Debian 快速参考手册

Osamu Aoki <osamu\#at\#debian.org>
译者：
Hao “Lyoo” Liu <iamlyoo\#at\#163.net>
Ming Hua <minghua\#at\#rice.edu>
肖盛文 <atzlinux\#at\#163.com>
Haifeng Chen <optical.dlz\#at\#gmail.com>
解彦博 <xieyanbo\#at\#gmail.com>
easthero <easthero\#at\#gmail.com>
‘作者’ on page 23

CVS, 星期四 一月 18 11:54:38 UTC 2007

摘要

Debian 快速参考手册 (http://qref.sourceforge.net/) 旨在对 Debian 系统进行简要介绍，如同一本**快速参考手册**，它是 Debian 参考手册 (http://qref.sourceforge.net/) 一文的节选。

Contents

Chapter 1

序言

本书最初是作为一本“快速参考手册”来写的，但是现在增加了很多内容。尽管如此，**保持文字简短紧凑**(keep it short and simple, KISS)是我的指导思想。

1.1 文档约定

文中许多信息通过简短的 `bash` 命令给出，以下是其排版格式约定：

参考：

- `bash(1)` 表示 Unix 风格 **manual** 页。
- `info libc` 表示 **GNU TEXINFO** 信息。

1.2 Debian 发行版(distributions)基本概念

Debian 同时维护 3 种不同的发行版。它们是：

- `stable` — 最广泛的用于架设产品化服务器，因为它只包含安全更新。
- `testing` — 推荐工作站用户使用的发行版，因为它包含有最近发布的桌面软件，这些软件已经接受了少量测试。
- `unstable` — 处在悬崖边缘的版本，供 Debian 开发者选用。

如果 `unstable` 发行版中的软件包不再出现 Release Critical(RC) 错误，大概一周后，它将自动升级到 `testing` 发行版。

Debian 发行版有代码名称。Woody 在2002年8月发布，在此之前，三个发行版对应为 Potato、Woody 和 Sid。在 Woody 发布后，三个发行版对应为 Woody、Sarge 和 Sid。当 Sarge 发布，`stable` 和 `unstable` 发行版对应为 Sarge 和 Sid；一个新的 `testing` 将被创建，(由 `stable` 复制而来) 并被分配一个代码名。

订阅低流量的邮件列表 `debian-devel-announce@lists.debian.org`，可以得到关于 Debian 的重要声明信息。

如果你想使用比发行版自带软件包更新的软件包版本，你可以按照‘发行版升级到 `stable`、`testing` 或 `unstable`’ on the next page 的描述，升级到一个新的发行版；或者你只升级选择的软件包。如果该软件包不能够容易的升级，你可以按照‘向 `stable` 系统引入软件包’ on page 17.

Chapter 2

发行版升级到 stable、testing 或 unstable

2.1 从 Potato 升级到 Woody

由于 Potato 版中的 APT 没有目前 `apt_preferences(5)` man 手册页所描述的所有功能，因此需要独立描述该过程。

在仅将 Woody 源放到 `/etc/apt/sources.list` 后，通过下面的方法升级 APT 系统和所需要的核心包到 Woody：

```
# apt-get update
# apt-get install libc6 perl libdb2 debconf
# apt-get install apt apt-utils dselect dpkg
```

然后升级剩下的系统到 Woody。

```
# apt-get upgrade
# apt-get dist-upgrade
```

2.2 准备升级工作

你可以用通过网络获取软件包的方式来将一个版本升级到另外的一个版本。这可以通过如下的方法来做。

生成一个干净的 `stable` 版存储列表：

```
# cd /etc/apt
# cp -f sources.list sources.list.old
# :>sources.list
# apt-setup noprobe
```

如果你想升级到 testing，你需要增加 testing 版的存储源到这个新的列表。如果你想升级到 unstable，你还需要增加 unstable 版的存储源。

```
# cd /etc/apt
# grep -e "^deb " sources.list >srcs
# :>sources.list
# cp -f srcs sources.list
# sed -e "s/stable/testing/" srcs >>sources.list
# sed -e "s/stable/unstable/" srcs >>sources.list
# apt-get update
# apt-get install apt apt-utils
```

调整 /etc/apt/sources.list 和 /etc/apt/preferences 的艺术请参阅 'Debian 软件包管理基础' on page 6。

2.3 升级

在按照描述的方法正确的设置 /etc/apt/sources.list 和 /etc/apt/preferences 文件后，你便可以开始升级了。

请注意，升级到 Debian testing 版有一个负面影响，包含安全补丁的软件包更新将会非常缓慢。因为这些软件包首先会被上载到 unstable，稍后才会移植到 testing。

软件包的实质性信息请参见 'Debian 软件包管理' on the facing page，如果你遇到问题，请查看 'APT 升级错误以及解决方法' on page 11。

2.3.1 使用 dselect

如果系统在许多软件包都包含了 -dev 等软件包，推荐使用下面的 dselect 操作方法进行控制软件包的细化操作。

```
# dselect update   # 升级前请先完成这步
# dselect select   # 选择附加软件包
```

运行 dselect 时当前所有软件包均被选中，dselect 会提示你基于 Depends，Suggests 和 Recommends 的附加软件包，如果不想添加任何软件包，只需输入 Q 退出 dselect。

```
# dselect install
```

在安装过程中，必须回答一些有关软件包配置的问题，准备好你的笔记本花点时间处理它们。参阅 'dselect' on page 8。

使用 dselect。**它能干得不赖 :)**

Chapter 3

Debian 软件包管理

高级包管理工具 aptitude 是目前首选的字符界面的 APT 前端程序。 它会记住哪些包是你安装的，哪些是为了满足依赖关系而安装的；在不被已安装包需要的情况下aptitude 会自动卸载后者。它内建一套高级的包过滤器，但是比较难上手。

synaptic 是目前首选的基于 GTK 的图形化 APT 前端程序。它的包过滤器比 aptitude 的好用多了。它包含了对 Debian Package Tags (http://debtags.alioth.debian.org/) 的实验性支持。

为了减少 Debian 仓库的网络负担和加快你下载的速度，你应该从 Debian 镜像下载。

如果你需要在你本地网络的许多台机器上安装相同的包。在使用 APT 下载包的时候，请考虑使用 squid 来设置本地 HTTP 代理。必要的话，可以设置环境变量 http_proxy 或者在 /etc/apt/apt.conf 里面设置 http 的值。

尽管 apt_preferences(5) 中描述的 APT 的 pinning 功能非常强大，但造成的影响是难以察觉和管理的。你应该把它作为一个高级功能来看待。

中描述的使用方法非常适合于需要同时确保系统的稳定性和使用最新软件的情况。

本章节是基于 Woody 之后的系统写的，某些东西只适合于 Sarge 或更新的系统。

3.1 介绍

如果你没有精力阅读完所有的开发者文档，那么先看看本章的内容，然后开始体验 Debian testing/unstable 的威力吧:-)

3.1.1 主要的包管理工具

```
dpkg       - Debian 包安装工具
apt-get    - APT 的命令行前端
aptitude   - APT 的高级的字符和命令行前端
synaptic   - 图形界面的 APT 前端
```

```
dselect    - 使用菜单界面的包管理工具
tasksel    - Task 安装工具
```

这些工具不是用来取代对方的，比如 dselect 同时使用 APT 和 dpkg。

APT 使用 /var/lib/apt/lists/* 来跟踪可用的软件包，而 dpkg 使用的是 /var/lib/dpkg/available。如果你使用了 aptitude 或者其他 APT 前端来安装软件包，同时你希望使用 dselect 来安装软件包，请不要忘记使用 dselect 菜单上的 [U]pdate(或者运行"dselect update")来更新 /var/lib/dpkg/available。

在处理依赖关系上 apt-get 会自动下载安装依赖的软件包，但是不会处理所安装软件推荐的或者建议的软件包。

相反 aptitude 可以设置成安装所安装软件推荐的或者建议的软件包。

dselect 给使用者列出所安装软件推荐或建议的软件包，可以进行单独选择。

3.1.2 方便的工具

```
dpkg-reconfigure  - 重新配置已安装的软件包
 (如果它是使用 debconf 进行配置的)
dpkg-source       - 管理源码包
dpkg-buildpackage - 自动生成包文件
apt-cache         - 在本地缓冲区检查包文件
```

3.2 Debian 软件包管理基础

3.2.1 设置 APT

参考 '准备升级工作' on page 3 来设置 sources.list。[1]

3.2.2 安装 tasks

你可以安装一些软件包集合，这些集合是由使 Debian 系统满足某些特定用途的典型软件包组成的。这些集合被称为"tasks"。

在初始化安装中，安装 tasks 最简单的方法就是使用 tasksel。注意在使用之前，你需要运行

```
dselect update
```

建议使用 aptitude 来安装 tasks。它能让你在选择好 tasks 并准备安装之前，删除 tasks 中的某些软件包。

[1]如够你使用 testing 或者 unstable，您可以移除 /etc/apt/sources.list 和 /etc/apt/preferences 中的 stable。因为 testing 最初拷贝自 stable。

3.2.3 aptitude

aptitude 是全新的可菜单操作的包安装工具，和 dselect 类似，但是是针对 APT 从头设计的。从大多数参数来讲，aptitude 完全可以作为 apt-get 的一个兼容的代替品。参阅 aptitude(1) 和 /usr/share/doc/aptitude/README.

一旦开始使用 aptitude，你最好继续使用它，而不是选择其他替代工具。否则你将失去 aptitude 包存的软件安装清单，你就不能享受自动删除多余软件包的功能了。

全屏状态下 aptitude 接受单键的命令，大多数是小写的。主要的几个功能键如下：

按键	动作
F10	菜单
?	按键命令帮助 (完整的清单)
u	更新软件包信息
+	标记软件包为升级或者新安装
-	标记软件包为删除 (保留配置文件)
_	标记软件包为完全删除 (删除配置文件)
=	保持软件包的当前版本，阻止其被升级
U	标记所有可以升级的软件包为升级
g	下载和安装选择的软件包
q	退出当前屏幕，保存改变
x	退出当前屏幕，忽略改变
Enter	查看一个软件包的信息
C	查看一个软件包的更新日志
l	改变软件包树状显示限制
/	搜索第一个匹配的软件包
\	重复最后一次搜索

和 apt-get 一样，aptitude 安装软件包的时候自动解决依赖问题。aptitude 还能安装即将安装的软件包推荐或者建议的软件包。你通过 F10 -> 选项 -> 处理依赖关系 在菜单上更改这一默认设置。

aptitude 的其他特点如下：

- aptitude 能访问所有版本的软件包。
- aptitude 的动作记录在 /var/log/aptitude。
- aptitude 能轻松的追踪陈旧的和本地建立的软件包，并在“过期的和在本地创建的软件包”上列出。
- aptitude 内建强大的包搜索和显示功能。熟悉 mutt 的用户很容易上手，因为这个显示方法的灵感来源于 mutt。参阅 /usr/share/doc/aptitude/README 中的“SEARCHING, LIMITING, AND EXPRESSIONS”
- aptitude 在全屏状态下有嵌入的 su 功能。普通用户都可以执行，直到安装或删除软件的时候再取得管理员权限。

3.2.4 dselect

从 stable 发行到现在为止(包含 Potato)，dselect 一直是主要的包维护工具。对于 Sarge，你可以考虑用 aptitude 代替。

当你启动程序的时候，dselect 会自动选择所有“Required”“Important”和“Standard”的包。

dselect 的用户界面是有些奇怪，但是大部分人已经习惯了。 它有四个主要命令：(指令都是大写的！)：

按键	动作
Q	退出。确认当前的选择并退出。 (忽略依赖关系)
R	撤销！ 我不是那个意思。
D	不管他！我不管你 dselect 怎么想的，照做就好了！
U	都照建议的来做

使用 D 和 Q，你可以选择有冲突的选项。请小心使用这个命令。

在 /etc/dpkg/dselect.cfg 中加上一行“expert”来减少干扰。

如果你的机器运行 dselect 的速度很慢，你可以考虑在速度快一点的机器上运行 dselect，确定你要安装的软件包之后，在慢的机器上通过 apt-get 来安装它们。

3.2.5 使用 APT 来维护发行版本

请编辑 /etc/apt/preferences 并加入以下内容来维持系统为 testing 版本：

```
Package: *
Pin: release a=testing
Pin-Priority: 800

Package: *
Pin: release a=stable
Pin-Priority: 600
```

要注意的是追踪 testing 版本可能带来延误安装安全性修正软件包的副作用。因为这些软件包都是上传到 unstable 一段时间后再移植到 testing。

更多复杂的例子请参考 apt_preferences(5)，可以让您做更多的事情，例如安装 unstable 的软件包的同时还能把系统维持在 testing。

关于限制特定软件在特定版本上，而其他软件随系统升级的设置在 examples subdirectory (http://www.debian.org/doc/manuals/debian-reference/examples/) 找到，即 preferences.testing 和 preferences.unstable。

如果你混用不同的发行版本，例如 testing 和 stable 或 unstable 和 stable,你终究还是会安装上 testing 或 unstale 版本的核心软件，例如 libc6,这样作无法确保系统中没有臭虫。你需要特别小心。

另外一个例子，preferences.stable，会强制降级所有的软件到 stable。

Debian 不支持将某个**软件包**降级到先前的发行版本。然而在新的软件包出问题时，你会发现你不得不安装旧的可用的软件包。你可以在本地的 /var/cache/apt/archives/ 或远端的 http://snapshot.debian.net/ 中找到先前的版本。请参考 '使用 dpkg 救助' on page 11。

从某个**发行版本**降级到先前的发行版本也是不被支持的，而且这样做往往造成很多问题。不过你愿意冒险的话，作为最后的手段这样做也是值得的。

3.2.6 aptitude, apt-get 和 apt-cache 命令

还是以上面使用 testing 发行版的用户为例,可使用下列命令来管系统：

- aptitude upgrade (或 apt-get upgrade 或 aptitude dist-upgrade 或 apt-get dist-upgrade)

  这样就会跟随 testing 版本 — 它们会跟踪 testing 版本的更新情况，对系统上所有软件包进行升级，并从 testing 处重新分析依赖关系并安装相关的包。[2]

- apt-get dselect-upgrade

  这个命令跟踪 testing 版本 — 根据 dselect 的选择对系统上的软件包进行升级。

- aptitude install *package*/unstable

  从 unstable 中安装 *package*，并由 testing 版本提供安装依赖的包。

- aptitude install -t unstable *package*

  通过设置 unstable 的 Pin-Priority 为 990，可以从 unstable 处安装 *package* 及其依赖的包。

- apt-cache policy *foo bar ...*

  检查 *foo bar ...* 软件包的状态。

- aptitude show *foo bar ...* | less(或 apt-cache show *foo bar ...* | less)

  查看 *foo bar ...* 软件包的有关信息。

- aptitude install *foo=2.2.4-1*

  安装 *foo* 软件包的特定版本 *2.2.4-1*。

[2] upgrade 和 dist-upgrade 仅仅在新的软件包依赖关系和旧的不同时才表现出不同的处理方式。请参考 apt-get(8) 来了解更多细节。aptitude upgrade 和 aptitude dist-upgrade 运行在 aptitude 的命令行模式。通过按键 e 你可以切换到全屏模式。

- `aptitude install` *`foo bar-`*
  安装 *foo* 软件包并删除 *bar* 软件包。
- `aptitude remove` *`bar`*
  删除 *bar* 软件包，但保留其配置文件。
- `aptitude purge` *`bar`*
  删除 *bar* 软件包及其所有配置文件。

在上面的例子中使用 `-u` 选项的作用是在实际升级之前将所有将要升级的软件包列出，并提示用户确认。下面的操作可将 `-u` 设置为默认行为：

```
$ cat >> /etc/apt/apt.conf << .
// Always show packages to be upgraded (-u)
APT::Get::Show-Upgraded "true";
.
```

使用 `--no-act` 可进行模拟升级，并不是进行真正的升级行为。

3.3 Debian 生存命令

掌握了这些知识，你就能够享受无尽的“升级”了 :-)

3.3.1 检测程序错误寻求帮助

如你使用某个软件包出现问题，在寻求帮助或发送错误报告之前请确认查看过下列站点 (`lynx`, `links` 和 `w3m` 都很好用):

```
$ lynx http://bugs.debian.org/
$ lynx http://bugs.debian.org/package-name  # 如果你知道软件包的名字
$ lynx http://bugs.debian.org/bugnumber     # 如果你知道错误序号
```

在 Google(www.google.com)中使用关键字“site:debian.org”搜索。

如有疑问，可阅读帮助文件。设置 `CDPATH` 如下：

```
export CDPATH=.:/usr/local:/usr/share/doc
```

然后输入

```
$ cd packagename
$ pager README.Debian # 如果存在的话
$ mc
```

3.3.2 APT 升级错误以及解决方法

从 unstable/testing 进行升级时可能出现 '升级' on page 4 中提到的软件包关联问题。多数情况下，是因为升级的软件包所需的新增的关联包没有安装。可使用如下方法解决：

```
# aptitude dist-upgrade
```

如果这招无效，可以重复下面的方法至到问题解决：

```
# aptitude -f upgrade          # 即使遇到错误也继续 upgrade
... 或
# aptitude -f dist-upgrade     # 即使遇到错误也继续 dist-upgrade
```

一些的确存在问题的升级脚本会引起持续出错。最好的解决方法是检查该软件包的安装脚本 /var/lib/dpkg/info/*packagename*.{post-,pre-}{install,removal} 然后运行：

```
# dpkg --configure -a     # 配置所有安装的软件包
```

如果脚本报告缺少配置文件，查看一下 /etc 中相关的配置文件。如果配置文件有 .dpkg-new 扩展名(或其它类似的扩展名)，去掉(mv)它的扩展名。

从 unstable/testing 进行升级时可能出现软件包关联问题。可用这个方法智取：

```
# aptitude -f install package # 重载坏关联
```

还可以用 equivs 包来解决此类问题。参阅 /usr/share/doc/equivs/README.Debian。

3.3.3 使用 dpkg 救助

如果你在使用 APT 的时候遇到死胡同了，那么可以从 Debian 的镜像站点下载软件包并使用 dpkg 来安装。如果你不能访问网络，可以在 /var/cache/apt/archives/ 中找到被缓存的软件包。

```
# dpkg -i fetchmail_6.2.5-4_i386.deb
```

如果你用这种方法安装软件包，但是遇到了依赖问题安装失败了，并且你确实需要安装这个软件包。你可以用 dpkg 的 --ignore-depends，--force-depends 和其他参数来安装软件包。dpkg(8) 有更详细的介绍。

3.3.4 恢复软件包选择状态的数据

如果 /var/lib/dpkg/status 因为某种原因坏掉了，Debian 系统将会完全丢失软件包选择状态的数据。赶快到 /var/lib/dpkg/status-old 或 /var/backups/dpkg.status.* 下找找旧的 /var/lib/dpkg/status 文件。

将 /var/backups/ 放在其它的分区是个好习惯，因为该目录包含了许多非常重要的系统数据。

如果旧的 /var/lib/dpkg/status 文件也坏了，仍可以从 /usr/share/doc/ 下的目录进行恢复这些信息。

```
# ls /usr/share/doc | \
  grep -v [A-Z] | \
  grep -v '^texmf$' | \
  grep -v '^debian$' | \
  awk '{print $1 " install"}' | \
  dpkg --set-selections
# dselect --expert # 重新安装系统，如果需要的话去除一些选项
```

3.3.5 /var 崩溃之后如何恢复系统

/var 目录包含着定时更新的数据如 mail，它们很容易遭破坏。将目录放到别的分区可降低风险，如果最坏的事情发生了，可以通过重建 /var 目录来挽救 Debian 系统。

从相同或旧版本的最简 Debian 系统中取得 /var 目录的内容框架，例如 var.tar.gz (http://people.debian.org/~osamu/pub/)，然后它放入受损系统的 root 目录，接着

```
# cd /
# mv var var-old       # 如果里面还有其他有用资料的话
# tar xvzf var.tar.gz # 使用 Woody 框架文件
# aptitude             # 或是用 dselect
```

上述步骤可使系统恢复工作。使用'恢复软件包选择状态的数据' on this page 中描述的技术加速软件包选择数据的恢复。([FIXME]：该过程需要更多的实践来检验)

3.3.6 为无法启动的系统安装软件包

使用 Debian 急救软盘 /CD 或从多启动 Linux 系统其它分区启动。将无法启动的系统挂载到 /target 并使用 dpkg 的 chroot 安装模式。

```
# dpkg --root /target -i packagefile.deb
```

接下来就可以着手配置并解决问题。

如是只是由于 lilo 损坏而造系统无法启动，可使用标准 Debian 急救盘启动。假设你的 root 分区位于 /dev/hda12 且想使用 runlevel 3，在启动提示符输入：

```
boot: rescue root=/dev/hda12 3
```

这样，你就可以使用软盘中内核启动系统，新系统的功能基本齐全。(可能丢失某些内核特性或模块)

3.3.7 如果 dpkg 命令出错怎么办

如果 dpkg 损坏就不能安装任何 .deb 文件。下面的操作可帮助你修复这种状况。(在第一行，你可将“links”替换成你喜欢的浏览器。)

```
$ links http://http.us.debian.org/debian/pool/main/d/dpkg/
  ... 下载完好的 dpkg_version_arch.deb
$ su
password: *****
# ar x dpkg_version_arch.deb
# mv data.tar.gz /data.tar.gz
# cd /
# tar xzfv data.tar.gz
```

对 i386，亦可用 http://packages.debian.org/dpkg 作为 URL。

3.4 Debian 必杀技

有了这些命令的**启迪**，你将会从无休止的升级冲突的地狱中解放出来，达到 Debian **天堂**。:-)

3.4.1 文件信息

在已安装的软件包中许找特定文件所属的软件包:

```
$ dpkg {-S|--search} pattern
```

或者搜索 Debian archive:

```
$ wget http://ftp.us.debian.org/debian/dists/sarge/Contents-i386.gz
$ zgrep -e pattern Contents-i386.gz
```

或是用专门的软件包命令：

```
# aptitude install dlocate
$ dlocate filename          # dpkg -L 和 dpkg -S 的高效代替品
...
# aptitude install auto-apt # 请求式软件包安装工具
# auto-apt update           # 为 auto-apt 建立 db 文件
$ auto-apt search pattern
                 # 在所有软件包中搜索 pattern，不论安装与否
```

3.4.2 软件包信息

搜索并显示包文件的信息。编辑 /etc/apt/sources.list，让 APT 指向正确的包文件。如果想了解 testing/unstable 中的相应软件包与当前系统安装的软件包有何差别，使用 apt-cache policy — 更好。

```
# apt-get   check             # 更新缓冲区并检查损坏的软件包
$ apt-cache search  pattern # 按文本描述搜索软件包
$ apt-cache policy  package # 软件包的 priority/dists 信息
$ apt-cache show -a package # 显示所有 dists 中软件包描述信息
$ apt-cache showsrc package # 显示相应源码包的信息
$ apt-cache showpkg package # 软件包调试信息
# dpkg  --audit|-C            # 搜索未完成安装的软件包
$ dpkg {-s|--status} package ... # 已安装软件包描述
$ dpkg -l package ...         # 已安装软件包的状态(每个占一行)
$ dpkg -L package ...         # 列出软件包安装的文件的名称
```

Woody 发布版没有为 apt-cache showsrc 建档，但该命令可用:)

你也这可这样查看软件包信息(我用 mc 浏览)：

```
/var/lib/apt/lists/*
/var/lib/dpkg/available
```

比较下面的文件可以确切了解最近的安装过程对系统造成了那些改变。

```
/var/lib/dpkg/status
/var/backups/dpkg.status*
```

3.4.3 使用 APT 无人值守安装

使用 APT 无人执守安装，要在 /etc/apt/apt.conf 中加上一行：/etc/apt/apt.conf:

```
Dpkg::Options {"--force-confold";}
```

另一种等价的方法是运行 apt-get -q -y *packagename*。这种方法可能产生严重的负作用，所以使用起来要小心。参阅 apt.conf(5)和 dpkg(1)。

安装完毕以后，可以用 '重新配置已安装的软件包' on the current page 中的方法配置特定的软件包。

3.4.4 重新配置已安装的软件包

使用下列方法重新配置已安装的软件包。

```
# dpkg-reconfigure --priority=medium package [...]
# dpkg-reconfigure --all   # 重新配置所有的软件包
# dpkg-reconfigure locales # 生成额外的 locales
# dpkg-reconfigure --p=low xserver-xfree86 # 重新配置 X 服务器
```

如果你想永久改变 debconf 对话框模式，可这么做。

某些程序用于生成特殊的配置脚本。[3]

```
apt-setup      - 创建 /etc/apt/sources.list
install-mbr    - 安装主引导(Master Boot Record)管理器
tzconfig       - 设定本地时间
gpmconfig      - 设置 gpm 鼠标 daemon
eximconfig     - 配置 Exim (MTA)
texconfig      - 配置 teTeX
apacheconfig   - 配置 Apache (httpd)
cvsconfig      - 配置 CVS
sndconfig      - 配置声音系统
...
update-alternatives - 设定默认启动命令，例如设定 vi 启动 vim
update-rc.d         - System-V init 脚本管理工具
update-menus        - Debian 菜单系统
...
```

3.4.5 删除和清除软件包

删除软件包但保留其配置文件：

```
# aptitude remove package ...
# dpkg   --remove package ...
```

删除软件包并清除配置文件：

```
# aptitude purge  package ...
# dpkg   --purge  package ...
```

[3]某些 *config 脚本在 Sarge 之后的发行版本里面消失了。软件包的配置功能已经转向 debconf 系统。

3.4.6 阻止旧软件包升级

举个例子，要阻止 `libc6` 和 `libc6-dev` 通过 `dselect` 或使用 `aptitude install` *package* 命令升级，可执行：

```
# echo -e "libc6 hold\nlibc6-dev hold" | dpkg --set-selections
```

这种方法不影响 `aptitude install` *package* 命令操作。要阻止 `aptitude upgrade` *package* 或 `aptitude dist-upgrade` 命令对软件包执行的强制自动降级行为，可在 `/etc/apt/preferences` 中加上：

```
Package: libc6
Pin: release a=stable
Pin-Priority: 2000
```

这里 "`Package:`" 后不能使用通配符如 "`libc6*`"，如果要保持所有与 `glibc` 源码包相关的二进制包的版本同步，可以明确的列出它们。

该命令可以显示处于 "阻止" 状态的软件包：

```
dpkg --get-selections "*"|grep -e "hold$"
```

3.4.7 stable/testing/unstable 混合系统

`apt-show-versions` 可以列出发行版中可用软件包的版本。

```
$ apt-show-versions | fgrep /testing | wc
... 你有多少 testing 软件包
$ apt-show-versions -u
... 列出可升级的软件包
$ aptitude install `apt-show-versions -u -b | fgrep /unstable`
... 将所有 unstable 软件包升级到最新版本
```

3.4.8 删除缓存包文件

使用 APT 安装软件包会在 `/var/cache/apt/archives` 目录留下缓存文件，要清除这些文件可使用：

```
# aptitude autoclean # 仅删除无用的包
# aptitude clean     # 删除所有的包
```

3.4.9 记录/拷贝系统配置

对软件包选择情况进行本地备份：

```
# debconf-get-selections > debconfsel.txt
# dpkg --get-selections "*" >myselections   # 或使用 \*
```

"*" 使 *myselections* 包含那些被指定“完全删除(purge)”的文件。

你可将这个文件发送到另一台电脑并在那儿按文件中的选择进行软件包安装。

```
# dselect update
# debconf-set-selections < debconfsel.txt
# dpkg --set-selections <myselections
# apt-get -u dselect-upgrade    # 或者 dselect install
```

3.4.10 向 stable 系统引入软件包

对 stable 系统进行部分升级，在软件运行环境中重新编译源码的确是个诱人的想法，这样可以避免由于关联关系不得不对大量软件包升级。首先，将下列镜像源加入 /etc/apt/sources.list：

```
deb-src http://http.us.debian.org/debian testing \
 main contrib non-free
deb-src http://http.us.debian.org/debian unstable \
 main contrib non-free
```

由于屏幕输出的限制，上述每条 deb-src 命令均分成了 2 行，实际上在 sources.list 中它们均为单行。

然后下载源码并在本地生成软件包：

```
$ apt-get update  # 更新软件包搜索列表
$ apt-get source package
$ dpkg-source -x package.dsc
$ cd package-version
  ... 查找需要的软件包(编译所需的关联包列在.dsc文件中)并安装它们，
      你还需要“fakerroot”软件包。

$ dpkg-buildpackage -rfakeroot

  ……或者(没有签名)
$ dpkg-buildpackage -rfakeroot -us -uc # 如果需要，再使用“debsign”

  ……然后安装
$ su -c "dpkg -i packagefile.deb"
```

通常，需要安装一些带“-dev”后缀的软件包以满足关联关系。debsign 在 devscripts 软件包中。auto-apt 可以轻松解决这些关联问题。请使用 fakeroot，如是没有必要，就别使用 root 帐号。

在 Woody 中，这些关联问题已被简化。例如，编译 pine 源码包：

```
# apt-get build-dep pine
# apt-get source -b pine
```

3.4.11 本地软件包文件

为了创建与 APT 和 dselect 系统兼容的本地软件包文件，需要创建 Packages，包中文件要放在特定的目录树中。

Debian 官方包文件喜欢存放于本地 deb 仓库，下面就来创建仓库：

```
# aptitude install dpkg-dev
# cd /usr/local
# install -d pool # 软件包存放的物理地址
# install -d dists/unstable/main/binary-i386
# ls -1 pool | sed 's/_.*$/ priority section/' | uniq > override
# 编辑 override # 调整 priority and section
# dpkg-scanpackages pool override /usr/local/ \
   > dists/unstable/main/binary-i386/Packages
# cat > dists/unstable/main/Release << EOF
Archive: unstable
Version: 3.0
Component: main
Origin: Local
Label: Local
Architecture: i386
EOF
# echo "deb file:/usr/local unstable main" \
   >> /etc/apt/sources.list
```

还有一种快速但随意的方法来创建本地 deb 仓库：

```
# aptitude install dpkg-dev
# mkdir /usr/local/debian
# mv /some/where/package.deb /usr/local/debian
# dpkg-scanpackages /usr/local/debian /dev/null | \
  gzip - > /usr/local/debian/Packages.gz
#  echo "deb file:/usr/local/debian ./" >> /etc/apt/sources.list
```

在 /etc/apt/sources.list 中设置相应镜像源入口地址，就可以通过 HTTP 或 FTP 方式远程访问存放在其中的包文件了。

3.4.12 转换或安装外来的二进制软件包

alien 可将其它格式的二进制软件包如 Redhat 的 rpm、Stampede 的 slp 、Slackware 的 tgz 和 Solaris 的 pkg 等转化成 Debian 的 deb 格式软件包，如果你想在自己的系统上使用别的 Linux 发行版中的软件包，可使用 alien 将它转化成系统首选的软件包格式后安装。alien 还支持 LSB 的软件包。

3.4.13 自动安装命令

auto-apt 是一种按需安装的软件包安装工具。

```
$ sudo auto-apt update
 ... 升级数据库
$ auto-apt -x -y run
进入 auto-apt 模式：/bin/bash
退出这个命令继而退出 auto-apt 模式。
$ less /usr/share/doc/med-bio/copyright # 访问不存在的文件
 ... 安装提供了这个文件的软件包。
 ... 同样安装依赖的包
```

3.4.14 校验已安装的软件包

debsums 可以校验已安装软件包的 MD5 编码，对某些软件包没有可用的 MD5 编码，系统管理员可使用一个临时的解决办法：

```
# cat >>/etc/apt/apt.conf.d/90debsums
DPkg::Post-Install-Pkgs {"xargs /usr/bin/debsums -sg";};
^D
```

来自 Joerg Wendland <joergland@debian.org> (untested).

3.5 其他 Debian 的特性

3.5.1 dpkg-divert 命令

使用文件**转移**(diversions)的方法可以强令 dpkg 将文件安装到 **转移** 目录而非默认目录。对于某个引起冲突的文件，可以在 Debian 软件包脚本中使用 **Diversions** 将它安装到别的目录。系统管理员还可以使用 diversion 来重载软件包配置文件，或者用来保留某些旧配置文件(这些文件没有在 **conffiles** 中登记)当安装新版软件时这些文件会被覆盖。。

```
# dpkg-divert [--add]  filename # 添加 “转移”
# dpkg-divert --remove filename # 删除 “转移”
```

记住，不到万不得已不要使用 dpkg-divert。

3.5.2 **equivs** 软件包

如果你从源码编译程序，最好将它做成本地 Debian 化软件包(*.deb)。最新的方法是使用 equivs。

```
Package: equivs
Priority: extra
Section: admin
Description: Circumventing Debian package dependencies
 This is a dummy package which can be used to create Debian
 packages, which only contain dependency information.
```

3.5.3 Alternative 命令

如果想用 vi 来启动 vim，请用 update-alternatives：

```
# update-alternatives --display vi
...
# update-alternatives --config vi
  Selection    Command
-----------------------------------------------
      1        /usr/bin/elvis-tiny
      2        /usr/bin/vim
*+    3        /usr/bin/nvi

Enter to keep the default[*], or type selection number: 2
```

Debian alternatives 系统中的这些项目，都是以符号连接的形式存放在 /etc/alternatives 下的。

想设置你喜爱的 X window 环境，执行 update-alternatives 来指定 /usr/bin/x-session-manager 和 /usr/bin/x-window-manager。

/bin/sh 是指向 /bin/bash 或 /bin/dash 的链接。想兼容旧的 Bash 脚本，使用 /bin/bash 比较保险，但更好还是使用 /bin/dash，因为它更符合 POSIX 标准。升级到 2.4 版 Linux 内核，系统一般将它设置为 /bin/dash。

3.5.4 运行级别 Runlevel

安装好之后，大部分 Debian 软件包的服务被设定为在 runlevel 2 到 5 时运行。所以，在没有定制过的 Debian 系统中，runleve 2、3、4、5、6 是没有区别的。Debian 保留这些给本地管理员使用。 to customize runlevels. 这样的 runlevels 系统和其他流行的 GNU/Linux 发行版本完全不同。 你可能要做的改变之一就是取消 runlevel 2 上的 xdm 和 gdm,使得在完成启动之后 X 显示管理去不会自动启动；然后你可以通过切换到 runlevel 3 来启动 X 显示管理器。

3.5.5 停止 daemon 服务

Debian 发行版非常注重系统安全，并期望系统管理员能担此重任。它将系统的易用性放在了第二位，许多 daemon 服务都定位在最高安全级别，因而，默认安装状态下系统只启动最少的(甚至没有)可用的服务。

如果拿不定主意(有关 Exim、DHCP...)，可执行 `ps aux` 或检查 `/etc/init.d/*` 和 `/etc/inetd.conf` 下的内容，还可以检查 `/etc/hosts.deny`。`pidof` 命令也很有用(参阅 `pidof(8)`)

在最近的 Debian 系统中，默认状态下 X11 不允许 TCP/IP(远程)连接。使用 SSH 进行 X 转发也是禁用的，

Appendix A

附录

A.1 作者

Debian 快速参考手册由 Osamu Aoki `<osamu\#at\#debian.org>` 发起，最初是一部个人的系统安装备忘录，而最终称之为“Quick Reference ...”。许多内容来自于“debian-user”邮件列表中的存档，同时也参考了《Debian Installation Manual》和《Debian Release Notes》。

由 Debian Documentation Project (`http://www.debian.org/doc/ddp`)(DDP)的积极参与者、也是目前《The Debian FAQ》的主要维护者的 Josip Rodin 提议，将本文档更名为“Debian 参考手册”并将《The Debian FAQ》中一些类似参考形式的章节内容合并过来。后来摘录出一部分内容形成了《Debian 快速参考手册》。

本文档的编辑、翻译、扩充工作由下列 QRFF 组员参与：

- 英文版初稿及《Quick Reference...》的初稿
  - Osamu Aoki `<osamu\#at\#debian.org>` (leader: all contents)
- 英文版校对和其它贡献
  - David Sewell `<dsewell\#at\#virginia.edu>` (extensive work: en style)
  - Thomas Hood `<jdthood\#at\#yahoo.co.uk>` (network related)
  - Brian Nelson `<nelson\#at\#bignachos.com>` (especially X related)
  - Jan Michael C Alonzo `<jmalonzo\#at\#spaceants.net>`
  - Daniel Webb `<webb\#at\#robust.colorado.edu>`
  - 所有翻译者反馈
- 法文版翻译
  - Guillaume Erbs `<gerbs\#at\#free.fr>` (leader: fr)
  - Renald Casagraude `<rcasagraude\#at\#interfaces.fr>`
  - Jean-Pierre Delange `<adeimantos\#at\#free.fr>`
  - Daniel Desages `<daniel\#at\#desages.com>`
- 意大利文版翻译
  - Davide Di Lazzaro `<mc0315\#at\#mclink.it>` (leader: it)
- 葡萄牙文(巴西)版翻译
  - Paulo Rogerio Ormenese `<pormenese\#at\#uol.com.br>` (leader: pt-br)
  - Andre Luis Lopes `<andrelop\#at\#ig.com.br>`

  - Marcio Roberto Teixeira <marciotex\#at\#pop.com.br>
  - Rildo Taveira de Oliveira <to_rei\#at\#yahoo.com>
  - Raphael Bittencourt Simoes Costa <raphael-bsc\#at\#bol.com.br>
  - Gustavo Noronha Silva <kov\#at\#debian.org> (coordinator)
- 西班牙文版翻译
  - Walter Echarri <wecharri\#at\#infovia.com.ar> (leader: es)
  - Jose Carreiro <ffx\#at\#urbanet.ch>
- 德文版翻译
  - Jens Seidel <tux-master\#at\#web.de> (leader: de)
  - Willi Dyck <wdyck\#at\#gmx.net>
  - Stefan Schroeder <stefan\#at\#fkp.uni-hannover.de>
  - Agon S. Buchholz <asb\#at\#kefk.net>
- 波兰版翻译 — PDDP (http://debian.linux.org.pl) 的下列成员：
  - Marcin Andruszkiewicz
  - Mariusz Centka <mariusz.centka\#at\#debian.linux.org.pl>
  - Bartosz Feński <fenio\#at\#debian.linux.org.pl> (leader: pl)
  - Radosław Grzanka <radekg\#at\#debian.linux.org.pl>
  - Bartosz 'Xebord' Janowski
  - Jacek Lachowicz
  - Rafał Michaluk
  - Leonard Milcin, Jr.
  - Tomasz Z. Napierała <zen\#at\#debian.linux.org.pl>
  - Oskar Ostafin <cx\#at\#debian.linux.org.pl>
  - Tomasz Piękoś
  - Jacek Politowski
  - Mateusz Prichacz <mateusz\#at\#debian.linux.org.pl>
  - Marcin Rogowski
  - Paweł Różański
  - Mariusz Strzelecki
  - Krzysztof Ścierski
  - Przemysław Adam Śmiejek <tristan\#at\#debian.linux.org.pl>
  - Mateusz Tryka <uszek\#at\#debian.linux.org.pl>
  - Cezary Uchto
  - Krzysztof Witkowski <tjup\#at\#debian.linux.org.pl>
  - Bartosz Zapałowski <zapal\#at\#debian.linux.org.pl>
- 简体中文版翻译
  - 刘浩(LYOO) <iamlyoo\#at\#163.net>
  - Ming Hua <minghua\#at\#rice.edu>
  - 肖盛文 <atzlinux\#at\#163.com> (leader: zh-cn)
  - Haifeng Chen <optical.dlz\#at\#gmail.com>
  - 解彦博 <xieyanbo\#at\#gmail.com>
  - easthero <easthero\#at\#gmail.com>
- 繁体中文版翻译
  - Asho Yeh <asho\#at\#debian.org.tw> (leader: zh-tw)
  - Tang Wei Ching <wctang\#at\#csie.nctu.edu.tw> (ex-leader: zh-tw)
- 日文版翻译

– Shinichi Tsunoda <tsuno\#at\#ngy.1st.ne.jp> (leader: ja)
– Osamu Aoki <osamu\#at\#debian.org>

A.2 保证

因为我不是个专家，所以不敢说对 Debian 或 Linux 了如指掌。文中有关系统安全的考虑仅适用于家庭使用。

本文档不能替代任何权威指南。

本文档不承诺任何保证。所有的商标均归属于其各自的拥有者。

A.3 反馈

欢迎对本文档提出意见和建议。请发邮件至 `debian-reference` 软件包或相关翻译包下的 Debian BTS system (`http://bugs.debian.org/`)，使用 `reportbug` 来发送错误报告会更方便。请将英文邮件发给 Osamu Aoki (`http://people.debian.org/~osamu/`) 他的邮箱是 <osamu\#at\#debian.org>，其它语言的邮件可发送给相应的翻译者。