定多个版本。

release选项依赖于APT仓库上的或者CD中的Release文件。如果你使用的 APT仓库并没有提供这个文件，这个选项就没有任何用处了。你可以在 /var/lib/apt/lists/中看到Release文件的内容。release的参数是： a(存档)、c(部件)、v(版本)、o(起源) 和l(标签)。

例如：

```
Package: *
Pin: release v=2.2*,a=stable,c=main,o=Debian,l=Debian
Pin-Priority: 1001
```

在这个例子中，我们选择了Debian版本2.2*(可以是2.2r2、2.2r3— —这些版本中通常 包含了对安全问题的修复和其它重要更新)，stable仓库，main (相应的还有contrib或者non-free)区段、起源和标签都是 Debian。origin(o=)定义了谁制作了这个Release文件，label(l=)定义了发行版的 名字：Debian自己就使用Debian而Progeny则使用Progeny。Release文件的例子如下 所示：

```
$ cat /var/lib/apt/lists/ftp.debian.org.br_debian_dists_potato_main_binary-i3
Archive: stable
Version: 2.2r3
Component: main
Origin: Debian
Label: Debian
Architecture: i386
```

Chapter 4

几个非常有用的工具

4.1 如何安装本地编译的软件包：equivs

有时，用户想使用某些软件的特殊版本，它们只以源代码的形式存在，没有现成 的Debian软件包。软件包管理系统在处理这类事务时可能会出问题。假设你想编 译新版本的邮件服务器，所有的事情都很正常，但是Debian中的很多软件包是依 赖于MTA(邮件传输代理)的。由于你是自己手工编译安装软件，软件包管理系统 对此一无所知。

现在是equivs登台的时候了。用它来安装软件包，它所做 的工作就是创建一个新的空软件包来实现关联，让软件包管理系统相信所有的依 赖关系都可以满足。

在我们开始以前，我必须提醒你，编译某个软件最安全的方法是对该软件现有 的Debian软件包进行修改后重新编译，如果你并不知道你正在干什么，劝你不要使 用equivs替换关联包。更多信息请参考'源码包操作' on page 29。

继续上面的例子，你安装好了新编译的postfix，接下来打算安装 mutt。突然你发现mutt想安装另外一 个MTA，但实际上你已经有了你的MTA。

转到某个目录(例如/tmp)执行：

```
# equivs-control name
```

将*name*替换为你创建的控制文件，控制文件按如下格式创建：

```
Section: misc
Priority: optional
Standards-Version: 3.0.1

Package: <enter package name; defaults to equivs-dummy>
Version: <enter version here; defaults to 1.0>
Maintainer: <your name and email address; defaults to username>
Pre-Depends: <packages>
Depends: <packages>
```

```
Recommends: <packages>
Suggests: <package>
Provides: <(virtual)package>
Architecture: all
Copyright: <copyright file; defaults to GPL2>
Changelog: <changelog file; defaults to a generic changelog>
Readme: <README.Debian file; defaults to a generic one>
Extra-Files: <additional files for the doc directory, comma-separated>
Description: <short description; defaults to some wise words>
 long description and info
 .
 second paragraph
```

我们只需按自己的需要修改相关项目就行了。文件中每个项目都描述得很清楚，我们不必在此逐行解释它们。现在开始修改吧：

```
Section: misc
Priority: optional
Standards-Version: 3.0.1

Package: mta-local
Provides: mail-transport-agent
```

行了，就是这样。mutt依赖于mail-transport-agent，这是所有MTA共同提供的一个虚拟包，我可以简单地将这个软件包命名为mail-transport-agent，不过我更愿意使用系统的虚拟包方案，使用Provides选项。

现在你可以开始构建软件包了：

```
# equivs-build name
dh_testdir
touch build-stamp
dh_testdir
dh_testroot
dh_clean -k
# Add here commands to install the package into debian/tmp.
touch install-stamp
dh_testdir
dh_testroot
dh_installdocs
dh_installchangelogs
dh_compress
dh_fixperms
dh_installdeb
dh_gencontrol
```

```
dh_md5sums
dh_builddeb
dpkg-deb: building package 'name' in '../name_1.0_all.deb'.
```

软件包已经被创建了，注意，软件包是创建在当前目录中的。

然后安装这个.deb文件。

众所周知，equivs的用法很多，譬如你可以创建一个my-favorites软件包，它依赖于你通常安装的软件包。尽情发挥你的想像力吧，当然还是要小心。

重要提示：在/usr/share/doc/equivs/examples目录下有控制文件的例子，最好看一下。

4.2 移除无用的地区配置(locale)文件：localepurge

许多Debian用户仅在固定地区使用Debian。例如，在巴西的Debian用户，通常使用pt_BR地区配置文件而不会关心es地区配置文件。

对于这类用户而言localepurge是一个非常有用的工具，你可以仅保留你当前所用的地区配置文件，删除其它无用的文件，从而释放大量硬盘空间。运行apt-get install localepurge就行了。

它配置起来非常容易，debconf的提问将引导用户一步一步完成设置。在回答第一个问题时请务必谨慎，如果回答错了，系统可能删掉所有的地区配置文件，包括你正在使用的这个。复原它们的唯一方法就是重装那些软件包。

4.3 如何知晓哪些软件包可以升级

apt-show-versions工具可以告诉你系统中哪些包可以更新以及其它一些有用的信息。-u选项可以显示可更新软件包列表：

```
$ apt-show-versions -u
libeel0/unstable upgradeable from 1.0.2-5 to 1.0.2-7
libeel-data/unstable upgradeable from 1.0.2-5 to 1.0.2-7
```

Chapter 5

获取软件包信息

有些基于APT系统的前端程序，能十分方便地获得系统软件包列表，列表包括可安 装或已安装的软件包，还可以显示某软件包属于哪个section，它的优先级是多少， 它的说明文档等等。

但是...在此我们想的学习如何使用APT本身来完成。你如何找出你想要安装的软件 包的名称?

我们完成这个任务的方法有很多。我们从apt-cache开始，APT系统使用 这个程序来维护它的数据库。下面我们通过一些实际操作来对它做个概览。

5.1 获得软件包名称

假设你十分怀念玩Atari 2600的好日子，你决定用APT安装一个Atari emulator， 随后再下载几个游戏，你可以这样：

```
# apt-cache search atari
atari-fdisk-cross - Partition editor for Atari (running on non-Atari)
circuslinux - The clowns are trying to pop balloons to score points!
madbomber - A Kaboom! clone
tcs - Character set translator.
atari800 - Atari emulator for svgalib/X/curses
stella - Atari 2600 Emulator for X windows
xmess-x - X binaries for Multi-Emulator Super System
```

我们找到了几个相关的软件包，以及有关的简单描述。想进一步获得某个软件包的 详细信息，你可以运行：

```
# apt-cache show stella
Package: stella
Priority: extra
Section: non-free/otherosfs
Installed-Size: 830
```

```
Maintainer: Tom Lear <tom@trap.mtview.ca.us>
Architecture: i386
Version: 1.1-2
Depends: libc6 (>= 2.1), libstdc++2.10, xlib6g (>= 3.3.5-1)
Filename: dists/potato/non-free/binary-i386/otherosfs/stella_1.1-2.deb
Size: 483430
MD5sum: 11b3e86a41a60fa1c4b334dd96c1d4b5
Description: Atari 2600 Emulator for X windows
 Stella is a portable emulator of the old Atari 2600 video-game console
 written in C++.  You can play most Atari 2600 games with it.  The latest
 news, code and binaries for Stella can be found at:
 http://www4.ncsu.edu/~bwmott/2600
```

屏幕上显示出这个软件包的详细信息及其用途的完整描述。如果你的系统中已安装 了某个软件包而系统又搜索到它的新版本，系统会将它们的详细信息一并列出。如 下例：

```
# apt-cache show lilo
Package: lilo
Priority: important
Section: base
Installed-Size: 271
Maintainer: Russell Coker <russell@coker.com.au>
Architecture: i386
Version: 1:21.7-3
Depends: libc6 (>= 2.2.1-2), debconf (>=0.2.26), logrotate
Suggests: lilo-doc
Conflicts: manpages (<<1.29-3)
Filename: pool/main/l/lilo/lilo_21.7-3_i386.deb
Size: 143052
MD5sum: 63fe29b5317fe34ed8ec3ae955f8270e
Description: LInux LOader - The Classic OS loader can load Linux and others
 This Package contains lilo (the installer) and boot-record-images to
 install Linux, OS/2, DOS and generic Boot Sectors of other OSes.
 .
 You can use Lilo to manage your Master Boot Record (with a simple text scree
 or call Lilo from other Boot-Loaders to jump-start the Linux kernel.

Package: lilo
Status: install ok installed
Priority: important
Section: base
Installed-Size: 190
Maintainer: Vincent Renardias <vincent@debian.org>
Version: 1:21.4.3-2
Depends: libc6 (>= 2.1.2)
```

```
Recommends: mbr
Suggests: lilo-doc
Description: LInux LOader - The Classic OS loader can load Linux and others
 This Package contains lilo (the installer) and boot-record-images to
 install Linux, OS/2, DOS and generic Boot Sectors of other OSes.
 .
 You can use Lilo to manage your Master Boot Record (with a simple text scree
 or call Lilo from other Boot-Loaders to jump-start the Linux kernel.
```

注意，首先列出的是可用软件包，接着列出的是已安装软件包。获取某个软件包的 常规信息可运行：

```
# apt-cache showpkg penguin-command
Package: penguin-command
Versions:
1.4.5-1(/var/lib/apt/lists/download.sourceforge.net_debian_dists_unstable_mai

Reverse Depends:
Dependencies:
1.4.5-1 - libc6 (2 2.2.1-2) libpng2 (0 (null)) libsdl-mixer1.1 (2 1.1.0) libs
Provides:
1.4.5-1 -
Reverse Provides:
```

如果仅想了解某软件包的与哪些软件包关联，可运行：

```
# apt-cache depends penguin-command
penguin-command
  Depends: libc6
  Depends: libpng2
  Depends: libsdl-mixer1.1
  Depends: libsdl1.1
  Depends: zlib1g
```

总之，有一系列工具可帮助我们找到我们想要的软件包。

5.2 使用dpkg查找软件包名称

另一个定位软件包的方法是知道软件包中某个关键文件的名称。例如，你编译时需 要某个“.h”头文件，查找提供该文件的软件包，你可以运行：

```
# dpkg -S stdio.h
libc6-dev: /usr/include/stdio.h
libc6-dev: /usr/include/bits/stdio.h
perl: /usr/lib/perl/5.6.0/CORE/nostdio.h
```

或者：

```
# dpkg -S /usr/include/stdio.h
libc6-dev: /usr/include/stdio.h
```

解系统中已安装软件的软件包名称十分有用，譬如当你想清理硬盘空间时，可以运行：

```
# dpkg -l | grep mozilla
ii  mozilla-browse 0.9.6-7        Mozilla Web Browser
```

这个命令的缺点是它会“截断”软件包的名字。在上例中，软件包的全称是mozilla-browser，解决这个问题可以使用COLUMNS环境变量：

```
[kov]@[couve] $ COLUMNS=132 dpkg -l | grep mozilla
ii  mozilla-browser             0.9.6-7                      Mozilla Web Brows
```

或显示成这样：

```
# apt-cache search "Mozilla Web Browser"
mozilla-browser - Mozilla Web Browser
```

5.3 如何“按需”安装软件包

你正在编译某段程序，突然，停住了！一条错误信息报告说你没有它需要的.h头文件。让auto-apt来救你吧，它问你是否要安装需要 的软件包，然后挂起编译进程，安装好软件包后再恢复编译进程。

你所要做的仅仅是：

```
# auto-apt run command
```

这里“command”指在运行过程中可能出现“需求文件不存在”问题的命令。例如：

```
# auto-apt run ./configure
```

一会儿，它就会告诉你要安装所需的软件包并自动转到apt-get处理。如果你正在 运行X，就会一个图形界面提示窗口。

为了提高效率auto-apt所用的数据库需要实时更新。可调用 auto-apt update，auto-apt updatedb和 auto-apt update-local来完成更新。

5.4 如何知道文件属于哪个软件包

如果你想安装某个软件包，但用apt-cache查不出它的名称，不过你知道这个程序的文件名，或这个软件包中某些文件的文件名，那么你可以用apt-file来查找软件包名称。如下所示：

```
$ apt-file search filename
```

它用起来很象dpkg -S，不过它还会列出包含该文件的已删除软件包。它也可以用来查找哪个软件包包含编译时所缺的文件，当然，解决这类问题auto-apt可能是更好的方案，请参考'如何“按需”安装软件包' on the facing page。

用这个命令，你可以列出软件包的内容：

```
$ apt-file list packagename
```

apt-file用一个数据库来存放所有软件包的内容信息，和auto-apt一样，这个数据库也需要实时更新，完成更新可以运行：

```
# apt-file update
```

缺省情况下，apt-file和auto-apt使用同一个数据库，参考'如何“按需”安装软件包' on the preceding page。

5.5 如何掌握软件包的变化情况

在每个软件包被安装以后，都会在文档目录(/usr/share/doc/packagename)生成一个changelog.Debian.gz的文件，这个文件记录了该软件包最后一次更新对系统做了哪些修改，你可以用zless阅读这些信息。不过当你对整个系统进行升级以后，逐个查看软件包的更新信息可不是件容易事。

有一个工具能帮你完成这项任务，它就是apt-listchanges。首先你要装上apt-listchanges软件包。在安装的过程中，为了进行配置，Debconf会问你一些问题，按你的要求回答它们就行了。

第一个问题是问你希望apt-listchanges如何来显示修改日志。你可以让它把信息通过邮件的方式发送给你，这对于自动更新是非常有用的。或者你可以让它在less等程序中显示修改日志，这样在继续升级前你就可以查看它们了。如果你不希望apt-listchanges在升级的时候自动的运行，可以回答none。

安装了apt-listchanges后，每当apt下载软件包之后(不论来源是Internet、光盘或是硬盘)都会显示这些软件包的系统更新信息。

Chapter 6

源码包操作

6.1 下载源码包

在自由软件的世界里，经常需要学习源码或为程序除错，所以你需要下载它们。APT提供了一套简便的方法帮你获得发布版中众多程序的源代码以及创建一个.debs所需的所有文件。

Debian源码的另一个普遍用途是将unstable发布版的新版程序进行改写以供别的发布版使用。例如，从stable发布版外引入新的软件包，需要重新生成.debs将它在原发布版中的关联关系迁移到新的发布版。

要完成这些工作，`/etc/apt/sources.list`文件中`deb-src`所指引用镜像源应该是unstable，别忘了将行首的注释符去掉。详情参考‘/etc/apt/sources.list文件’ on page 3。

用下面的命令下载源码包：

```
$ apt-get source packagename
```

通常会下载三个文件：一个`.orig.tar.gz`、一个`.dsc`和一个`.diff.gz`。对于Debian专用的软件包，不会下载最后一个文件，第一个文件的文件名中没有“`orig`”项。

dpkg-source通过`.dsc`文件中的信息，将源码包解包到 *packagename-version* 目录，下载下来的源码包中有一个`debian/`目录，里面是创建.deb包所需的文件。

想要下载的源码包自动编译成软件包，只需在命令行中加上`-b`，如下：

```
$ apt-get -b source packagename
```

如果你不打算在下载后就立刻创建.deb文件，你可以在之后用下面的命令创建：

```
$ dpkg-buildpackage -rfakeroot -uc -b
```

上述命令应当在下载后为软件包创建的目录中执行。要安装用这种方式构建好的软件包，只能直接使用软件包管理器，例如：

```
# dpkg -i file.deb
```

apt-get的source命令与它的其它命令有所不同，普通用户就可以运行source命令。文件被下载到用户调用apt-source package命令时所处的目录中。

6.2 编译源码包所需的软件包

通常，编译源码包时要用到某些头文件和共享库，所有的源码包的控制文件中都有一个域“Build-Depends：”，域中指出了编译该源码包需要哪些附加包。

APT提供了一个简单的方法下载这些附加包，你只需运行apt-get build-dep package，其中“package”就是你打算编译的源码包名称。见下例：

```
# apt-get build-dep gmc
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  comerr-dev e2fslibs-dev gdk-imlib-dev imlib-progs libgnome-dev libgnorba-de
  libgpmg1-dev
0 packages upgraded, 7 newly installed, 0 to remove and 1  not upgraded.
Need to get 1069kB of archives. After unpacking 3514kB will be used.
Do you want to continue? [Y/n]
```

这些将要被安装的包是用于正确编译gmc的。注意这个命令不能用来搜索某个软件的源码包，你得另外运行apt-get source下载源码包。

如果你想做的是检查要编译一个软件包需要哪些其它的软件包，apt-cache show可以显示它(从那考‘获取软件包信息’ on page 23，在众多信息之中，Build-Depends一行会列出那些需要的软件包。

```
# apt-cache showsrc package
```

Chapter 7

如何处理错误

7.1 一般错误

错误总是发生，大部分是因为用户的粗心，下面列举一些常见错误及处理方法。

如果在运行apt-get install package时，你的系统报告如下信息：

```
Reading Package Lists... Done
Building Dependency Tree... Done
W: Couldn't stat source package list 'http://people.debian.org unstable/ Pack
W: You may want to run apt-get update to correct these missing files
E: Couldn't find package penguineyes
```

上次你修改/etc/apt/sources.list后，忘了运行apt-get update更新。

如果出现这样的信息：

```
E: Could not open lock file /var/lib/dpkg/lock - open (13 Permission denied)
E: Unable to lock the administration directory (/var/lib/dpkg/), are you root
```

如果你没有root权限，运行除source外的其它apt-get命 令，如会出现上面的错误信息。这是因为你是普通用户 。

当你同时运行两个apt-get进程，或者当你试图运行apt-get时已有一个的dpkg进程处于激活状态，系统也 会报告与上面相似的错误信息。唯一能与其它命令同时运行的只有source 命令。

如果在安装过程中出现中断，然后你发现该软件包既不能重装又不能删除，试试下 面两个命令：

```
# apt-get -f install
# dpkg --configure -a
```

再试着安装那个软件包，如果不行再次运行上述命令后再试。这两个命令对于那些使用unstable的玩家非常有用。

如果你在运行`apt-get update`时看到“E: Dynamic MMap ran out of room”，那么在`/etc/apt/apt.conf`加入如下内容：

```
APT::Cache-Limit 10000000;
```

7.2 在哪儿获得帮助?

如果你发现自己有太多疑问，没关系，有大量的Debian软件包管理系统文档供你参考。`--help`和帮助文档能为你提供巨大的帮助，这些文档位于`/usr/share/doc`目录中，如`/usr/share/doc/apt`。

如果本文档没法帮你排忧解难，可以去Debian邮件列表找找答案，在相关栏目内你会获得更多信息。Debian的网址是：http://www.debian.org。

记住只有Debian用户才能这些邮件列表和资源，其它操作系统的用户请到相关系统发布者建立的社区中获取更多资源。

Chapter 8

哪些发行版支持APT?

下面是部分使用APT系统的发行版的名称：

Debian GNU/Linux (http://www.debian.org) - APT正是为这个发行版开发的

Conectiva (http://www.conectiva.com.br) - 这是第一个将APT移植到rpm上的发行版

Libranet (http://www.libranet.com)

Mandrake (http://www.mandrake.com)

PLD (http://www.pld.org.pl)

Vine (http://www.vinelinux.org)

APT4RPM (http://apt4rpm.sf.net)

Alt Linux (http://www.altlinux.ru/)

Red Hat (http://www.redhat.com/)

Sun Solaris (http://www.sun.com/)

SuSE (http://www.suse.de/)

Yellow Dog Linux (http://www.yellowdoglinux.com/)

Chapter 9

致谢

非常感谢你们，我Debian-BR项目组的好朋友们！还有你Debian，始终在我身边帮助我，给我动力不停工作为全人类做出贡献，帮助我树立拯救世界的理想。:)

我还要感谢CIPSGA，他们给予我们项目组乃至整个自由软件项目巨大的帮助，是我们灵感的源泉。

特别感谢：

Yooseong Yang <yooseong@debian.org>

Michael Bramer <grisu@debian.org>

Bryan Stillwell <bryan@bokeoa.com>

Pawel Tecza <pawel.tecza@poczta.fm>

Hugo Mora <h.mora@melix.com.mx>

Luca Monducci <luca.mo@tiscali.it>

Tomohiro KUBOTA <kubota@debian.org>

Pablo Lorenzzoni <spectra@debian.org>

Steve Langasek <vorlon@netexpress.net>

Arnaldo Carvalho de Melo <acme@conectiva.com.br>

Erik Rossen <rossen@freesurf.ch>

Ross Boylan <RossBoylan@stanfordalumni.org>

Matt Kraai <kraai@debian.org>

Aaron M. Ucko <ucko@debian.org>

Jon Åslund <d98-jas@nada.kth.se>

Chapter 10

本使用指南的新版本

本操作手册由Debian-BR项目组 (http://www.debian-br.org)撰写，我们希望它能为Debian用户提供有效的帮助。

你可以从Debian文档项目页面获得本文档的新版本：http://www.debian.org/doc/ddp。

对本文档有任何意见或建议可直接发email给我：<kov@debian.org>。(中文用户请发给译者)