debian

Debian 参考手册

Debian 参考手册（版本 2.76）(2019-03-21 15:39:20 UTC) 旨在作为一份 Debian 系统安装后的用户指南，为 Debian 系统的使用与管理提供广泛的概览。它通过为非开发者编写的 shell 命令示例来涵盖系统管理的方方面面。

COLLABORATORS

	TITLE : Debian 参考手册	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Osamu Aoki (青木修)	November 19, 2020	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

List of Tables

Abstract

序言

Debian 参考手册（版本 2.76）(2019-03-21 15:39:20 UTC) 旨在为作为一份安装后用户指南，为 Debian 系统的使用与管理提供宽泛的概览。

本书的目标读者：愿意学习 shell 脚本，但是不准备为了理解 GNU/Linux 系统是如何运作的而阅读其所有 C 语言源代码的人。

如需系统安装指导信息，请见：

- Debian GNU/Linux 当前稳定版安装手册
- Debian GNU/Linux 当前测试版安装手册

免责声明

所有担保条款具有免责效力。所有商标均为其各自商标所有者的财产。

Debian 系统本身是一个变化的事物。这导致其文档难于及时更新并且正确。虽然是以 Debian 系统当前的不稳定版本作为写作该文档的基础，但当你阅读本文的时候，部分内容仍然可能已经过时。

请把本文档作为第二参考。本文档不能够代替任何官方指导手册。文档作者和文档贡献者对在本文档中的错误、遗漏或歧义，不承担责任后果。

什么是 **Debian**

Debian 项目是一个由个人组成的团体，该团体的成员均把创建一个自由操作系统作为共同事业。Debian 的发布具有下列特征：

- 承诺软件自由：Debian 社群契约和 Debian 自由软件指导方针（DFSG）
- 基于因特网上无酬劳的志愿者的工作发布：https://www.debian.org
- 大量预编译的高质量软件包
- 专注于稳定性和安全性，同时易于获取安全更新
- 在 `unstable` 和 `testing` 仓库中注重软件包最新版本的平滑升级
- 支持大量硬件架构

Debian 系统中的自由软件来自GNU, Linux, BSD, X, ISC, Apache, Ghostscript, Common Unix Printing System , Samba, GNOME, KDE, Mozilla, LibreOffice, Vim, TeX, LaTeX, DocBook, Perl, Python, Tcl, Java, Ruby, PHP, Berkeley DB, MariaDB, PostgreSQL, SQLite, Exim, Postfix, Mutt, FreeBSD, OpenBSD, Plan 9 以及许多更加独立的自由软件项目。Debian 将上述各种各样的自由软件集成到一个系统里面。

关于本文档

指导原则

写作本文档时，遵循下列指导原则。

- 仅提供概览，而忽略边界情况。（**Big Picture** 原则）
- 保持文字简短紧凑。（**KISS** 原则）
- 不重复造轮子。（使用链接指向已有参考）
- 专注于使用非图形的工具和控制台。（使用 **shell** 例子）
- 保持客观。（使用 popcon 等等。）

提示
我试图阐明操作系统底层和体系结构的各方面内容。

预备知识

警告
阅读本文档，你需要通过自己的努力去查找本文档未提及的问题答案。本文档仅仅提供有效的起点。

你必须自己从以下原始材料查找解决方案。

- Debian 管理员手册
- Debian 网站（https://www.debian.org）上的通用信息
- "`/usr/share/doc/<package_name>`" 目录下的文档
- Unix 风格的 **manpage**: "`dpkg -L <package_name> |grep '/man/man.*/'`"
- GNU 风格的 **info page**: "`dpkg -L <package_name> |grep '/info/'`"
- 错误报告：http://bugs.debian.org/<package_name>
- Debian Wiki（https://wiki.debian.org/）用于变化和特定的话题
- Linux 文档项目（TLDP，http://tldp.org/）的 HOWTO
- 国际开放标准组织的 UNIX 系统主页（http://www.unix.org/）上的单一 UNIX 规范
- 自由的百科全书：维基百科（https://www.wikipedia.org/）

注意
软件包的详细文档，你需要安装软件包名用"`-doc`" 作为后缀名的相应文档包来得到。

排版约定

本文通过如下使用 bash(1) shell 命令例子的简要方式来提供信息。

```
# <以 root 账户运行的命令>
$ <以普通用户账户运行的命令>
```

这些 shell 提示符区分了所使用的帐户。为了可读性，在本手册中 shell 提示符相关的环境变量被设置为“PS1='\$'”和“PS2=' '。这与实际安装的系统所使用的 shell 提示符很有可能会不同。

注意
参见在 bash(1) 中对环境变量“$PS1”和“$PS2”的解释。

要求系统管理员执行的操作，须用祈使句描述，如“在 shell 中输入命令字符串后，键入 Enter 键。”

这些描述列或类似信息在表格有一个名词短语，后面会紧跟软件包短描述，这些短语会省略掉前面的”a” 和”the”。它们也可以包含一个不定式短语作名词短语，在联机帮助的短命令描述约定后面不带”to”。有些人可能觉得这看起来有点可笑，这里故意保留这种风格是为了让文档看起来尽可能的简单。这些名词短语在短命令描述约定里并不会采用首字母大写的方式。

注意
无论专有名词和命令名位于何处，保持其英文字母大小写不变。

在文本段落中引用的命令片断由双引号括起来的打印机字体进行标记，就像”aptitude safe-upgrade”。

在文本段落中引用的来自配置文件的文本数据由双引号括起来的打印机字体进行标记，就像”deb-src”。

命令和置于其后的圆括号内的手册页章节数（可选），由打字机字体进行标记，就像 bash(1)。我们鼓励您这样通过输入以下命令来获得信息。

```
$ man 1 bash
```

manpage 会在打字机字体后面括号中显示 manpage 页章节号，如 sources.list(5)。建议你通过键入以下命令来获取帮助信息。

```
$ man 5 sources.list
```

info page 页是由双引号之间的打字机字体来标注，如 `info make`。建议你通过键入以下的命令来获取帮助信息。

```
$ info make
```

文件名将由双引号括起来的打印机字体进行标记，就像”/etc/passwd”。对于配置文件，你可以输入下列的命令来获取它的信息。

```
$ sensible-pager "/etc/passwd"
```

目录名将由双引号括起来的打印机字体进行标记，就像”/etc/apt”。你可以输入下列的命令来浏览目录的内容。

```
$ mc "/etc/apt/"
```

软件包名称将由打印机字体进行标记，就像 vim。你可以输入下列的命令来获取它的信息。

```
$ dpkg -L vim
$ apt-cache show vim
$ aptitude show vim
```

一个文档可能通过文件名来指示它的位置，文件名将由双引号括起来的打印机字体进行标记，就像"/usr/share/doc/base-
和" /usr/share/doc/base-passwd/users-and-groups.html"，或通过它的URL，就像https://www.debian.org。
你可以通过输入下列命令来阅读文档。

```
$ zcat "/usr/share/doc/base-passwd/users-and-groups.txt.gz" | sensible-pager
$ sensible-browser "/usr/share/doc/base-passwd/users-and-groups.html"
$ sensible-browser "https://www.debian.org"
```

环境变量将由双引号括起来的打印机字体进行标记，并带有"$" 前缀，就像"$TERM"。你可以输入下列命令来获取它的当前值。

```
$ echo "$TERM"
```

popcon 流行度

popcon 数据被用来客观地衡量每个包的流行度。它的下载时间为 2019-03-21 15:37:51 UTC，包含了超过 168993 个二进制软件包和 26 个架构的全部 200711 份提交。

注意
请注意 amd64 不稳定（unstable）版的软件仓库中只包含当前 60425 软件包。popcon 数据包含许多旧系统安装报告。

以“V:”开头表示“votes”的 popcon 数值计算方式为“1000 *（当前运行在 PC 上的包的 popcon 提交）/（总的 popcon 提交）”。

以“I:”开头表示“安装数”的 popcon 数值计算方式为“1000 *（当前安装在 PC 上的包的 popcon 提交）/（总的 popcon 提交）”。

注意
流行度评比 popcon 数据不应视为对包的重要性的绝对度量。有许多因素可以影响统计数据。例如，参与流行度评比的某些系统可能有像“/bin”的目录，挂载的时候带“noatime”选项以提升系统性能，这样的系统有效的禁用了“投票（vote）”功能。

软件包大小

软件包的大小数据同样表明了对每个包的客观衡量。它基于“apt-cache show”或“aptitude show”命令（目前在 amd64 架构的不稳定版）报告的“安装大小”。报告的大小单位是 KiB（Kibibyte= 表示 1024 Bytes 的单位）。

注意
包大小是一个小数值的包可能显示了这个在“不稳定”版的包是一个虚拟包，它包含关于依赖关系的重要内容，会安装其他的包。虚拟包使能平稳过度或分割一个包。

注意
包大小后面跟着”(*)” 表明这个软件包在不稳定版本中是缺失的同时使用了实验性版本中的软件包大小来替代。

给本文档报告 Bug

如果你发现本文档有任何问题，请使用 reportbug(1) 向 `debian-reference` 软件包报告 bug。对纯文件版本或源代码的改进建议，请使用”`diff -u`” 包含在 bug 报告里面。

一些对新使用者的提醒

这里给出对新用户的一些提醒信息：

- 备份你的数据
- 妥善保存你的密码和安全信息
- KISS（保持简单而傻瓜式）
  - 不要在系统中过度设计（overengineering）
- 阅读你的日志文件
  - 第一条错误信息才是最重要的
- RTFM（阅读手册与指导）
- 在问问题前，先在互联网上搜索
- 当不是必须要使用 root 的时候，就不要使用 root
- 不要胡乱折腾软件包管理系统
- 不要输入任何你不理解的命令
- （在完全地检查过安全问题之前）不要随意修改文件权限
- 在测试过你所做的修改之前不要关闭 root shell
- 总是准备好备用启动介质（USB 启动盘、启动光盘等）

一些对新使用者的引导

从 Debian 邮件列表来的一些有趣引文，说不定可以帮助新使用者启蒙。

- “这是 Unix。它给你足够的绳索来吊死你自己。” --- Miquel van Smoorenburg `<miquels at cistron.nl>`
- “Unix 是用户友好的……它仅仅选择谁是它的朋友。” --- Tollef Fog Heen `<tollef at add.no>`

维基百科文章”Unix 哲学” 列出了一些有趣的指导。

Chapter 1

GNU/Linux 教程

我认为学习一个计算机系统，就像学习一门新的外语。虽然教程和文档是有帮助的，但你必须自己练习。为了帮助你平滑起步，我详细说明一些基本要点。

Debian GNU/Linux中最强大的设计来自Unix操作系统，一个多用户多任务的操作系统。你必须学会利用这些特性以及 Unix 和 GNU/Linux 的相似性。

别回避面向 Unix 的文档，不要只是依赖于 GNU/Linux 文档，这样做会剥夺你了解许多有用的信息。

注意
如果你在任何类 Unix 系统中使用过一段时间的命令行工具，你可能已经掌握了这份文档中的内容。那请把它当做一个实战检验和回顾。

1.1 控制台基础

1.1.1 shell 提示符

启动系统之后，如果你没有安装 X 窗口系统和显示管理器（例如 `gdm3`），那么你就会看对字符登录界面。假设你的主机名为 `foo`，那么登录提示符将如下所示。

```
foo login:
```

如果你安装了一个 GUI 环境，例如 GNOME 或 KDE，那么你能够用 Ctrl-Alt-F1 进入登录提示符，同时你可以通过 Alt-F7 回到 GUI 环境（更多详情请参阅下文第 1.1.6 节）。

在登录提示符下，你输入你的用户名，例如 `penguin`，然后按回车键，接下来输入你的密码并再次按回车键。

注意
遵循 Unix 传统，Debian 系统下的用户名和密码是大小写敏感的。用户名通常由小写字母组成。第一个用户账号通常在安装期间进行创建。额外的用户账号由 root 用户用 adduser(8) 创建。

系统以保存在"`/etc/motd`" 中的欢迎信息（Message Of The Day）来开始，同时显示一个命令提示符。

```
Debian GNU/Linux jessie/sid foo tty1
foo login: penguin
Password:
Last login: Mon Sep 23 19:36:44 JST 2013 on tty3
Linux snoopy 3.11-1-amd64 #1 SMP Debian 3.11.6-2 (2013-11-01) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
foo:~$
```

现在，你就在 shell 下。shell 解析你的命令。

1.1.2 X 系统下的 shell 提示符

如果你安装了带有显示管理器的 X Window System，例如通过在安装 Debian 时选择“桌面环境”所安装GNOME 的 gdm3，那么你在启动系统后将使用图形登录界面。输入你的用户名和密码可以登录到非特权用户帐号。使用 Tab 键（跳格键）可以在用户名和密码之间移动，也可以使用鼠标左击。

要在 X 窗口下获得 shell 提示符，你必须启动一个 `x` 终端模拟器程序，例如 gnome-terminal(1)、rxvt(1) 或 xterm(1)。在 GNOME 桌面环境下，你可以点击“应用程序” → “附件” → “终端”来打开终端。

你还可以看下下面的第 1.1.6 节章节。

在其它一些桌面系统（如 `fluxbox`）下面，可能没有明显的开始菜单入口。如果是这种情况，试下右击桌面屏幕并希望能有弹出菜单。

1.1.3 root 账户

root 账户也被称作超级用户或特权用户。用这个账户，你能够履行下面的系统管理任务。

- 读、写和删除系统上的任何文件，不顾它们的文件权限
- 设置系统上任何文件的所有者和权限
- 设置系统上任何非特权用户的密码
- 免用户密码登录任何帐户

无限权力的 root 账户，要求你慎重和负责任的使用。

警告
千万不要和其他人共享 root 密码.

注意
一个文件（包括硬件设备，如 CD-ROM 等，这些对 Debian 系统来说都只是一个文件）的权限可能会导致非 root 用户无法使用或访问它。虽然在这种情况下，使用 root 帐户是一个快速的方法，但正确的解决方法应该是对文件权限和用户组的成员进行合适的设置（参见第 1.2.3 节）。

1.1.4 root shell 提示符

这里有一些基本的方法可以让你在输入 root 密码后获得 root 的 shell 提示符。

- 在字符界面的登录提示符，键入 `root` 作为用户名登录。
- 在 GNOME 桌面环境下点击“应用程序” → “附件” → “Root 终端”。
- 在任意用户的 shell 提示符下输入“`su -l`”。
  - 这不会保存当前用户的环境设定。
- 在任意用户的 shell 提示符下输入“`su`”。
  - 这会保存当前用户的一些环境设定。

1.1.5 GUI 系统管理工具

如果你的桌面菜单没有适当的权限启动系统管理工具，你可以在 X 终端模拟器（例如 gnome-terminal(1)、rxvt(1) 或 xterm(1)）中 root 的 shell 提示符下启动它。参见第 1.1.4 节和第 7.8.5 节。

警告
永远不要在显示管理器（例如 gdm3(1)）的提示符下输入 `root` 来使用 root 账户启动 X 显示/会话管理器。

警告
永远不要在显示关键信息的 X Window 下运行不受信任的远程 GUI 程序，因为它可能会监听你的 X 屏幕。

1.1.6 虚拟控制台

在默认的 Debian 系统中，有 6 个可切换的类 VT100字符控制台，可以直接在 Linux 主机上启动 shell。除非你处于 GUI 环境下，否则你可以同时按下左 `Alt` 键和 `F1`—`F6` 之一的键在虚拟控制台间切换。每一个字符控制台都允许独立登陆账户并提供多用户环境。这个多用户环境是伟大的 Unix 的特性，很容易上瘾。

如果你处于 X Window 系统中，你可以通过 `Ctrl-Alt-F1` 键前往字符控制台 1，也就是同时按下左 `Ctrl` 键、左 `Alt` 键和 `F1` 键。你可以按下 `Alt-F7` 回到 X Window System，它一般运行在虚拟控制台 7。

你也可以使用命令行切换到另一个虚拟控制台，例如切换到控制台 1。

```
# chvt 1
```

1.1.7 怎样退出命令行提示符

在命令行输入 `Ctrl-D`，即同时按下左侧`-Ctrl-`键和 `d-`键，即可关闭 shell 活动。如果你正处于字符控制台，你将会返回到登录提示行。尽管这些控制字符“control D” 使用了大写字母，你并不需要按住 Shift-键。`Ctrl-D` 也可以简写为 `^D`。或者，你也可以键入”exit” 退出命令行。

如果你位于 x 终端模拟器 (1) 中，你可以使用这个关闭 `x` 终端模拟器窗口。

1.1.8 怎样关闭系统

就像任何其他的现代操作系统一样，Debian 会通过内存中的缓存数据进行文件操作以提高性能，因此在电源被安全地关闭前需要适当的关机过程，通过将内存中的数据强制写入硬盘来维持文件的完整性。如果软件的电源控制可用，那么关机过程中会自动关闭系统电源。（否则，你可能需要在关机过程之后按电源键几秒钟。）

在普通多用户模式模式下，可以使用命令行关闭系统。

```
# shutdown -h now
```

在单用户模式下，可以使用命令行关闭系统。

```
# poweroff -i -f
```

另外，如果在“/etc/inittab”中含有“ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -h now”，那么你可以按下 Ctrl-Alt-Delete（同时按下左 Ctrl 键、左 Alt 键和 Delete）来关机。参见 inittab(5) 获取更多细节。

参见第 6.9.6 节。

1.1.9 恢复一个正常的控制台

当做了一些滑稽的事（例如“cat< 二进制文件 >”）后，屏幕会发狂，你可以在命令行输入“reset”。你可能无法在屏幕上看到你输入的命令。你也可以输入“clear”来清屏。

1.1.10 建议新手安装的额外软件包

尽管连无需任何桌面环境的 Debian 系统最小安装都提供了基本的 Unix 功能，但对新手而言，使用 apt-get(8) 安装一些基于字符终端的命令行和 curses 软件包（例如 mc 和 vim）依旧是一个不错的主意。

```
# apt-get update
 ...
# apt-get install mc vim sudo
 ...
```

如果你已经安装了这些软件包，那么不会有新的软件包被安装。

软件包	流行度	大小	说明
mc	V:66, I:237	1450	文本模式的全屏文件管理器
sudo	V:517, I:739	3795	给普通用户授予部分 root 权限的程序
vim	V:119, I:395	2799	Unix 文本编辑器 Vi 的改进版，一个程序员的文本编辑器（标准版）
vim-tiny	V:64, I:969	1343	Unix 文本编辑器 Vi 的改进版，一个程序员的文本编辑器（精简版）
emacs25	V:4, I:14	75	GNU 项目的 Emacs，基于 Lisp 的扩展文本编辑器
w3m	V:80, I:433	2323	文本模式的万维网浏览器
gpm	V:11, I:19	497	Unix 风格的文本控制台复制粘贴工具（守护进程）

Table 1.1: 有趣的文本模式程序包列表

您也可以考虑阅读一些其他的信息文档。

你可以用下面的命令安装这些包。

软件包	流行度	大小	说明
doc-debian	I:859	166	Debian 项目文档，(Debian 常见问题) 和其它文档
debian-policy	I:62	4245	Debian 策略手册和相关文档
developers-reference	I:6	1308	Debian 开发者指导方针和信息
maint-guide	I:4	989	Debian 新维护者手册
debian-history	I:1	4059	Debian 项目历史
debian-faq	I:852	1277	Debian 常见问题

Table 1.2: 软件包信息文档列表

```
# apt-get install package_name
```

1.1.11 额外用户账号

如果你不想用你自己的主用户账户来进行下面的练习操作，你可以使用下面的方式创建一个练习用户账户，比如说，创建一个用户名为 fish 的账号。

```
# adduser fish
```

回答所有问题。

这将创建一个名为 fish 的新账号。在你练习完成后，你可以使用下面的命令删除这个用户账号和它的用户主目录。

```
# deluser --remove-home fish
```

1.1.12 sudo 配置

对于典型的单用户工作站，例如运行在笔记本电脑上的桌面 Debian 系统，通常简单地配置 sudo(8) 来使为非特权用户 (例如用户 penguin) 只需输入用户密码而非 root 密码就能获得管理员权限。

```
# echo "penguin  ALL=(ALL) ALL" >> /etc/sudoers
```

另外，可以使用下列命令使非特权用户 (例如用户 penguin) 无需密码就获得管理员权限。

```
# echo "penguin  ALL=(ALL) NOPASSWD:ALL" >> /etc/sudoers
```

这些技巧只对你管理的单用户工作站中那个唯一的用户有用。

警告
在多用户工作站中不要建立这样的普通用户账户，因为它会导致非常严重的系统安全问题。

小心
在上述例子中，用户 penguin 的密码及账号要有和 root 账号密码同样多的保护。

小心
在这种情况下，管理员权限被赋予那些有权对工作站进行系统管理任务的人。永远不要让你的公司行政管理部门或你的老板进行管理（例如给予他们权限），除非他们获得了授权并有这样的能力。

注意
为了对受限的设备和文件提供访问权限，你应该考虑使用组来提供受限访问，而不是通过 sudo(8) 来使用 root 权限。

注意
随着越来越细致周密的配置，sudo(8) 可以授予一个共享系统上的其它用户有限的管理权限而不共享 root 密码。这可以帮助对有多个管理员的主机进行责任追究，你可以了解到是谁做什么。另一方面，你可能不想任何人有这样的权限。

1.1.13 动手时间

现在你已经准备好在 Debian 系统上开工了，只要你使用非特权用户账号就不会有风险。

这是因为 Debian 系统（即使是默认安装）会设置适当的文件权限来防止非特权用户对系统造成破坏。当然，可能仍然有一些漏洞可以利用，但关心这些问题的人不应该阅读这一节，而应该去阅读 Debian 安全手册。

我们使用下面的方式，把 Debian 系统当作一个 类 Unix 系统来学习。

- 第 1.2 节 (基本概念)
- 第 1.3 节 (生存方式)
- 第 1.4 节 (基本方式)
- 第 1.5 节 (shell 机制)
- 第 1.6 节 (文本处理方式)

1.2 类 Unix 文件系统

在 GNU/Linux 和其他类 Unix操作系统中，文件被组织到目录中。所有的文件和目录排放在以“/”为根的巨大的树里。叫它树是因为如果你画出文件系统，它看起来就像一棵树，但是它是颠倒过来的。

这些文件和目录可以分散在多个设备中。mount(8) 用于把某个设备上找到的文件系统附着到巨大的文件树上。相反的，umount(8) 把它再次分离。在最近的 Linux 内核里，mount(8) 带某些参数时可以把文件树的一部分绑定到另外的地方，或者可以把文件系统挂载为共享的、私有的、从设备、或不可绑定的。对每个文件系统支持的挂载选项可以在`/usr/share/doc/linux-doc-*/Documentation/filesystems/`找到。

Unix 系统上叫做目录，某些其他系统上叫做文件夹。请同样留意，在任何 Unix 系统上，没有的驱动器的概念，例如“`A:`”。这只有一个文件系统，并且所有东西都包含在内。这相对于 Windows 来说是一个巨大的优点。

1.2.1 Unix 文件基础

下面是一些 Unix 文件基础。

- 文件名是 区分大小写的。也就是说，”`MYFILE`” 和”`MyFile`” 是不同的文件。

- 根目录意味着文件系统的根，简单的称为“/”，不要把它跟 root 用户的家目录“/root”混淆了。
- 每个目录都有一个名字，它可以包含任意字母或除了/”以外的符号。根目录是个特例。它的名字是“/”(称作“斜线”或“根目录”)，并且它不能被重命名。
- 每个文件或目录都被指定一个全限定文件名，绝对文件名，或路径，按顺序给出必须经过的目录从而到达相应目录。这三个术语是同义的。
- 所有的全限定文件名以“/”目录开始，并且在每个目录或文件名之间有一个“/”。第一个“/”是最顶层目录，其他的“/”用于分隔跟着的子目录。直到到达最后的入口，即实际文件的名称。这些话可能会令人困惑。用下面这个全限定文件名作为例子:“/usr/share/keytables/us.map.gz”。不过，人们也把它的基名“us.map.gz”单独作为文件名。
- 根目录有很多分支，例如“/etc/”和“/usr/”。这些子目录依次分出更多的子目录，例如“/etc/init.d/”和“/usr/local/”。这整体叫做“目录树”。你可以把一个绝对文件名想象成从“/”这棵树的基到某个分支 (一个文件) 的结尾的一条路径。你也听到人们谈论目录树，就好像它是一个包含所有直系后代的“家庭”树的一个图，这个图叫做根目录 (“/”): 因此子目录有父目录，并且一条路径显示了一个文件完整的祖先。也有相对路径从其他地方开始，而不是从根目录。你应该还记得目录“../”指向父目录。这个术语也适用于其他类似目录的结构，如分层数据结构。
- 对于一个物理设备, 是没有一个特定的目录路径名来对应的组成部分. 这不同于RT-11, CP/M,OpenVMS,MS-DOS,AmigaOS, 以及微软的 Windows，这些系统存在一个路径包含了一个设备名字，比如”C:\”。(尽管如此, 路径条目确实存在引用了物理设备作为正常的文件系统的一部分. 参考第 1.2.2 节。)

注意
虽然你可以在文件名中使用任意的字幕或者符号, 但是在实际情况下这样做是一个坏主意. 最好避免使用一些在命令行里面含有特殊意义的字符, 比如空格, 制表符, 换行符, 和其它的特殊字符: { } ( ) [ ] ’ ` ” \ / >< | ; ! #&^ * % @ $. 如果你想有一个区分度良好的命名, 比较好的选择是利用时期, 连字符和下划线. 你也可以每个单词的首字母大写, 这叫大驼峰命名法, 比如这样”LikeThis”. 经验丰富的 Linux 用户会趋向于在文件名中不使用空格.

注意
这个”root” 可能既表示” 超级用户 root” 又表示” 根目录”(/root) . 应该根据上下文确定它的用法.

注意
单词 **path** 不仅表示包含全限定文件名, 也可能表示命令搜索的路径. 通常路径真实的意思是需要通过上下文来明确.

关于文件层次的最佳详细实践在文件系统层次标准 (”/usr/share/doc/debian-policy/fhs/fhs-2.3.txt.gz” 和 hier (7)). 你应该记住以下的一些标准作为开始学习的步骤.

目录	目录用途
/	根目录
/etc/	系统范围的配置文件
/var/log/	系统日志文件
/home/	所有非特权用户的用户目录

Table 1.3: 重要目录的用途列表

1.2.2 文件系统深入解析

按照 **UNIX** 系统的传统，Debian GNU / Linux 的文件系统是在物理数据存储设备诸如磁盘或其他存储设备上，与硬件设备的交互，如控制台和远程串口终端都是以统一的方式呈现在“/ dev /”下面。

每个文件、目录、命名管道（一种两个程序间共享数据的方法）或 Debian GNU/Linux 系统上的物理设备都有一个叫做 inode的数据结构，描述了其相关属性，例如拥有它的用户（所有者），它属于的组，最后一次访问时间，等等。把所有东西都表示在文件系统中的想法是来源于 Unix，现代的 Linux 内核则将这个思路进行了扩展。现在，甚至有关计算机上正在运行的进程的信息都可以在文件系统中找到。

这个对物理实体和内部进程的统一和抽象是非常强大的，因为这允许我们用同样的命令对许多完全不同的设备进行同样的操作。甚至可以通过向链接到运行进程的特殊文件写入数据来改变内核的运行方式。

提示
如果您需要识别文件树和物理实体之间的对应关系，请尝试不带参数运行 mount(8)。

1.2.3 文件系统权限

类 Unix系统的文件系统权限被定义给三类受影响的用户。

- 拥有这个文件的用户（**u**）
- 这个文件所属组的其他用户（**g**）
- 所有其余的用户（**o**），同样称为“世界”和“所有人”

对文件来说，每个对应权限允许下列动作。

- 可读（**r**）权限允许所有者检查文件的内容。
- 可写（**w**）权限允许所有者修改文件内容。
- 可执行（**x**）权限允许所有者把文件当做一个命令运行。

对于目录来说，每个对应权限允许下列动作。

- 可读（**r**）权限允许所有者列出目录内的内容。
- 可写（**w**）权限允许所有者添加或删除目录里面的文件。
- 可执行（**x**）权限允许所有者访问目录里的文件。

在这里，一个目录的可执行权限意味着不仅允许读目录里的文件，还允许显示他们的属性，例如大小和修改时间。

ls(1) 用于显示文件和目录的权限信息（更多）。当运行时带有“-l”选项，它将按给定顺序显示下列信息。

- 文件类型（第一个字母）
- 文件的访问权限（9 个字符，三个字符组成一组按照用户、组、其他的顺序表示）
- 链接到文件的硬链接数
- 文件所有者的用户名
- 这个文件所属的组名
- 以字符（字节）为单位的文件大小
- 文件的日期和时间（mtime）

字符	说明
-	普通文件
d	目录
l	符号链接
c	字符设备节点
b	块设备节点
p	命名管道
s	套接字

Table 1.4: “ls -l”输出的第一个字符列表

- 文件的名字

chown(1) 用于 root 账户修改文件的所有者。chgrp(1) 用于文件的所有者或 root 账户修改文件所属的组。chmod(1) 用于文件的所有者或 root 账户修改文件和文件夹的访问权限。操作一个 foo 文件的基本语法如下。

```
# chown <newowner> foo
# chgrp <newgroup> foo
# chmod  [ugoa][+-=][rwxXst][,...] foo
```

例如，你可以按照下面使一个目录树被用户 foo 所有，并共享给组 bar。

```
# cd /some/location/
# chown -R foo:bar .
# chmod -R ug+rwX,o=rX .
```

有三个更加特殊的权限位。

- **Set-User-ID(SUID)** 位（**s** 或 **S** 替换用户的 **x**）
- **Set-Group-ID(SGID)** 位（**s** 或 **S** 替换组的 **x**）
- 粘滞位（**t** 或 **T** 替代其他用户的 **x**）

如果“ls -l”对这些位的输出是大写字母，则表示这些输出下面的执行位未设置。

给一个可执行文件设置 **Set-User-ID** 位将允许一个用户以他自己的 ID 运行这个可执行文件（例如 **root** 用户）。类似的，给一个可执行文件设置了 **Set-Group-ID** 位将允许一个用户以文件所属组的 ID 运行该文件。（例如 **root** 组）。由于这些设置可能导致安全风险，设置它们为可用的时候需要格外留意。

在一个目录上设置“**Set-Group-ID**”将打开类 BSD 的文件创建计划，所有在目录里面创建的文件将属于目录所属的组。

给一个目录设置“粘滞位”将保护该目录内的文件不被其所有者之外的一个用户删除。为了保护一个在像“/tmp”这样所有人可写或同组可写的目录下文件内容的安全，不仅要去除可写权限，还要给其所在目录设置粘滞位。否则，该文件可以被任意对其所在目录有写权限的用户删除并创建一个同名的新文件。

这里有一点有趣的文件权限例子。

```
$ ls -l /etc/passwd /etc/shadow /dev/ppp /usr/sbin/exim4
crw------T 1 root root   108, 0 Oct 16 20:57 /dev/ppp
-rw-r--r-- 1 root root     2761 Aug 30 10:38 /etc/passwd
-rw-r----- 1 root shadow   1695 Aug 30 10:38 /etc/shadow
-rwsr-xr-x 1 root root   973824 Sep 23 20:04 /usr/sbin/exim4
$ ls -ld /tmp /var/tmp /usr/local /var/mail /usr/src
drwxrwxrwt 14 root root  20480 Oct 16 21:25 /tmp
drwxrwsr-x 10 root staff  4096 Sep 29 22:50 /usr/local
```

```
drwxr-xr-x 10 root root   4096 Oct 11 00:28 /usr/src
drwxrwsr-x  2 root mail   4096 Oct 15 21:40 /var/mail
drwxrwxrwt  3 root root   4096 Oct 16 21:20 /var/tmp
```

chmod(1) 有另一种数值模式来描述文件权限。这种数字模式使用 3 到 4 位八进制（底为 8）数。

数字	说明
第一个可选数字	**Set-User-ID** (=4), **Set-Group-ID** (=2) 和 粘滞位 (=1) 之和
第二个数字	用户的可读 (=4), 可写 (=2) 和 可执行 (=1) 权限之和
第三个数字	组权限同上
第四个数字位	其他用户权限同上

Table 1.5: chmod(1) 命令文件权限的数字模式

这听起来很复杂实际上相当简单。如果你把“`ls -l`”命令输出的前几列（2-10），看成以二进制（底为 2）表示文件的权限（“-”看成 0，“rwx”看成 1），你应该可以理解用数字模式值的最后 3 位数字对文件权限的八进制表示。

尝试下列例子

```
$ touch foo bar
$ chmod u=rw,go=r foo
$ chmod 644 bar
$ ls -l foo bar
-rw-r--r-- 1 penguin penguin 0 Oct 16 21:39 bar
-rw-r--r-- 1 penguin penguin 0 Oct 16 21:35 foo
```

提示
如果你需要在 shell 脚本中访问“`ls -l`”显示的信息，你需要使用相关命令，如 test(1)，stat(1) 和 readlink(1)。shell 内置命令，如“`[`”或“`test`”，可能也会用到。

1.2.4 控制新建文件的权限：umask

什么权限将应用到新建文件受 shell 内置命令 `umask` 的限制。参见 dash(1)，bash(1)，和内建命令 (7)。

```
(文件权限) = (请求的文件权限) & ~(umask 值)
```

umask 值	创建的文件权限	创建的目录权限	用法
`0022`	`-rw-r--r--`	`-rwxr-xr-x`	仅所属用户可写
`0002`	`-rw-rw-r--`	`-rwxrwxr-x`	仅所属组可写

Table 1.6: **umask** 值举例

Debian 默认使用用户私人组（UPG）。每当一个新用户添加到系统的时候都会创建一个 UPG。UPG 的名字和创建它的用户相同，这个用户是这个 UPG 的唯一成员。自从每个用户都有自己的私人组之后，把 umask 设置成 `0002` 变得更安全了。（在某些 Unix 变体中，把所有普通用户设置到一个叫 **`users`** 的组是非常常见的做法，在这种情况下，出于安全考虑把 umask 设为 `0022` 是一个好主意）

提示
通过把“`umask 002`”写入 `~/.bashrc` 文件打开 UPG。

1.2.5 一组用户的权限（组）

为了使组权限应用到一个特定用户，这个用户需要通过使用“sudo vigr”编辑 /etc/group 以及使用“sudo vigr -s”编辑 /etc/gshadow 成为该组的成员。你需要注销之后重新登录（或运行“exec newgrp”）以启用新的组配置。

注意
或者，你可以通过添加一行“auth optional pam_group.so”到“/etc/pam.d/common-auth”以及配置“/etc/security/group.conf”，使得在身份验证过程动态添加用户到组。（参见第 4 章。）

在 Debian 系统中，硬件设备是另一种文件。如果你从一个用户账户访问某些设备出现问题，例如 CD-ROM 和 USB 记忆棒，你需要使这个用户成为相关组的成员。

一些著名的由系统提供的组允许其成员不需要 root 权限访问某些特定的文件和设备。

组	可访问文件和设备的描述
dialout	完全及直接的访问串口端口（“/dev/ttyS[0-3]”）
dip	有限的访问串口，创建到信任点的拨号 **IP** 连接
cdrom	CD-ROM, DVD+/-RW 驱动器
audio	音频设备
video	视频设备
scanner	扫描仪
adm	系统监控日志
staff	一些用于初级管理工作的目录：“/usr/local”，“/home”

Table 1.7: 关于文件访问的由系统提供的著名组列表

提示
你需要属于 dialout 组才能重配置调制解调器、拨号到任意地方，等等。但如果 root 用户在“/etc/ppp/peers/”为受信任点创建了预定义配置文件的话，你只需要属于 dip 组，就可以创建拨号 **IP** 来连接到那些受信任的点上，需使用的命令行工具包括 pppd(8)、pon(1) 以及 poff(1)。

某些著名的由系统提供的组允许它们的成员不带 root 权限运行特定的命令。

组	可访问命令
sudo	不带它们的密码运行 sudo
lpadmin	执行命令以从打印机数据库添加、修改、移除打印机

Table 1.8: 著名的由系统提供用于特定命令运行的组列表

由系统提供的用户和组的完整列表，参见由 base-passwd 包提供的“/usr/share/doc/base-passwd/users-and-gro
中，当前版本的“用户和组”。

用户和组系统的管理命令，参见 passwd(5)，group(5)，shadow(5)，newgrp(1)，vipw(8)，vigr(8)，以及 pam_group(8)。

1.2.6 时间戳

GNU/Linux 文件有三种类型的时间戳。

注意
ctime 不是文件创建时间。

类型	含义（历史上 **Unix** 的定义）
mtime	文件修改时间 (`ls -l`)
ctime	文件状态修改时间 (`ls -lc`)
atime	文件最后被访问的时间 (`ls -lu`)

Table 1.9: 时间戳类型列表

注意
atime 在 GNU/Linux 系统上的真实值可能和历史上 Unix 的定义有所不同。

- 覆盖一个文件，将会改变该文件所有的 **mtime**, **ctime**, 和 **atime** 属性。
- 改变文件的所有者或者权限，将改变文件的 **ctime** 和 **atime** 属性。
- 在历史上的 Unix 系统中，读取一个文件将改变文件的 **atime** 属性。
- 读一个文件，将改变文件的 **atime** 属性；在 GNU/Linux 系统上，这仅发生在其文件系统使用“`strictatime`”参数挂载的情况下。
- 如果 GNU/Linux 系统的文件系统使用”`relatime`” 选项挂载，第一次读文件，或者随后读文件，将改变该文件的 **atime** 属性. (从 Linux 2.6.30 开始的默认行为)
- 如果 GNU/Linux 系统的文件系统使用”`noatime`” 挂载，则读一个文件，不会改变这个文件的 **atime** 属性。

注意
为了在正常的使用场景中能够提升文件系统的读取效率，新增了”`noatime`” 和”`relatime`” 这两个加载选项。如使用了”`strictatime`” 选项，即使简单的文件读操作都伴随着更新 **atime** 属性这个耗时的写操作。但是 **atime** 属性除了 mbox(5) 文件以外却很少用到。详情请看 mount(8)。

使用 touch(1) 命令修改已存在文件的时间戳。

对于时间戳，`ls` 命令输出字段在非英语区域（“`fr_FR.UTF-8`”）与旧的区域（“`C`”）不同。

```
$ LANG=fr_FR.UTF-8  ls -l foo
-rw-rw-r-- 1 penguin penguin 0 oct. 16 21:35 foo
$ LANG=C  ls -l foo
-rw-rw-r-- 1 penguin penguin 0 Oct 16 21:35 foo
```

提示
参考第 9.2.5 节自定义“`ls -l`”输出。

1.2.7 链接

有两种方法把一个文件“`foo`”链接到一个不同的文件名“`bar`”。

- 硬链接
  - 对现有文件重复名称
  - “`ln foo bar`”
- 符号链接或 symlink

– 通过名字指向另一个文件的特殊文件
–“ln -s foo bar”

请参阅下面的示例，rm 命令结果中链接数的变化和细微的差别。

```
$ umask 002
$ echo ”Original Content” > 1 foo
$ ls -li foo
1449840 -rw-rw-r-- 1 penguin penguin 17 Oct 16 21:42 foo
$ ln foo bar     # 硬链接
$ ln -s foo baz  # 符号链接
$ ls -li foo bar baz
1449840 -rw-rw-r-- 2 penguin penguin 17 Oct 16 21:42 bar
1450180 lrwxrwxrwx 1 penguin penguin  3 Oct 16 21:47 baz -> 2 foo
1449840 -rw-rw-r-- 2 penguin penguin 17 Oct 16 21:42 foo
$ rm foo
$ echo ”New Content” > 3 foo
$ ls -li foo bar baz
1449840 -rw-rw-r-- 1 penguin penguin 17 Oct 16 21:42 bar
1450180 lrwxrwxrwx 1 penguin penguin  3 Oct 16 21:47 baz -> 4 foo
1450183 -rw-rw-r-- 1 penguin penguin 12 Oct 16 21:48 foo
$ cat bar
Original Content
$ cat baz
New Content
```

硬链接可以在同一个文件系统内创建，并共用同一个 inode 号，由 ls(1) 带“-i”选项显示的。

符号链接总是名义上具有“rwxrwxrwx”的文件访问权限，如上面例子所示，实际的有效访问权限由它所指向的文件确定。

小心
除非你有非常好的理由，否则不要创建一个复杂的符号链接或硬链接通常是个好主意。符号链接的逻辑组合可能导致文件系统噩梦般的无限循环。

注意
通常使用符号链接比使用硬链接更合适，除非你有一个好理由使用硬链接。

“.”目录链接到它所在的目录，因此任何新建目录的链接数从 2 开始。“..”目录链接到父目录，因此目录的链接数随着新的子目录的创建而增加。

如果你刚从 Windows 迁移到 Linux，你很快将清楚 Unix 的文件名链接相较于 Windows 最相近的“快捷方式”是多么精心设计的。由于它是在文件系统中实现的，应用无法看到链接文件跟原始文件之间的区别。在硬链接这种情况，这真的是毫无差别。

1.2.8 命名管道（先进先出）

命名管道是一个像管道一样的文件。你把内容放进了文件，它从另一端出来。因此，它被称为 FIFO，即先进先出：你从管道这端先放进去的东西会从另一端先出来。

如果对一个命名管道进行写入操作，写入的过程不会被终止，直到写入的信息从管道中被读取出来。读取过程将会持续到没有信息可以读取为止。管道的大小始终是零，它不存储数据，它只是连接两个过程，像 shell 提供的”1| 2” 语法功能一样。然而，一旦管道有了名称，这两个进程就可以不必在同一个命令行，甚至由同一个用户运行。管道是 UNIX 的一个非常有影响力的创新。

尝试下列例子

```
$ cd; mkfifo mypipe
$ echo "hello" >mypipe & # 放到后台运行
[1] 8022
$ ls -l mypipe
prw-rw-r-- 1 penguin penguin 0 Oct 16 21:49 mypipe
$ cat mypipe
hello
[1]+  Done                    echo "hello" >mypipe
$ ls mypipe
mypipe
$ rm mypipe
```

1.2.9 套接字

套接字被广泛应用于所有的互联网通信，数据库和操作系统本身。它类似于命名管道（FIFO）并且允许进程之间甚至不同计算机之间进行信息交换。对于套接字，这些进程不需要在同一时间运行，也不需要是同一个父进程的子进程。它是进程间通信（IPC）的一个节点。信息的交换可能会通过网络发生在不同主机之间。最常见的两种是 互联网套接字 和 UNIX 域套接字 。

提示
通过"`netstat -an`"命令可以很方便的查看系统已经打开了那些套接字。

1.2.10 设备文件

设备文件包括系统的物理设备和虚拟设备，如硬盘、显卡、显示屏、键盘。虚拟设备的一个例子是控制台，用“`/dev/console`”来描述。

设备文件有两种类型。

- 字符设备
  - 每次访问一个字符
  - 一个字符等于一个字节
  - 如键盘、串口…
- 块设备
  - 通过更大的单元–块，进行访问
  - 一个块 > 一个字节
  - 如硬盘等…

你可以读写块设备文件，尽管该文件可能包含二进制数据，读取后显示出无法理解的乱码。向文件写入数据，有时可以帮助定位硬件连接故障。比如，你可以将文本文件导入打印机设备“`/dev/lp0`”，或者将调制解调命令发送到合适的串口“`/dev/ttyS0`”。但是，除非这些操作都小心完成，否则可能会导致一场大灾难。所以要特别小心。

注意
常规访问打印机，使用 lp(1)。

设备的节点数可以通过执行 ls(1) 得到，如下所示。

```
$ ls -l /dev/sda /dev/sr0 /dev/ttyS0 /dev/zero
brw-rw---T  1 root disk     8,  0 Oct 16 20:57 /dev/sda
brw-rw---T+ 1 root cdrom   11,  0 Oct 16 21:53 /dev/sr0
crw-rw---T  1 root dialout  4, 64 Oct 16 20:57 /dev/ttyS0
crw-rw-rw-  1 root root     1,  5 Oct 16 20:57 /dev/zero
```

- ”/dev/sda” 的主设备号是 8，次设备号是 0。它可以被 disk 群组的用户读写。
- ”/dev/sr0” 的主设备号是 11，次设备号是 0。它可以被 cdrom 群组的用户读写。
- ”/dev/ttyS0” 的主设备号是 4，次设备号是 64。它可以被 dailout 群组的用户读写。
- ”/dev/zero” 的主设备号是 1，次设备号是 5。它可以被任意用户读写。

在现代 Linux 系统中，处在”/dev” 之下的文件系统会自动被 udev() 机制填充。

1.2.11 特殊设备文件

还有一些特殊的设备文件。

设备文件	操作	响应描述
/dev/null	读取	返回“文件结尾字符（EOF）“
/dev/null	写入	无返回（一个无底的数据转存深渊）
/dev/zero	读取	返回”\0 空字符”（与 ASCII 中的数字 0 不同）
/dev/random	读取	从真随机数产生器返回一个随机字符，供应真熵（缓慢）
/dev/urandom	读取	从能够安全加密的伪随机数产生器返回一个随机字符
/dev/full	写入	返回磁盘已满（ENOSPC）错误

Table 1.10: 特殊设备文件列表

这些特别设备文件经常和 shell 数据重定向联合使用（参考第 1.5.8 节）。

1.2.12 procfs 和 sysfs

procfs和sysfs两个伪文件系统，分别加载于”/proc” 和”/sys” 之上，将内核中的数据结构暴露给用户空间。或者说，这些条目是虚拟的，他们打开了深入了解操作系统运行的方便之门。

目录”/proc” 为每个正在运行的进程提供了一个子目录，目录的名字就是进程标识符（PID）。需要读取进程信息的系统工具，如 ps()，可以从这个目录结构获得信息。

”/proc/sys” 之下的目录，包含了可以更改某些内核运行参数的接口。（你也可以使用专门的 sysctl() 命令修改，或者使用其预加载/配置文件”/etc/sysctl.conf”。）

当人们看到这个特别大的文件”/proc/kcore” 时，常常会惊慌失措。这个文件于你的的电脑内存大小相差不多。它被用来调试内核。它是一个虚拟文件，指向系统内存，所以不必担心它的大小。

”/sys” 以下的目录包含了内核输出的数据结构，它们的属性，以及它们之间的链接。它同时也包含了改变某些内核运行时参数的接口。

参考”proc.txt(.gz)”,”sysfs.txt(.gz)”,以及其他相关的 Linux 内核文档（”/usr/share/doc/linux-doc-*/Doc
这些文件由 linux-doc-* 软件包提供。

1.2.13 tmpfs

tmpfs是一个临时文件系统，它的文件都保存在虚拟内存中。必要时，位于内存页缓存的 tmpfs 数据可能被交换到硬盘中的交换分区。

系统启动早期阶段，”/run” 目录挂载为 tmpfs。这样即使”/” 挂载为只读，它也是可以被写入的。它为过渡态文件提供了新的存储空间，同时也替代了Filesystem Hierarchy Standar2.3 版中说明的目录位置：

- ”/var/run” → ”/run”
- ”/var/lock” → ”/run/lock”
- ”/dev/shm” → ”/run/shm”

参考”tmpfs.txt(.gz)”, 文件位于 Linux 内核文档 (”/usr/share/doc/linux-doc-*/Documentation/filesystem 目录之下，由软件包 linux-doc-* 提供。

1.3 Midnight Commander (MC)

Midnight Commander (MC) 是一个 Linux 终端或其它终端环境下的 GNU 版” 瑞士军刀” 。它为新手们提供了一个菜单式样的终端使用体验，这更易于学习运用标准的 Unix 命令。

你可能需要按照下面的命令来安装标题为” mc ” 的 Midnight Commander 包.

```
$ sudo apt-get install mc
```

使用 mc(1) 命令那个来浏览 Debian 系统。这是最好的学习方式。请使用光标键和回车键来翻看一些感兴趣的内容。

- ”/etc” 及其子目录
- ” /var/log ” 及其子目录
- ” /usr/share/doc ” 及其子目录
- ” /sbin ” 和” /bin ”

1.3.1 自定义 MC

为了在退出 MC 的时候更改目录并 cd 到其它目录，我建议修改”~/.bashrc” 包含一个由 mc 包提供的脚本。

```
. /usr/lib/mc/mc.sh
```

查看 mc(1) (在”-P” 选项里) 的原因。(如果你不能理解我这里说所讲的，你可以稍后回头再看)

1.3.2 启动 MC

MC 可以这样启动起来。

```
$ mc
```

MC 通过菜单覆盖了所有的文件操作，因此而让用户更省心省力。只需要按 F1 就可以跳转到帮助界面。你只需要按光标键和功能键就可以使用 MC。

注意
某些终端比如 gnome-terminal(1)，功能键的按键触发消息可能会被终端程序截取。在 `gnome-terminal` 里可以通过”Edit” → ”Keyboard Shortcuts” 选项设置来禁止这类消息。

如果你遇到字符编码问题，显示出来都是乱码，通过添加”`-a`” 到 MC 命令行或许有助于避免此类问题。

如果这样不能解决 MC 中的显示问题，可以参考第 9.4.6 节.

1.3.3 MC 文件管理

默认的两个目录面板里包含了文件列表。另一个有用的模式是设置右边窗口为” 信息” 来读取文件访问权限信息。接下来是一些必要的快捷键。守护进程 gpm(8) 运行的时候，你也可以在字符命令行里用鼠标来操作。(在 MC 里进行复制和粘贴操作的时候一定要按住 shift 键。)

快捷键	键绑定功能
`F1`	帮助菜单
`F3`	内部文件查看器
`F4`	内部编辑器
`F9`	激活下拉菜单
`F10`	退出 Midnight Commander
`Tab`	在两个窗口间移动
`Insert` 或 `Ctrl-T`	用于多文件操作的标记文件，如副本
`Del`	删除文件 (注意---设置 MC 为安全删除模式)
光标键	自我解释

Table 1.11: MC 快捷键绑定

1.3.4 MC 命令行技巧

- `cd` 命令在选中的屏幕中改变目录。
- `Ctrl-Enter` or `Alt-Enter` 拷贝文件名到命令行。使用 cp(1) 和 mv(1) 两个命令来进行处理。
- `Alt-Tab` 显示文件名自动补全提示。
- 通过添加 MC 命令参数可以指定开始目录；例如，”`mc /etc /root`”。
- `Esc` + `n-key` → `Fn` (i.e., `Esc` + `1` → `F1`, etc.; `Esc` + `0` → `F10`)
- 先按 `Esc` 键和同时按 `Alt` 是一样；例如, 输入 `Esc` + `c` 和同时 `Alt-C` 是一样的。`Esc` 被称为 meta 键，有时候也称之为”`M-`”。

1.3.5 MC 内部编辑器

这个内置编辑器有一个有意思的粘贴方案。摁 `F3` 开始选择起始点，再摁 `F3` 选择终点并高亮选择区。此刻你可以移动你的光标，使用 F6 将选区移动到当前光标下，F5 则将选区复制到当前光标下。 `F2` 保存文件。 `F10` 退出。多数光标键以直观的方式工作。

MC 编辑器可以直接以下面的命令方式启动。

```
$ mc -e filename_to_edit
```

```
$ mcedit filename_to_edit
```

这不是一个多窗口编辑器，但是能通过复用终端来达到同样的效果。在两个窗口间复制，需要用到 Alt-F < n > 来切换虚拟终端并使用”File→Insert file” 或者”File→Copy to file” 来移动文本。

内部编辑器可以被外部编辑器替代。

同样，许多程序使用环境变量 `$EDITOR` 或 `$VISUAL` 来决定编辑器的使用。如果你准备使用 vim(1) 或者 nano(1) 来开始，你或许需要将下面的代码加入”`~/.bashrc`” 来对 `mcedit` 进行设置。

```
export EDITOR=mcedit
export VISUAL=mcedit
```

如果可能的话我推荐用”`vim`”。

如果你使用 vim(1) 并不顺手，你可以在大部分系统中继续使用 mcedit(1) 来进行工作。

1.3.6 MC 内部查看器

MC 是一个非常智能的查看器。这是一个在文档中搜索文本的好工具。我经常使用它在`/usr/share/doc` 目录中查找文件。这是浏览大量 Linux 信息的最快方式。这个查看器可以通过下列命令中的任何一个来直接启动。

```
$ mc -v path/to/filename_to_view
```

```
$ mcview path/to/filename_to_view
```

1.3.7 自动启动 MC

在文件中输入回车，用适当的程序来处理文件的内容 (查看第 9.3.11 节)。这是 MC 一个非常方便的用法。

文件类型	对回车键的响应
可执行文件	执行命令
帮助文档	管道内容查看器软件
html 文件	管道内容网页浏览器
”`*.tar.gz`” 和”`*.deb`” 文件	浏览其内容就像查看子目录一样

Table 1.12: MC 中对回车键的响应

为让这些查看器和虚拟文件特征生效，可查看的文件不能够被设置为可执行。使用 chmod(1) 或通过 MC 文件菜单改变他们的状态。

1.3.8 MC 中的 FTP 虚拟文件系统

MC 能够使用 FTP 跨因特网访问文件。在菜单按 F9, 然后输入”p” 来激活 FTP 虚拟文件系统。按”`username:passwd@hostnam`
的形式输入 URL, 就会像本地目录一样来检索远程目录。

试着打开”[deb.debian.org/debian]” 来浏览 Debian 的文件组织结构。

1.4 类 Unix 工作环境基础

虽然 MC 差不多可以让你做任何事情，但学会从 shell 提示下使用命令行工具也是非常重要的，可以让你变得熟悉类 Unix 工作环境。

1.4.1 登录 shell

你可以通过 chsh(1) 选择你的登录 shell。

软件包	流行度	大小	**POSIX shell**	说明
bash	V:826, I:999	6462	Yes	Bash: GNU Bourne Again SHell (事实上的标准)
tcsh	V:10, I:34	1311	No	TENEX C Shell: 一个 Berkeley csh 的增强版本
dash	V:929, I:988	212	Yes	Debian Almquist Shell, 擅长 shell 脚本
zsh	V:38, I:70	2401	Yes	Z shell: 有许多增强的标准 shell
mksh	V:6, I:12	1383	Yes	Korn shell 的一个版本
csh	V:2, I:11	343	No	OpenBSD C Shell, Berkeley csh 的一个版本
sash	V:0, I:5	1054	Yes	有内置命令的 Stand-alone shell (并不意味着标准的"/bin/sh")
ksh	V:4, I:20	3294	Yes	Korn shell的真正的 AT&T 版本
rc	V:0, I:3	154	No	AT&T Plan 9 rc shell 的一个实现
posh	V:0, I:0	190	Yes	Policy-compliant Ordinary SHell 策略兼容的普通 shell(pdksh 派生)

Table 1.13: shell 程序列表

> 提示
> 虽然类 POSIX 共享基本语法，但他们在 shell 变量和全局扩展等基本事情上，行为可以不同。细节请查阅他们的文档。

在本教程中，交互式的 shell 总是指 bash.

1.4.2 定制 bash

你可以通过"~/.bashrc"来定制 bash(1) 的行为。

尝试下列例子。

```
# enable bash-completion
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

# CD upon exiting MC
. /usr/lib/mc/mc.sh

# set CDPATH to a good one
CDPATH=.:/usr/share/doc:~:~/Desktop:~
```

```
export CDPATH

PATH="${PATH+$PATH:}/usr/sbin:/sbin"
# set PATH so it includes user's private bin if it exists
if [ -d ~/bin ] ; then
  PATH="~/bin${PATH+:$PATH}"
fi
export PATH

EDITOR=vim
export EDITOR
```

提示
你可以在第 9 章中的第 9.2.7 节找到更多关于 bash 的定制技巧。

提示
bash-completion 软件包能够让 bash 进行命令补全.

1.4.3 特殊按键

在类 Unix环境，有一些具有特殊含义的按键。请注意，普通的 Linux 字符控制台，只有左手边的 Ctrl 和 Alt 键可以正常工作。其中有几个值得记住的按键。

快捷键	描述
Ctrl-U	删除光标前到行首的字符
Ctrl-H	删除光标前的一个字符
Ctrl-D	终止输入（如果你在使用 shell，则退出 shell）
Ctrl-C	终止一个正在运行的程序
Ctrl-Z	通过将程序移动到后台来暂停程序
Ctrl-S	停止屏幕输出
Ctrl-Q	激活屏幕输出
Ctrl-Alt-Del	重启/关闭系统，参见 inittab(5)
Left-Alt-key（或 Windows-key）	Emacs 和相似 UI 的元键（meta-key）
Up-arrow	开始在 bash 中的命令历史搜索
Ctrl-R	开始在 bash 中的增量命令历史搜索
Tab	在 bash 命令行中补全文件名
Ctrl-V Tab	在 bash 命令行中输出 Tab 而不是进行补全

Table 1.14: bash 的按键绑定列表

提示
Ctrl-S 的终端功能可能被 stty(1) 禁用。

1.4.4 Unix 类型的鼠标操作

Unix 类型的鼠标操作基于 3 键的鼠标系统。

在现代滚轮鼠标上的中央滚轮，被认为是中间键，并可以被当做中间键使用。在 2 键鼠标系统的情况下，同时按左键和右键就相当于按中间键。为了在 Linux 字符控制台使用鼠标，你需要把 gpm(8) 作为后台守护进程（daemon）运行。

操作	响应
左击并拖动鼠标	选择并复制到剪贴板
单击左键	选择开头
单击右键	选择末尾并拷贝到剪贴板
单击中键	粘贴剪切板的内容到光标处

Table 1.15: Unix 样式的鼠标操作列表

1.4.5 分页程序

less(1) 命令是一个增强版的分页程序（文件内容查看器）。它按照指定的命令参数或标准输出来读取文件。在用 `less` 命令查看的时候如果需要帮助可以按“h”。它的功能比 more(1) 命令更丰富，通过在脚本的开头执行”`eval $(lesspipe)`”或”`eval $(lessfile)`”它的功能还能变得更加强大。详细请参考”`/usr/share/doc/less/LESSOPEN`”。”`-R`” 选项可以实现原始的字符输出还可以启用 ANSI 颜色转义序列。详细请参考 less(1)。

1.4.6 文本编辑器

在使用类 Unix 系统过程中，各种类似于Vim 或 Emacs的工具，你应该精通其中的一个。

我认为习惯于使用 Vim 命令是一个明智的选择，因为 Linux/Unix 系统里一般都附带了 Vi 编辑器。(实际上最初的 `vi` 以及后来的 `nvi` 这类工具程序很常见。因为在 Vim 里提供了 `F1` 帮助键，在同类工具中它的功能更强大，所以我选择 Vim 而不是其它新出的一些工具。)

假设你不是用 Emacs 就是用XEmacs 作为你的编辑器，其实还有更好的选择，尤其是在编程的时候。Emacs 还有很多其他的特点，包括新手导读，目录编辑器，邮件客户端等等。当编写脚本或程序的时候，它能自动识别当前工作模式所对应的格式，让使用更加便利。一些人甚至坚持认为 Linux 系统里最需要配备的就是 Emacs。花十分钟来学习 Emacs 可以为后面的工作剩下更多时间。在此强烈推荐学习使用 Emacs 时候直接使用 GNU Emacs 参考手册。

在实践应用中所有这些程序都会有一个教程，输入”`vim`” 和 F1 键就可以启动 Vim。建议你最好阅读一下前面的 35 行。移动光标到”`|tutor|`” 并按 `Ctrl-]` 就可以看到在线培训教程。

注意
好的编辑器，像 Vim 和 Emacs，可以处理 UTF-8 及其它不常用编码格式的文本。有个建议就是在 X 环境下使用 UTF-8 编码，并安装要求的程序和字体。编辑器里可以选择独立于 X 环境的编码格式。关于多字节文本可以查阅参考文档。

1.4.7 设置默认文本编辑器

Debian 有许多不同的编辑器。我们建议安装上面提到的 `vim` 软件包。

Debian 通过命令“`/usr/bin/editor`”提供了对系统默认编辑器的统一访问，因此其它程序（例如 reportbug(1)）可以调用它。你可以通过下列命令改变它。

```
$ sudo update-alternatives --config editor
```

对于新手，我建议使用“`/usr/bin/vim.basic`”代替“`/usr/bin/vim.tiny`”，因为它支持格式高亮。

提示
许多程序使用环境变量“`$EDITOR`”或“`$VISUAL`”来决定使用那个编辑器（参见第 1.3.5 节和第 9.3.11 节）。出于 Debian 系统的一致性考虑，它们被设置到“`/usr/bin/editor`”。(在历史上，“`$EDITOR`”是“`ed`”，“`$VISUAL`”是“`vi`”。)

1.4.8 定制 vim

你可以通过“~/.vimrc”来定制 vim(1) 的行为。

尝试下列例子

```
" ------------------------------
" Local configuration
"
set nocompatible
set nopaste
set pastetoggle=<f2>
syn on
if $USER == "root"
 set nomodeline
 set noswapfile
else
 set modeline
 set swapfile
endif
" filler to avoid the line above being recognized as a modeline
" filler
" filler
```

1.4.9 记录 shell 活动

shell 命令的输出有可能滚动出了屏幕，并可能导致你无法再查看到它。将 shell 活动记录到文件中再来回顾它是个不错的主意。当你执行任何系统管理任务时，这种记录是必不可少的。

记录 shell 活动的基本方法是在 script(1) 下运行 shell。

尝试下列例子

```
$ script
Script started, file is typescript
```

在 script 下使用任何 shell 命令。

按 Ctrl-D 来退出 script。

```
$ vim typescript
```

参见第 9.2.3 节。

1.4.10 基本的 Unix 命令

让我们来学习基本的 Unix 命令。在这里，我指的是一般意义上的“UNIX”。任何 UNIX 克隆系统通常都会提供等价的命令。Debian 系统也不例外。如果有一些命令不像你想的那样起作用，请不要担心。如果 shell 中使用了别名，其对应的命令输出会不同。这些例子并不意味着要以这个顺序来执行。

尝试使用非特权用户账号来使用下列的命令。

注意
Unix 有一个惯例，以“.”开头的文件将被隐藏。它们一般为包含了配置信息和用户首选项的文件。

命令	说明
pwd	显示当前/工作目录的名称
whoami	显示当前的用户名
id	显示当前用户的身份（名称、uid、gid 和相关组）
file <foo>	显示“<foo>”文件的文件类型
type -p <commandname>	显示“<commandname>”命令的文件所处位置
which <commandname>	同上
type <commandname>	显示“<commandname>”命令的相关信息
apropos <key-word>	查找与“<key-word>”有关的命令
man -k <key-word>	同上
whatis <commandname>	用一行解释“<commandname>”命令
man -a <commandname>	显示“<commandname>”命令的解释（Unix 风格）
info <commandname>	显示“<commandname>”命令相当长的解释（GNU 风格）
ls	显示目录内容（不包含以. 点号开头的文件和目录）
ls -a	显示目录内容（包含所有文件和目录）
ls -A	显示目录内容（包含几乎所有文件和目录，除了“..”和“.”）
ls -la	显示所有的目录内容，并包含详细的信息
ls -lai	显示所有的目录内容，并包含 inode 和详细的信息
ls -d	显示当前目录下的所有目录
tree	使用树状图显示目录内容
lsof <foo>	列出处于打开状态的文件”<foo>”
lsof -p <pid>	列出被某进程打开的文件: ”<pid>”
mkdir <foo>	在当前目录中建立新目录“<foo>”
rmdir <foo>	删除当前目录中的“<foo>”目录
cd <foo>	切换到当前目录下或变量“$CDPATH”中的“<foo>”目录
cd /	切换到根目录
cd	切换到当前用户的家目录
cd /<foo>	切换到绝对路径为“/<foo>”的目录
cd ..	切换到上一级目录
cd ~<foo>	切换到用户“<foo>”的家目录
cd -	切换到之前的目录
</etc/motd pager	使用默认的分页程序来显示“/etc/motd”的内容
touch <junkfile>	建立一个空文件“<junkfile>”
cp <foo> <bar>	将一个现有文件“<foo>”复制到一个新文件“<bar>”
rm <junkfile>	删除文件“<junkfile>”
mv <foo> <bar>	将一个现有文件“<foo>”重命名成“<bar>”（“<bar>”必须不存在）
mv <foo> <bar>	将一个现有文件“<foo>”移动到新的位置“<bar>/<foo>”（必须存在“<bar>”目录）
mv <foo> <bar>/<baz>	移动一个现有文件“<foo>”到新位置并重命名为“<bar>/<baz>”（必须存在“bar”目录，且不存在“bar>/<baz> 文件”）
chmod 600 <foo>	使其他人无法读写现有文件“<foo>”（并且所有人都无法执行该文件）
chmod 644 <foo>	使其他人对现有文件“<foo>”可读但不可写（并且所有人都无法执行该文件）
chmod 755 <foo>	使其他人对“<foo>”可读而不可写（并且所有人都能执行该文件）
find . -name <pattern>	使用 shell “<pattern>”查找匹配的文件名（速度较慢）
locate -d . <pattern>	使用 shell “<pattern>”查找匹配的文件名（速度较快，使用定期生成的数据库）
grep -e ”<pattern>” *.html	在当前目录下以“.html”结尾的所有文件中，查找匹配“<pattern>”的文件并显示
top	全屏显示进程信息，输入“q”退出
ps aux | pager	显示所有正在运行的进程的信息（BSD 风格）
ps -ef | pager	显示所有正在运行的进程的信息（Unix system-V 风格）
ps aux | grep -e ”[e]xim4*”	显示所有正在运行“exim”和“exim4”的进程
ps axf | pager	显示所有正在运行的进程的信息（ASCII 风格）
kill <1234>	杀死 ID 为“<1234>”的进程
gzip <foo>	使用 Lempel-Ziv 编码（LZ77）将“<foo>”压缩为“<foo>.gz”
gunzip <foo>.gz	将“<foo>.gz”解压为“<foo>”
bzip2 <foo>	使用 Burrows-Wheeter 块排序压缩算法和 Huffman 编码将“<foo>”压缩为“<foo>.bz2”（压缩效果比 gzip 更好）

注意
对于 cd 命令，参见 builtins(7)。

注意
基本的 Debian 系统的默认分页程序是 more(1)，它无法往回滚动。通过命令“apt-get install less”安装 less 软件包后，less(1) 会成为默认的分页程序，它可以通过方向键往回滚动。

注意
”[” 和”]” 在正则表达式”ps aux | grep -e ”[e]xim4*”” 命令中，可以避免 grep 在结果中排除它自己，正则表达式中的”4*” 意思是空或字符”4”，这样可以让 grep 既找到”exim” 也找到”exim4”。虽然”*” 可以用于命令名称匹配和正则表达式中，但是它们的含义是不一样的。欲详细了解正则表达式可以参考 grep(1)。

作为训练，请使用上述的命令来遍历目录并探究系统。如果你有任何有关控制台命令的问题，请务必阅读手册。

尝试下列例子

```
$ man man
$ man bash
$ man builtins
$ man grep
$ man ls
```

手册的风格可能让人有点难以习惯，因为它们都相当简洁，尤其是比较老旧、非常传统的那些手册。但是，一旦你习惯了它，你来欣赏它们的简洁。

请注意，许多类 Unix 命令（包含来自 GNU 和 BSD 的）都可以显示简短的帮助信息，你可以使用下列的其中一种方式来查看它（有时不带任何参数也可以）。

```
$ <commandname> --help
$ <commandname> -h
```

1.5 简单 shell 命令

现在，你对如何使用 Debian 系统已经有一些感觉了。让我们更深入了解 Debian 系统的命令执行机制。在这里，我将为新手做一般的讲解。精确的解释参见 bash(1)。

一般的命令由有序的组件构成。

1. 设置变量值（可选）
2. 命令名
3. 参数（可选）
4. 重定向（可选：>, >>, <, << 等等）
5. 控制操作（可选：&&，||，< 换行符 >, ;, &, (,)）

1.5.1 命令执行和环境变量

一些环境变量的值会改变部分 Unix 命令的行为。

环境变量的默认值由 PAM 系统初始化，其中一些会被某些应用程序重新设定。

- 显示管理器（例如 `gdm3`）会重新设定环境变量。
- Shell 脚本启动的时候会重置"`~/.bash_profile`"和"`~/.bashrc`"中的环境变量。

1.5.2 "$LANG"变量

"`$LANG`"变量的完整的语言环境值由 3 部分组成："`xx_YY.ZZZZ`"。

语言环境值	说明
xx	ISO 639 语言代码（小写）例如"en"
YY	ISO 3166 国家代码（大写）例如"US"
ZZZZ	编码，总是设置为"UTF-8"

Table 1.17: 语言环境值的 3 个部分

对于语言代码和国家代码，参加"`info gettext`"中的相关描述。

对于现代 Debian 系统中的编码，你应该总是设定为 **`UTF-8`**，除非你有足够的理由和背景知识并且特别想使用过时的编码。

对于语言环境配置的细节，参见第 8.4 节。

> 注意
> "`LANG=en_US`"既不是"`LANG=C`"也不是"`LANG=en_US.UTF-8`"。它是"`LANG=en_US.ISO-8859-1`"（参见第 8.4.1 节）。

语言环境推荐	语言（地区）
en_US.UTF-8	英语（美国）
en_GB.UTF-8	英语（大不列颠）
fr_FR.UTF-8	法语（法国）
de_DE.UTF-8	德语（德国）
it_IT.UTF-8	意大利语（意大利）
es_ES.UTF-8	西班牙语（西班牙）
ca_ES.UTF-8	加泰隆语（西班牙）
sv_SE.UTF-8	瑞典语（瑞典）
pt_BR.UTF-8	葡萄牙语（巴西）
ru_RU.UTF-8	俄语（俄国）
zh_CN.UTF-8	汉语（中华人民共和国）
zh_TW.UTF-8	汉语（中国台湾）
ja_JP.UTF-8	日语（日本）
ko_KR.UTF-8	韩语（韩国）
vi_VN.UTF-8	越南语（越南）

Table 1.18: 语言环境推荐列表

使用 shell 命令行按顺序执行下列典型的命令。

```
$ date
Sun Jun  3 10:27:39 CST 2007
$ LANG=fr_FR.UTF-8 date
dimanche 3 juin 2007, 10:27:33 (UTC+0800)
```

这里，date(1) 程序执行时使用了与环境变量“$LANG”不同的值。

- 第一个命令，“$LANG”设置为系统的默认语言环境值“en_US.UTF-8”。
- 第二个命令，“$LANG”设置为法语的 UTF-8 语言环境值“fr_FR.UTF-8”。

大多数的命令在执行时并没有预先定义环境变量。对于上面的例子，你也可以选择如下的方式。

```
$ LANG=fr_FR.UTF-8
$ date
dimanche 3 juin 2007, 10:27:33 (UTC+0900)
```

正如你所看到的，命令的输出受环境变量的影响，上面产生的是法语输出。如果你想这个环境变量能在子进程中被继承的话 (例如执行 shell 脚本时)，你需要使用下面的命令导出（**export**）它。

```
$ export LANG
```

注意
在使用常规的控制台终端的时候，环境变量"$LANG" 通常会被桌面环境变量通过 **exported** 方式重置。如果要测试 export 带来的影响，这个可能不是一个很好的例子。

提示
提交一个 BUG 报告的时候，如果使用的是非英语的环境，在"LANG=en_US.UTF-8" 环境下对命令进行运行和检查会更好一些。

可以通过命令 locale(5) 和 locale(7) 来查看"$LANG" 及相关的环境变量。

注意
建议最好用变量"$LANG" 来配置系统环境变量，只有在逼不得已的情况下才用 $LC_*" 开头的变量。

1.5.3 "$PATH" 变量

当你在 Shell 里输入命令的时候，Shell 会在"$PATH" 变量所包含的目录列表里进行搜索，"$PATH" 变量的值也叫作 Shell 的搜索路径。

在默认的 Debian 安装过程中，所使用的用户账号的"$PATH" 环境变量可能不包括"/sbin" 和"/usr/sbin" 目录。例如，ifconfig 命令就需要指定完整的路径"/sbin/ifconfig"。(类似地，ip 命令是在"/bin" 目录下)

可以在 Bash 脚本文件"~/.bash_profile" 或"~/.bashrc" 中改变"$PATH" 环境变量的值。

"$HOME" 变量的值	程序运行环境
/	初始进程执行的程序（守护进程）
/root	root 用户权限 Shell 执行的程序
/home/<normal_user>	普通用户权限 Shell 执行的程序
/home/<normal_user>	普通用户 GUI 桌面菜单执行的程序
/home/<normal_user>	用 root 用户权限来执行程序"sudo program"
/root	用 root 用户权限执行程序"sudo -H program"

Table 1.19: "$HOME" 变量值列表

1.5.4 "$HOME" 变量

很多命令在用户目录中都存放了用户指定的配置，然后通过配置的内容来改变它的执行方式，用户目录通常用"$HOME"变量来指定。

提示
Shell 扩展"~/" 为转入当前用户的主目录，也就是"$HOME/"。Shell 扩展"~foo/" 为 foo 的目录，也就是"/home/foo/"。

1.5.5 命令行选项

一些命令附带参数。这些参数以"-" 或"--" 开头，通常称之为选项，用来控制命令的执行方式。

```
$ date
Mon Oct 27 23:02:09 CET 2003
$ date -R
Mon, 27 Oct 2003 23:02:40 +0100
```

这里的命令参数"-R" 改变 date(1) 命令输出为 RFC2822 标准的日期字符格式。

1.5.6 Shell 通配符

经常有这种情况你期望命令成串自动执行而不需要挨个输入，将文件名扩展为 **glob**，(有时候被称为 通配符)，以此来满足这方面的需求。

shell glob 模式	匹配规则描述
*	不以"." 开头的文件名 (段)
.*	以"." 开头的文件名 (段)
?	精确字符
[…]	包含在括号中的任意字符都可以作为精确字符
[a-z]	"a" 到"z" 之间的任意一个字符都可以作为精确字符
[^…]	除了包含在括号中的任意字符 (" 1^ 2" 除外)，其它字符都可以作为精确字符

Table 1.20: Shell glob 模式

尝试下列例子

```
$ mkdir junk; cd junk; touch 1.txt 2.txt 3.c 4.h .5.txt ..6.txt
$ echo *.txt
1.txt 2.txt
$ echo *
1.txt 2.txt 3.c 4.h
$ echo *.[hc]
3.c 4.h
$ echo .*
. .. .5.txt ..6.txt
$ echo .*[^.]*
.5.txt ..6.txt
$ echo [^1-3]*
4.h
$ cd ..; rm -rf junk
```

参见 glob(7)。

注意
与 shell 通用的文件名匹配方式不同，使用” -name ”选项的 find (1)，其 shell 模式” * ”，匹配以” . ”开始的文件名. (新POSIX 的特性)

注意
BASH 可以使用内置的 shopt 选项如” dotglob ”，” noglob ”,” nocaseglob ”,” nullglob ”,” extglob ”定制全局行为, 使用 bash (1) 查看详细说明。

1.5.7 命令的返回值

每个命令都会返回它的退出状态 (变量:“$?”) 作为返回值。

命令的退出状态	数字返回值	逻辑返回值
success	zero, 0	**TRUE**
error	non-zero, -1	**FALSE**

Table 1.21: 命令的退出代码

尝试下列例子。

```
$ [ 1 = 1 ] ; echo $?
0
$ [ 1 = 2 ] ; echo $?
1
```

注意
请注意，**success** 是逻辑 **TRUE** ，0 (zero) 则是它的值。这有些不直观，需要在这里提一下。

命令常见用法	说明
`command &`	在子 shell 的后台 中执行 `command`
`command1 | command2`	通过管道将 `command1` 的标准输出作为 `command2` 的标准输入（并行执行）
`command1 2>&1 | command2`	通过管道将 `command1` 的标准输出和标准错误作为 `command2` 的标准输入（并行执行）
`command1 ; command2`	按顺序执行 `command1` 和 `command2`
`command1 && command2`	执行 `command1`；如果成功，按顺序执行 `command2`（如果 `command1` 和 `command2` 都执行成功了，返回 success ）
`command1 || command2`	执行 `command1`；如果不成功，按顺序执行 `command2`（如果 `command1` 或 `command2` 执行成功，返回 success ）
`command > foo`	将 `command` 的标准输出重定向到文件 `foo`（覆盖）
`command 2> foo`	将 `command` 的标准错误重定向到文件 `foo`（覆盖）
`command >> foo`	将 `command` 的标准输出重定向到文件 `foo`（附加）
`command 2>> foo`	将 `command` 的标准错误重定向到文件 `foo`（附加）
`command > foo 2>&1`	将 `command` 的标准输出和标准错误重定向到文件 `foo`
`command < foo`	将 `command` 的标准输入重定向到文件 `foo`
`command << delimiter`	将 `command` 的标准输入重定向到下面的命令行，直到遇到“`delimiter`”（here document）
`command <<- delimiter`	将 `command` 的标准输入重定向到下面的命令行，直到遇到“`delimiter`”（here document，命令行中开头的制表符会被忽略）

Table 1.22: Shell 命令常见用法

1.5.8 典型的顺序命令和 shell 重定向

让我们试着记住下面 Shell 命令里部分命令行所使用的命令习语。

Debian 系统是一个多任务的操作系统。后台任务让用户能够在一个 shell 中执行多个程序。后台进程的管理涉及 shell 的内建命令：`jobs`、`fg`、`bg` 和 `kill`。请阅读 bash(1) 中的章节：“SIGNALS”、“JOB CONTROL”和“builtins(1)”。

尝试下列例子

```
$ </etc/motd pager
```

```
$ pager </etc/motd
```

```
$ pager /etc/motd
```

```
$ cat /etc/motd | pager
```

尽管 4 个 shell 重定向的例子都会显示相同的结果，但最后一个例子毫无意义地运行了额外的 `cat` 命令浪费了资源。

shell 允许你使用 `exec` 通过任意一个文件描述符来打开文件。

```
$ echo Hello >foo
$ exec 3<foo 4>bar  # 打开文件
$ cat <&3 >&4       # 标准输入重定向到 3，标准输出重定向到 4
$ exec 3<&- 4>&-    # 关闭文件
$ cat bar
Hello
```

预定义的文件描述符 0-2。

设备	说明	文件描述符
stdin	标准输入	0
stdout	标准输出	1
stderr	标准错误	2

Table 1.23: 预定义的文件描述符

1.5.9 命令别名

你可以为经常使用的命令设置一个别名。

尝试下列例子

```
$ alias la='ls -la'
```

现在，“la”是“ls -al”的简写形式，并同样会以长列表形式列出所有的文件。

你可以使用 alias 来列出所有的别名（参见 bash(1) 中的“SHELL BUILTIN COMMANDS”）。

```
$ alias
...
alias la='ls -la'
```

你可以使用 type 来确认命令的准确路径或类型（参见 bash(1) 中的“SHELL BUILTIN COMMANDS”）。

尝试下列例子

```
$ type ls
ls is hashed (/bin/ls)
$ type la
la is aliased to ls -la
$ type echo
echo is a shell builtin
$ type file
file is /usr/bin/file
```

ls 在最近被使用过，而“file”没有，因此“ls”标记为“hashed”（被录入哈希表），即 shell 有一个内部的记录用来快速访问“ls”所处的位置。

提示
参见第 9.2.7 节。

1.6 类 Unix 的文本处理

在类 Unix 的工作环境中，文本处理是通过使用管道组成的标准文本处理工具链完成的。这是另一个重要的 Unix 创新。

1.6.1 Unix 文本工具

这里有一些在类 Unix 系统中经常使用到的标准文本处理工具。

- 没有使用正则表达式：
  - cat(1) 连接文件并输出全部的内容。
  - tac(1) 连接文件并反向输出。
  - cut(1) 选择行的一部分并输出。
  - head(1) 输出文件的开头。
  - tail(1) 输出文件的末尾。
  - sort(1) 对文本文件的行进行排序。
  - uniq(1) 从已排序的文件中移除相同的行。
  - tr(1) 转换或删除字符。
  - diff(1) 对文件的行进行对比。
- 使用基础的正则表达式（**BRE**）：
  - grep(1) 匹配满足 pattern 的文本。
  - ed(1) 是一个原始行编辑器。
  - sed(1) 是一个流编辑器。
  - vim(1) 是一个屏幕编辑器。
  - emacs(1) 是一个屏幕编辑器。（有些扩展的 **BRE**）
- 使用扩展的正则表达式（**ERE**）：
  - egrep(1) 匹配满足多个 pattern 的文本。
  - awk(1) 进行简单的文本处理。
  - tcl(3tcl) 可以进行任何你想得到的文本处理：参见 re_syntax(3)。经常与 tk(3tk) 一起使用。
  - perl(1) 可以进行任何你想得到的文本处理。参见 perlre(1)。
  - `pcregrep` 软件包中的 pcregrep(1) 可以匹配满足 Perl 兼容正则表达式（PCRE）模式的文本。
  - 带有 `re` 模块的 python(1) 可以进行任何你想得到的文本处理。参见“`/usr/share/doc/python/html/index.html`”。

如果你不确定这些命令究竟做了什么，请使用“`man command`”来自己把它搞清楚吧。

注意
排序的顺序和表达式的范围取决于语言环境。如果你想要获得一个命令的传统行为，可以在命令之前使用“`LANG=C`”让 **C** 语言环境代替 **UTF-8**（参见第 1.5.2 节和第 8.4 节）。

注意
Perl 正则表达式（perlre(1)）、Perl 兼容正则表达式（PCRE）和 Python 的 `re` 模块提供的正则表达式与一般的 **ERE** 相比多了许多通用的扩展。

1.6.2 正则表达式

正则表达式被使用在许多文本处理工具中。它们类似 shell 的通配符，但更加复杂和强大。

正则表达式描述要匹配的模式，它是由文本字符和元字符构成的。

元字符仅仅是带有特殊含义的字符。它们有两种主要的形式，**BRE** 和 **ERE**，使用哪种取决于上述的文本工具。

emacs 中的正则表达式基本上是 **BRE** 但含有 **ERE** 中的元字符“+”和“?”。因此，在 `emacs` 中没必要使用“\”来转义它们。

grep(1) 可以使用正则表达式来进行文本搜索。

尝试下列例子

BRE	ERE	正则表达式的描述
\ . [ ] ^ $ *	\ . [ ] ^ $ *	通用的元字符
\+ \? \(\) \{ \} |		BRE 独有的“\”转义元字符
	+ ? ( ) { } |	ERE 独有的不需要“\”转义的元字符
c	c	匹配非元字符“c”
\c	\c	匹配一个字面意义上的字符“c”，即使“c”本身是元字符
.	.	匹配任意字符，包括换行符
^	^	字符串的开始位置
$	$	字符串的结束位置
\<	\<	单词的开始位置
\>	\>	单词的结束位置
[abc…]	[abc…]	匹配在“abc...”中的任意字符
[^abc…]	[^abc…]	匹配除了“abc...”中的任意字符
r*	r*	匹配零个或多个“r”
r\+	r+	匹配一个或多个“r”
r\?	r?	匹配零个或一个“r”
r1|r2	r1|r2	匹配一个“r1”或“r2”
\(r1|r2\)	(r1|r2)	匹配一个“r1”或“r2“，并作为括号内的正则表达式

Table 1.24: BRE 和 ERE 中的元字符

```
$ egrep 'GNU.*LICENSE|Yoyodyne' /usr/share/common-licenses/GPL
GNU GENERAL PUBLIC LICENSE
GNU GENERAL PUBLIC LICENSE
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
```

提示
参见第 9.2.7 节。

1.6.3 替换表达式

对于替换表达式，一些字符有特殊的含义。

替换表达式	替换表达式替换的文本
&	正则表达式所匹配的内容（在 emacs 中使用 \& ）
\n	前 n 个括号的正则表达式匹配的内容（“n”是数字）

Table 1.25: 替换表达式

对 Perl 替换字符串来说，应使用“$&”而非“&”，应使用“$n”而非“\n”。

尝试下列例子

```
$ echo zzz1abc2efg3hij4 | \
sed -e 's/\(1[a-z]*\)[0-9]*\(.*\)$/=&=/'
zzz=1abc2efg3hij4=
$ echo zzz1abc2efg3hij4 | \
sed -e 's/\(1[a-z]*\)[0-9]*\(.*\)$/\2===\1/'
```

```
zzzefg3hij4===1abc
$ echo zzz1abc2efg3hij4 | \
perl -pe 's/(1[a-z]*)[0-9]*(.*)$/$2===$1/'
zzzefg3hij4===1abc
$ echo zzz1abc2efg3hij4 | \
perl -pe 's/(1[a-z]*)[0-9]*(.*)$/=$&=/'
zzz=1abc2efg3hij4=
```

请特别注意这些括号正则表达式的格式，以及这些被匹配的文本在不同的工具中是如何被替换的。

这些正则表达式在一些编辑器中也可以用来移动光标和替换文本。

在 shell 命令行行末的反斜杠“\”会跳脱一个换行符（作为空白符），并将光标移动到下一行的行首。

请阅读所有相关手册来学习这些命令。

1.6.4 正则表达式的全局替换

ed(1) 命令可以在“file”中将所有的“FROM_REGEX”替换成“TO_TEXT”。

```
$ ed file <<EOF
,s/FROM_REGEX/TO_TEXT/g
w
q
EOF
```

sed(1) 命令可以在“file”中将所有的“FROM_REGEX”替换成“TO_TEXT”。

```
$ sed -i -e 's/FROM_REGEX/TO_TEXT/g' file
```

vim(1) 命令可以通过使用 ex(1) 命令在“file”中将所有的“FROM_REGEX”替换成“TO_TEXT”。

```
$ vim '+%s/FROM_REGEX/TO_TEXT/gc' '+w' '+q' file
```

提示
上面的“c”标志可以确保在每次替换时都进行交互式的确认。

多个文件（“file1”，“file2”和“file3”）可以使用 vim(1) 或 perl(1) 通过正则表达式进行类似的处理。

```
$ vim '+argdo %s/FROM_REGEX/TO_TEXT/ge|update' '+q' file1 file2 file3
```

提示
上面的“e”标志是为了防止“No match”错误中断替换。

```
$ perl -i -p -e 's/FROM_REGEX/TO_TEXT/g;' file1 file2 file3
```

在 perl(1) 例子中，”-i” 是在每一个目标文件的原处编辑，”-p” 是表示循环所有给定的文件。

提示
使用参数“-i.bak”代替“-i”，可以在文件名后添加“.bak”再保存。对于复杂的替换，这使得从错误中恢复变得容易。

注意
ed(1) 和 vim(1) 使用 **BRE**； perl(1) 使用 **ERE**。

1.6.5 从文本文件的表格中提取数据

下面有一个文本文件“DPL”，里面含有 2004 年以前 Debian 项目的领导者名字和起始日期，并以空格分隔。

```
Ian     Murdock   August  1993
Bruce   Perens    April   1996
Ian     Jackson   January 1998
Wichert Akkerman  January 1999
Ben     Collins   April   2001
Bdale   Garbee    April   2002
Martin  Michlmayr March   2003
```

提示
参见 “Debian 简史” 获取最新的 Debian 领导阶层历史。

Awk 经常被用来从这种类型的文件中提取数据。

尝试下列例子

```
$ awk '{ print $3 }' <DPL                    # month started
August
April
January
January
April
April
March
$ awk '($1=="Ian") { print }' <DPL           # DPL called Ian
Ian     Murdock   August  1993
Ian     Jackson   January 1998
$ awk '($2=="Perens") { print $3,$4 }' <DPL # When Perens started
April 1996
```

Shell（例如 Bash）也可以用来分析这种文件。

尝试下列例子

```
$ while read first last month year; do
    echo $month
  done <DPL
... 第一个 AWK 例子的一些输出
```

内建命令 read 使用“$IFS”（内部域分隔符）中的字符来将行分隔成多个单词。

如果你将“$IFS”改变为“:”，你可以很好地使用 shell 来分析“/etc/passwd”。

```
$ oldIFS="$IFS"   # save old value
$ IFS=':'
$ while read user password uid gid rest_of_line; do
    if [ "$user" = "bozo" ]; then
      echo "$user's ID is $uid"
    fi
  done < /etc/passwd
bozo's ID is 1000
$ IFS="$oldIFS"   # restore old value
```

（如果要用 Awk 做到相同的事，使用“FS=':'”来设置域分隔符。）

IFS 也被 shell 用来分割参数扩展、命令替换和算术扩展的结果。这不会出现在双引号或单引号中。IFS 的默认值为 <空格 >、<tab> 和 < 换行符 >。

请谨慎使用 shell 的 IFS 技巧。当 shell 将脚本的一部分解释为对它的输入时，会发生一些奇怪的事。

```
$ IFS=":,"                        # use ":" and "," as IFS
$ echo IFS=$IFS,   IFS="$IFS"     # echo is a Bash builtin
IFS=  , IFS=:,
$ date -R                         # just a command output
Sat, 23 Aug 2003 08:30:15 +0200
$ echo $(date -R)                 # sub shell --> input to main shell
Sat  23 Aug 2003 08 30 36 +0200
$ unset IFS                       # reset IFS to the default
$ echo $(date -R)
Sat, 23 Aug 2003 08:30:50 +0200
```

1.6.6 用于管道命令的小片段脚本

下面的脚本作为管道的一部分，可以做一些细致的事情。

使用 find(1) 和 xargs(1)，单行 shell 脚本能够在多个文件上循环使用，可以执行相当复杂的任务。参见第 10.1.5 节和第 9.3.9 节.

当使用 shell 交互模式变得太麻烦的时候，请考虑写一个 shell 脚本 (参见第 12.1 节).

脚本片段（在一行内输入）	命令效果
`find /usr -print`	找出"/usr" 下的所有文件
`seq 1 100`	显示 1 到 100
`| xargs -n 1 <command>`	把从管道过来的每一项作为参数，重复执行命令
`| xargs -n 1 echo`	把从管道过来的，用空格隔开的项，分隔成多行
`| xargs echo`	把从管道过来的所有行合并为一行
`| grep -e <regex_pattern>`	从管道过来，包含有 <regex_pattern> 的行，提取出来
`| grep -v -e <regex_pattern>`	把从管道过来，不包含有 <regex_pattern> 的行，提取出来
`| cut -d: -f3 -`	把从管道过来，用":" 分隔的第三列提取出来 (passwd 文件等。)
`| awk '{ print $3 }'`	把用空格隔开的第三列提取出来
`| awk -F'\t' '{ print $3 }'`	把用 tab 键隔开的第三列提取出来
`| col -bx`	删除退格键，扩展 tab 键为空格键
`| expand -`	扩展 tab 键到空格键
`| sort| uniq`	排序并删除重复行
`| tr 'A-Z' 'a-z'`	将大小字母转换为小写字母
`| tr -d '\n'`	将多行连接为一行
`| tr -d '\r'`	删除换行回车符
`| sed 's/^/# /'`	在每行行首增加一个"#" 符
`| sed 's/\.ext//g'`	删除".ext"
`| sed -n -e 2p`	显示第二行
`| head -n 2 -`	显示最前面两行
`| tail -n 2 -`	显示最后两行

Table 1.26: 管道命令的小片段脚本列表

Chapter 2

Debian 软件包管理

注意
这一章假定最新的稳定版的代号为: `buster` 。

Debian 是一个志愿者组织，它建立一致的自由软件的预编译二进制包并从档案库中分发它们。

许多远程镜像站提供了 HTTP 和 FTP 的方式来访问 Debian 档案库。也可以使用 CD-ROM/DVD 。

Debian 软件包管理系统，当使用适当时，可以让用户从档案库安装统一设置的二进制软件包到系统中。现在，有 60425 个可用于 amd64 架构的软件包。

Debian 软件包管理系统有丰富的历史，有许多可供选择的前端用户程序和后端访问方式。现在，我们推荐下列的这些。

- apt(8) 用于所有的交互式命令行操作，包含软件包的安装、移除和升级。自 Debian Jessie（Debian 8）起可用。
- apt-get(8) 用于从脚本中调用 Debian 软件包管理系统。它在 `apt` 不可用时也可作为一个备选选项（常见于较旧的 Debian 系统）。
- aptitude(8) 使用一个交互式的文本界面来管理已安装的软件包和搜索可用的软件包。

2.1 Debian 软件包管理的前提

2.1.1 软件包配置

下面是 Debian 系统软件包配置的一些要点。

- Debian 尊重系统管理员的手动配置。换句话说，软件包配置系统不会为了方便而去更改那些配置。
- 每个软件包都带有自己的配置脚本，它使用标准用户接口 debconf(7) 来帮助软件包初始化安装过程。
- Debian 开发者通过软件包配置脚本，尽力使你能有一个完美的升级体验。
- 系统管理员可以使用软件包工具的全部功能。但在默认的安装中会禁用那些具有安全风险的。
- 如果你手动激活了一些具有安全隐患的服务，你有责任遏制风险。
- 高深的配置可以由系统管理员手动启用。这可能会对用于系统配置的通用流行帮助程序造成干扰。

软件包	流行度	大小	说明
apt	V:875, I:999	3997	高级软件包工具 (APT), dpkg 的前端，提供了“http”、“ftp”和“file”的档案库访问方式（包含 apt、apt-get 和 apt-cache 命令）
aptitude	V:104, I:571	4366	aptitude(8)，基于终端的交互式软件包管理工具
tasksel	V:36, I:974	378	Debian 系统上对安装进行选择的工具（APT 的前端）
unattended-upgrades	V:299, I:421	299	用于 APT 的增强软件包，会自动安装安全更新
dselect	V:3, I:43	2603	基于终端的软件包管理工具（之前的标准，APT 的前端和其它老的访问方式）
dpkg	V:940, I:999	6651	用于 Debian 的软件包管理系统
synaptic	V:60, I:438	7810	图形化的软件包管理工具（GNOME 的 APT 前端）
apt-utils	V:397, I:996	1130	APT 实用程序：apt-extracttemplates(1)、apt-ftparchive(1) 和 apt-sortpkgs(1)
apt-listchanges	V:401, I:849	399	软件包历史更改提醒工具
apt-listbugs	V:8, I:11	457	在每次 APT 安装前列出严重的 bug
apt-file	V:17, I:78	90	APT 软件包搜索工具——命令行界面
apt-rdepends	V:0, I:5	40	递归列出软件包依赖

Table 2.1: Debian 软件包管理工具列表

2.1.2 基本的注意事项

警告
不要从任何的混合套件中安装软件包。它可能会打破软件包的一致性，这需要你要深厚的系统管理知识，例如 ABI 编译器、库版本和解释器特性等等。

Debian 系统管理员中的新手应该保持在只进行安全更新的 **stable** 版本。我的意思是，最好避免下列的一些行为作为一项预防措施，直到你十分了解 Debian 系统。下面有一些提醒。

- 在“/etc/apt/sources.list”中不要包含 **testing** 或 **unstable**。
- 在“/etc/apt/sources.list”里不要在标准的 Debian 中混合使用其它非 Debian 的档案库，例如 Ubuntu。
- 不要建立“/etc/apt/preferences”。
- 不了解会造成的全部影响，就不要通过配置文件改变软件包管理工具的默认行为。
- 不要使用“dpkg -i <random_package>”安装任何软件包。
- 绝不使用“dpkg --force-all -i <random_package>”安装任何软件包。
- 不要删除或修改“/var/lib/dpkg/”中的文件。
- 不要让从源码直接安装的程序覆盖系统文件。
  - 如果需要的话，将它们安装到“/usr/local”或“/opt”中。

上述对 Debian 软件包管理系统做的行为所导致的不兼容影响可能会使你的系统无法使用。

负责有关键任务的服务器的严谨的 Debian 系统系统管理员，应该使用额外的预防措施。

- 没有在安全的条件下使用你特定的配置进行彻底地测试，就不要从 Debian 安装任何软件包（包含安全更新）。
  - 你作为系统管理员要对你的系统负责到底。
  - Debian 系统长久的稳定史并无法保证什么。

2.1.3 持续升级的生活

尽管我在上面进行了警告，我知道许多阅读这份文档的人还是想要使用 Debian 的 testing 或 unstable 套件来作为他们自行管理的桌面环境的主系统。这是因为这些套件运行得很好，更新频繁，并提供了最新的特性。

小心
对于你的生产服务器，建议使用带有安全更行的 stable 套件。对于你只进行有限管理的桌面 PC 也是同样如此，例如你母亲的 PC 。

你只需要在“/etc/apt/sources.list”中简单地将发行版字符串设置为套件名：“testing”或“unstable”；或者代号：“bullseye”或“sid”。这会使你享受持续升级的生活。

使用 testing 或 unstable 是很有乐趣的，但会带来一些风险。尽管 Debian 系统的 unstable 套件在大多数时候看起来都非常稳定，但在 Debian 系统的 testing 和 unstable 套件中存在一些软件包问题，并且它们中的一部分是不容易解决的。这对你来说可能会很痛苦。有时候，你可能会有一个损坏的软件包或缺少某些功能几个星期。

这里有一些方法，可以使你简单快速地从 Debian 软件包的 bug 中恢复。

- 通过将 Debian 系统的 stable 套件安装到另一个分区，可以使系统能够进行双启动
- 制作安装 CD 便于用于 救援启动
- 考虑安装 apt-listbugs ，这可以在升级之前检查 Debian Bug 跟踪系统（BTS）的信息
- 对软件包系统的基础设施有足够的了解来解决问题
- 建立一个 chroot 或类似的环境来提前运行最新的系统（参见第 9.10 节）

（如果你无法做到这些预防措施中的任何一个，那你可能还没做好使用 testing 和 unstable 套件的准备。）

菩萨使用下面的内容拯救一个人，使他从挣扎于持续升级地狱的因果报应中脱困，并让他达到 Debian 的极乐世界。

2.1.4 Debian 档案库基础

让我们从系统用户的角度来看看 Debian 档案库。

提示
Debian 档案库官方政策的定义参见 Debian 政策文档，第 2 章——Debian 档案库。

对于典型的 HTTP 访问，档案库在“/etc/apt/sources.list”文件中像下面那样指定，例如，现在 stable = buster 系统。

```
deb http://deb.debian.org/debian/ buster main contrib non-free
deb-src http://deb.debian.org/debian/ buster main contrib non-free

deb http://security.debian.org/ buster/updates main contrib
deb-src http://security.debian.org/ buster/updates main contrib
```

这里，我倾向于使用代号“buster”来代替套件名“stable”，以避免下一个 stable 版本发布时出现意外。

“/etc/apt/sources.list”的含义在 sources.list(5) 中进行了描述，下面是一些要点。

- “deb”的那行定义了二进制软件包。

- “deb-src”的那行定义了源代码软件包。
- 第一个参数是 Debian 档案库的根 URL 。
- 第二个参数是发行版名称：可以使用套件名或代号。
- 第三个和之后的参数是 Debian 档案库的有效档案库范围名称。

如果只是用 aptitude（它不访问源代码相关的元数据），“deb-src”那行可以安全地删掉（或者在文件开头添加“#”来将它注释掉）。这可以加速档案库元数据的更新。URL 可以是“http://”、“ftp://”、“file://”……

提示
如果在上述的例子中，使用了“sid”代替“buster”，那么“/etc/apt/sources.list”中的“deb: http://security.debian.org/ …”这行就不需要了。因为没有用于“sid”（unstable）的安全更新的档案库。

下面是配置文件所使用的 Debian 档案库站点的 URL 和套件名或代号的列表。

档案库 **URL**	套件名（代号）	目的
http://deb.debian.org/debian/	stable (buster)	stable (buster) release 版本
http://deb.debian.org/debian/	testing (bullseye)	testing (bullseye) release 版本
http://deb.debian.org/debian/	unstable (sid)	unstable (sid) release 版本
http://deb.debian.org/debian/	experimental	experimental pre-release 版本（可选，只适用于开发者）
http://deb.debian.org/debian/	stable-proposed-updates	用于下一个稳定版本的更新（可选）
http://security.debian.org/	stable/updates	用于 stable release 版本的安全更新（重要）
http://security.debian.org/	testing/updates	用于 testing release 版本的安全更新（重要）
http://deb.debian.org/debian/	buster-updates	用于 buster 的垃圾邮件过滤器、IM 客户端等的兼容更新
http://deb.debian.org/debian/	buster-backports	用于 buster 的较新的 backported 软件包（可选）

Table 2.2: Debian 档案库站点列表

小心
只有带有安全更新的纯净的 **stable** release 版本可以提供最佳的稳定性。运行大多数 **stable** release 版本的软件包之中混合一些来自 **testing** 或 **unstable** release 版本的软件包会比运行纯净的 **unstable** release 版本冒更大的风险，这是因为库版本的不匹配导致的。如果在 **stable** release 版本下你真的需要一些程序的最新版本，请使用来自 buster-updates 和 http://backports.debian.org（参见第 2.7.4 节）的软件包。使用这些软件包时必须额外小心。

小心
在”deb” 行中，你只需列出 stable，testing 或者 unstable 套件中的一个即可，如果你在”deb” 行中混合了 stable，testing 和 unstable 套件，APT 程序的执行速度将会变慢并且只有最新的档案库是有用的。只有在”/etc/apt/preferences” 文件带有明确目标的时候，混合的列表才是有意义的。（查看第 2.7.3 节）。

提示
对于使用 stable 和 testing 套件的 Debian 系统而言，在“/etc/apt/sources.list”中包含带有“http://security.debian.org/”的一行是不错的主意。它会启用安全更新。

注意
Debian 安全团体将会修正 stable 档案库的安全缺陷。这些行为是十分严格可靠的。testing 档案库中的缺陷，不一定会被 Debian 测试安全团体修正。由于一些原因，这些行为相对 stable 档案库没有那么严格, 您可能需要等待已修正的 unstable 软件包移植到 testing。unstable 档案库的缺陷，交由各个维护者修改。经常维护的 unstable 软件包通常处于相当好的状况，因为它利用了上流最新的安全修正。有关 Debian 怎样处理安全缺陷，请参见 Debian 安全常问问题 。

区域	软件包数量	软件包组件标准
main	59430	遵从 Debian 自由软件指导方针 (DFSG)，并且不依赖于 non-free
contrib	343	遵从 Debian 自由软件指导方针 (DFSG)，但依赖于 non-free
non-free	652	不遵从 Debian 自由软件指导方针 (DFSG)

Table 2.3: Debian 归档区域 (area) 列表

上述软件包的数量是 amd64 架构的。main 区域提供 Debian 系统 (参见第 2.1.5 节)。

通过把你的浏览器指向档案库 URL，这些 URL 在 dists 或 pool 之后是各不相同的，Debian 档案库能够被有规划的组织。

发行版可以用套件或代号来指定。发行版在许多文档中也被当做是套件的同义词。套件和代号的关系总结如下。

时间	**suite = stable**	**suite = testing**	**suite = unstable**
在 buster 发布后	codename = buster	codename = bullseye	codename = sid
在 bullseye 发布后	codename = bullseye	codename = bookworm	codename = sid

Table 2.4: 套件和代号的关系

代号的历史参见 Debian FAQ: 6.2.1 以前用过哪些代号名？

在较严格的 Debian 档案术语，“部分 section”这一词特指按应用领域来分类的软件包类别。(但是，主要部分 (”main section”) 这一词有时会用来描述 Debian 档案区中，名为”main 主要” 的区域。)

Debian 开发者 (DD) 每次上传软件包到 unstable 档案库 (通过 incoming 处理)，都必须确保上传的软件包与最新的 unstable 档案库中的最新软件包兼容。

如果 DD 故意打破重要的库升级等的这种兼容性，这通常会在 Debian 开发者邮件列表等进行公告。

在 Debian 档案库维护脚本将软件包从 unstable 档案库移动到 testing 档案库前，档案库维护脚本不仅检查时间 (约 10 天) 和软件包的 RC bug 报告的状态，还尝试确保它们可以和最新的 testing 档案库中的软件兼容。这个过程使得 testing 档案库非常正确可用。

通过由发布团队领导的逐步冻结档案库的过程，并进行一些手动干预，使 testing 档案库完全一致，无缺陷。然后，将旧的 testing 档案库的代码名称分配给新的 stable 档案库，并为新的 testing 档案库创建新的代码名称。新的 testing 档案库最初的内容和新发布的 stable 档案库的内容完全相同。

unstable 和 testing 档案库都可能会遭受由以下几个因素导致的临时的小故障。

- 损坏的软件包被上传到档案库 (多见于 unstable)
- 延迟接受新的软件包到档案库 (多见于 unstable)
- 档案库时间同步问题 (testing 和 unstable)
- 手动干预档案库，例如移除软件包 (多见于 testing) 等。

因此，如果你决定使用这些档案库，你应该能够修复或忍受这些类型的小故障。

小心
在新的 stable 版本发布后的几个月，大多数桌面用户应该使用带有安全更新的 stable 档案库，即使他们通常使用 unstable 或 testing 档案库。在这个过渡期中，unstable 和 testing 档案库不适合大多数人。你使用 unstable 档案库的系统是很难保持良好的工作状态的，因为它会遭受核心软件包的大量升级狂潮。testing 档案库不大有用，因为它包含有和没有安全支持的 stable 档案库相同的内容（Debian testing 安全公告 2008-12）。一个月左右的时间后，如果你仔细点的话，unstable 档案库或许可以使用。

提示
跟踪 testing 档案库时，由一个已移除的软件包引起的问题通常可以安装 unstable 档案库中相同的软件包（已修复 bug）来解决。

档案库的定义参见 Debian 政策文档。

- 部分
- "优先级"
- "基本系统"
- "极重要的软件包"

2.1.5 Debian 是 100% 的自由软件

Debian 是 100% 的自由软件，因为：

- Debian 默认只安装自由软件，这尊重了用户的自由。
- Debian 在 main 中只提供自由软件。
- Debian 建议只运行来自 main 的自由软件。
- 在 main 中的软件包没有依赖于在 non-free 或 contrib 中的软件包。

有人想知道下列的两个事实是否互相矛盾。

- "Debian 将始终是 100% 的自由软件"。（Debian 社群契约中的第一条）
- Debian 服务器上有一些 non-free 和 contrib 软件包。

因为下列原因，这并不矛盾。

- Debian 系统具有 100% 的自由，并且它的软件包位于 Debian 服务器的 main 区域。
- Debian 系统之外的软件包位于 Debian 服务器的 non-free 和 contrib 区域。

在 Debian 社群契约的第 4 条和第 5 条对这进行了明确的解释：

- 我们将优先考虑我们的用户及自由软件
  - 我们由我们的用户及自由软件社群的需要所导向。我们将优先考虑他们的利益。我们将在多种计算环境中支持我们的用户的操作需要。我们不反对在 Debian 系统上使用非自由软件，我们也不会尝试向创建和使用这部分软件的用户索取费用。我们允许他人，在没有我们的资金的参与下，制造包括 Debian 以及商业软件的增值套件。为了达成这些目标，我们将提供集成的、高质量的、100% 自由的软件，而不附加任何可能阻止在这些方面使用的法律限制。

- 哪些作品不符合我们的自由软件规范
  - 我们明了，某些我们的用户需要使用不符合 Debian 自由软件指导方针的作品。我们为这些作品，在我们的档案库中留出了“contrib”以及“non-free”目录。在这些目录下的软件包，并不属于 Debian 系统尽管它们已被配置成可以在 Debian 下使用。我们鼓励光盘制造商阅读这些目录下的软件的许可证，以判断他们是否可以在光盘中发行这些软件。所以，尽管非自由软件并非 Debian 系统的一部分，我们仍支持它们的使用，并且我们为非自由软件提供了公共资源 (诸如我们的缺陷跟踪系统以及邮件列表)。

用户应该了解使用 non-free 和 contrib 中的软件包所需要冒的风险：

- 使用类似的软件包会失去自由
- 失去 Debian 对软件包的支持（这些软件包无法访问源代码，Debian 不能进行完全的支持。）
- 污染你 100% 自由的 Debian 系统

Debian 自由软件指导方针为 Debian 设立了自由软件标准。Debian 对软件包中的软件做了最广泛的解释，包含文档、固件、图标和图形数据。这使得 Debian 的自由软件标准非常严格。

为了满足 main 严格的自由软件标准，Debian 曾经提供了 去掉 Mozilla 商标 的软件包（例如 Firefox 、Thunderbird 和 Seamonkey），它们移除了 logo 和一些图形数据；并将它们分别用 Iceweasel 、Icedove 和 Iceape 替代。在这些问题被解决后，这些软件包随着 Debian Stretch（Debian 9）的发布恢复了其原本的名称。

典型的 non-free 和 contrib 软件包包含了下列类型的自由分发的软件包：

- 在GNU Free Documentation License下的文档包，包含不变的部分，比如 GCC 和 Make 的。（大多数都可以在 non-free/doc 找到.)
- 包含没有源代码的二进制数据的固件软件包,例如在第 9.9.6 节中列出的 non-free 软件包。(多见于 non-free/kernel 部分。)
- 游戏和字体软件包，对商业使用和/或内容修改进行了限制。

请注意，non-free 和 contrib 软件包的数量少于 main 软件包的 2%。允许访问 non-free 和 contrib 并不会模糊软件包的来源。使用 aptitude(8) 的全屏交互式界面可以提供完全的可见性和完全的控制，可以让你决定安装来自某个部分的软件包，来使你的系统保持自由。

2.1.6 软件包依赖关系

Debian 系统通过其控制文件字段中的版本化二进制依赖声明机制来提供一致的二进制软件包集合。下面有一些它们的简单定义。

- “依赖”
  - 绝对的依赖，所有在这里列出的软件包都必须同时或提前安装。
- ” 预依赖”
  - 类似于 Depends，但列出的软件包必须提前完成安装。
- ” 推荐”
  - 这里表示一个强，但不是绝对的依赖关系。大多数用户不会想要这个包，除非在这里列出的所有包都已经安装。
- ” 建议”
  - 较弱的依赖。这个软件包的大多数用户可能会从安装所列的软件包中受益，但没有它们也可以有适当的功能。
- ” 增强”

  - 这里表明一个像推荐的弱依赖关系，不装也没关系。
- ”破损”
  - 表明一个软件包不兼容一些版本规范。一般的解决方法就是升级列出的所有软件包。
- ”冲突”
  - 这表明了绝对的不兼容。为了安装这个软件包必须移除所有列出的软件包。
- ”替代”
  - 这表明这个文件安装的文件会替代所列的软件包的文件。
- ”提供”
  - 表明这个软件包会提供所列的软件包所有的文件和功能。

注意
请注意，同时将“Provides”、“Conflicts”和“Replaces”定义到一个虚拟的软件包是一个明智的配置。这确保了在任何一个时间只能安装一个提供该虚拟包的真正软件包。

包含源代码依赖关系的官方定义位于 the Policy Manual: Chapter 7 - Declaring relationships between packages。

2.1.7 包管理的事件流

这是 APT 提供的软件包管理的简单事件流摘要。

- 更新（“`apt update`”、“`aptitude update`”或“`apt-get update`”）：
  1. 从远程档案库获取档案库元数据
  2. 重建和更新 APT 使用的本地元数据
- 升级（“`apt upgrade`”和“`apt full-upgrade`”,或“`aptitude safe-upgrade`”和“`aptitude full-upgrade`”,或“`apt-get upgrade`”和“`apt-get dist-upgrade`”）：
  1. 选择候选版本，它所安装的软件包通常都是最新的可用版本（例外参见第 2.7.3 节）
  2. 解决软件包依赖关系
  3. 如果候选版本与已安装的版本不同，会从远程档案库获取所选择的二进制软件包
  4. 解包所获取的二进制软件包
  5. 运行 **preinst** 脚本
  6. 安装二进制文件
  7. 运行 **postinst** 脚本
- 安装（“`apt install ...`”、“`aptitude install ...`”或者”`apt-get install …`”）：
  1. 选择命令行中列出的包
  2. 解决软件包依赖关系
  3. 从远程服务器获取已选二进制包
  4. 解包所获取的二进制软件包
  5. 运行 **preinst** 脚本
  6. 安装二进制文件

7. 运行 **postinst** 脚本

- 移除（"apt remove …"，"aptitude remove …" 或 "apt-get remove …"）：

1. 选择命令行中列出的包
2. 解决软件包依赖关系
3. 运行 **prerm** 脚本
4. 移除已安装的文件，除了配置文件
5. 运行 **postrm** 脚本

- 清除（"apt purge"，"aptitude purge …" 或 "apt-get purge …"）：

1. 选择命令行中列出的包
2. 解决软件包依赖关系
3. 运行 **prerm** 脚本
4. 移除已安装的文件，包含配置文件
5. 运行 **postrm** 脚本

这里，为了大局，我特意省略了技术细节。

2.1.8 对包管理问题的第一个回应

你应该阅读优良的官方文档。第一个阅读的文档是 Debian 特定的"/usr/share/doc/<package_name>/README.Debian"，同时也应该查询"/usr/share/doc/<package_name>/"中的其它文档。如果你设置 shell 为第 1.4.2 节，输入下列命令。

```
$ cd <package_name>
$ pager README.Debian
$ mc
```

你可能需要安装以"-doc"后缀命名的对应文档软件包来获取详细的信息。

如果你在使用一个特定的软件包时出现了问题，一定要首先检查 Debian bug 跟踪系统（BTS）网站。

网站	命令
Debian bug 跟踪系统（BTS）的主页	sensible-browser "http://bugs.debian.org/"
软件包名称已知的 bug 报告	sensible-browser "http://bugs.debian.org/<package_name>"
bug 编号已知的 bug 报告	sensible-browser "http://bugs.debian.org/<bug_number>"

Table 2.5: 解决特定软件包问题的主要网站

使用 Google 搜索，在关键字中包含"site:debian.org"，"site:wiki.debian.org"，"site:lists.debian.org"等等。

当你要发送一份 bug 报告时，请使用 reportbug(1) 命令。

2.2 基础软件包管理操作

在 Debian 系统中有许多基于 APT 的软件包管理工具可以在 Debian 系统上进行基于仓库的软件包管理操作。在这里，我们将介绍 3 种基本的软件包管理工具：apt，apt-get / apt-cache 和 aptitude。

对于涉及软件包安装或更新软件包元数据的软件包管理操作，你必须有 root 权限。

2.2.1 apt vs. apt-get / apt-cache vs. aptitude

尽管 aptitude 是作者主要使用的一个非常好的可互动工具，但你应该知道下列警示：

- 不建议在新版本发布后在 stable Debian 系统上使用 aptitude 命令来进行跨版本的系统升级。
  - 建议使用"apt full-upgrade" 或“apt-get dist-upgrade”来进行这个操作。参见 Bug #411280。
- aptitude 命令有时候会为了 testing 或 unstable Debian 系统升级清除大量软件包。
  - 这个情况吓坏了许多的系统管理员。请不要惊慌。
  - 这似乎大多数是由元软件包的依赖或推荐的软件包版本偏差造成的，例如 gnome-core。
  - 要解决这个问题，可以在 aptitude 命令菜单中选择“取消待执行的动作”，退出 aptitude，并使用“apt full-upgrade”。

apt-get 和 apt-cache 是最基础的基于 APT 的软件包管理工具。

- apt-get 和 apt-cache 只提供命令行用户界面。
- apt-get 是进行跨版本的主系统升级等操作的最合适工具。
- apt-get 提供了一个强大的软件包依赖解析器。
- apt-get 对硬件资源的要求不高。它消耗更少的内存并且运行速度更快。
- apt-cache 提供了一个 标准的正则表达式来搜索软件包名称和描述。
- apt-get 和 apt-cache 可以使用 /etc/apt/preferences 来管理软件包的多个版本，但这非常繁琐。

apt 命令是一个用于软件包管理的高级命令行界面。它基本上是 apt-get、apt-cache 和类似命令的一个封装，被设计为针对终端用户交互的界面，它默认启用了某些适合交互式使用的选项。

- apt 工具在用户使用 apt install 安装软件包时提供了一个友好的进度条。
- 在成功安装下载的软件包后，apt 将默认删除缓存的 .deb 软件包。

提示
建议用户使用新的 apt(8) 命令用于 交互式的使用场景，而在 shell 脚本中使用 apt-get(8) 和 apt-cache(8) 命令。

---

aptitude 命令是最通用的基于 APT 的软件包管理工具。

- aptitude 提供了一个全屏的交互式文本用户界面。
- aptitude 同样也提供了一个命令用户界面。
- aptitude 是用于日常软件包管理（例如检查已安装的软件包和搜索可用的软件包）的最合适工具。
- aptitude 对硬件资源的要求更高。它消耗更多的内存并且运行速度更慢。
- aptitude 提供一个增强的正则表达式来搜索所有的软件包元数据。
- aptitude 可以管理软件包的多个版本，并且不使用 /etc/apt/preferences，这会十分直观。

apt 语法	**aptitude** 语法	**apt-get** / **apt-cache** 语法	说明
apt update	aptitude update	apt-get update	更新软件包档案库元数据
apt install foo	aptitude install foo	apt-get install foo	安装“foo”软件包的候选版本以及它的依赖
apt upgrade	aptitude safe-upgrade	apt-get upgrade	安装已安装的软件包的候选版本并且不移除任何其它的软件包
apt full-upgrade	aptitude full-upgrade	apt-get dist-upgrade	安装已安装的软件包的候选版本，并且需要的话会移除其它的软件包
apt remove foo	aptitude remove foo	apt-get remove foo	移除“foo”软件包，但留下配置文件
apt autoremove	N/A	apt-get autoremove	移除不再需要的自动安装的软件包
apt purge foo	aptitude purge foo	apt-get purge foo	清除“foo”软件包的配置文件
apt clean	aptitude clean	apt-get clean	完全清除本地仓库的软件包检索文件
apt autoclean	aptitude autoclean	apt-get autoclean	清除本地仓库中过时软件包的软件包检索文件
apt show foo	aptitude show foo	apt-cache show foo	显示“foo”软件包的详细信息
apt search < 正则表达式 >	aptitude search <regex>	apt-cache search <regex>	搜索匹配 <regex> 的软件包
N/A	aptitude why <regex>	N/A	解释匹配 <regex> 的软件包必须被安装的原因
N/A	aptitude why-not <regex>	N/A	解释匹配 <regex> 的软件包不必安装的原因
N/A	aptitude search '~i!~M'	apt-mark showmanual	列出手动安装的软件包

Table 2.6: 使用 apt(8), aptitude(8) 和 apt-get(8) / apt-cache(8) 的命令行基本软件包管理操作

2.2.2 命令行中的基础软件包管理操作

下面是使用 apt(8), aptitude(8) 和 apt-get(8) / apt-cache(8) 的命令行基本软件包管理操作。

注意
虽然 aptitude 命令提供了丰富的功能，例如增强的软件包解析器，但它的复杂程度导致了（或可能导致）一些退步，例如 Bug #411123、Bug #514930 及 Bug #570377。如有疑问，请使用 apt,apt-get 和 apt-cache 命令来替代 aptitude 命令。

注意
因为在 lenny 版本之后的 Debian 系统中，apt、apt-get 和 aptitude 会共享自动安装的软件包的状态（参见第 2.5.5 节），因此你可以混合使用这些工具而不会出现严重的麻烦（参见 Bug #594490）。

“aptitude why <regex>”可以通过“aptitude -v why <regex>”列出更多的信息。类似的信息可以通过”apt rdepends <package>”或“apt-cache rdepends <package>”获取。

当 aptitude 命令在命令行模式下启动后遇到了一些问题（例如软件包冲突），你可以在之后的提示中按下“e”键切换到全屏的交互模式。

你可以在“aptitude”后面使用的命令选项。

命令选项	说明
-s	模拟命令的结果
-d	仅下载，不进行安装/更新
-D	在自动安装和删除前，显示简要的说明

Table 2.7: aptitude(8) 中重要的命令选项

更多内容参见 aptitude(8) 和位于“/usr/share/doc/aptitude/README”的“aptitude 用户手册”。

提示
dselect 软件包依旧可用，并且曾是之前发布的版本中首选的全屏交互式软件包管理工具。

2.2.3 aptitude 的交互式使用

要使用交互式的软件包管理，你可以像下面那样以交互模式启动 aptitude。

```
$ sudo aptitude -u
Password:
```

这将更新档案库信息的本地副本，并以菜单的形式全屏显示软件包列表。aptitude 将它的配置放在“~/.aptitude/config”。

提示
如果你想用 root 的配置而非使用者的，可以在上面的例子中使用“sudo -H aptitude …”代替“sudo aptitude …”。

提示
当 aptitude 以交互模式启动时，会自动设置待执行的动作。如果您不喜欢，您可以通过菜单：“动作” → “取消待执行的动作”来取消它。

2.2.4 aptitude 的按键绑定

在全屏模式下浏览软件包状态和设置动作的按键如下。

快捷键	键绑定功能
F10 或 Ctrl-t	菜单
?	显示按键帮助（更加完整的清单）
F10 → 帮助 → 用户手册	显示用户手册
u	更新软件包档案库信息
+	标记该软件包以便升级或安装
-	标记该软件包以便移除（保留配置文件）
_	标记该软件包以便清除（移除配置文件）
=	将软件包设为保持状态
U	标记所有可升级包（动作如同 **full-upgrade**）
g	开始 下载并 安装所选择包
q	退出该界面并保存变更
x	退出该界面并清除变更
Enter	查看软件包的信息
C	查看软件包的变更记录
l	变更软件包的显示限制
/	搜寻匹配的第一个软件包
\	重复上一个搜索

Table 2.8: aptitude 的按键绑定

可以通过命令行指定文件名称，也可以通过按“l”或“/”之后在菜单提示下输入下列所述的 aptitude 正则表达式。aptitude 正则表达式可以使用“~n”开头后接软件包名称的字符串来精确匹配软件包名称。

提示
你需要在可视化界面中按下“U”键让所有的已安装软件包升级到可用版本。否则只有选中的软件包和一些与之有依赖关系的软件包才能被升级到可用版本。

2.2.5 aptitude 软件包视图

aptitude(8) 全屏交互模式下，软件包列表里的软件包会像下面的例子那样显示。

```
idA   libsmbclient                              -2220kB 3.0.25a-1  3.0.25a-2
```

该行的从左到右的含义如下。

- “状态”标签（第一个字母）
- “动作”标签（第二个字母）
- “自动”标签（第三个字母）
- 软件包名称
- 该“动作”对磁盘空间的变化
- 软件包当前版本
- 软件包可用版本

提示
您可以在帮助菜单中找到完整的标签列表，按“?”即可在帮助菜单底部显示。

可用版本的选择是依据当前的本地首选项（参见 apt_preferences(5) 和第 2.7.3 节）。

软件包视图的几种类型都可以在“视图 ”菜单下找到。

视图	状态	视图描述
软件包视图	良好	参见表 2.10 (默认)
检查推荐结果	良好	列出推荐安装但还没有安装的软件包
平面软件包列表	良好	不分类地列出软件包 (用于正则表达式)
Debtags 浏览器	非常有用	列出由 debtags 进行分类的软件包
分类浏览器	已弃用	列出按照类别分类的软件包（用 Debtags 浏览器替代）

Table 2.9: aptitude 视图

注意
请帮助我们改进用 debtags 标记的软件包！

标准“软件包视图”分类软件包的方法与带有一些额外功能的 dselect 有点像。

分类	视图描述
可升级软件包	按照 section → area → 软件包的顺序显示列出软件包
新软件包	同上
已安装软件包	同上
未安装软件包	同上
过期的和在本地创建的软件包	同上
虚拟软件包	列出同样功能的软件包
软件集	列出一个特定任务所需的不同功能的软件包

Table 2.10: 标准软件包视图的分类

提示
软件集视图可以用来为你的任务选出最佳的软件包。

2.2.6 aptitude 搜索方式选项

aptitude 提供了几个可以使用正则表达式来搜索软件包的选项。

- shell 命令行:
  - “aptitude search '<aptitude_regex>'”列出安装状态、软件包名称和匹配软件包的剪短描述
  - “aptitude show '<package_name>'”列出软件包的详细描述
- 全屏交互模式:
  - "l" 可以限制匹配软件包的视图
  - "/" 搜索匹配的软件包

– ”\” 向后搜索匹配的软件包
– ”n” 查找下一个
– ”N” 查找上一个

提示
字符串 <package_name> 被看作软件包名称的精确字符串匹配，除非它是以”~” 开头的正则表达式。

2.2.7 aptitude 正则表达式

aptitude 正则表达式是类 mutt 的拓展 **ERE**（参见第 1.6.2 节），aptitude 具体的特殊匹配规则扩展如下。

- 正则表达式使用的是 **ERE**，就跟 egrep(1)、awk(1) 和 perl(1) 这些典型的类 Unix 文本工具中所使用的“^”、“.*”、“$”等是相同的。
- 依赖关系 <type> 是一种特定的软件包相互关系（depends、predepends、recommends、suggests、conflicts、replaces、provides）。
- 默认的 <type> 依赖关系是“depends”。

提示
当 <regex_pattern> 为空字符串时，请立即在命令后面添加”~T”。

下面是一些快捷方式。

- ”~P<term>” == ”~Dprovides:<term>”
- ”~C<term>” == ”~Dconflicts:<term>”
- ”…~W term” == ”(…|term)”

用户熟悉 mutt 的快速选择，因为 mutt 的灵感来源于表达式语法。参见“用户手册”“/usr/share/doc/aptitude/README”中的“SEARCHING, LIMITING, AND EXPRESSIONS”。

注意
lenny 版本的 aptitude(8) 中，新的长格式语法，例如“?broken”，在正则表达式中可以用来等效为它旧的短格式“~b”。现在空格字符“ ”被认为是除了波浪字符“~”外的另一个正则表达式终止字符。新的长格式语法参见“用户手册”。

2.2.8 aptitude 的依赖解决

如果通过菜单“F10 → 选项 → 首选项 → 正在处理依赖关系”进行相应的设置，则在 aptitude 中选择一个软件包时，不仅会将其“Depends:”列表中的软件包选上，“Recommends:”列表中的软件包也会被选上。在 aptitude 下，这些自动安装的软件包在不再需要时会自动移除。

aptitude 命令中控制“自动安装”行为的标签也可以通过 apt 软件包中的 apt-mark(8) 命令来设置。

2.2.9 软件包活动日志

你可以在日志文件里查询到软件包活动历史。

事实上，很难从这些日志上快速获得有用的信息。较简便的方法参见第 9.2.10 节。

扩展匹配规则描述	正则表达式
匹配软件包名称	`~n<regex_name>`
匹配描述	`~d<regex_description>`
匹配软件集名称	`~t<regex_task>`
匹配 debtag	`~G<regex_debtag>`
匹配维护者	`~m<regex_maintainer>`
匹配软件包的 section	`~s<regex_section>`
匹配软件包版本	`~V<regex_version>`
匹配档案库	`~A{buster,bullseye,sid}`
匹配来源	`~O{debian,…}`
匹配优先级	`~p{extra,important,optional,required,standard}`
匹配必要的软件包	`~E`
匹配虚拟软件包	`~v`
匹配新的软件包	`~N`
匹配待执行的动作	`~a{install,upgrade,downgrade,remove,purge,hold,keep}`
匹配已安装软件包	`~i`
匹配带有 **A** 标签的已安装软件包（自动安装的软件包）	`~M`
匹配不带有 **A** 标签的已安装软件包（管理员选择的软件包）	`~i!~M`
匹配已安装并且是可升级的软件包	`~U`
匹配已删除但未清除的软件包	`~c`
匹配已移除，已清除或可移除的软件包	`~g`
匹配破坏依赖关系的软件包	`~b`
匹配破坏 <type> 依赖关系的软件包	`~B< 类型 >`
匹配 <pattern> 软件包的 <type> 依赖关系	`~D[< 类型 >:]< 模式 >`
匹配 <pattern> 软件包破坏的 <type> 依赖关系	`~DB[< 类型 >:]< 模式 >`
匹配依赖于 <pattern> 软件包的 <type> 依赖的软件包	`~R[< 类型 >:]< 模式 >`
匹配依赖于 <pattern> 软件包破坏的 <type> 依赖的软件包	`~RB[< 类型 >:]< 模式 >`
匹配其它已安装软件包所依赖的软件包	`~R~i`
匹配没有被其它已安装软件包所依赖的软件包	`!~R~i`
匹配其它已安装软件包所依赖或建议安装的软件包	`~R~i|~R 推荐:~i`
匹配 <pattern> 过滤版本之后的软件包	`~S 过滤 < 模式 >`
匹配所有软件包（真）	`~T`
不匹配软件包（假）	`~F`

Table 2.11: aptitude 正则表达式

文件	内容
`/var/log/dpkg.log`	dpkg 级的软件包活动日志
`/var/log/apt/term.log`	通用 APT 活动日志
`/var/log/aptitude`	aptitude 命令活动日志

Table 2.12: 软件包活动日志文件

2.3 aptitude 操作范例

下面是一些 aptitude(8) 的操作范例。

2.3.1 通过正则表达式匹配软件包名称来列出软件包

下面的命令列出了通过正则表达式匹配软件包名称来列出软件包。

```
$ aptitude search '~n(pam|nss).*ldap'
p libnss-ldap - 使用 LDAP 作为名称服务的 NSS 模块
p libpam-ldap - 允许 LDAP 接口的插入式验证模块
```

这种方式查找精确的软件包名称很方便。

2.3.2 使用正则表达式匹配浏览

在" 新扁平软件包列表" 中使用"l" 提示查看，正则表达式"~dipv6" 可以限制性地匹配软件描述，并交互式地展示信息。

2.3.3 完整地清理已删除软件包

您能清除所有已移除软件包的剩余配置文件。

检查以下命令的结果。

```
# aptitude search '~c'
```

如果您确认所列出的软件包应当被完整删除，请运行以下命令。

```
# aptitude purge '~c'
```

您可能想要在交互模式中做类似的操作进行细粒度的控制。

在" 新软件包视图" 使用"l" 提示并输入正则匹配式"~c" ，这将仅匹配软件包，比如，" 移除但不清空配置" 。所有符合匹配的软件包可以在顶层标题上使用"[" 显示。

当您在顶层标题如“未安装的包”中输入“_”，当前标题下的软件包只有匹配正则式才会被清除。您还可以使用“=”来交互式地排除软件包以避免删除它们。

这种技术方便易用且适用于许多其他的命令键。

2.3.4 调整自动/手动安装状态

下面是调整软件包的自动/手动安装状态的方法（在使用非 aptitude 软件包管理器之后）。

1. 用 root 以交互模式运行 aptitude。
2. 用"u" 命令更新可用的软件包列表，"U" 命令标记所有可升级的软件包以执行升级，"f" 命令清除新软件包列表，"g" 命令执行所有可升级的软件包以执行升级。
3. 按下“l”，并输入“~i(~R~i|~Rrecommends:~i)”来限制软件包的显示，按下“M”将“已安装软件包”的状态改为自动安装。

4. 按下“l”，并输入“~required|~pimportant|~pstandard|~E”来限制软件包的显示，按下“m”将“已安装软件包”的状态改为手动安装。
5. 按下“l”，并输入“~i!~M”来限制软件包的显示，在“已安装软件包”上按下“[”来陈列无用的软件包，按下“-”将它们移除。
6. 按下“l”，并输入“~i”来限制软件包的显示，之后在“软件集”上按下“m”将那些软件包标记为手动安装。
7. 退出 aptitude。
8. 用 root 用户执行”apt-get -s autoremove|less” 命令，来查看有那些软件包是不再需要的。
9. 在交互模式下重启 aptitude 程序，用“m”命令标记所需要的软件包。
10. 用 root 用户重新执行”apt-get -s autoremove|less” 这个命令来复查移除的包中是不是只含有自己所希望移除的软件包。
11. 用 root 用户执行”apt-get autoremove|less” 命令来自动移除不再需要的软件包。

在你所需要执行的”Tasks” 上，运行”m” 命令是一个可选的操作，目的就是为了防止大量软件包被卸载的情况出现。

2.3.5 全面的系统升级

注意
当你迁移到新的发行版的时候，虽然正如下面所描述的那样，Debian 是可升级的，但是你还是应该考虑纯净的安装新的系统。这给了你机会去移除废弃的软件包同时还可以接触到最新软件包的完美集合体。当然，在做迁移之前，你也应该对你的系统做完整的备份，并把它移到安全的地方去 (查看第 10.2 节)。“我” 也建议用不同的分区做另外一个启动项，来实现平稳的升级。

你可以通过改变”/etc/apt/sources.list” 的内容使之指向新的发行版所在地址的方法来进行系统的全面升级，然后运行”apt update; apt dist-upgrade” 命令。

从 stable 升级到 testing 或者 unstable，你应该用”bullseye” 或者”sid” 替换”/etc/apt/sources.list” 文件里的”buster”，参考第 2.1.4 节。

事实上，由于一些软件包版本变迁的问题，你可能会遇到一些困难，主要是由于软件包的依赖问题。升级之后的差异越大，你越有可能遇到麻烦。在新版本发行后，系统从旧的 stable 过渡到新的 stable，你可以查看 Release Notes 然后按照里面的步骤去做，来尽可能的减少麻烦。

在它正式发布之前，你决定要从先前的 stable 迁移到将要发布的 testing，这里没有 Release Notes 可以帮到你。在前一个 stable 发布以后， stable 发行版跟将要发布的 testing 发行版之间的差异可能变得相当大同时也使得升级系统变得更加的复杂。

在全面升级系统的时候，你应该谨慎的操作，同时你也应该从邮件列表中获取最新的资料然后根据你的常识作出正确的判断。

1. 查看先前的“发行说明”。
2. 备份整个系统 (尤其是数据和配置信息)。
3. 当 bootloader 坏了的时候，手边应该有可以引导电脑启动的存储介质。
4. 事先通知系统上的用户。
5. 用 script(1) 记录升级的过程。
6. 用“unmarkauto”命令来保留你想要的软件包，例如”aptitude unmarkauto vim” 这个命令是用来防止移除 vim 这个软件的。
7. 为了减少软件包之间可能会发生的冲突，应该尽量减少要安装的软件包的数目，例如，移除桌面环境这个软件包。

8. 移除”/etc/apt/preferences” 文件（禁用 apt-pinning）。
9. 试着一步步的升级：oldstable → stable → testing → unstable.
10. 升级”/etc/apt/sources.list” 文件，使其指向新的档案库然后运行”aptitude update” 命令。
11. 可选的安装选项，首先是新的 **core packages**，例如“aptitude install perl”。
12. 运行”apt-get -s dist-upgrade” 命令来评估升级造成的影响。
13. 最后运行”apt-get dist-upgrade” 命令。

小心
在 stable 版本升级的时候，跳过主要的 Debian 发行版是不明智的。

小心
在先前的“发行手册”里，GCC, Linux Kernel, initrd-tools, Glibc, Perl, APT tool chain 等等，有一些关于系统全面升级的重要注意事项。

关于 unstable 版本的日常升级，查看第 2.4.3 节。

2.4 高级软件包管理操作

2.4.1 命令行中的高级软件包管理操作

下面列出了一些其它的软件包管理操作，这些操作对于 aptitude 过于高级或缺失所需的功能。

注意
对于一个支持多架构的软件包，你可能需要为一些命令指定架构名称。例如，使用“dpkg -L libglib2.0-0:amd64”来列出 amd64 架构的 libglib2.0-0 软件包的内容。

小心
系统管理员应该小心使用低级的软件包工具（例如“dpkg -i …”和“debi …”），它们不会自动处理所需的软件包依赖。dpkg 的命令行选项“--force-all”和类似的选项（参见 dpkg(1)）只适用于高手。没有完全理解它们的效果却使用它们会破坏你的整个系统。

请注意以下几点。

- 所有的系统配置和安装命令都需要以 root 运行。
- 不同于使用正则表达式的 aptitude（参见第 1.6.2 节），其它的软件包管理命令使用类似于 shell glob 的通配符（参见第 1.5.6 节）。
- apt-file(1) 由 apt-file 软件包提供，并且需要先运行“apt-file update”。
- configure-debian(8) 由 configure-debian 软件包提供，它运行 dpkg-reconfigure(8) 作为后端。
- dpkg-reconfigure(8) 使用 debconf(1) 作为后端来运行软件包脚本。

命令	操作
COLUMNS=120 dpkg -l <package_name_pattern>	列出已安装软件包的列表用于错误报告
dpkg -L <package_name>	显示一个已安装软件包的内容
dpkg -L <package_name> | egrep '/usr/share/man/man.*/.+'	列出一个已安装软件包的 man 手册页
dpkg -S <file_name_pattern>	列出匹配文件名的已安装软件包
apt-file search <file_name_pattern>	列出档案库中匹配文件名的软件包
apt-file list <package_name_pattern>	列出档案库中匹配的软件包的内容
dpkg-reconfigure <package_name>	重新配置软件包
dpkg-reconfigure -p=low <package_name>	通过最详细的方式来重新配置软件包
configure-debian	以全屏菜单的形式重新配置软件包
dpkg --audit	部分安装软件包的审计系统
dpkg --configure -a	配置所有部分安装的软件包
apt-cache policy <binary_package_name>	显示一个二进制软件包的可用版本、优先级和档案库信息
apt-cache madison <package_name>	显示一个软件包的可用版本和档案库信息
apt-cache showsrc <binary_package_name>	显示一个二进制软件包的源代码软件包信息
apt-get build-dep <package_name>	安装构建软件包所需要的软件包
aptitude build-dep <package_name>	安装构建软件包所需要的软件包
apt-get source <package_name>	（从标准档案库）下载源代码
dget <URL for dsc file>	（从其它档案库）下载源代码软件包
dpkg-source -x <package_name>_<version>-<debian_version>.dsc	从源代码软件包集合（“*.orig.tar.gz”和 “*.debian.tar.gz”/”*.diff.gz”）中构建代码树
debuild binary	从本地的源代码树中构建软件包
make-kpkg kernel_image	从内核源代码树中构建一个内核软件包
make-kpkg --initrd kernel_image	从启用了 initramfs 的内核代码树中构建一个内核软件包
dpkg -i <package_name>_<version>-<debian_version>_<arch>.deb	安装一个本地的软件包到系统中
apt install /path/to/<package_filename>.deb	安装本地软件包到系统中，同时尝试自动解决依赖
debi <package_name>_<version>-<debian_version>_<arch>.dsc	安装本地软件包到系统中
dpkg --get-selections '*' >selection.txt	保存 dpkg 级别的软件包选择状态信息
dpkg --set-selections <selection.txt	使用 dpkg 设置软件包选择状态
echo <package_name> hold | dpkg --set-selections	使用 dpkg 将一个软件包的包选择状态设置为 **hold**（相当于”aptitude hold < 包名 >”）

Table 2.13: 高级软件包管理操作

- ”apt-get build-dep”、”apt-get source”和”apt-cache showsrc”命令需要”/etc/apt/sources.list”中存在”deb-src” 条目。
- dget(1) 、 debuild(1) 和 debi(1) 需要 devscripts 软件包。
- 参见第 2.7.13 节里使用”apt-get source” 的打包（重打包）过程。
- make-kpkg 命令需要 kernel-package 软件包（参见第 9.9 节）。
- 通用打包参见第 12.11 节.

2.4.2 验证安装的软件包文件

已经安装 debsums 软件包的，能使用 debsums(1) 命令通过”/var/lib/dpkg/info/*.md5sums” 文件中的 MD5sum 值，验证已安装的文件。参见第 10.3.5 节来获得 MD5sum 是怎样工作的信息。

注意
因为 MD5sum 数据库可能被侵入者篡改，debsums(1) 作为安全工具使用有限。这种工具用于校验管理者造成的本地修改或媒体错误造成的损坏是很不错的。

2.4.3 预防软件包故障

许多用户更想使用 **unstable**，因为它有新的功能和软件包。但这会使得系统更容易遇到严重的软件包 bug 。

安装软件包 apt-list bugs 可以避免您的系统遭遇严重 bugs，在通过 APT 系统升级时，它会自动检查 Debian BTS 里的严重 bug。

安装 apt-listchanges 软件包，在使用 APT 系统升级时它会在“NEWS.Debian”中提供重要新闻。

2.4.4 搜索软件包元数据

尽管近来浏览 Debian 网站 https://packages.debian.org/ 是搜索软件包元数据更加简单的方法，但我们依旧来看看更传统的方法。

grep-dctrl(1) 、 grep-status(1) 和 grep-available(1) 命令被用来搜索具有 Debian 软件包控制文件格式的任何文件。

“dpkg -S <file_name_pattern>”被用来搜索由 dpkg 安装的软件包中包含匹配文件的。但它会忽略维护者的脚本创建的文件。

如果你需要对 dpkg 元数据进行更复杂的搜索，你需要在“/var/lib/dpkg/info/”目录下运行“grep -e regex_pattern *”命令。这会使你在软件包脚本和安装查询文本中搜索提及的单词。

如果你想递归查找软件包依赖，你应该使用 apt-rdepends(8) 。

2.5 Debian 软件包内部管理

让我们来学习 Debian 软件包管理的内部工作原理。这应该能够帮助你独立解决一些软件包问题。

2.5.1 档案库元数据

每个发行版的元数据文件都保存在 Debian 镜像站的“dist/<codename>”下面，例如“http://deb.debian.org/debian 档案库的结构可以通过网络浏览器来浏览。其中有 6 种关键的元数据。

为了减少网络流量，在最近的档案库中，这些元数据存储为压缩了的差分文件。

文件	位置	内容
`Release`	发行版的顶层	档案库描述和完整性信息
`Release.gpg`	发行版的顶层	"`Release`" 文件的签名文件，使用档案库密钥签名
`Contents-<architecture>`	发行版的顶层	列出在相关架构中所有软件包的全部文件
`Release`	每个发行版/区域/架构组合的顶部	归档描述使用 apt_preferences(5) 的规则
`Packages`	每个发行版/区域/二进制架构组合的顶部	连接 `debian/control` 获得二进制包
`Sources`	每个发行版/区域/源代码组合的顶部	连接 `debian/control` 获取源代码包

Table 2.14: Debian 档案库元数据的内容

2.5.2 顶层“Release”文件及真实性

提示
顶层“`Release`”文件用于签署 **secure APT** 系统下的归档文件。

每个 Debian 档案库的网址都有一个这样的“`Release`”文件，例如“`http://deb.debian.org/debian/dists/unstable`”，内容如下。

```
Origin: Debian
Label: Debian
Suite: unstable
Codename: sid
Date: Sat, 14 May 2011 08:20:50 UTC
Valid-Until: Sat, 21 May 2011 08:20:50 UTC
Architectures: alpha amd64 armel hppa hurd-i386 i386 ia64 kfreebsd-amd64 kfreebsd-i386 mips ↩
    mipsel powerpc s390 sparc
Components: main contrib non-free
Description: Debian x.y Unstable - Not Released
MD5Sum:
 bdc8fa4b3f5e4a715dd0d56d176fc789 18876880 Contents-alpha.gz
 9469a03c94b85e010d116aeeab9614c0 19441880 Contents-amd64.gz
 3d68e206d7faa3aded660dc0996054fe 19203165 Contents-armel.gz
...
```

注意
在第 2.1.4 节里，你能够发现我使用"suite" 和"codename" 的逻辑。“发行版”被用来同时谈及"suite" 和"codename". 所有由档案库提供的归档"area" 名，会被列在"Components" 下.

顶层文件"`Release`" 的完整性，是由叫 secure apt 的加密架构来验证.

- 加密签名文件"`Release.gpg`" 是由顶层授权文件"`Release`" 和加密的 Debian 档案库公钥创建。
- 公开的 Debian 档案库公钥能够放入“`/etc/apt/trusted.gpg`”;
  - 这样的操作可以由安装最新的 `base-files` 软件包的方式完成，或者
  - 手动使用 `gpg` 或者 `apt-key` 工具 获取发布在 ftp-master.debian.org 上最新的档案库公钥。

- **secure APT** 系统验证下载的顶层文件"Release"的完整性。加密验证过程用到了"Release.gpg"文件和在"/etc/apt/tr
  里的 Debian 档案库公钥.

所有"Packages"和"Sources"文件的完整性是由在顶层"Release"文件里的 MD5sum 值来验证。所有软件包文件的完整性由"Packages"和"Sources"文件里的 MD5sum 值来验证. 参见 debsums(1) 和第 2.4.2 节.

因加密签名验证比计算 MD5sum 值消耗更多的 CPU，使用 MD5sum 值来验证每一个软件包，使用加密签名来验证顶层的"Release"文件，这种方式提供 较好安全性的同时，也有比较好的性能 (参见第 10.3 节).

2.5.3 档案库层的“Release”文件

提示
档案库层的“Release”文件将用作 apt_preferences(5) 的规则。

归档层次的"Release"文件,其全部归档位置在"/etc/apt/sources.list"中的"deb"行中指定,如以下的"http://deb
或"http://deb.debian.org/debian/dists/sid/main/binary-amd64/Release"。

```
Archive: unstable
Origin: Debian
Label: Debian
Component: main
Architecture: amd64
```

小心
对于"Archive:"章节，系列名称 ("stable"，"testing"，"unstable", ⋯) 用于 Debian archive ，而代号 ("trusty", "xenial", "artful", ⋯) 用于 Ubuntu archive。

对于部分档案库, 比如说 experimental 和 buster-backports, 它们包含的软件包不会被自动安装，这是因为有额外的行,例如在"http://deb.debian.org/debian/dists/experimental/main/binary-amd64/Release"里面有如下额外的一行。

```
Archive: experimental
Origin: Debian
Label: Debian
NotAutomatic: yes
Component: main
Architecture: amd64
```

请注意，普通的档案库没有"NotAutomatic: yes", 默认的 Pin-Priority 值是 500, 而对于有"NotAutomatic: yes"的特殊档案库, 默认的 Pin-Priority 值是 1 (参见 apt_preferences(5) 和第 2.7.3 节).

2.5.4 获取用于软件包的元数据

当使用 APT 工具时，如 aptitude, apt-get, synaptic, apt-file, auto-apt，我们需要更新包含 Debian 档案库信息元数据的本地拷贝。这些本地拷贝的文件名称，和在"/etc/apt/sources.list"文件里面的 distribution, area, architecture 相应名称一致。(参见第 2.1.4 节).

- "/var/lib/apt/lists/deb.debian.org_debian_dists_<distribution>_Release"

- ”/var/lib/apt/lists/deb.debian.org_debian_dists_<distribution>_Release.gpg”
- ”/var/lib/apt/lists/deb.debian.org_debian_dists_<distribution>_<area>_binary-<architectur
- ”/var/lib/apt/lists/deb.debian.org_debian_dists_<distribution>_<area>_source_Sources”
- ”/var/cache/apt/apt-file/deb.debian.org_debian_dists_<distribution>_Contents-<architectur (apt-file)

前 4 种类型的文件是所有相关的 APT 命令共享的，并且可以通过“apt-get update”或“aptitude update”在命令行中进行更新。如果在“/etc/apt/sources.list”中有相应的“deb”行，则“软件包”元数据会进行更新。如果在“/etc/apt/sources.list”中有相应的“deb-src”行，则“源代码”元数据会进行更新。

”Packages” 和”Sources” 的元数据文件包含有“Filename:”字段，指向二进制和源代码包文件的位置。目前，这些软件包都统一放在”pool/” 目录树下，这样可以改善跨版本发布的传输。

“软件包”元数据的本地副本可以使用 aptitude 来进行交互式的搜索。专门的搜索命令 grep-dctrl(1) 可以搜索“软件包”和“源代码”元数据的本地副本。

”Contents-<architecture>” 元数据的本地拷贝，能够被”apt-file update” 更新，它的位置和其它 4 个不同。参见 apt-file(1). (auto-apt 的”Contents-<architecture>.gz” 文件的本地拷贝默认也使用不同的位置。)

2.5.5 APT 的软件包状态

除了远程获取元数据，lenny 之后的 APT 工具还会将它在本地产生的安装状态信息保存在“/var/lib/apt/extended_sta 中，APT 会使用它们来追踪自动安装的所有软件包。

2.5.6 aptitude 的软件包状态

除了远程获取元数据，aptitude 命令还会将它在本地产生的安装状态信息保存在“/var/lib/aptitude/pkgstates” 中，这些信息只能被 aptitude 使用。

2.5.7 获取的软件包的本地副本

所有通过 APT 机制远程获取的软件包都被保存在“/var/cache/apt/archives”中，直到它们被清除。

aptitude 的这个缓存文件清理策略，能够在”Options” → ”Preferences” 下设置，也可以通过它的菜单，”Actions” 下的”Clean package cache” 或”Clean obsolete files” 来执行强制清理。

2.5.8 Debian 软件包文件名称

Debian 软件包文件有特定的名称结构。

软件包类型	名称结构
二进制软件包（亦称 deb）	<package-name>_<epoch>:<upstream-version>-<debian.versi
用于 debian-installer 的二进制软件包（亦称 udeb）	<package-name>_<epoch>:<upstream-version>-<debian.versi
源代码软件包（上游源代码）	<package-name>_<epoch>:<upstream-version>-<debian.versi
1.0 源代码软件包 (Debian 改变)	<package-name>_<epoch>:<upstream-version>-<debian.versi
3.0 (quilt 补丁管理工具) 源代码软件包 (Debian 改变)	<package-name>_<epoch>:<upstream-version>-<debian.versi
源代码软件包（说明）	<package-name>_<epoch>:<upstream-version>-<debian.versi

Table 2.15: Debian 软件包的名称结构

提示
这里仅叙述了基本的源码包格式。更多内容请参考 dpkg-source(1)。

名称组件	可用的字符（正则表达式）	存在状态
<package-name>	[a-z,A-Z,0-9,.,+,-]+	必需
<epoch>:	[0-9]+:	可选
<upstream-version>	[a-z,A-Z,0-9,.,+,-,:]+	必需
<debian.version>	[a-z,A-Z,0-9,.,+,~]+	可选

Table 2.16: Debian 软件包名称中每一个组件可以使用的字符

注意
你可以用 dpkg(1) 提供的命令检查软件包版本，例如，"dpkg --compare-versions 7.0 gt 7.~pre1 ; echo $?"。

注意
debian-installer (d-i) 使用 udeb 作为它的二进制软件包的文件扩展名，而非普通的 deb。一个 udeb 软件包是从 deb 软件包中剥离了一些不必要的内容（例如文档），从而节省空间同时也放宽软件包政策的要求。deb 和 udeb 软件包会共享相同的软件包结构。“u”表示微小。

2.5.9 dpkg 命令

dpkg(1) 是 Debian 软件包管理中最底层的工具。它非常强大，必须小心使用。

当安装名为“<package_name>”的软件包时，dpkg 会按照下列的顺序处理它。

1. 解包 deb 文件（等同于“ar -x”）
2. 使用 debconf(1) 执行“<package_name>.preinst”
3. 将软件包安装到系统中（等同于“tar -x”）
4. 使用 debconf(1) 执行“<package_name>.postinst”

debconf 系统提供带有 I18N 和 L10N（第 8 章）支持的标准化用户交互。

“status”文件也被例如 dpkg(1)、“dselect update”和“apt-get -u dselect-upgrade”等工具使用。

专门的搜索命令 grep-dctrl(1) 可以搜索“status”和“available”元数据的本地副本。

提示
在debian 安装器环境下，udpkg 命令用于打开 udeb 软件包，udpkg 命令是 dpkg 命令的一个精简版本。

2.5.10 update-alternatives 命令

Debian 系统使用 update-alternatives(1) 让用户可以不受干扰地安装多种重叠的程序。例如，如果同时安装了 vim 和 nvi 软件包，你可以使 vi 命令选择运行 vim。

文件	内容说明
/var/lib/dpkg/info/<package_name>.list	列出软件包安装的所有文件和目录
/var/lib/dpkg/info/<package_name>.md5sums	列出软件包安装的文件的 MD5 哈希值
/var/lib/dpkg/info/<package_name>.preinst	软件包安装之前运行的软件包脚本
/var/lib/dpkg/info/<package_name>.postinst	软件包安装之后运行的软件包脚本
/var/lib/dpkg/info/<package_name>.prerm	软件包移除之前运行的软件包脚本
/var/lib/dpkg/info/<package_name>.postrm	软件包移除之后运行的软件包脚本
/var/lib/dpkg/info/<package_name>.config	用于 debconf 系统的软件包脚本
/var/lib/dpkg/alternatives/<package_name>	update-alternatives 命令使用的替代信息
/var/lib/dpkg/available	所有软件包的可用性信息
/var/lib/dpkg/diversions	dpkg(1) 使用的文件移动信息，由 dpkg-divert(8) 设置
/var/lib/dpkg/statoverride	dpkg(1) 使用的文件状态改变信息，由 dpkg-statoverride(8) 设置
/var/lib/dpkg/status	所有软件包的状态信息
/var/lib/dpkg/status-old	“var/lib/dpkg/status”文件的第一代备份
/var/backups/dpkg.status*	第二代备份，以及“var/lib/dpkg/status”文件更旧的备份

Table 2.17: dpkg 创建的重要文件

```
$ ls -l $(type -p vi)
lrwxrwxrwx 1 root root 20 2007-03-24 19:05 /usr/bin/vi -> /etc/alternatives/vi
$ sudo update-alternatives --display vi
...
$ sudo update-alternatives --config vi
  Selection    Command
 -----------------------------------------------
      1        /usr/bin/vim
*+    2        /usr/bin/nvi

Enter to keep the default[*], or type selection number: 1
```

Debian 选择系统在”/etc/alternatives/” 目录里通过符号链接来维持它的选择。选择进程使用”/var/lib/dpkg/alter 目录里面的相应文件。

2.5.11 dpkg-statoverride 命令

当安装一个软件包时，由 dpkg-statoverride(8) 命令提供的 状态修改，是告诉 dpkg(1) 对 文件使用不同的属主或权限的一个方法。如果使用了”--update” 选项，并且文件存在，则该文件会被立即设置为新的属主和模式。

小心
系统管理员使用 chmod 或 chown 命令直接修改某个软件包文件的属主或权限，将会在下次软件包升级时，被重置。

注意
本人在此使用了文件一词，但事实上也可用于 dpkg 所处理的任何文件系统对象，包括目录，设备等。

2.5.12 dpkg-divert 命令

dpkg-divert(8) 命令提供的文件转移，是迫使 dpkg(1) 将文件不安装到其默认位置，而是安装到转移的位置。dpkg-divert 是软件包维护脚本。不建议系统管理员使用这个命令。

2.6 从损坏的系统中恢复

当运行 unstable 系统，系统管理员会遇到从错误的软件包管理进行恢复的情形。

小心
下面的一些方法具有很高的风险。在此先对你进行警告！

2.6.1 不兼容旧的用户配置

如果一个桌面 GUI 程序在重要的上游版本升级后变得不稳定，你应该怀疑这是旧的本地配置文件（由它创建的）所导致的。如果它在新建的用户账号下运行稳定，那么这个假设就得到了证实。（这是一个打包的 bug 并且打包者通常会避免它。）

为了恢复稳定，你应该移除相应的本地配置文件并重新启动 GUI 程序。你可能需要阅读旧的配置文件内容以便之后恢复配置信息。（别将它们删得太快了。）

2.6.2 具有相同文件的不同软件包

文档级的软件包管理系统，比如说 aptitude(8) 或 apt-get(1), 使用软件包依赖，当出现相同文件时，不会尝试去安装软件包。(参见第 2.1.6 节).

软件包维护者的错误，或者系统管理员配置了不一致的档案库混合源，(参见第 2.7.2 节)，都会出现不正确的软件包依赖情况。如果在出现相同文件的情况下，你通过 aptitude(8) 或 apt-get(1) 安装软件包，dpkg(1) 在对软件包解包时，确定会给调用程序返回错误，并不会覆盖已经存在的文件。

小心
使用第三方软件包会导致重大的系统风险，因为其通过使用 root 权限运行维护者脚本能够对你的系统做任何事。dpkg(1) 命令只防止解包时的覆盖行为。

可以先通过删除旧的令人讨厌的软件包，<old-package>，来解决这类错误的安装问题。

```
$ sudo dpkg -P <old-package>
```

2.6.3 修复损坏的软件包脚本

当软件包脚本中的一个命令由于某些原因返回错误，脚本也将由于错误而退出，软件包管理系统忽略它们的行为，并导致部分安装的软件包。当一个软件包在它的删除脚本中有错误时，该软件包将会成为不可能删除的软件包，处理这些问题，都会变得相当棘手。

对于“<package_name>”的软件包脚本问题，你应该查看下列的软件包脚本。

- ”/var/lib/dpkg/info/<package_name>.preinst”
- ”/var/lib/dpkg/info/<package_name>.postinst”
- ”/var/lib/dpkg/info/<package_name>.prerm”
- ”/var/lib/dpkg/info/<package_name>.postrm”

使用下列的方法，以 root 编辑损坏的软件包脚本。

- 在行首添加“#”可以禁用出错的行
- 在出错行的行尾添加“|| true”可以强制返回成功

使用下列命令来配置所有部分安装的软件包。

```
# dpkg --configure -a
```

2.6.4 使用 dpkg 命令进行救援

因为 dpkg 是非常底层的软件包工具，它可以在很糟糕的情况下进行工作，例如无法启动系统且没有网络连接。让我们假定 foo 软件包损坏了，并且需要更换。

你可以在软件包缓存目录：“/var/cache/apt/archives/”中找到旧的 foo 软件包的无 bug 版本。(如果找不到，你可以从档案库 https://snapshot.debian.org/ 中下载它，或从具有软件包缓存功能的机器中拷贝它。)

如果你能够启动系统，你可以通过下列命令来安装它。

```
# dpkg -i /path/to/foo_<old_version>_<arch>.deb
```

提示
如果你系统损坏较小，你也可以使用更高层的 APT 系统来降级整个系统，就像第 2.7.10 节中做的那样。

如果你的系统无法从硬盘启动，你应该寻找其它方式来启动它。

1. 使用 Debian 安装光盘以救援模式启动系统。
2. 将硬盘上无法启动的系统挂载到“/target”。
3. 通过下列命令安装旧版本的 foo 软件包。

```
# dpkg --root /target -i /path/to/foo_<old_version>_<arch>.deb
```

即使位于硬盘上的 dpkg 命令已损坏，该命令依旧可以执行。

提示
任何由硬盘、live GNU/Linux CD、可启动的 USB 驱动或网络启动上的另一系统启动的 GNU/Linux 系统到可以类似地用来救援损坏的系统。

如果由于依赖问题,无法用这种方式安装软件包,并且你真的必须真么做,你可以使用 dpkg 的“--ignore-depends”、“--fo
和其它选项来无视依赖。如果你这么做了，之后你必须认真努力地修复依赖关系。更多细节参见 dpkg(8)。

注意
如果你的系统严重损坏了，你应该将系统完整备份到一个安全的地方 (参见第 10.2 节) 并进行一次全新的安装。这是耗时较少且效果较好的办法。

2.6.5 恢复软件包选择数据

如果“/var/lib/dpkg/status”因为某种原因出现错误，Debian 系统会丢失软件包选择数据并受到严重影响。寻找位于“/var/lib/dpkg/status-old”或“/var/backups/dpkg.status.*”中旧的“/var/lib/dpkg/status”文件。

给“/var/backups/”分配一个单独的分区是一个好习惯，因为这个目录包含了许多重要的系统数据。

对于严重的损坏，我建议备份系统后重新安装。即使失去“/var/”中的所有数据，你依旧可以从“/usr/share/doc/”目录恢复一些信息来引导你进行新的安装。

重新安装最小（桌面）系统。

```
# mkdir -p /path/to/old/system
```

将旧系统挂载到“/path/to/old/system/”。

```
# cd /path/to/old/system/usr/share/doc
# ls -1 >~/ls1.txt
# cd /usr/share/doc
# ls -1 >>~/ls1.txt
# cd
# sort ls1.txt | uniq | less
```

然后你就可以根据软件包名称来进行安装了。（可能会有一些非软件包名称，例如“texmf”。）

2.7 软件包管理技巧

2.7.1 如何挑选 Debian 软件包

你可以根据 aptitude 这个包管理工具中的软件包描述或者是任务面板下的列表信息，来查找你所需要的软件包。

当遇到 2 个以上的类似的软件包时，先前没有经过反复的尝试，你不知道安装哪一个的时候，应该用常识来判断。我认为以下几点是首选的软件包应该具有的特征。

- 重要性：是 > 否
- 类型：main > contrib > non-free
- 优先级：需要 > 重要 > 标准 > 可选 > 额外
- 任务：在任务下有软件包的列表信息，例如“桌面环境”
- 软件包是被与之有依赖关系的软件包所选择的（例如 python2.4 依赖 python）
- 流行度：在投票或者安装指数上有着更高的分数
- 更新日志：维护者经常的更新
- BTS (缺陷跟踪系统): 没有 RC 级别的缺陷（没有危险、重大严重的缺陷）
- BTS (缺陷跟踪系统): 有维护者对缺陷报告反馈
- BTS (缺陷跟踪系统): 有着更多的近期修复的 bug 数目
- BTS (缺陷跟踪系统): 遗留的非严重（non-wishlist）缺陷数量较少

Debian 是一个使用分布式开发模式的志愿项目，它的档案库包含了许多不同关注点和不同质量的软件包。你必须做出自己的选择。

2.7.2 混合源档案库中的软件包

小心
从混合源档案库中安装软件包是不被 Debian 官方发行版所支持的，除了官方支持的档案库的特殊组合以外，例如 `stable` 的 security updates 和 buster-updates。

这里有一个列子，在原有只跟踪 `testing` 的场景，操作包含在 `unstable` 里发现的新的上游软件包版本。

1. 临时更改"`/etc/apt/sources.list`" 文件，使之指向单一的"`unstable`" 发行版路径。
2. 运行"`aptitude update`" 命令。
3. 运行"`aptitude install <package-name>`" 命令。
4. 恢复到原始"`/etc/apt/sources.list`" 文件，使之指向 `testing` 路径。
5. 运行"`aptitude update`" 命令。

使用这个手工方法，你不需要创建"`/etc/apt/preferences`" 文件，也不需要担心 apt-pinning。但这个方法仍然是非常麻烦的。

小心
当使用混合档案源的时候，因为 Debian 不会确保软件之间的兼容性，所以你必须自己去解决兼容性问题。如果软件之间存在不兼容性，系统可能会损坏。你必须能够判断这些操作所需的技术要求。使用任意混合的档案源是完全可选的操作，我并不鼓励你去使用它。

从不同的档案库中安装软件包的一般规则如下。

- 非二进制软件包 ("`Architecture: all`") 的安装是更保险的。
  - 文档软件包：没有特别的要求
  - 解释程序的软件包：兼容的解释器必须是可用的
- 二进制软件包 (non "`Architecture: all`") 通常会面临很多障碍，它的安装不保险的。
  - 库文件版本的兼容性 (包括"`libc`")
  - 与之相关的有用的程序版本的兼容性
  - 内核 ABI 的兼容性
  - C++ ABI 的兼容性
  - ...

注意
为了使软件包的安装变得更保险，一些商业的非自由的二进制程序包可能会提供完整的静态链接库。你还是应该检查 ABI 的兼容性问题等等。

注意
为避免短期出现坏的软件包，从非官方支持的档案库安装二进制软件包通常是一个坏注意。即使你在使用 apt-pinning 情况下，也是这样的。(参见第 2.7.3 节). 你应当考虑使用 chroot 或类似技术 (参见第 9.10 节) 来运行不同档案库的程序。

2.7.3 调整候选版本

没有"/etc/apt/preferences" 文件，APT 系统使用版本字符串来选择最新的可用版本作为 候选版本。这是通常的状态，也是 APT 系统最推荐的使用方法。所有官方支持的档案库集合，并不要求"/etc/apt/preferences" 文件，因此，一些不应当被作为自动更新源的软件包，被标记为 **NotAutomatic**，并被适当处理。

提示
版本字符串的比较规则可以被验证，例子如下，"dpkg --compare-versions ver1.1 gt ver1.1~1; echo $?" (参见 dpkg(1))。

如果经常从混合源档案库中安装软件包 (参见第 2.7.2 节), 你可以通过创建"/etc/apt/preferences" 文件并且在其中写入关于调整候选版本的软件包选取规则的合适条目 (如 apt_preferences(5) 中所示) 来自动化这些复杂的操作。这被称为 **apt-pinning**。

警告
新手用 apt-pinning 命令会造成比较大的问题。你必须避免使用这个命令除非确实需要它。

小心
当使用 apt-pinning 命令时，因为 Debian 不会确保软件之间的兼容性，所以你必须自己确认其兼容性。apt-pinning 是完全可选的操作，我并不建议去使用它。

小心
档案库层级的 Release 文件 (参见第 2.5.3 节) 使用 apt_preferences(5) 的规则. 对于 Debian 通用档案库 和 Debian 安全档案库，apt-pinning 只在"suite" 名下工作。(这点和 Ubuntu 档案库不同.) 例如，你在"/etc/apt/preferences" 文件里面，可以使用"Pin: release a=unstable" ，但不能使用"Pin: release a=sid".

小心
当使用非 Debian 的档案库作为 apt-pinning 的一部分时，你应该检查它们的用途和可信度。例如，Ubuntu 和 Debian 是不能混在一起的。

注意
即使不创建"/etc/apt/preferences" 文件，在不用 apt-pinning 命令的情况下，你也可以进行相当复杂的系统工作 (参见第 2.6.4 节和第 2.7.2 节)。

如下是关于 **apt-pinning** 技术的简化说明。

可用的软件包源在"/etc/apt/sources.list" 文件里面定义，APT 系统从可用的软件包源里面选择 Pin-Priority 值最大的，作为升级软件包的候选版本. 如果一个软件包的 Pin-Priority 大于 1000, 这个版本限制为只能 升级，关闭了软件包降级功能 (参见第 2.7.10 节).

每个软件包的 Pin-Priority 值是在"/etc/apt/preferences" 文件中的"Pin-Priority" 条目中定义或者是使用它的默认值。

目标发行版档案库可以通过以下几种方法来设置。

Pin-Priority	apt-pinning 对软件包的影响
1001	安装该软件包，即使是一个降级软件包的指令
990	用作目标发行版档案库的默认值
500	用作常规档案库的默认值
100	用于 NotAutomatic 和 ButAutomaticUpgrades 档案库的默认值
100	用于已安装软件包
1	用于 NotAutomatic 档案库的默认值
-1	即使被推荐，也绝不安装这个软件包

Table 2.18: 用于 **apt-pinning** 技术的值得注意的 Pin-Priority 值列表。

- ”`/etc/apt/apt.conf`” 配置文件中写入”`APT::Default-Release "stable";`”
- 命令行选项，例如: ”`apt-get install -t testing some-package`”

NotAutomatic 和 **ButAutomaticUpgrades** 的档案是由档案库服务器上档案层级的 Release 文件来设置，(参见第 2.5.3 节)，同时包含”`NotAutomatic: yes`” 和”`ButAutomaticUpgrades: yes`”. 而 **NotAutomatic** 档案也是由档案库服务器上的档案层级的 Release 文件来设置，但只包含”`NotAutomatic: yes`”.

来自众多档案源的 < 软件包 > 的 **apt-pinning** 情况可以通过”`apt-cache policy <package>`” 命令显示。

- ”`Package pin:`” 开头的行，列出了软件包版本的 **pin** ，如果 <package> 相关的 pin 已经定义, 例如, ”`Package pin: 0.190`”.
- 没有”`Package pin:`” 的行存在，如果没有 <package> 相关的定义。
- 与 <package> 相关的 Pin-Priority 值列在所有版本字符串的右边，比如，”`0.181 700`”.
- ”`0`” 是列在所有版本字符串的右边，如果没有 <package> 相关的定义。例如, ”`0.181 0`”.
- 档案库 (在”`/etc/apt/preferences`” 文件作为”`Package: *`” 定义) 的 Pin-Priority 值，列在所有档案库路径的左边，例如，”`100 http://deb.debian.org/debian/ buster-backports/main Packages`”.

2.7.4 更新和向后移植

buster-updates 和 backports.debian.org 档案库提供了 `stable` (`buster`) 发行版的更新软件包。

为了去使用这些档案库，你需要在”`/etc/apt/sources.list`” 文件里写入如下所示的档案库列表。

```
deb http://deb.debian.org/debian/ buster main contrib non-free
deb http://security.debian.org/ buster/updates main contrib
deb http://deb.debian.org/debian/ buster-updates main contrib non-free
deb http://deb.debian.org/debian/ buster-backports main contrib non-free
```

并不需要在”`/etc/apt/preferences`” 文件中显式设置 Pin-Priority 值. 当新的包可用时，默认配置提供了更合理的更新 (请见第 2.5.3 节).

- 所有已安装的旧软件包都可以通过 `buster-updates` 档案库升级到新软件包。
- 只有从 `buster-backports` 档案库中手动安装的旧软件包才会通过 `buster-backports` 档案库升级到新软件包。

当你想要从 `buster-backports` 档案库中手动的安装一个名叫”`<package-name>`” 的软件及其依赖包的时候，你应该在目标档案库之前加一个 “-t” 参数。

```
$ sudo apt-get install -t buster-backports <package-name>
```

2.7.5 阻止推荐的软件包的安装

如果不想要引入推荐的特定软件包，你必须创建”/etc/apt/preferences” 文件并且像如下所示的那样在文件的顶部明确列出这些软件包。

```
Package: <package-1>
Pin: version *
Pin-Priority: -1

Package: <package-2>
Pin: version *
Pin-Priority: -1
```

2.7.6 使用带有 **unstable** 软件包的 **testing** 版本

如下是一个关于 **apt-pinning** 技术的例子，当使用 testing 的时候，实现 unstable 中的特定的较新的上游版本软件包的日常升级。你应该按如下所示的在”/etc/apt/sources.list” 文件中列出所有需要的档案库。

```
deb http://deb.debian.org/debian/ testing main contrib non-free
deb http://deb.debian.org/debian/ unstable main contrib non-free
deb http://security.debian.org/ testing/updates main contrib
```

按如下所示的设置”/etc/apt/preferences” 文件。

```
Package: *
Pin: release a=unstable
Pin-Priority: 100
```

当想要在此配置下从 unstable 档案库中安装”<package-name>” 软件及它的依赖包时，你执行带有”-t” 选项 (unstable 的 Pin-Priority 值变为 990) 的转换目标发行版的命令。

```
$ sudo apt-get install -t unstable <package-name>
```

在此配置下,执行”apt-get update” 和”apt-get dist-upgrade”(或者”aptitude safe-upgrade” 和”aptitude full-upgrade”) 命令, 会从 testing 档案库升级那些从 testing 档案库安装的软件包并且从 unstable 档案库升级那些从 unstable 档案库中安装的软件包。

小心
小心不要去移除”/etc/apt/sources.list” 文件中的”testing” 档案库。如果文件中没有”testing” ，APT 系统会使用更加新的 unstable 档案库升级软件包。

提示
我通常会在上述操作后，马上注释掉”/etc/apt/sources.list” 文件中的”unstable” 档案库记录。这避免了因为处理”/etc/apt/sources.list” 文件中的众多记录而造成的升级缓慢虽然同时也阻止了那些从 unstable 档案库中安装的软件包通过 unstable 升级。

提示
如果"/etc/apt/preferences"文件中"Pin-Priority: 1"替代了"Pin-Priority:100"，即使"/etc/apt/sources.list"文件中的"testing"记录被删除了，Pin-Priority 值为 100 的已安装软件包也不会通过 unstable 档案库升级。

如果你希望自动跟踪 unstable 里某些特殊的软件包，而在安装时不再使用初始化选项"-t unstable"，你必须创建"/etc/apt/preferences"文件，并在该文件顶部按下面的方式清晰的列出所有那些软件包。

```
Package: <package-1>
Pin: release a=unstable
Pin-Priority: 700

Package: <package-2>
Pin: release a=unstable
Pin-Priority: 700
```

如下是为每个特定的软件包设置 Pin-Priority 值。例如，为了使用最新的 unstable 的英文版"Debian Reference"，你应该在"/etc/apt/preferences"文件中写入以下条目。

```
Package: debian-reference-en
Pin: release a=unstable
Pin-Priority: 700

Package: debian-reference-common
Pin: release a=unstable
Pin-Priority: 700
```

提示
即使你使用的是 stable 档案库，apt-pinning 技术仍然是有效的。根据我以前的经验，从 unstable 档案库安装的文档包一直是安全的。

2.7.7 使用带有 experimental 软件包的 unstable 版本

这是使用 **apt-pinning** 的另一个示例，该示例主要使用 unstable 源，但包含了 experimental 源, 该源可用于安装上游更新的软件包. 需要包含在"/etc/apt/sources.list"文件中的列表如下:

```
deb http://deb.debian.org/debian/ unstable main contrib non-free
deb http://deb.debian.org/debian/ experimental main contrib non-free
deb http://security.debian.org/ testing/updates main contrib
```

由于 experimental 源是非自动（**NotAutomatic**）的源 (参见第 2.5.3 节)，其默认的 Pin-Priority 值被设置为 1 (<<100) . 并不需要在"/etc/apt/preferences"文件中设置 Pin-Priority 值，只需要指定 experimental 源，除非你需要在下次更新时自动升级时更新特定软件包.

2.7.8 自动下载和升级软件包

apt 软件包有自己的 cron 脚本"/etc/cron.daily/apt",它支持自动下载软件包。可以安装 unattended-upgrades 软件包来增强这个脚本,使它能够自动升级软件包。可以通过"/etc/apt/apt.conf.d/02backup"和"/etc/apt/apt.co 中的参数来进行自定义，相关说明位于"/usr/share/doc/unattended-upgrades/README"中。

unattended-upgrades 软件包主要用于 stable 系统的安全更新。如果自动升级损坏 stable 系统的风险小于被入侵者利用已被安全更新修复的安全漏洞，你应该考虑使用自动更新，配置参数如下。

```
APT::Periodic::Update-Package-Lists ”1”;
APT::Periodic::Download-Upgradeable-Packages ”1”;
APT::Periodic::Unattended-Upgrade ”1”;
```

如果你运行的是 unstable 系统，你应该不会想要使用自动更新，因为它肯定会在某天损坏系统。即使位于这样的 unstable 情况下，你可能依旧想提前下载软件包以节省交互式升级的时间，其配置参数如下。

```
APT::Periodic::Update-Package-Lists ”1”;
APT::Periodic::Download-Upgradeable-Packages ”1”;
APT::Periodic::Unattended-Upgrade ”0”;
```

2.7.9 限制 APT 的下载带宽

如果你想限制 APT 的下载带宽到 800Kib/sec (=100KiB/sec)，你应该像下面那样设置 APT 的配置参数。

```
APT::Acquire::http::Dl-Limit ”800”;
```

2.7.10 紧急降级

小心
降级在 Debian 设计上就不被官方支持。仅仅是在紧急恢复过程中需要做的一部分工作。尽管憎恨这种情形，但降级在很多场景下工作得也不错。对于重要系统，你应当在恢复操作后备份所有重要数据，并从零开始重新安装一个新的系统。

你可以通过控制候选版本从新的档案库降级到旧的档案库 (参见第 2.7.3 节)，从而使损坏的系统恢复。下面是一种懒惰的方法，可以避免许多冗长的“dpkg -i <broken-package>_<old-version>.deb”命令 (参见第 2.6.4 节)。

搜索“/etc/apt/sources.list”文件中像下面那样使用 unstable 的行。

```
deb http://deb.debian.org/debian/ sid main contrib non-free
```

使用下面的行替换它，从而改为使用 testing。

```
deb http://deb.debian.org/debian/ bullseye main contrib non-free
```

按如下所示的设置”/etc/apt/preferences” 文件。

```
Package: *
Pin: release a=testing
Pin-Priority: 1010
```

运行“apt-get update; apt-get dist-upgrade”使整个系统的软件包强制降级。

在紧急降级后，移除“/etc/apt/preferences”这个特殊的文件。

提示
这是一个好方法，移除（不是清除！）尽可能多地软件包，来减少依赖问题。你可能需要手动移除和安装一些软件包来使系统降级。需要特别注意 Linux 内核、引导程序、udev、PAM、APT 和网络相关的软件包以及它们的配置文件。

2.7.11 上传软件包的是谁？

尽管“/var/lib/dpkg/available”和“/usr/share/doc/package_name/changelog”中列出的维护者姓名提供了关于“软件包运作的幕后者是谁”这一问题的一些信息，但软件包的实际上传者依旧不明。devscripts 软件包中的 who-uploads(1) 可以识别 Debian 源软件包的实际上传者。

2.7.12 equivs 软件包

如果你从源代码编译了一个程序来代替 Debian 软件包，最好将它做成一个真正的本地 Debian 软件包（*.deb）并使用私人档案库。

如果你选择从源代码编译一个程序并将它安装到“/usr/local”，你可能需要使用 equivs 作为最后步骤来满足缺失的软件包依赖。

```
Package: equivs
Priority: optional
Section: admin
Description: Circumventing Debian package dependencies
 This package provides a tool to create trivial Debian packages.
 Typically these packages contain only dependency information, but they
 can also include normal installed files like other packages do.
 .
 One use for this is to create a metapackage: a package whose sole
 purpose is to declare dependencies and conflicts on other packages so
 that these will be automatically installed, upgraded, or removed.
 .
 Another use is to circumvent dependency checking: by letting dpkg
 think a particular package name and version is installed when it
 isn't, you can work around bugs in other packages' dependencies.
 (Please do still file such bugs, though.)
```

2.7.13 移植一个软件包到 stable 系统

对于部分升级的 stable 系统，使用源软件包在运行环境中重新构建一个软件包是不错的选择。这可以避免因为依赖关系导致大量软件包升级。

在 stable 系统的“/etc/apt/sources.list”文件中添加下列条目。

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

如下安装编译所需的软件包并下载源软件包。

```
# apt-get update
# apt-get dist-upgrade
# apt-get install fakeroot devscripts build-essential
# apt-get build-dep foo
$ apt-get source foo
$ cd foo*
```

如果需要向后移植，可以从 backport 的软件包中更新一些工具链软件包，例如 dpkg 和 debhelper。

执行下列命令。

```
$ dch -i
```

更新软件包版本，例如在“debian/changelog”中附加一个“+bp1”

像下面那样构建软件包并将它们安装到系统中。

```
$ debuild
$ cd ..
# debi foo*.changes
```

2.7.14 用于 APT 的代理服务器

因为镜像整个 Debian 档案库的子区会浪费硬盘和网络带宽，当你管理许多 LAN 上的系统时，为 APT 部署一个本地代理服务器是个好主意。APT 可以通过配置来使用通用 web（http）代理服务器，例如 squid（参见第 6.10 节），细节参见 apt.conf(5) 和“/usr/share/doc/apt/examples/configure-index.gz”。环境变量“$http_proxy 会覆盖“/etc/apt/apt.conf”文件中设置的代理服务器。

这里有一些 Debian 档案库的专用代理工具。你应该在使用它们之前检查 BTS。

软件包	流行度	大小	说明
approx	V:0, I:0	4729	缓存 Debian 档案库文件的代理服务器（已编译的 OCaml 程序）
apt-cacher	V:1, I:1	289	为 Debian 软件包和源代码文件进行缓存代理（Perl 程序）
apt-cacher-ng	V:4, I:5	1421	分发软件包的缓存代理（C++ 编译的程序）

Table 2.19: Debian 档案库的专用代理工具

小心
当 Debian 重构它的档案库结构时，这些专用的代理工具往往需要软件包维护者重写代码，并可能在一段时间内无法使用。另一方面，通用 web（http）代理服务器更强健并且更容易应对这种改变。

2.7.15 小型公共软件包档案库

提示
手动建立软件仓库是极其复杂的。有数个软件仓库管理工具可供用户选用。网上有一个 详尽的列表 可供参阅。

下面是一个建立小型公共软件包档案库的示例，兼容了 **secure APT** 系统（参见第 2.5.2 节）。让我们进行一些假定。

- 账号名：“foo”
- 主机名：“www.example.com”
- 所需软件包：apt-utils、gnupg 和其它软件包
- URL: ”http://www.example.com/~foo/” (→ ”/home/foo/public_html/index.html”)
- 软件包架构：“amd64”

在该服务器上使用如下方式为 Foo 创建一个 APT 源钥匙对。

```
$ ssh foo@www.example.com
$ gpg --gen-key
...
$ gpg -K
...
sec   1024D/3A3CB5A6 2008-08-14
uid                  Foo (ARCHIVE KEY) <foo@www.example.com>
ssb   2048g/6856F4A7 2008-08-14
$ gpg --export -a 3A3CB5A6 >foo.public.key
```

公布”foo.public.key” 文件，即公钥 ID 为”3A3CB5A6” 的源公钥文件，该文件可用于 Foo 源的发布

使用如下方式创建一个名为”Origin: Foo” 的源目录树。

```
$ umask 022
$ mkdir -p ~/public_html/debian/pool/main
$ mkdir -p ~/public_html/debian/dists/unstable/main/binary-amd64
$ mkdir -p ~/public_html/debian/dists/unstable/main/source
$ cd ~/public_html/debian
$ cat > dists/unstable/main/binary-amd64/Release << EOF
Archive: unstable
Version: 4.0
Component: main
Origin: Foo
Label: Foo
Architecture: amd64
EOF
$ cat > dists/unstable/main/source/Release << EOF
Archive: unstable
Version: 4.0
Component: main
Origin: Foo
Label: Foo
Architecture: source
EOF
$ cat >aptftp.conf <<EOF
APT::FTPArchive::Release {
  Origin ”Foo”;
  Label ”Foo”;
  Suite ”unstable”;
  Codename ”sid”;
  Architectures ”amd64”;
  Components ”main”;
  Description ”Public archive for Foo”;
};
EOF
$ cat >aptgenerate.conf <<EOF
Dir::ArchiveDir ”.”;
```

```
Dir::CacheDir ".";
TreeDefault::Directory "pool/";
TreeDefault::SrcDirectory "pool/";
Default::Packages::Extensions ".deb";
Default::Packages::Compress ". gzip bzip2";
Default::Sources::Compress "gzip bzip2";
Default::Contents::Compress "gzip bzip2";

BinDirectory "dists/unstable/main/binary-amd64" {
  Packages "dists/unstable/main/binary-amd64/Packages";
  Contents "dists/unstable/Contents-amd64";
  SrcPackages "dists/unstable/main/source/Sources";
};

Tree "dists/unstable" {
  Sections "main";
  Architectures "amd64 source";
};
EOF
```

通过 dupload，你可以自动重复更新你服务器系统上的 APT 档案库内容。

当“~/.dupload.conf”包含如下内容时，在客户端执行“dupload -t foo changes_file”将所有的软件包文件放入“~foo/public_html/debian/pool/main/”。

```
$cfg{'foo'} = {
  fqdn => "www.example.com",
  method => "scpb",
  incoming => "/home/foo/public_html/debian/pool/main",
  # The dinstall on ftp-master sends emails itself
  dinstall_runs => 1,
};

$cfg{'foo'}{postupload}{'changes'} = "
  echo 'cd public_html/debian ;
  apt-ftparchive generate -c=aptftp.conf aptgenerate.conf;
  apt-ftparchive release -c=aptftp.conf dists/unstable >dists/unstable/Release ;
  rm -f dists/unstable/Release.gpg ;
  gpg -u 3A3CB5A6 -bao dists/unstable/Release.gpg dists/unstable/Release'|
  ssh foo@www.example.com  2>/dev/null ;
  echo 'Package archive created!'";
```

dupload(1) 触发 **postupload** 钩子脚本，该脚本为每次上传创建软件包文件.

你可以通过下面的方法将这个小型公共档案库添加到你客户端系统的 apt 源中。

```
$ sudo bash
# echo "deb http://www.example.com/~foo/debian/ unstable main" \
   >> /etc/apt/sources.list
# apt-key add foo.public.key
```

提示
如果档案库位于本地文件系统中，你可以使用“deb file:///home/foo/debian/ …”。

2.7.16 记录和复制系统配置

你可以通过下面命令建立软件包和 debconf 选择状态的本地副本。

```
# dpkg --get-selections '*' > selection.dpkg
# debconf-get-selections    > selection.debconf
```

这里，“*”使“selection.dpkg”也包含“purge”的软件包。

你可以将这两个文件移动到另一个电脑，并通过下列命令安装它们。

```
# dselect update
# debconf-set-selections < myselection.debconf
# dpkg --set-selections  < myselection.dpkg
# apt-get -u dselect-upgrade    # or dselect install
```

如果你需要管理一个集群中的许多服务器，并且它们的配置几乎相同，你应该考虑使用专门的软件包，例如 fai 来管理整个系统。

2.7.17 转换或安装一个外来的二进制软件包

alien(1) 可以将 Red Hat 的 rpm、Stampede 的 slp、Slackware 的 tgz 和 Solaris 的 pkg 二进制软件包文件格式转换为 Debian 的 deb 软件包。如果你想使用来自另一个发行版的软件包，你可以使用 alien 转换并安装它。alien 也支持 LSB 软件包。

警告
不应该用 alien(1) 来替代必要的系统软件包，例如 sysvinit、libc6、libpam-modules 等等。实际上，alien(1) 应该只用于 **non-free** 中仅提供二进制文件的软件包，并且它们应该兼容 LSB 或被静态链接。对于自由软件，你应该使用它们的源软件包来制作一个真正的 Debian 软件包。

2.7.18 不使用 **dpkg** 解压软件包

在任何的 Unix-like 环境中都可以不使用 dpkg(1)，而使用标准的 ar(1) 和 tar(1) 来解压“dpkg*.deb”软件包。

```
# ar x /path/to/dpkg_<version>_<arch>.deb
# ls
total 24
-rw-r--r-- 1 bozo bozo  1320 2007-05-07 00:11 control.tar.gz
-rw-r--r-- 1 bozo bozo 12837 2007-05-07 00:11 data.tar.gz
-rw-r--r-- 1 bozo bozo     4 2007-05-07 00:11 debian-binary
# mkdir control
# mkdir data
# tar xvzf control.tar.gz -C control
# tar xvzf data.tar.gz -C data
```

其它的“*.deb”软件包可以使用 dpkg-deb(1) 解压来获得上面的“dpkg*.deb”软件包；或像上面那样使用标准的 ar(1) 和较新的带有 xz(1) 解压支持的 GNU tar(1)。

你也可以使用 mc 命令来浏览软件包内容。

2.7.19 更多关于软件包管理的文档

你可以从下面的文档中了解软件包管理的更多信息。

- 软件包管理的主要文档：
  - aptitude(8)、dpkg(1)、tasksel(8)、apt(8)、apt-get(8)、apt-config(8)、apt-key(8)、sources.list(5)、apt.conf(5) 和 apt_preferences(5)；
  - 来自 `apt-doc` 软件包的“`/usr/share/doc/apt-doc/guide.html/index.html`”和“`/usr/share/doc/apt-do`
  - 来自 `aptitude-doc-en` 软件包的“`/usr/share/doc/aptitude/html/en/index.html`”。
- Debian 档案库的官方详细文档：
  - ”Debian Policy Manual Chapter 2 - The Debian Archive”，
  - ”Debian Developer’s Reference, Chapter 4 - Resources for Debian Developers 4.6 The Debian archive”，
  - ”The Debian GNU/Linux FAQ, Chapter 6 - The Debian FTP archives”。
- 为 Debian 用户构建一个 Debian 软件包的教程：
  - “Debian 新维护人员手册”（已过时）。
  - “Debian 维护者指导”。

Chapter 3

系统初始化

作为系统管理员，粗略地了解 Debian 系统的启动和配置方式是明智的。尽管准确的细节在安装的软件包及对应的文档中，但这些知识对我们大多数人来说都是必须掌握的。

笔者基于自己和其他人的过往及现在的知识，尽己所能地提供关于 Debian 系统的知识要点及其配置的快速概览作为读者的参考。由于 Debian 系统在不断地更新中，系统的状况可能已经有所变化。在对系统做任何修改之前，请参考各个软件包的最新文档。

提示
bootup(7) 介绍了基于 `systemd` 的系统启动流程。(近期的 Debian)

提示
boot(7) 介绍了基于 UNIX System V Release 4 的系统启动流程。(旧版的 Debian)

3.1 启动过程概述

计算机系统从上电事件到能为用户提供完整的操作系统 (OS) 功能为止，需要经历几个阶段的启动过程。

为简便起见，笔者将讨论范围限定在具有默认安装的典型 PC 平台上。

典型的启动过程像是一个四级的火箭。每一级火箭将系统控制权交给下一级。

- 第 3.1.1 节
- 第 3.1.2 节
- 第 3.1.3 节
- 第 3.1.4 节

当然，这些阶段可以有不同的配置。比如，你编译了自己的内核，则可能会跳过迷你 Debian 系统的步骤。因此，在读者亲自确认之前，请勿假定自己系统的情况也是如此。

注意
对于 SUN 或 Macintosh 系统等非传统 PC 平台来说，ROM 上的 BIOS 及磁盘上的分区可能大不相同 (第 9.5.2 节)。对于这种情况，请另寻对应平台相关的文档。

3.1.1 第一阶段：BIOS

BIOS 是启动过程的第一阶段，在上电事件后开始。CPU 的程序计数器在上电事件后被初始化为一个特定的内存地址，驻留在只读存储器（ROM）中的 BIOS 就是从这个特定的内存地址开始执行。

BIOS 执行硬件的基本初始化（POST: 上电自检）并将系统控制权交给你指定的下一步骤。BIOS 通常和硬件一同提供。

BIOS 启动屏幕通常指示了进入 BIOS 配置界面所需的按键。流行的按键是 F1、F2、F10、Esc、Ins 和 Del 键。假如你的启动屏幕被一个漂亮的图形界面隐藏，你可以按下某些按键（比如 ESC）取消隐藏。这些按键高度依赖于硬件。

硬件位置和 BIOS 启动的代码的优先级可以在 BIOS 配置界面中选择。通常，在已选择的设备（硬盘、软件、CD-ROM ……）中，最先找到的设备的最开始的几个扇区将被加载到内存，并执行其中的初始化代码。初始化代码可以是以下任意一种。

- 引导加载代码
- 类似 FreeDOS 这样的过滤型操作系统的内核代码
- 能够加载到如此小的空间中的目标操作系统的内核代码

通常，系统从主硬件的特定分区中引导。传统 PC 硬盘的最开始两个扇区中包含了主引导记录（MBR）。在 MBR 的末尾记录了磁盘分区信息及引导选择。BIOS 中执行的首段引导加载代码占据了 MBR 的其余部分。

3.1.2 第二阶段：引载加载程序

引导加载程序是启动过程的第二阶段，由 BIOS 启动。引导加载程序将系统内核映像和 initrd 映像加载到内存并将控制权交给它们。initrd 映像是根文件系统映像，其支持程度依赖于所使用的引导加载程序。

Debian 系统通常使用 Linux 内核作为其默认的系统内核。当前 2.6/3.x 版本 Linux 内核的 initrd 镜像从技术上说是 initramfs（初始化 RAM 文件系统）镜像。基本的 initrd 镜像是 root 文件系统中各个文件使用 cpio 压缩得到的。内核可以在启动流程中非常早的阶段，在加载基本的 initrd 镜像之前即更新微码。以未压缩 cpio 格式存储微码二进制文件的 initrd 镜像和基本 initrd 镜像两部分可以联合组成一个 initrd 镜像，从而帮助实现上述功能。

提示
您可以使用 lsinitramfs(8) 和 unmkinitramfs(8) 这两个工具检查 initrd 镜像文件的内容，它们由 `initramfs-tools-core` 软件包提供。另见 https://wiki.debian.org/initramfs 以了解更多信息。

Debian 系统默认将 PC 平台的 GRUB 引导加载程序的第一阶段代码安装在 MBR 中。可用的引导加载程序和配置选项如下。

警告
假如没有从 `grub-rescue-pc` 软件包中的映像制作出来的可引导修复盘（U 盘、CD 或软盘），请勿玩弄引导加载程序。即使硬盘上没有可正常工作的引导加载程序，可引导修复盘也能引导你的系统。

传统 GRUB 的菜单配置文件位于 `/boot/grub/menu.lst`。例如，文件中有如下的配置条目。

```
title           Debian GNU/Linux
root            (hd0,2)
kernel          /vmlinuz root=/dev/hda3 ro
initrd          /initrd.img
```

GRUB 第 2 版的菜单配置文件位于 `/boot/grub/grub.cfg`。此文件由 `/usr/sbin/update-grub` 根据”`/etc/grub.d/*`”中的模板及”`/etc/default/grub`” 中的设置自动生成。例如，文件中有如下的配置条目。

软件包	流行度	大小	**initrd**	引导加载程序	说明
grub-legacy	V:0, I:2	729	支持	传统 GRUB	可智能识别磁盘分区和文件系统（例如 vfat、ext3⋯）。
grub-pc	V:27, I:825	532	支持	GRUB 第 2 版	可智能识别磁盘分区和文件系统（例如 vfat、ext4⋯）。（默认安装）
grub-rescue-pc	V:0, I:1	6286	支持	GRUB 第 2 版	此为 GRUB 第 2 版的可引导修复映像（CD 和软盘）（PC / BIOS 版本）
lilo	V:0, I:3	693	支持	Lilo	依赖于数据在硬盘上的扇区位置。（较老）
syslinux	V:4, I:54	344	支持	Isolinux	可识别 ISO9660 文件系统。引导 CD 使用此项。
syslinux	V:4, I:54	344	支持	Syslinux	可识别 MSDOS 文件系统（FAT）。引导软盘使用此项。
loadlin	V:0, I:1	83	支持	Loadlin	新系统从 FreeDOS 或 MSDOS 中启动。
mbr	V:0, I:9	49	不支持	Neil Turton 的 MBR	此为取代 MSDOS MBR 的自由软件。只可识别硬盘分区。

Table 3.1: 引导加载程序列表

```
menuentry "Debian GNU/Linux" {
        set root=(hd0,3)
        linux /vmlinuz root=/dev/hda3
        initrd /initrd.img
}
```

这些示例中，GRUB 参数的含义如下。

GRUB 参数	说明
`root`	使用主磁盘的第 3 个分区，在传统 GRUB 中将此参数设置为"`(hd0,2)`"，在 GRUB 第 2 版中将此参数设置为"`(hd0,3)`"
`kernel`	使用位于"`/vmlinuz`"的内核，同时将"`root=/dev/hda3 ro`"作为参数传递给内核
`initrd`	使用位于"`/initrd.img`"的 initrd/initramfs 映像

Table 3.2: GRUB 参数的含义

注意
传统 GRUB 使用的分区号为 Linux 内核及各种实用工具使用的分区号减 1。GRUB 第 2 版修复了这个问题。

提示
在标识一个块设备时，可能需要使用 UUID（参见第 9.5.3 节）而不是类似"`/dev/hda3`"这样的文件名，例如"`root=UUID=81b289d5-4341-4003-9602-e254a17ac232 ro`"。

提示
如果使用了 GRUB，内核的启动参数可以在 `/boot/grub/grub.cfg` 里面设置。在 Debian 系统里，你不应该直接编辑 `/boot/grub/grub.cfg`。你可以通过编辑 `/etc/default/grub` 文件中 `GRUB_CMDLINE_LINUX_DEFAULT` 的值并运行 update-grub(8) 来更新 `/boot/grub/grub.cfg`。

提示
通过使用链式引导技术，你可以在一个引导装载程序中启动另一个引导装载程序。

参见“`info grub`”及 grub-install(8)。

3.1.3 第三阶段：迷你 Debian 系统

迷你 Debian 系统是启动流程的第三阶段，由引导加载程序启动。它会在内存中运行系统内核和根文件系统。这是启动流程的一个可选准备阶段。

注意
“迷你 Debian 系统”是笔者自创的术语，用于在本文档中描述启动流程的第三个阶段。这个系统通常被称为 initrd 或 initramfs 系统。内存中类似的系统在 Debian 安装程序中使用。

`/init` 程序是内存中的根文件系统上执行的第一个程序。这个程序在用户空间把内核初始化，并把控制权交给下一阶段。迷你 Debian 系统能够在主引导流程之前添加内核模块或以加密形式挂载根文件系统，使引导流程更加灵活。

- 如果 initramfs 是由 `initramfs-tools` 创建，则”`/init`” 程序是一个 shell 脚本程序。
  - 通过给内核添加“`break=init`” 等启动参数，你可以中断这部分启动流程以获取 root shell。更多中断条件请参见”`/init`“脚本。这个 shell 环境已足够成熟，你可通过它很好地检查机器的硬件。
  - 迷你 Debian 系统中可用的命令是精简过的，且主要由一个称为 busybox(1) 的 GNU 工具提供。
- 如果 initramfs 是由 `dracut` 创建，则”`/init`” 程序是一个二进制 `systemd` 程序。
  - 迷你 Debian 系统中可用的命令是一个精简过的 systemd(1) 环境。

小心
当在一个只读的根文件系统上时，使用 `mount` 命令需要添加 `-n` 选项。

3.1.4 第四阶段：常规 Debian 系统

常规 Debian 系统是启动流程的第四阶段，由迷你 Debian 系统启动。迷你 Debian 系统的内核在此环境下继续运行。根文件系统将由内存切换到实际的硬盘文件系统上。

init 程序是系统执行的第一个程序（PID=1），它启动其它各种程序以完成主引导流程。init 程序的默认路径是”`/sbin/init`“，但可通过内核启动参数修改，例如”`init=/path/to/init_program`”。

默认的 init 程序一直在变化中：

- `squeeze` 之前的 Debian，使用简单的 SysV 风格的 init。
- `wheezy` 版本的 Debian 对 SysV 风格的 init 做了改进：使用 LSB 头将启动步骤排序，同时并行执行启动脚本。
- `jessie` 版本的 Debian 将默认 init 切换成 systemd，以使用事件驱动和并行初始化。

提示
你的系统中实际使用的 init 命令可以使用“`ps --pid 1 -f`”命令确认。

软件包	流行度	大小	说明
systemd	V:750, I:858	13484	基于事件且支持并发的 init(8) 守护进程（可替代 sysvinit）
systemd-sysv	V:733, I:852	122	systemd 需用的用以代替 sysvinit 的手册页和符号链接
systemd-cron	V:0, I:1	139	提供 cron 后台守护进程（daemon）和 anacron 功能的 systemd 单元
init-system-helpers	V:745, I:876	133	在 sysvinit 和 systemd 之间进行转换的帮助工具
initscripts	V:188, I:509	213	用于初始化和关闭系统的脚本
sysvinit-core	V:10, I:13	263	类 System V 的 init(8) 工具
sysv-rc	V:334, I:520	121	类 System V 的运行级别修改机制
sysvinit-utils	V:729, I:999	131	类 System V 的实用工具（startpar(8)，bootlogd(8)，⋯⋯）
lsb-base	V:886, I:999	49	Linux 标准规范 3.2 版的 init 脚本功能
insserv	V:403, I:510	148	利用 LSB init.d 脚本依赖性来组织启动步骤的工具
uswsusp	V:5, I:10	714	使用 Linux 提供的用户态软件 suspend 的工具
kexec-tools	V:1, I:7	271	用于 kexec(8) 重启（热启动）的 kexec 工具
systemd-bootchart	V:0, I:0	123	启动流程性能分析器
bootchart2	V:0, I:1	94	启动流程性能分析器
pybootchartgui	V:0, I:1	177	启动流程性能分析器（可视化）
mingetty	V:0, I:3	35	仅包含控制台的 getty(8)
mgetty	V:0, I:1	319	可智能调制解调的 getty(8) 替代品

Table 3.3: Debian 系统启动工具列表

提示
在 Debian jessie 版本后，"/sbin/init" 是一个到"/lib/systemd/systemd" 的符号链接。

提示
有关启动流程加速的最新信息，请参见 Debian 维基：启动流程加速词条。

3.2 Systemd 初始化

本节描述系统是怎样通过 PID=1 的 systemd(1) 程序来启动（即初始化进程）。

systemd 初始化进程基于单元配置文件 (参见 systemd.unit(5)) 来并行派生进程，这些单元配置文件使用声明样式来书写，代替之前的类 SysV 的过程样式。这些单元配置文件从下面的一系列路径来加载 (参见 systemd-system.conf(5))：

- "/lib/systemd/system": OS 默认配置文件
- "/etc/systemd/system": 系统管理员的配置文件，它将忽略操作系统默认的配置文件
- "/run/systemd/system": 运行时产生的配置文件，它将忽略安装的配置文件

他们的相互依赖关系通过"Wants=", "Requires=", "Before=", "After=", ⋯等指示来配置，(参见 systemd.unit(5) 里的"MAPPING OF UNIT PROPERTIES TO THEIR INVERSES")。资源控制也是被定义 (参见 systemd.resource-control(5)).

根据单元配置文件的后缀来区分它们的类型：

- ***.service** 描述由 systemd 控制和监管的进程. 参见 systemd.service(5).
- ***.device** 描述在 sysfs(5) 里面作为 udev(7) 设备树展示的设备。参见 systemd.device(5).

- ***.mount** 描述由 systemd 控制和监管的文件系统挂载点。参见 systemd.mount(5).
- ***.automount** 描述由 systemd 控制和监管的文件系统自动挂载点。参见 systemd.automount(5).
- ***.swap** 描述由 systemd 控制和监管的 swap 文件或设备。参见 systemd.swap(5).
- ***.path** 描述被 systemd 监控的路径，用于基于路径的活动。参见 systemd.path(5).
- ***.socket** 描述被 systemd 控制和监管的套接字，用于基于套接字的活动。参见 systemd.socket(5).
- ***.timer** 描述被 systemd 控制和监管的计时器，用于基于时间的活动。参见 systemd.timer(5).
- ***.slice** 管理 cgroups(7) 的资源。参见 systemd.slice(5).
- ***.scope** 使用 systemd 的总线接口来程序化的创建，用以管理一系列系统进程。参见 systemd.scope(5).
- ***.target** 把其它单元配置文件分组，在启动的时候，来创建同步点。参见 systemd.target(5).

系统启动时 (即，init)，systemd 进程会尝试启动”/lib/systemd/system/default.target (通常是到”graphical.ta
的符号链接)。首先，一些特殊的 target 单元 (参见 systemd.special(7))，比如”local-fs.target”、”swap.target”
和”cryptsetup.target” 会被引入以挂载文件系统。之后，其它 target 单元也会根据单元依赖关系而被引入。详细
情况，请阅读 bootup(7)。

systemd 提供向后兼容的功能。在”/etc/init.d/rc[0123456S].d/[KS]<name>” 里面的 SysV 风格的启动脚本
仍然会被分析；telinit(8) 会被转换为 systemd 的单元活动请求。

小心
模拟的运行级别 2 到 4 全部被符号链接到了相同的“multi-user.target”。

3.2.1 主机名

内核维护系统主机名。在启动的时候，通过 systemd-hostnamed.service 启动的系统单位设置系统的主机名，此
主机名保存在”/etc/hostname”。这个文件应该只包含系统主机名，而不是全称域名。

不带参数运行 hostname(1) 命令可以打印出当前的主机名。

3.2.2 文件系统

硬盘和网络文件系统的挂载选项可以在”/etc/fstab” 中设置，参见 fstab(5) 和第 9.5.7 节。

加密文件系统的配置设置在“/etc/crypttab”中。参见 crypttab(5)

软 RAID 的配置 mdadm(8) 设置在”/etc/mdadm/mdadm.conf”. 参见 mdadm.conf(5).

警告
每次启动的时候，在挂载了所有文件系统以后，”/tmp”, ”/var/lock”, 和”/var/run” 中的临时文件会被清
空。

3.2.3 网络接口初始化

对于使用 systemd 的现代 Debian 桌面系统，网络接口通常由两个服务进行初始化：lo 接口通常在“networking.service”
处理，而其它接口则由“NetworkManager.service”处理。

参见第 5 章来获取怎样来配置它们的信息。

3.2.4 内核消息

在控制台上显示的内核错误信息，能够通过设置他们的阈值水平来配置。

```
# dmesg -n3
```

错误级别值	错误级别名称	说明
0	KERN_EMERG	系统不可用
1	KERN_ALERT	行为必须被立即采取
2	KERN_CRIT	危险条件
3	KERN_ERR	错误条件
4	KERN_WARNING	警告条件
5	KERN_NOTICE	普通但重要的条件
6	KERN_INFO	信息提示
7	KERN_DEBUG	debug 级别的信息

Table 3.4: 内核错误级别表

3.2.5 系统消息

在 systemd 下, 内核和系统的信息都通过日志服务 systemd-journald.service (又名 journald) 来记录，放在"/var/log/journal" 下的不变的二进制数据，或放在"/run/log/journal/" 下的变化的二进制数据. 这些二进制日志数据，可以通过 journalctl(1) 命令来访问。

在 systemd 下, 系统日志工具 rsyslogd(8) 改变它的行为来读取变化的二进制数据 (代替 systemd 之前的默认的"/dev/log")，并能够创建传统的不变的 ASCII 系统日志数据。

"/etc/default/rsyslog" 和"/etc/rsyslog.conf" 能够自定义系统消息的日志文件和屏幕显示。参见 rsyslogd(8) 和 rsyslog.conf(5)，也可以参见第 9.2.2 节。

3.2.6 systemd 下的系统管理

systemd 不仅仅提供系统初始化，还提供通用的系统管理功能。比如说日志记录，登录管理，时间管理，网络管理等等。

systemd(1) 通过几个命令来管理：

- systemctl(1) 命令控制 systemd 的系统和服务管理器 (命令行) ,
- systemsdm(1) 命令控制 systemd 的系统和服务管理器 (图形界面),
- journalctl(1) 命令查询 systemd 日志,
- loginctl(1) 命令控制 systemd 登录管理器,
- systemd-analyze(1) 分析系统启动性能。

这里有一个典型的 systemd 管理命令片段列表。确切含义，请阅读相关 man 手册页。

这里, 上面例子中的"$unit"，可以是一个单元名 (后缀 .service 和 .target 是可选的)，或者，在很多情况下，也可以是匹配的多个单元 (shell 式样的全局通配符"*", "?", "[]"，通过使用 fnmatch(3) ，来匹配目前在内存中的所有单元的基本名称).

上面列子的系统状态改变命令，通常是通过"sudo" 来处理，用以获得需要的系统管理权限。

"systemctl status $unit|$PID|$device" 的输出使用有颜色的点 ("●") 来概述单元状态，让人看一眼就知道。

操作	类型	命令片段
用于服务管理的图形界面	图形界面	"systemadm" (systemd-ui 软件包)
列出所有 target 单元配置	单元	"systemctl list-units --type=target"
列出所有 service 单元配置	单元	"systemctl list-units --type=service"
列出所有单元配置类型	单元	"systemctl list-units --type=help"
列出内存中所有 socket 单元	单元	"systemctl list-sockets"
列出内存中所有 timer 单元	单元	"systemctl list-timers"
启动"$unit"	单元	"systemctl start $unit"
停止"$unit"	单元	"systemctl stop $unit"
重新加载服务相关的配置	单元	"systemctl reload $unit"
停止和启动所有"$unit"	单元	"systemctl restart $unit"
启动"$unit" 并停止所有其它的	单元	"systemctl isolate $unit"
转换到" 图形" (图形界面系统)	单元	"systemctl isolate graphical"
转换到" 多用户" (命令行系统)	单元	"systemctl isolate multi-user"
转换到" 应急模式" (单用户命令行系统)	单元	"systemctl isolate rescue"
向"$unit" 发送杀死信号	单元	"systemctl kill $unit"
检查"$unit" 服务是否是活动的	单元	"systemctl is-active $unit"
检查"$unit" 服务是否是失败的	单元	"systemctl is-failed $unit"
检查"$unit\|$PID|device" 的状态	单元	"systemctl status $unit\|$PID|$device"
显示"$unit\|$job" 的属性	单元	"systemctl show $unit\|$job"
重设失败的"$unit"	单元	"systemctl reset-failed $unit"
列出所有单元服务的依赖性	单元	"systemctl list-dependencies --all"
列出安装在系统上的单元文件	单元文件	"systemctl list-unit-files"
启用"$unit" (增加符号链接)	单元文件	"systemctl enable $unit"
禁用"$unit" (删除符号链接)	单元文件	"systemctl disable $unit"
取消遮掩"$unit" (删除到"/dev/null" 的符号链接)	单元文件	"systemctl unmask $unit"
遮掩"$unit" (增加到"/dev/null" 的符号链接)	单元文件	"systemctl mask $unit"
获取默认的 target 设置	单元文件	"systemctl get-default"
设置默认 target 为"graphical" (图形系统)	单元文件	"systemctl set-default graphical"
设置默认的 target 为"multi-user" (命令行系统)	单元文件	"systemctl set-default multi-user"
显示工作环境变量	环境变量	"systemctl show-environment"
设置环境变量"variable" 的值为"value"	环境变量	"systemctl set-environment variable=value"
取消环境变量"variable" 的设置	环境变量	"systemctl unset-environment variable"
重新加载所有单元文件和后台守护进程 (daemon)	生命周期	"systemctl daemon-reload"
关闭系统	系统	"systemctl poweroff"
关闭和重启系统	系统	"systemctl reboot"
挂起系统	系统	"systemctl suspend"
休眠系统	系统	"systemctl hibernate"
查看"$unit" 的工作日志	日志	"journalctl -u $unit"
查看"$unit" 的工作日志 ("tail -f" 式样)	日志	"journalctl -u $unit -f"
显示每一个初始化步骤所消耗的时间	分析	"systemd-analyze time"
列出所有单元的初始化时间	分析	"systemd-analyze blame"
加载"$unit" 文件并检测错误	分析	"systemd-analyze verify $unit"
跟踪 cgroups(7) 的启动过程	Cgroup	"systemd-cgls"
跟踪 cgroups(7) 的启动过程	Cgroup	"ps xawf -eo pid,user,cgroup,args"

- 白色的”●” 表示一个” 不活动” 或” 变为不活动中” 的状态。
- 红色的”●” 表示“失败”或者“错误”状态。
- 绿色”●” 表示“活动”、“重新加载中”或“激活中”状态。

3.2.7 定制 systemd

使用默认安装，通过 systemd 启动的过程中，在 network.target 启动后，很多网络服务 (参见第 6 章) 作为后台守护进程（daemon）启动。”sshd” 也不列外。让我们修改为按需启动”sshd” 作为一个定制化的例子。

首先，禁用系统安装的服务单元。

```
$ sudo systemctl stop sshd.service
$ sudo systemctl mask sshd.service
```

传统 Unix 服务的按需套接字激活（on-demand socket activation）系统由 indetd 超级服务来提供。在 systemd 下, 相同功能能够通过增加 ***.socket** 和 ***.service** 单元配置文件来启用。

sshd.socket 用来定义一个监听的套接字

```
[Unit]
Description=SSH Socket for Per-Connection Servers

[Socket]
ListenStream=22
Accept=yes

[Install]
WantedBy=sockets.target
```

sshd@.service 作为 sshd.socket 匹配的服务文件

```
[Unit]
Description=SSH Per-Connection Server

[Service]
ExecStart=-/usr/sbin/sshd -i
StandardInput=socket
```

然后重新加载。

```
$ sudo systemctl daemon-reload
```

3.3 udev 系统

对于 Linux 内核 2.6 版和更新版本，udev 系统 提供了自动硬件发现和初始化机制。(参见 udev(7)). 在内核发现每个设备的基础上，udev 系统使用从 sysfs 文件系统 (参见第 1.2.12 节) 的信息启动一个用户进程，使用 modprobe(8) 程序 (参见第 3.3.1 节) 加载支持它所要求的内核模块, 创建相应的设备节点。

提示
如果由于某些理由，"/lib/modules/<kernel-version>/modules.dep" 没有被 depmod(8) 正常生成，模块可能不会被 udev 系统按期望的方式加载。执行"depmod -a" 来修复它。

设备节点的名字，可以通过"/etc/udev/rules.d/" 里的 udev 文件来配置. 当前默认的规则倾向创建动态生成的名字，除了光驱和网络设备外，会生成非静态的设备名。通过添加和光驱、网络设备类似的个性化规则，你也可以为 USB 盘之类的其它设备，生成静态设备名。参见"Writing udev rules" 或"/usr/share/doc/udev/writing_udev_rules/index.h

由于 udev 系统是一个正在变化的事物，我在其它文档进行了详细描述，在这里只提供了最少的信息。

提示
"/etc/fstab" 里面的挂载规则, 设备节点不必需是静态的。你能够使用 UUID 来挂载设备，来代替"/dev/sda" 之类的设备名. 参见第 9.5.3 节.

3.3.1 内核模块初始化

通过 modprobe(8) 程序添加和删除内核模块，使我们能够从用户进程来配置正在运行的 Linux 内核。udev 系统 (参见第 3.3 节) 自动化它的调用来帮助内核模块初始化。

下面的非硬件模块和特殊的硬件驱动模块，需要被预先加载，把它们在"/etc/modules" 文件里列出 (参见 modules(5)).

- TUN/TAP 模块提供虚拟的 Point-to-Point 网络设备 (TUN) 和虚拟的 Ethernet 以太网网络设备 (TAP),
- netfilter 模块提供 netfilter 防火墙能力 (iptables(8), 第 5.10 节),
- watchdog timer 驱动模块。

modprobe(8) 程序的配置文件是按 modprobe.conf(5) 的说明放在"/etc/modprobes.d/" 目录下，(如果你想避免自动加载某些内核模块，考虑把它们作为黑名单放在"/etc/modprobes.d/blacklist" 文件里.)

"/lib/modules/<version>/modules.dep" 文件由 depmod(8) 程序生成，它描述了 modprobe(8) 程序使用的模块依赖性.

注意
如果你在启动时出现模块加载问题，或者 modprobe(8) 时出现模块加载问题, "depmod -a" 可以通过重构"modules.dep" 来解决这些问题。

modinfo(8) 程序显示 Linux 内核模块信息。

lsmod(8) 程序以好看的格式展示"/proc/modules" 的内容, 显示当前内核加载了哪些模块。

提示
你能够精确识别你系统上的硬件。参见第 9.4.3 节.

提示
你可以在启动时配置硬件来激活期望的硬件特征。参见第 9.4.4 节.

提示
你可以重新编译内核来增加你的特殊设备的支持。参见第 9.9 节.

Chapter 4

认证

当用户（或程序）需要访问系统时，需要进行认证，确认身份是受信任。

警告
PAM 的配置错误可能会锁住你的系统。你必须有一个准备好的救援 CD，或者设立一个替代的 boot 分区。为了恢复系统，你需要使用它们启动系统并纠正错误。

警告
本章是基于 2013 年发布的 Debian 7.0 (wheezy) 编写的，所以其内容正在变得过时。

4.1 一般的 Unix 认证

一般的 Unix 认证由 PAM（Pluggable Authentication Modules，即可插入的验证模块）下的 pam_unix(8) 模块提供。它的 3 个重要文件如下，其内的条目使用“:”分隔。

文件	权限	用户	组	说明
/etc/passwd	-rw-r--r--	root	root	（明文的）用户账号信息
/etc/shadow	-rw-r-----	root	shadow	安全加密的用户账号信息
/etc/group	-rw-r--r--	root	root	组信息

Table 4.1: pam_unix(8) 使用的 3 个重要配置文件

“/etc/passwd”包含下列内容。

```
 ...
user1:x:1000:1000:User1 Name,,,:/home/user1:/bin/bash
user2:x:1001:1001:User2 Name,,,:/home/user2:/bin/bash
 ...
```

如 passwd(5) 中所述，这个文件中被“:”分隔的每项含义如下。

- 登录名

- 密码形式说明
- 数字形式的用户 ID
- 数字形式的组 ID
- 用户名或注释字段
- 用户家目录
- 可选的用户命令解释器

“/etc/passwd”的第二项曾经被用来保存加密后的密码。在引入了“/etc/shadow”后，该项被用来说明密码形式。

内容	说明
（空）	无需密码的账号
x	加密后的密码保存在“/etc/shadow”
*	无法登陆的账号
!	无法登陆的账号

Table 4.2: “/etc/passwd”第二项的内容

“/etc/shadow”包含下列内容。

```
 ...
user1:$1$Xop0FYH9$IfxyQwBe9b8tiyIkt2P4F/:13262:0:99999:7:::
user2:$1$vXGZLVbS$ElyErNf/agUDsm1DehJMS/:13261:0:99999:7:::
 ...
```

如 shadow(5) 中所述，这个文件中被“:”分隔的每项含义如下。

- 登录名
- 加密后的密码（开头的“1”表示使用 MD5 加密。“*”表示无法登陆。）
- 最后一次修改密码的时间，其表示从 1970 年 1 月 1 日起的天数
- 允许用户再次修改密码的天数间隔
- 用户必须修改密码的天数间隔
- 密码失效前的天数，在此期间用户会被警告
- 密码失效后的天数，在次期间密码依旧会被接受
- 账号失效的时间，其表示从 1970 年 1 月 1 日起的天数
- …

“/etc/group”包含下列内容。

```
group1:x:20:user1,user2
```

如 group(5) 中所述，这个文件中被“:”分隔的每项含义如下。

- 组名称

- 加密后的密码（不会被真正使用）
- 数字形式的组 ID
- 使用“,”分隔的用户名列表

注意
“/etc/gshadow”为“/etc/group”提供了与“/etc/shadow”相似的功能，但没有被真正地使用。

注意
如果”auth optional pam_group.so” 这行添加到了”/etc/pam.d/common-auth”，并且在”/etc/security/group.conf” 里进行了设置，一个用户的实际组就可以被动态添加。参见 pam_group(8).

注意
base-passwd 软件包包含了一份用户和组的官方文档:“/usr/share/doc/base-passwd/users-and-groups.html

---

4.2 管理账号和密码信息

下面是一些管理账号信息的重要命令。

命令	功能
getent passwd <user_name>	浏览“<user_name>”的账号信息
getent shadow <user_name>	浏览用户”<user_name>” 隐藏的账户信息
getent group <group_name>	浏览“<group_name>”的组信息
passwd	管理账号密码
passwd -e	为激活的账号设置一次性的密码
chage	管理密码有效期信息

Table 4.3: 管理账号信息的命令

其中的一些功能只能被 root 使用。密码和数据的加密参见 crypt(3)。

注意
在设置了 PAM 和 NSS 的系统上（例如 Debian salsa 机器），本地的“/etc/passwd”、“/etc/group”和“/etc/shadow”可能不会被系统激活使用。上述的命令即使处于这种环境下依旧是有效的。

4.3 好密码

在系统安装时建立一个账号或使用 passwd(1) 命令时，你应该选择一个好密码，它应该由 6 到 8 个字符组成，其中包含下列根据 passwd(1) 设定的每个组合中的一个或多个字符。

- 小写字母
- 数字 0 到 9
- 标点符号

警告
密码中不要使用可以猜到的词。账号名、身份证号码、电话号码、地址、生日、家庭成员或宠物的名字、字典单词、简单的字符序列（例如“12345”或“qwerty”）等都是糟糕的选择。

4.4 设立加密的密码

下面是一些用于 生成加盐的加密密码 的独立工具。

软件包	流行度	大小	命令	功能
whois	V:42, I:516	355	mkpasswd	具备 crypt(3) 库所有特性的前端
openssl	V:808, I:992	1452	openssl passwd	计算密码哈希 (OpenSSL). passwd(1ssl)

Table 4.4: 生成密码的工具

4.5 PAM 和 NSS

现代的类 Unix 系统（例如 Debian 系统）提供 PAM（Pluggable Authentication Modules，插入式验证模块）和 NSS（Name Service Switch，名称服务切换）机制给本地系统管理员，使他们能够配置自己的系统。它们的功能可以概括为以下几点。

- PAM 给应用软件提供了一个灵活的认证机制，因此涉及到了密码数据的交换。
- NSS 提供了一个灵活的名称服务机制，它经常被 C 标准库使用，使例如 ls(1) 和 id(1) 这样的程序获得用户和组名称。

PAM 和 NSS 系统必须保持配置一致。

PAM 和 NSS 系统中重要的软件包如下。

软件包	流行度	大小	说明
libpam-modules	V:842, I:999	1059	插入式验证模块（基础服务）
libpam-ldap	V:1, I:14	244	允许 LDAP 接口的插入式验证模块
libpam-cracklib	I:16	116	启用 cracklib 支持的插入式验证模块
libpam-systemd	V:454, I:760	393	用于 logind 注册用户会话的插入式验证模块（PAM）
libpam-doc	I:1	1031	插入式验证模块（html 和文本文档）
libc6	V:937, I:999	12333	GNU C 库：同样提供“名称服务切换”服务的共享库
glibc-doc	I:13	2995	GNU C 库：帮助页面
glibc-doc-reference	I:5	13278	GNU C 库：参考手册，有 info、pdf 和 html 格式（non-free）
libnss-mdns	I:557	119	用于解析组播 DNS 名称的 NSS 模块
libnss-ldap	I:12	255	NSS 模块，用于使用 LDAP 作为一个名称服务的
libnss-ldapd	I:19	152	NSS 模块，用于使用 LDAP 作为一个名称服务的（libnss-ldap 的新 fork）

Table 4.5: PAM 和 NSS 系统中重要的软件包

- libpam-doc 中“The Linux-PAM System Administrators’ Guide”是了解 PAM 配置的必要文档。

- `glibc-doc-reference` 中的“System Databases and Name Service Switch”是了解 NSS 配置的重要文档。

注意
你可以使用“`aptitude search 'libpam-|libnss-'`”命令查看更多的相关软件包。NSS 缩写也可能意味着“Network Security Service，网络安全服务”，它不同于“Name Service Switch，名称服务切换”。

注意
PAM 是为每个程序初始化环境变量为系统默认值的最基础方法。

在 systemd 下, `libpam-systemd` 软件包被安装用来管理用户登录，通过在 `systemd` 控制组层，为 logind 注册用户会话来实现。参见 systemd-logind(8), logind.conf(5), 和 pam_systemd(8).

4.5.1 PAM 和 NSS 访问的配置文件

下面是一些 PAM 和 NSS 访问的重要配置文件。

配置文件	功能
`/etc/pam.d/<program_name>`	为“`<program_name>`”程序设置 PAM 配置；参加 pam(7) 和 pam.d(5)
`/etc/nsswitch.conf`	为每个服务条目设置 NSS 配置。参见 nsswitch.conf(5)
`/etc/nologin`	通过 pam_nologin(8) 模块限制用户登陆
`/etc/securetty`	通过 pam_securetty(8) 模块限制 root 访问 tty
`/etc/security/access.conf`	通过 pam_access(8) 模块设置访问限制
`/etc/security/group.conf`	通过 pam_group(8) 模块设置基于组的限制
`/etc/security/pam_env.conf`	通过 pam_env(8) 模块设置环境变量
`/etc/environment`	通过带有“`readenv=1`”参数的 pam_env(8) 模块设置额外的环境变量
`/etc/default/locale`	通过带有“`readenv=1 envfile=/etc/default/locale`”参数的 pam_env(8) 模块设置语言环境值（在 Debian 系统中）
`/etc/security/limits.conf`	通过 pam_linits(8) 模块设置资源限制（ulimit、core 等等）
`/etc/security/time.conf`	通过 pam_time(8) 模块设置时间限制
`/etc/systemd/logind.conf`	设置 `systemd` 的登录管理器配置 (参见 logind.conf(5) 和 systemd-logind.service(8))

Table 4.6: PAM 和 NSS 访问的配置文件

密码选择的限制是通过 PAM 模块 pam_unix(8) 和 pam_cracklib(8) 来实现的。它们可以通过各自的参数进行配置。

提示
PAM 模块在文件名中使用后缀“`.so`”。

4.5.2 现代的集中式系统管理

现代的集中式系统管理可以使用集中式的轻量目录访问协议（LDAP）服务器进行部署，从而通过网络管理许多类 Unix 和非类 Unix 系统。轻量目录访问协议的开源实现是 OpenLDAP 软件。

LDAP 服务器使用带有 PAM 和 NSS 的 `libpam-ldap` 和 `libnss-ldap` 软件包为 Debian 系统提供账号信息。需要一些动作来启用 LDAP（我没有使用过这个设置，并且下面的信息纯粹是第二手的信息。请在这种前提下阅读下列内容。）。

- 你通过运行一个程序，例如独立的 LDAP 守护进程 slapd(8)，来建立集中式的 LDAP 服务器。
- 你在“/etc/pam.d/”目录中的 PAM 配置文件里，使用“pam_ldap.so”替代默认值“pam_unix.so”。
  - Debian 使用“/etc/pam_ldap.conf”作为 libpam-ldap 的配置文件，“/etc/pam_ldap.secret”作为保存 root 密码的文件。
- 你在“/etc/nsswitch.conf”文件中改变 NSS 配置，使用“ldap”替代默认值（“compat”或“file”）。
  - Debian 使用“/etc/libnss-ldap.conf”作为 libnss-ldap 的配置文件。
- 为了密码的安全，你必须让 libpam-ldap 使用 SLL（或 TLS）连接。
- 为了确保 LDAP 网络开销数据的完整性，你必须让 libpam-ldap 使用 SLL（或 TLS）连接。
- 为了减少 LDAP 网络流量，你应该在本地运行 nscd(8) 来缓存任何 LDAP 搜索结果。

参见由 libpam-doc 软件包提供的 pam_ldap.conf(5) 中的文档和“/usr/share/doc/libpam-doc/html/”，以及 glibc-doc 软件包提供的“info libc 'Name Service Switch'”。

类似地，你可以使用其它方法来设置另一种集中式的系统。

- 同 Windows 系统集成用户和组。
  - 通过 winbind 和 libpam_winbind 软件包访问 Windows domain 服务.
  - 参见 winbindd(8) 和 Integrating MS Windows Networks with Samba。
- 同古老的类 Unix 系统集成用户和组。
  - 通过 nis 软件包访问 NIS (之前叫 YP) 或 NIS+。
  - 参见 The Linux NIS(YP)/NYS/NIS+ HOWTO。

4.5.3 “为什么 GNU su 不支持 wheel 组”

这是在旧的“info su”底部 Richard M. Stallman 所说的一句名言。别担心：Debian 系统中当前的 su 命令使用了 PAM，这样当在“/etc/pam.d/su”中启用了带有“pam_wheel.so”的行后，就能够限制非 wheel 组的用户 su 到 root 组的能力。

4.5.4 严格的密码规则

安装 libpam-cracklib 软件包你能够强制使用严格的密码规则，例如，通过在“/etc/pam.d/common-password”中添加下列行。

对于 squeeze 发行版：

```
password required pam_cracklib.so retry=3 minlen=9 difok=3
password [success=1 default=ignore] pam_unix.so use_authtok nullok md5
password requisite pam_deny.so
password required pam_permit.so
```

4.6 其它的访问控制

注意
参见第 9.3.15 节来限制内核的安全警告密钥（SAK）功能。

4.6.1 sudo

sudo(8) 程序是为了使一个系统管理员可以给用户受限的 root 权限并记录 root 活动而设计的。sudo 只需要一个普通用户的密码。安装 sudo 软件包并通过设置“/etc/sudoers”中的选项来使用它。参见“/usr/share/doc/sudo/examples/sud 和第 1.1.12 节中的配置示例。

我将 sudo 用于单用户系统（参见第 1.1.12 节）是为了防止自己可能做出的愚蠢行为。就我个人而言，我认为使用 sudo 会比使用 root 账号操作系统来得好。例如，下列命令将“<some_file>”的拥有者改变为“<my_name>”。

```
$ sudo chown <my_name> <some_file>
```

当然如果你知道 root 密码（比如自行安装 Debian 的用户所做的），任何用户账号都可以使用“su -c”让任何命令以 root 运行。

4.6.2 PolicyKit

PolicyKit 是在类 Unix 操作系统中控制整个系统权限的一个操作系统组件。

较新的 GUI 图形界面程序设计时便考虑到了不作为特权进程来运行。它们通过 PolicyKit 来和特权进程通信，从而执行管理操作。

在 Debian 系统中，PolicyKit 限制了属于 sudo 组的用户账号的这种操作。

参见 polkit(8)。

4.6.3 SELinux

Security-Enhanced Linux (SELinux) 是一个收紧权限模块的框架，它比普通的类 Unix 安全模块 mandatory access control (MAC) 策略更严格。root 权限在某些条件下被限制。

4.6.4 限制访问某些服务端的服务

对系统安全而言，尽可能的禁用服务程序，是一个好的主意。网络服务是危险的。有不使用的服务，不管是直接由后台守护进程（daemon）激活，还是通过super-server 程序激活，都被认为是安全风险。

许多程序, 比如说 sshd(8), 使用基于 PAM 的访问控制。也还有许多方式来限制访问一些服务端的程序。

- 配置文件: ”/etc/default/<program_name>”
- 后台守护进程（daemon）的服务单元配置
- PAM (Pluggable Authentication Modules)
- super-server 使用”/etc/inetd.conf”
- TCP wrapper 使用”/etc/hosts.deny” 和”/etc/hosts.allow” , tcpd(8)
- Sun RPC” 使用 /etc/rpc.conf”
- atd(8) 使用”/etc/at.allow” 和”/etc/at.deny”
- crontab(1) 使用”/etc/cron.allow” 和”/etc/cron.deny”
- Network firewall 或netfilter 框架

参见第 3.2.6 节, 第 4.5.1 节, 和第 5.10 节.

提示
NFS 和其它基于 RPC 的程序，需要激活 Sun RPC 服务。

提示
如果你远程访问最新的 Debian 系统有问题，看下在”/etc/hosts.deny” 里是否存在”ALL: PARANOID” 这样讨厌的配置，请把它注释掉。(但是你必须注意这种行为所带来的安全风险。)

4.7 安全认证

注意
这里的信息也许不能完全满足你的安全需求，但这里应当是一个好的起点。

4.7.1 确保互联网上的的密码安全

许多流行的传输层服务都使用纯文本来传输包括密码验证信息在内的各类消息。使用纯文本在公网上传输密码是很糟糕的做法，因为这样传输的密码很容易在网上被他人截获。为了确保整个沟通过程，包括密码信息在内都使用加密传输来确保安全，您可以在“传输层安全（Transport Layer Security，TLS）”协议或者其前身，“安全套接字层（Secure Sockets Layer，SSL）”协议之上运行这些服务。

不安全的服务名	端口	安全的服务名	端口
www (http)	80	https	443
smtp (邮件)	25	ssmtp (smtps)	465
ftp-data	20	ftps-data	989
ftp	21	ftps	990
telnet	23	telnets	992
imap2	143	imaps	993
pop3	110	pop3s	995
ldap	389	ldaps	636

Table 4.7: 安全和不安全的服务端口列表

加密消耗 CPU 时间。作为对 CPU 有益的替代方案，你可以保持使用纯文本通讯，仅仅使用安全认证协议加密密码，比如说：POP 使用”Authenticated Post Office Protocol” (APOP)，SMTP 和 IMAP 使用”Challenge-Response Authentication Mechanism MD5” (CRAM-MD5)。(你的邮件客户端通过互联网上你的邮件服务器发送邮件时，最近流行使用新的递交端口 587 来代替传统的 SMTP 端口 25，这样可以避免在使用 CRAM-MD5 认证自己时，网络提供商阻塞 25 端口。)

4.7.2 安全 Shell

安全 Shell (SSH) 程序使用安全认证来提供不安全网络上两个不可信任主机之间的安全加密通讯。它由 OpenSSH 客户端, ssh(1), 和 OpenSSH 后台守护进程（daemon）, sshd(8) 组成.SSH 使用端口转发特性，可以给 POP 和 X 之类的不安全的协议通讯建立隧道，使其可以在互联网上安全传输。

客户端可以使用如下方式来认证自己：基于主机的认证、公钥认证、质疑应答认证、密码认证。使用公钥认证，可以实现远程免密码登录。参见第 6.9 节.

4.7.3 互联网额外的安全方式

即使你运行 Secure Shell (SSH) 和 Point-to-point tunneling protocol (PPTP) 这样的安全服务，在互联网上，仍然有机会使用野蛮暴力猜测密码攻击进入。使用防火墙策略 (参见第 5.10 节)，并和下面的安全工具一起，可以提升安全形势。

软件包	流行度	大小	说明
knockd	V:0, I:2	89	小的 port-knock 后台守护进程（daemon）knockd(1) 和客户端 konck(1)
fail2ban	V:103, I:114	1735	禁用造成多个认证错误的 IP
libpam-shield	V:0, I:0	115	把尝试猜测密码的远程攻击者关在外面

Table 4.8: 提供额外安全方式的工具列表

4.7.4 root 密码安全

为阻止人们使用 root 权限访问你的机器，你需要做下面的操作。

- 阻止对硬盘的物理访问
- 锁住 BIOS 来阻止从可移动介质启动
- 为 GRUB 交互式会话设置密码
- 锁住 GRUB 菜单，禁止编辑

如果可以物理访问硬盘，则可以使用下面的步骤，相对简单的重置密码。

1. 将硬盘拿到一个可以设置 BIOS 从 CD 启动的电脑。
2. 使用紧急介质启动系统（Debian 启动磁盘, Knoppix CD, GRUB CD, ⋯）。
3. 用读写访问挂载根分区。
4. 编辑根分区的"`/etc/passwd`"文件，使 `root` 账户条目的第二段为空。

对于 `grub-rescue-pc`，即使用紧急介质启动的电脑，如果有编辑 GRUB 菜单条目 (参见第 3.1.2 节) 的权限，在启动时，使用下面的步骤更加简单。

1. 使用内核参数启动系统来修改一些事情，比如说，"`root=/dev/hda6 rw init=/bin/sh`".
2. 编辑"`/etc/passwd`"文件，使 `root` 账户条目的第二段为空。
3. 重启系统。

系统的 root shell 现在可以无密码访问了。

注意
一旦某人拥有 root shell 访问权限，他能够访问任何内容，并可以重设系统上的任何密码。此外，他可以使用 `john` 和 `crack` 等软件包的暴力破解工具来比较所有用户的密码 (参见第 9.4.11 节)。被破解的密码，可以用来和其它系统进行比较。

为避免这些相关问题，仅有的理论上的软件解决方案是使用 dm-crypt 和 initramfs (参见第 9.8 节) 加密 root 分区 (或"`/etc`"分区)。这样的话，你总是需要密码来启动系统。

Chapter 5

网络设置

提示
关于 GNU/Linux 网络的通用手册，请查看Linux 网络管理员手册。

提示
关于 Debian 专属的网络手册，请查看Debian 管理员手册—网络配置。

警告
为代替使用传统的网络接口名称的方案 (”eth0”, ”eth1”, ”wlan0”, ⋯), 新的 systemd 使用”enp0s25” 之类的”可预测网络接口名称”.

警告
本章是基于 2013 年发布的 Debian 7.0 (wheezy) 编写的，所以其内容正在变得过时。

提示
尽管本手册仍用旧的 ifconfig(8) 命令和 IPv4 协议当作网络配置的例子，Debian 在 wheezy 发行版后转向使用 ip(8) 命令和 IPv4+IPv6 协议。欢迎大家提供补丁，更新这个手册。

提示
systemd环境下，可以用networkd来配置网络。请参考 systemd-networkd(8)。

5.1 基本网络架构

让我们来回顾一下现代 Debian 操作系统中的基本网络架构。

软件包	流行度	大小	类型	说明
ifupdown	V:627, I:995	217	配置:: ifupdown	用来启动/关闭网络的标准工具（Debian 特有）
ifplugd	V:4, I:21	209	同上	自动管理有线网络
ifupdown-extra	V:0, I:1	100	同上	网络测试脚本，加强”ifupdown” 软件包的功能
ifmetric	V:0, I:1	37	同上	设置网络接口的路由度量
guessnet	V:0, I:0	422	同上	脚本文件，利用”/etc/network/interfaces” 文件来加强”ifupdown” 的功能
ifscheme	V:0, I:0	58	同上	映射脚本文件，增强”ifupdown” 软件包的功能
network-manager	V:380, I:471	11584	配置:: NM	NetworkManager（守卫进程）：自动管理网络
network-manager-gnome	V:159, I:408	5921	同上	NetworkManager（GNOME 前端）
wicd	I:31	35	配置:: wicd	有线和无线网络管理器（元软件包）
wicd-cli	V:0, I:1	59	同上	有线和无线网络管理器（命令行客户端）
wicd-curses	V:0, I:4	175	同上	有线和无线网络管理器（文本界面客户端）
wicd-daemon	V:26, I:35	962	同上	有线和无线网络管理器（守护进程）
wicd-gtk	V:21, I:33	574	同上	有线和无线网络管理器（GTK+ 客户端）
iptables	V:270, I:995	2569	配置:: Netfilter	封包过滤和网络地址转换管理工具（Netfilter）
iproute2	V:671, I:871	2585	配置:: iproute2	iproute2, IPv6 和其他高级网络配置：ip(8),tc(8) 等等
ifrename	V:1, I:2	125	同上	根据不同的静态标准来重命名网络接口：ifrename(8)
ethtool	V:110, I:259	393	同上	显示或更改以太网设备的设定
iputils-ping	V:254, I:996	100	测试:: iproute2	测试能否连接远程主机，通过主机名或IP 地址（iproute2）
iputils-arping	V:26, I:392	51	同上	测试能否连接远程主机，通过ARP地址
iputils-tracepath	V:5, I:102	68	同上	跟踪访问远程主机的路径
net-tools	V:299, I:744	979	配置:: net-tools	NET-3 网络工具箱（net-tools，IPv4 网络配置）：ifconfig(8) 等等。
inetutils-ping	V:0, I:2	350	测试:: net-tools	测试能否连接远程主机，通过hostname或IP 地址（传统方式，GNU）
arping	V:1, I:28	73	同上	测试能否连接远程主机，通过ARP地址（传统方法）
traceroute	V:63, I:960	154	同上	跟踪连接远程主机的路径（传统方法，控制台）
isc-dhcp-client	V:255, I:973	673	配置:: 底层	DHCP 客户端
wpasupplicant	V:310, I:539	3352	同上	WPA 和 WPA2 客户端支持（IEEE 802.11i）
wpagui	V:0, I:3	786	同上	wpa_supplicant Qt 图形界面客户端
wireless-tools	V:192, I:274	297	同上	操控 Linux 无线扩展的工具
ppp	V:264, I:510	1020	同上	使用 chat 连接 PPP/PPPoE
pppoeconf	V:0, I:9	290	配置:: 辅助	配置助手，以便于使用 PPPoE 连接
pppconfig	V:1, I:2	805	同上	配置助手，以便于使用 chat 连接 PPP
wvdial	V:0, I:6	249	同上	配置助手，以便于使用 wvdial 和 ppp 连接 PPP
mtr-tiny	V:6, I:55	152	测试:: 底层	追踪连接远程主机的路径（文本界面）
mtr	V:5, I:40	206	同上	追踪连接远程主机的路径（文本界面和 GTK+ 界面）
gnome-nettool	V:3, I:91	2105	同上	获取常见网络信息的工具（GNOME）
nmap	V:36, I:292	4510	同上	网络映射/端口扫描（Nmap，控制台）
zenmap	V:4, I:12	2939	同上	网络映射/端口扫描（GTK+）
tcpdump	V:21, I:200	1192	同上	网络流量分析（Tcpdump，控制台）
wireshark	I:61	69	同上	网络流量分析（Wireshark,GTK+）
tshark	V:3, I:35	398	同上	网络流量分析 (控制台）
tcptrace	V:0, I:1	392	同上	根据 tcpdump 的输出生成的连接数据统计
snort	V:0, I:1	1920	同上	灵活的网络入侵侦测系统（Snort）
ntopng	V:1, I:2	950	同上	在网页浏览器中展示网络流量
dnsutils	V:71, I:591	719	同上	BIND软件包提供的网络客户端程序：nslookup(8),nsupdate(8),dig(8)
dlint	V:0, I:12	53	同上	利用域名服务器查询来查看DNS域信息
dnstracer	V:0, I:2	56	同上	跟踪DNS查询直至源头

5.1.1 主机名解析

主机名解析，目前也是由 NSS (名字服务转换 Name Service Switch) 机制来支持。这个解析的流程如下。

1. "/etc/nsswitch.conf" 文件里的"hosts: files dns" 这段规定主机名解析顺序。(代替"/etc/host.conf" 文件里的"order" 这段原有的功能。)
2. files 方式首先被调用。如果主机名在"/etc/hosts" 文件里面发现，则返回所有有效地址并退出。("/etc/host.conf 文件包含"multi on".)
3. dns 方式被调用。如果主机名通过查询"/etc/resolv.conf" 文件里面写的 互联网域名系统 Domain Name System (DNS) 来找到，则返回所有有效地址并退出。

例如, "/etc/hosts" 看起来如下。

```
127.0.0.1 localhost
127.0.1.1 <host_name>

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

每一行由 IP 地址 开始，接下来是相关联的主机名.

在这个例子的第二行 127.0.1.1 IP 地址也许不会在其它类 Unix 系统发现。Debian Installer 为没有永久 IP 地址的系统创建这个条目，作为某些软件（如 GNOME）的一个变通方法，见文档 bug #719621.

<host_name> 匹配在"/etc/hostname" 里定义的主机名。

对于有永久 IP 地址的系统，这个永久 IP 地址应当代替这里的 127.0.1.1。

对于有永久 IP 地址和有 域名系统 Domain Name System (DNS)提供完全资格域名 fully qualified domain name (FQDN) 的系统，规范名 <host_name>.<domain_name> 应当被用来代替 <host_name>.

如果 resolvconf 软件包没有安装，"/etc/resolv.conf" 是一个静态文件。如果安装了，它是一个符号链接。此外，它包含有解析策略的初始化信息。如 DNS 是 IP="192.168.11.1", 则包含如下。

```
nameserver 192.168.11.1
```

resolvconf 软件包使这个"/etc/resolv.conf" 文件成为一个符号链接，并通过钩子脚本自动管理其内容。

对于典型 adhoc 局域网环境下的 PC 工作站，除了基本的 files 和 dns 方式之外，主机名还能够通过组播 DNS (mDNS, 零配置网络 Zeroconf) 进行解析。

- Avahi 提供 Debian 下的组播 DNS 发现框架。
- 它和 Apple Bonjour / Apple Rendezvous 相当.
- libnss-mdns 插件包提供 mDNS 的主机名解析，GNU C 库 (glibc) 的 GNU 名字服务转换 Name Service Switch (NSS) 功能支持 mDNS。
- "/etc/nsswitch.conf" 文件应当有像"hosts: files mdns4_minimal [NOTFOUND=return] dns mdns4" 这样的一段.
- ".local"结尾的主机名，使用 pseudo-top-level domain (TLD) 来解析.

- mDNS IPv4 本地连接组播地址”224.0.0.251” 或它相应的 IPv6 地址”FF02::FB” 被用来作为”.local” 结尾名字的 DNS 查询。

较老的 Windows 系统安装 winbind 软件包来提供旧的 NETBios over TCP/IP 主机名解析。为启用这个功能，”/etc/nsswitch 文件应当有这样的一段: ”hosts: files mdns4_minimal [NOTFOUND=return] dns mdns4 wins”。(现代 Windows 系统通常使用 dns 方式来进行主机名解析。)

注意
域名系统 Domain Name System 中的扩展通用顶级域名 expansion of generic Top-Level Domains (gTLD) 还在进行中。在局域网内，选择一个域名时，请提防名字冲突 name collision。

5.1.2 网络接口名称

网络接口名称，比如说 eth0, 是在 Linux 内核里分配给每一个硬件的，当这个硬件被内核发现的时候，通过用户层的配置机制 udev (参见第 3.3 节) 来分配. 网卡接口名称也就是 ifup(8) 和 interfaces(5) 里的 **physical interface**。

为了保证每个网络接口名称在每次重启后一致，会用到 MAC 地址 等, 有一个规则文件”/etc/udev/rules.d/70-persist 这个文件是由”/lib/udev/write_net_rules” 程序自动生成，是由”persistent-net-generator.rules” 规则文件来运行. 你可以修改该文件来改变命名规则。

小心
当编辑”/etc/udev/rules.d/70-persistent-net.rules” 规则文件时，你必须保持每条规则在单独的一行中，并且 MAC 地址 要小写。比如说，如果你发现”FireWire device” 和”PCI device” 在这个文件中，你也许想命名”PCI device” 作为 eth0，并配置它为首要网络接口。

5.1.3 局域网网络地址范围

让我们重新提醒下在 rfc1918 里规定的局域网 local area networks (LANs)IPv4 32 位地址在各类地址的保留范围. 这些地址保证不会与因特网上专有的地址冲突。

类别	网络地址	子网掩码	子网掩码/位数	# 子网数
A	10.x.x.x	255.0.0.0	/8	1
B	172.16.x.x —172.31.x.x	255.255.0.0	/16	16
C	192.168.0.x —192.168.255.x	255.255.255.0	/24	256

Table 5.2: 网络地址范围列表

注意
如果这些地址分配到一个主机，那么这个主机一定不能够直接访问互联网，必须通过一个作为网关的代理服务或通过 网络地址转换 Network Address Translation (NAT). 消费局域网环境，宽带路由器通常使用 NAT。

5.1.4 网络设备支持

尽管 Debian 系统支持大多数硬件设备，但依旧有一些网络设备需要 DFSG non-free 固件来支持它们。参见第 9.9.6 节。

5.2 现代的桌面网络配置

对于使用 systemd 的现代 Debian 桌面系统，网络接口通常由两个服务进行初始化：lo 接口通常在“networking.service”处理，而其它接口则由“NetworkManager.service”处理。

Debian squeeze 和新的发行版都可以通过后台守护进程（daemon）管理软件来管理网络连接，例如 NetworkManager (NM)（network-manager 和相关软件包）或 Wicd（wicd 和相关软件包）。

- 它们有自己的 GUI 和命令行程序来作为用户界面。
- 它们有自己的后台守护进程（daemon）作为它们的系统后端。
- 它们使你可以简单地将系统连接到网络。
- 它们使你可以简单地管理有线和无线网络的配置。
- 它们允许你配置网络而不依赖传统的 ifupdown 软件包。

注意
不要在服务器上使用这些自动网络配置工具。它们主要针对于笔记本电脑上的移动桌面用户。

这些现代的网络配置工具需要进行适当的配置，以避免与传统 ifupdown 软件包发生冲突，它的配置文件位于“/etc/network/interfaces”。

注意
这些自动网络配置工具的一些功能可能会带来令人烦扰的问题。它们不像传统的 ifupdown 软件包那样健壮。检查 network-manager 的 BTS 和 wicd 的 BTS，来查看当前的问题和限制。

5.2.1 图形界面的网络配置工具

Debian 系统 NM 和 Wicd 的官方文档分别位于“/usr/share/doc/network-manager/README.Debian”和“/usr/share/doc/wicd/README.Debian”。

本质上，如下操作即可完成桌面的网络配置。

1. 通过下列命令使桌面用户 foo 归属“netdev”组（另外，例如 GNOME 和 KDE 这样的现代桌面环境会通过 D-bus 自动完成该操作）。

```
$ sudo adduser foo netdev
```

2. 使“/etc/network/interfaces”的配置保持下面那样简洁。

```
auto lo
iface lo inet loopback
```

3. 通过下列命令重新启动 NM 或 Wicd。

```
$ sudo /etc/init.d/network-manager restart
```

```
$ sudo /etc/init.d/wicd restart
```

4. 通过图形界面配置网络。

注意
只有不列在“/etc/network/interfaces”中的接口会被 NM 或 Wicd 管理，以避免与 ifupdown 的冲突。

提示
如果你想扩展 NM 的网络配置功能，请寻找适当的插件模块和补充软件包，例如 network-manager-openconnect、network-manager-openvpn-gnome、network-manager-pptp-gnome、mobile-broadband-provider-info、gnome-bluetooth 等等。这同样适用于 Wicd。

小心
这些自动网络配置工具可能无法兼容“/etc/network/interfaces”中传统的 ifupdown 的深奥配置，例如第 5.6 节和第 5.7 节中的那些配置。检查 network-manager 的 BTS 和 wicd 的 BTS 来查看当前的问题和限制。

5.3 没有图像界面的现代网络配置

使用 systemd 的系统中，可以在 /etc/systemd/network/ 里配置网络。参见 systemd-resolved(8)、resolved.conf(5) 和 systemd-networkd(8)。

这个允许在没有图像界面的情况下配置现代网络。

DHCP 客户端的配置可以通过创建”/etc/systemd/network/dhcp.network” 文件来进行设置。例如：

```
[Match]
Name=en*

[Network]
DHCP=yes
```

一个静态网络配置能够通过创建”/etc/systemd/network/static.network” 来设置. 比如：

```
[Match]
Name=en*

[Network]
Address=192.168.0.15/24
Gateway=192.168.0.1
```

5.4 传统的网络连接和配置

如果第 5.2 节中描述的方法无法满足你的需要，那你应该使用结合了许多普通工具的传统网络连接和配置方法。

传统网络连接的每个方法都是特定的（参见第 5.5 节）。

用于 Linux 底层网络配置的程序有两种类型（参见第 5.8.1 节）。

- 来自 Linux NET-3 网络系统的旧 net-tools 程序（ifconfig(8)…… ）。它们中的大多数都已经过时了。
- 来自现在的 Linux 网络系统的新 Linux iproute2 程序（ip(8)…… ）。

尽管底层程序十分强大，但它们使用繁琐。因此创建了高层网络配置系统。

ifupdown 软件包是 Debian 中这种高层网络配置系统的实际标准。它让你可以简单地通过例如“ifup eth0”这样的命令来打开网络。它的配置文件位于“/etc/network/interfaces”中并且其典型内容如下。

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp
```

resolvconf 软件包是为了使 ifupdown 系统支持自动重写解析器配置文件“/etc/resolv.conf”来使网络地址解析平滑地重新配置。现在，大多数 Debian 网络配置软件包都使用了 resolvconf 软件包（参见“/usr/share/doc/resolvc

ifupdown 软件包的辅助脚本，例如 ifplugd、guessnet、ifscheme 等，是为了进行网络环境的自动动态配置，例如位于有线局域网中的移动电脑。这些相对来说比较难使用，但在当前的 ifupdown 系统下工作良好。

详细的案例参见第 5.6 节和第 5.7 节。

5.5 网络连接方式（传统）

小心
在这节描述的连接测试方式仅仅用于测试目的。不应当直接用于日常的网络连接。建议你使用 NM, Wicd, 或 ifupdown 包代替。(参见第 5.2 节和第 5.6 节).

一台电脑典型的网络连接方式和连接路径，能够使用下面的内容概述。

PC	连接方式	连接路径
串口 (ppp0)	PPP	modem POTS 拨号接入点 ISP
以太网口 (eth0)	PPPoE/DHCP/Static	宽带-modem 宽带链路 宽带接入点 ISP
以太网口 (eth0)	DHCP/Static	LAN 网络地址转换 (NAT) 的宽带路由器 (宽带-modem …)

Table 5.3: 网络连接方式和连接路径列表

每种连接方式配置脚本汇总。

网络连接缩略语意义如下。

注意
通过电视线缆的广域网服务，通常使用 DHCP 或 PPPoE。ADSL 和 FTTP 通常使用 PPPoE。你需要咨询你的互联网服务提供商来获得广域网连接使用的精确配置。

注意
当宽带路由器用来搭建家庭局域网环境时，局域网上的电脑需要使用宽带路由器上的 网络地址转换（NAT），来连接到广域网。在这样的情况下，局域网上的电脑网络接口需使用静态 IP 或者宽带路由器提供的 DHCP 服务。宽带路由器必须按 ISP 的指引来配置连接到广域网。

连接方式	配置	后端包
PPP	`pppconfig` 创建固定的 chat	`pppconfig`, `ppp`
PPP (选用)	`wvdialconf` 创建启发式的 chat	`ppp`, `wvdial`
PPPoE	`pppoeconf` 创建固定的 chat	`pppoeconf`, `ppp`
DHCP	在”`/etc/dhcp/dhclient.conf`”里描述	`isc-dhcp-client`
静态 IP (IPv4)	在”`/etc/network/interfaces`”里描述	`iproute` 或 `net-tools` (旧)
静态 IP (IPv6)	在”`/etc/network/interfaces`”里描述	`iproute`

Table 5.4: 网络连接配置列表

缩略语	说明
POTS	普通老式电话服务
BB	宽带
BB-service	比如说，数字用户线路（DSL），电视线缆，光纤到户 (FTTP)
BB-modem	比如说，DSL modem, 线缆 modem, 或光纤网络终端 (ONT)
LAN	局域网
WAN	广域网
DHCP	动态主机配置协议
PPP	点到点协议
PPPoE	以太网上的点到点协议
ISP	互联网服务提供商

Table 5.5: 网络连接缩略语列表

5.5.1 以太网 DHCP 连接

典型的现代家庭和小的商业网络，也就是局域网，使用一些消费等级的宽带路由器连接到广域网（因特网）。路由器后面的局域网通常使用路由器上运行的动态主机配置协议（DHCP）服务端提供的服务。

仅仅需要安装 `isc-dhcp-client` 包为以太网提供动态主机配置协议 (DHCP)服务。

参见 dhclient.conf(5).

5.5.2 以太网静态 IP 连接

静态 IP 以太网不需要特别的配置动作。

5.5.3 使用 pppconfig 的 PPP 连接

配置脚本 `pppconfig` 配置 PPP 连接，仅需要交互式的选择下面内容。

- 电话号码
- ISP 用户名
- ISP 密码
- 端口速率
- modem 通信端口
- 认证方式

文件	功能
/etc/ppp/peers/<isp_name>	pppconfig 生成针对 <isp_name> 的 pppd 配置文件
/etc/chatscripts/<isp_name>	pppconfig 生成针对 <isp_name> 的 chat 配置文件
/etc/ppp/options	pppd 常用的执行参数
/etc/ppp/pap-secret	PAP 的鉴权认证数据 (有安全风险)
/etc/ppp/chap-secret	CHAP 的鉴权认证数据 (更安全)

Table 5.6: 使用 pppconfig 的 PPP 连接配置文件列表

小心
"<isp_name>" 是" 互联网服务提供商"，假设 pon 和 poff 命令调用时，没有参数。

你可以使用下面的底层网络配置工具测试配置。

```
$ sudo pon <isp_name>
...
$ sudo poff <isp_name>
```

参见"/usr/share/doc/ppp/README.Debian.gz".

5.5.4 使用 wvdialconf 的另一种可选的 PPP 连接

一个不同的使用 pppd(8) 方案是使用来自 wvdial 包的 wvdial(1)。代替 pppd 运行 chat(8) 来拨号和协商连接，wvdial 进行拨号和初始化协商，然后启动 pppd 进行余下操作。

配置脚本 wvdialconf 配置 PPP 连接，仅需要交互式的选择下面内容。

- 电话号码
- ISP 用户名
- ISP 密码

在大部分情况下，wvdial 能够成功建立连接并自动维护认证鉴权数据。

文件	功能
/etc/ppp/peers/wvdial	wvdialconf 生成针对 wvdial 的 pppd 配置文件
/etc/wvdial.conf	wvdialconf 生成配置文件
/etc/ppp/options	pppd 常用的执行参数
/etc/ppp/pap-secret	PAP 的鉴权认证数据 (有安全风险)
/etc/ppp/chap-secret	CHAP 的鉴权认证数据 (更安全)

Table 5.7: 使用 wvdialconf 的 PPP 连接配置文件列表

你可以使用下面的底层网络配置工具测试配置。

```
$ sudo wvdial
...
$ sudo killall wvdial
```

参见 wvdial(1) 和 wvdial.conf(5).

5.5.5 使用 pppoeconf 的 PPPoE 以太网连接

当你的互联网提供商提供 PPPoE 连接，并且你决定把电脑直接连接到广域网，那你的电脑网络必须使用 PPPoE 来配置。PPPoE 表示以太网上的 PPP。配置脚本 `pppoeconf` 交互式的配置 PPPoE 连接。

配置文件在下面。

文件	功能
`/etc/ppp/peers/dsl-provider`	`pppoeconf` 生成针对 `pppoe` 的 `pppd` 配置文件
`/etc/ppp/options`	`pppd` 常用的执行参数
`/etc/ppp/pap-secret`	PAP 的鉴权认证数据 (有安全风险)
`/etc/ppp/chap-secret`	CHAP 的鉴权认证数据 (更安全)

Table 5.8: 使用 pppoeconf 的 PPPoE 连接配置文件列表

你可以使用下面的底层网络配置工具测试配置。

```
$ sudo /sbin/ifconfig eth0 up
$ sudo pon dsl-provider
...
$ sudo poff dsl-provider
$ sudo /sbin/ifconfig eth0 down
```

参见"`/usr/share/doc/pppoeconf/README.Debian`".

5.6 使用 ifupdown 进行基本网络配置（旧）

Debian 系统上的传统 TCP/IP 网络 搭建，`ifupdown` 软件包是作为一个上层工具来使用。有两个典型场景。

- 像移动电脑上的 动态 **IP** 系统，你可以使用 `resolvconf` 包搭建 TCP/IP 网络，它能够使你快速切换你的网络配置。(参见第 5.6.4 节).
- 像服务器上的 静态 **IP** 系统，你不需要 `resolvconf` 包来搭建你的 TCP/IP 网络，并保持你的系统简单 (参见第 5.6.5 节).

如果你想设置高级配置，这些传统的设置方法，是相当有用的。在下面的内容中发现细节。

`ifupdown` 包提供 Debian 系统中标准的高层网络配置框架。在本节中，我们通过 `ifupdown` 的简单介绍和许多典型列子来学习基本的网络配置。

5.6.1 简单的命令语法

`ifupdown` 包包含有两个命令: ifup(8) 和 ifdown(8). 它们提供专注于"/etc/network/interfaces" 配置文件的上层网络配置。

命令	操作
`ifup eth0`	如果"`iface eth0`" 节存在，使用 `eth0` 的配置来启动网络接口 `eth0`
`ifdown eth0`	如果"`iface eth0`" 节存在，使用 `eth0` 的配置来关闭网络接口 `eth0`

Table 5.9: 使用 ifupdown 进行基本网络配置的命令列表

警告
请不要使用 ifconfig(8) 和 ip(8) 这类的底层网络配置工具命令来配置一个 **up** 状态的接口。

注意
并没有一个叫 `ifupdown` 的命令。

5.6.2 ”/etc/network/interfaces” 基本语法

在 interfaces(5) 里解释的”`/etc/network/interfaces`” 关键语法，能够用下面的表格概括。

节	说明
”`auto <interface_name>`”	当系统启动时，启动接口 < interface_name>
”`allow-auto <interface_name>`”	同上
”`allow-hotplug <interface_name>`”	当内核从接口检测到一个热拔插事件时，启动接口 <interface_name>
”`iface <config_name> …`” 开头的行	定义 <config_name> 的网络配置
”`mapping <interface_name_glob>`” 开头的行	定义 <config_name> 的映射值来匹配 <interface_name>
”#” 号开始的行	像注释一样忽略 (行尾注释不被支持)
”\” 反斜杠结尾的行	扩展配置到下一行

Table 5.10: ”`/etc/network/interfaces`” 里面的节列表

以 **`iface`** 开头行的节，有下面的语法。

```
iface <config_name> <address_family> <method_name>
 <option1> <value1>
 <option2> <value2>
 ...
```

对于基本配置，**`mapping`** 节没有被使用，你可以使用网络接口名作为网络配置名。(参见第 5.7.5 节).

警告
在”`/etc/network/interfaces`” 里，不要为一个网络接口重复定义”`iface`” 节。

5.6.3 回环网络接口

在启动系统的时候，”`/etc/network/interfaces`” 文件里下面的配置条目启动了回环网络接口 `lo` 。(通过 **`auto`** 节).

```
auto lo
iface lo inet loopback
```

这节内容在”`/etc/network/interfaces`” 文件里面一直存在。

5.6.4 使用 DHCP 的网络接口

按第 5.5.1 节准备系统后，在”/etc/network/interfaces” 里面，按下面的内容创建配置条目后，网络接口的 DHCP 便配置好了。

```
allow-hotplug eth0
iface eth0 inet dhcp
```

当 Linux 内核检测到物理接口 eth0, **allow-hotplug** 节促使 ifup 启动接口，**iface** 促使 ifup 使用 DHCP 来配置接口。

5.6.5 使用静态 IP 地址的网络接口

在”/etc/network/interfaces” 文件里面创建配置条目，来配置静态 IP 网络接口。如下所示。

```
allow-hotplug eth0
iface eth0 inet static
 address 192.168.11.100
 netmask 255.255.255.0
 gateway 192.168.11.1
 dns-domain example.com
 dns-nameservers 192.168.11.1
```

当 Linux 内核检测到 eth0, **allow-hotplug** 节促使 ifup 启动接口，**iface** 节促使 ifup 使用静态 IP 来配置接口。

这里，我假设下面的配置。

- 局域网的 IP 地址范围: 192.168.11.0 - 192.168.11.255
- 网关的 IP 地址: 192.168.11.1
- 电脑的 IP 地址: 192.168.11.100
- resolvconf 包: 已安装
- 域名: ”example.com”
- DNS 服务器的 IP 地址: 192.168.11.1

当 resolvconf 包没有安装时，DNS 相关的配置，需要按下面的方式手工编辑”/etc/resolv.conf” 。

```
nameserver 192.168.11.1
domain example.com
```

小心
用在上面例子里的 IP 地址，不意味着照抄。你应当按你实际网络配置调整 IP 地址。

5.6.6 无线局域网接口基础

无线 LAN (简称 WLAN) 提供快速的无线连接，使用基于 IEEE 802.11 标准集的非授权无线宽带扩频通信技术。

无线接口跟以太网接口非常像，但在初始化时，要求提供一些网络 ID 和密钥数据。他们的上层网络工具差不多和以太网接口一样，除开接口名有一点点不同，按使用的不同内核驱动，像 `eth1`， `wlan0`， `ath0`， `wifi0` ……

提示
`wmaster0` 设备是主设备，它仅仅只是新的 mac80211 Linux API 里，SoftMAC 使用的一个内部设备。

这里有一些需要记住的 WLAN 关键词。

缩略语	全称	说明
NWID	Network ID	802.11 之前 WaveLAN 网络使用的 16 位网络号 (强烈不赞成使用)
(E)SSID	(Extended) Service Set Identifier	无线接入点 (APs) 的网络名称，互连形成一个完整的 802.11 无线局域网, 域名 ID
WEP, (WEP2)	Wired Equivalent Privacy	使用 40 位密钥的第一代 64 位 (128 位) 无线加密标准 (不赞成使用)
WPA	Wi-Fi Protected Access	第二代无线加密标准 (实现大部分 802.11i)，和 WEP 兼容
WPA2	Wi-Fi Protected Access 2	第三代无线加密标准 (完全的 802.11i)，与 WEP 不兼容

Table 5.11: WLAN 缩写词列表

实际选择使用的协议是由你配置的无线路由器所限制。

5.6.7 使用 WPA/WPA2 的无线局域网接口

你需要安装 `wpasupplicant` 包来支持 WLAN 使用新的 WPA/WPA2.

使用 DHCP 的无线局域网连接，"`/etc/network/interfaces`" 文件的条目需要按下面的内容设置。

```
allow-hotplug ath0
iface ath0 inet dhcp
 wpa-ssid homezone
 # hexadecimal psk is encoded from a plaintext passphrase
 wpa-psk 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
```

参见"`/usr/share/doc/wpasupplicant/README.modes.gz`".

5.6.8 使用 WEP 的无线局域网接口

你需要安装 `wireless-tools` 包来支撑老的 WEP 无线局域网。(你的消费等级的路由器可能仍然使用不安全的架构，但这总比没有好。)

小心
请注意：WEP 无线局域网上的网络流量，可以被其他人监听。

使用 DHCP 的无线局域网连接，”/etc/network/interfaces” 文件的条目需要按下面的内容设置。

```
allow-hotplug eth0
iface eth0 inet dhcp
 wireless-essid Home
 wireless-key1 0123-4567-89ab-cdef
 wireless-key2 12345678
 wireless-key3 s:password
 wireless-defaultkey 2
 wireless-keymode open
```

参见”/usr/share/doc/wireless-tools/README.Debian”.

5.6.9 PPP 连接

你需要按之前的描述配置 PPP 连接 (参见第 5.5.3 节). 然后，按下面的方式给第一个 PPP 设备 ppp0 增加“/etc/network/in
文件里的条目.

```
iface ppp0 inet ppp
 provider <isp_name>
```

5.6.10 另一种 PPP 连接

你需要按之前的描述先配置好使用 wvdial 的另外一种 PPP 连接 (参见第 5.5.4 节). 然后，按下面的方式给第一个 PPP 设备 ppp0 增加 “/etc/network/interfaces” 文件里的条目.

```
iface ppp0 inet wvdial
```

5.6.11 PPPoE 连接

使用 PPPoE 直接连接到广域网的电脑，你需要按之前的描述用 PPPoE 连接配置系统 (参见第 5.5.5 节). 然后，按下面的方式给第一个 PPPoE 设备 eth0 增加 “/etc/network/interfaces” 文件里的条目.

```
allow-hotplug eth0
iface eth0 inet manual
 pre-up /sbin/ifconfig eth0 up
 up ifup ppp0=dsl
 down ifdown ppp0=dsl
 post-down /sbin/ifconfig eth0 down
# The following is used internally only
iface dsl inet ppp
 provider dsl-provider
```

5.6.12 ifupdown 网络配置状态

”/etc/network/run/ifstate” 文件保存了由 ifupdown 软件包管理的当前所有的活动网络接口的期望状态。但不幸的是，即使 ifupdown 系统没有按期望的启动某个网络接口，”/etc/network/run/ifstate” 文件仍然会把它列为激活状态。

如果对一个网络接口的 ifconfig(8) 命令输出没有如下列子中的一行，那它就不能够作为 IPV4 网络的一部分使用.

```
inet addr:192.168.11.2  Bcast:192.168.11.255  Mask:255.255.255.0
```

注意
对于连接到 PPPoE 的以太网设备，ifconfig(8) 命令的输出看起来像上面的列子。

5.6.13 网络重新配置基础

当你试图重新配置接口,如 eth0 时,你必须首先用”**sudo ifdown eth0**” 命令关闭它. 这将从”/etc/network/run/ifsta
文件里面移除 eth0 条目。(如果 eth0 之前没有被适当配置，或没有激活，这个命令将导致出现一些错误信息。迄今为止，对于简单的单用户工作站，在任何时间执行这个操作，看起来都是安全的。)

你现在可以按需要重新配置网络接口 eth0 ，重写”/etc/network/interfaces” 文件的内容。

然后，你可以使用”**sudo ifup eth0**” 命令，重新激活 eth0 。

提示
你可以简单的执行”**sudo ifdown eth0;sudo ifup eth0**” 来初始化或重新初始化网络接口。

5.6.14 ifupdown-extra 包

ifupdown-extra 包提供简易的网络连接测试，和 ifupdown 包一道使用.

- network-test(1) 命令能够在 shell 里使用。
- 自动脚本将运行每一个 ifup 执行的命令。

network-test 命令把你从麻烦的执行分析网络问题的底层命令中解放出来。

自动脚本安装在” /etc/network/*/ ” 并执行下面的操作。

- 检查网络线缆连接
- 检查重复 IP 地址使用
- 按”/etc/network/routes” 的定义，建立系统静态路由
- 检查网络的网关是否可以到达
- 在”/var/log/syslog” 文件里面记录结果

系统日志记录对管理远程系统的网络问题非常有用。

提示
ifupdown-extra 包的自动化行为是由”/etc/default/network-test” 来配置. 部分自动化检查，会减慢一点系统启动速度，因为这些脚本需要一些时间来监听 ARP 答复.

5.7 使用 ifupdown 的高级网络配置（旧）

ifupdown 软件包，使用其高级用法，其功能就能够超出在第 5.6 节所描述的内容。

描述在这里的这些功能是完全可选的。我，由于懒惰和极简主义，几乎不使用这些令人烦扰的东西。

小心
如果你不能够通过第 5.6 节里的信息建立网络连接, 使用下面的信息，你就会使你的情形变得比较糟糕。

5.7.1 ifplugd 软件包

ifplugd 软件包是一个老的自动网络配置工具，它仅能够管理以太网连接。解决了移动 PC 等拔插以太网线缆的问题。如果你有安装 NetworkManager 或 Wicd (参见第 5.2 节)，就不需要这个软件包。

这个软件包运行一个后台守护进程（daemon） 来代替 **auto** 或 **allow-hotplug** 的功能 (参见表 5.10)，并启动网络连接上的接口。

以太网端口怎样使用 ifplugd 软件包，比如 eth0，请看下面。

1. 删除”/etc/network/interfaces” 里面的节: ”auto eth0” 或”allow-hotplug eth0”.
2. 保留”/etc/network/interfaces” 里的节: ”iface eth0 inet …” 和”mapping …”.
3. 安装 ifplugd 软件包.
4. 运行”sudo dpkg-reconfigure ifplugd”.
5. 把 eth0 作为” 由 ifplugd 监控的静态网卡”.

现在，网络按你希望的方式重新配置了。

- 在打开电源或发现硬件的时候启动，接口不会自己启动自己。
  - 快速启动过程，没有长的 DHCP 等待时间。
  - 没有适当 IPv4 地址的接口不会被滑稽的激活 (参见第 5.6.12 节).
- 发现以太网线缆时，启动接口。
- 在拔掉以太网线缆后启动一段时间，然后接口自动关闭。
- 在插入另外的以太网线缆时，接口在新的网络环境下启动。

提示
ifplugd(8) 命令的参数能够设置其行为，比如说重新配置接口的延时。

5.7.2 ifmetric 软件包

ifmetric 软件包使我们能够根据经验来维护路由度量值，即使是 DHCP 的路由度量值。

下面设置 eth0 接口的值，让其在有 wlan0 接口的情况下，更加适当。

1. 安装 ifmetric 软件包。

2. 在”/etc/network/interfaces” 里，增加一行”metric 0”，紧挨着放在”iface eth0 inet dhcp” 这行下面。
3. 在”/etc/network/interfaces” 里，增加一行”metric 1” ，紧挨着放在”iface wlan0 inet dhcp” 这行下面.

metric 0 意味着最高路由优先级，是默认值。大的 metric 值意味着较低的路由优先级。具有最低 metric 值的活动的接口 IP 地址，成为原始路由。参见 ifmetric(8)。

5.7.3 虚拟接口

单个物理以太网接口能够配置为使用不同的 IP 地址的多个虚拟接口。这样做的目的，通常是把接口连接到几个 IP 子网。比如说，只有一个网卡的基于 IP 地址的虚拟 web 主机，就是这样一个应用。

举个例子，让我们假设下面的情况。

- 你主机上的单个以太网接口连接到以太网集线器（不是宽带路由器）。
- 以太网集线器同时连接到互联网和本地局域网。
- 局域网使用子网 192.168.0.x/24.
- 你主机的物理接口 eth0 使用 DHCP 提供的 IP 地址来连接互联网。
- 你的主机使用 192.168.0.1 作为局域网的虚拟接口 eth0:0 的地址。

”/etc/network/interfaces” 里下面的节配置你的网络。

```
iface eth0 inet dhcp
 metric 0
iface eth0:0 inet static
 address 192.168.0.1
 netmask 255.255.255.0
 network 192.168.0.0
 metric 1
```

小心
虽然这个配置列子，并使用 netfilter/iptables (参见第 5.10 节) 的网络地址转换 (NAT)，能够给只有单个网络接口的局域网提供廉价的路由器，但这样设置，没有真正的防火墙能力。你应当使用 2 块物理网卡的 NAT 来使本地网络更安全，隔离不安全的互联网。

5.7.4 高级命令语法

ifupdown 软件包提供高级网络配置，使用网络配置名和网络接口名。我使用的术语和 ifup(8) 以及 interfaces(5) 有少量不同。

在第 5.6.1 节里的基本网络配置命令，需要网络配置名来标识匹配”/etc/network/interfaces” 里 **iface** 节的网络接口名。

高级网络配置命令能够按下面的方式区分”/etc/network/interfaces” 里的网络配置名和网络接口名。

man 手册页术语	本文术语	下面文本的列子	说明
物理接口名	网络接口名	`lo`, `eth0`, `<interface_name>`	Linux 内核给出的名字 (使用 udev 机制)
逻辑接口名	网络配置名	`config1`, `config2`, `<config_name>`	在”`/etc/network/interfaces`”里紧跟着 **`iface`** 的名字

Table 5.12: 网络设备术语列表

命令	操作
`ifup eth0=config1`	使用配置 `config1` 启动网络接口 `eth0`
`ifdown eth0=config1`	使用配置 `config1` 关闭网络接口 `eth0`
`ifup eth0`	使用 **`mapping`** 节选择的 `eth0` 配置启动网络接口
`ifdown eth0`	使用 **`mapping`** 节选择的 `eth0` 配置关闭网络接口

Table 5.13: ifupdown 高级网络配置命令列表

5.7.5 映射节 mapping stanza

为了避免复杂，我们在第 5.6.2 节里省略了解释”`/etc/network/interfaces`” 里的 **`mapping`** 节。

```
mapping <interface_name_glob>
 script <script_name>
 map <script_input1>
 map <script_input2>
 map ...
```

这给 `/etc/network/interfaces` 文件提供了一个高级特征，可以自动选择映射脚本 `<script_name>` 定义的配置。

让我们来跟随下面的执行。

```
$ sudo ifup eth0
```

当”`<interface_name_glob>`” 匹配”`eth0`”, 这个执行过程执行下面的命令来自动配置 `eth0`。

```
$ sudo ifup eth0=$(echo -e '<script_input1> \n <script_input2> \n ...' | <script_name> eth0 ↩
    )
```

这里，” `map` ” 脚本输入行是可选和可以重复的。

注意
`mapping` 节工作的匹配模式，类似于 shell 文件名匹配。(参见第 1.5.6 节).

5.7.6 手动的可切换网络配置

以下是如何在几个网络配置中进行手动切换，而无需像第 5.6.13 节中那样重写“`/etc/network/interfaces`”文件。

对于你需要访问的所有网络配置，你需要在“`/etc/network/interfaces`”文件中像下面那样创建一个单独的节。

```
auto lo
iface lo inet loopback

iface config1 inet dhcp

iface config2 inet static
 address 192.168.11.100
 netmask 255.255.255.0
 gateway 192.168.11.1
 dns-domain example.com
 dns-nameservers 192.168.11.1

iface pppoe inet manual
 pre-up /sbin/ifconfig eth0 up
 up ifup ppp0=dsl
 down ifdown ppp0=dsl
 post-down /sbin/ifconfig eth0 down

# The following is used internally only
iface dsl inet ppp
 provider dsl-provider

iface pots inet ppp
 provider provider
```

请注意，**iface** 后面标识的 网络配置名称不用于标识 网络接口名称。另外，也没有 **auto** 或 **allow-hotplug** 节来根据事件自动启动网络接口 eth0。

现在，你可以切换网络配置了。

让我们通过 DHCP 将你的 PC 移动到局域网。你可以通过下列命令开启由网络配置名称（逻辑接口名称）config1 指定的网络接口（物理接口）eth0。

```
$ sudo ifup eth0=config1
Password:
...
```

eth0 接口已开启，由 DHCP 配置并连接到了局域网。

```
$ sudo ifdown eth0=config1
...
```

eth0 接口已关闭并断开局域网连接。

让我们通过静态 IP 使你的 PC 移动到局域网。你可以通过下列命令开启由网络配置名称 config2 指定的网络接口 eth0。

```
$ sudo ifup eth0=config2
...
```

开启 eth0 接口，使用静态 IP 配置并连接到局域网。像 dns-* 这样的额外参数会配置“/etc/resolv.conf”的内容。如果安装了 resolvconf，“/etc/resolv.conf”会更容易管理。

```
$ sudo ifdown eth0=config2
...
```

eth0 接口再次关闭并断开局域网连接。

让我们将你的 PC 移动到 PPPoE 服务器的 BB-modem 上的一个端口。你可以通过下列命令开启由网络配置名称 pppoe 指定的网络接口 eth0。

```
$ sudo ifup eth0=pppoe
...
```

eth0 接口已开启，由 PPPoE 配置直接连接到 ISP。

```
$ sudo ifdown eth0=pppoe
...
```

eth0 接口再次关闭并断开连接。

让我们将你的 PC 移动到使用 POTS 和 modem 的位置，而非局域网或 BB-modem。你可以通过下列命令开启由网络配置名称 ppp0 指定的网络接口 eth0。

```
$ sudo ifup ppp0=pots
...
```

开启 ppp0 接口，并使用 PPP 连接到互联网。

```
$ sudo ifdown ppp0=pots
...
```

关闭 ppp0 接口并断开网络。

你应该检查“/etc/network/run/ifstate”文件，查看 ifupdown 系统当前网络配置的状态。

警告
如果你有多个网络接口，你可能需要调整 eth*、ppp* 等的末尾数字。

5.7.7 ifupdown 系统的脚本

ifupdown 系统会自动运行安装在“/etc/network/*/”中的脚本，而且会传递环境变量给脚本。

这里，每一个环境变量，”$IF_<OPTION>”, 是在相应的选项名字 <option1> 和 <option2> 前增加”$IF_” 来创建, 把字母转换为大写字母，将中划线替换为下划线，忽略非字母数字的字符。

提示
<address_family>, <method_name>, <option1> 和 <option2> 的说明，请参见第 5.6.2 节。

ifupdown-extra 软件包 (参见第 5.6.14 节) 使用这些环境变量来扩展 ifupdown 软件包的功能. ifmetric 软件包 (参见第 5.7.2 节) 安装”/etc/network/if-up.d/ifmetric” 脚本，这个脚本通过”$IF_METRIC” 变量来设置 metric 路由度量值. guessnet 软件包 (参见第 5.7.8 节), 提供简单和功能强大的框架，用于通过 mapping 映射机制自动选择网络配置，这个软件包也使用了这些环境变量。

环境变量	传递值
"$IFACE"	处理中的接口的物理名称（接口名称）
"$LOGICAL"	处理中的接口的逻辑名称（配置名称）
"$ADDRFAM"	接口的 <address_family>
"$METHOD"	接口的 <method_name>（例如“static”）
"$MODE"	如果是 ifup 运行的，则值为“start”；如果是 ifdown 运行的，则值为“stop”
"$PHASE"	根据“$MODE”，但有更细致的区分，共分为 pre-up、post-up、pre-down 和 post-down 阶段
"$VERBOSITY"	指示是否使用了“--verbose”；是为 1，否为 0
"$PATH"	命令搜索路径:“/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/
"$IF_<OPTION>"	位于 **iface** 这节的相应选项值

Table 5.14: ifupdown 系统传递的环境变量

注意
使用这些环境变量进行个性化网络配置的列子，你可以查看"/usr/share/doc/ifupdown/examples/*" 里的列子脚本，以及 ifscheme 和 ifupdown-scripts-zg2 软件包使用的脚本. 这些额外的脚本有部分功能和 ifupdown-extra 与 guessnet 软件包的基本功能重叠. 如果你安装了这些额外脚本，你应当个性化这些脚本来避免互相影响。

5.7.8 guessnet Mapping 映射

为了替代在第 5.7.6 节描述的手工选择配置，你可以使用在第 5.7.5 节描述的映射机制，自动选择个性化脚本来进行配置网络。

guessnet 软件包提供的 guessnet-ifupdown(8) 命令，是被设计作为映射脚本，并提供强力框架来增强 ifupdown 系统.

- 在 **iface** 节下的每一个网络配置，列出测试条件作为 **guessnet** 选项的值。
- 映射机制选择第一个没有错误结果的 **iface** 作为网络配置。

guessnet-ifupdown 使用的映射脚本和 ifupdown 的原始网络配置架构,这两种对"/etc/network/interfaces" 文件的用法，不会造成负面的影响，因为 **guessnet** 选项仅仅只导出额外的环境变量到 ifupdown 系统运行的脚本。细节参见 guessnet-ifupdown(8).

注意
当多个 **guessnet** 选项行需要在"/etc/network/interfaces" 里出现时, 选项行使用 **guessnet1**, **guessnet2**, 这类的开头, 因为 ifupdown 软件包不允许选项行开头字符串重复。

5.8 底层网络配置

5.8.1 Iproute2 命令

Iproute2 命令集提供完整的底层网络配置能力。有个从旧的 net-tools 命令集到新的 iproute2 命令集的转换表。

参见 ip(8) 和 IPROUTE2 工具套件 Howto.

旧的 **net-tools**	新的 **iproute2**	操作
ifconfig(8)	`ip addr`	一个设备上的协议（IP 或 IPv6）地址
route(8)	`ip route`	路由表条目
arp(8)	`ip neigh`	ARP 或 NDISC 缓存条目
`ipmaddr`	`ip maddr`	多播地址
`iptunnel`	`ip tunnel`	IP 隧道
nameif(8)	ifrename(8)	基于 MAC 地址的网络接口名
mii-tool(8)	ethtool(8)	以太网设备设置

Table 5.15: 从旧的 `net-tools` 命令集到新的 `iproute2` 命令集转换表

5.8.2 安全的底层网络操作

你可以按下面的方式安全的使用底层网络命令，这些命令不会改变网络配置。

命令	说明
`ifconfig`	显示活动的网络接口连接和地址状态
`ip addr show`	显示活动的网络接口连接和地址状态
`route -n`	用数字地址显示全部路由表
`ip route show`	用数字地址显示全部路由表
`arp`	显示当前 ARP 缓存表的内容
`ip neigh`	显示当前 ARP 缓存表的内容
`plog`	显示 ppp 后台守护进程（daemon）日志
`ping yahoo.com`	检查到"`yahoo.com`"的因特网连接
`whois yahoo.com`	在域名数据库里面检查谁注册了"`yahoo.com`"
`traceroute yahoo.com`	跟踪到"`yahoo.com`"的因特网连接
`tracepath yahoo.com`	跟踪到"`yahoo.com`"的因特网连接
`mtr yahoo.com`	跟踪到"`yahoo.com`"的因特网连接（重复的）
`dig [@dns-server.com] example.com [{a|mx|any}]`	查询由"`dns-server.com`"提供服务的"`example.com`"域名的 DNS 记录：`"a"`, `"mx"` 或`"any"` 记录
`iptables -L -n`	查看包过滤
`netstat -a`	找出所有打开的端口
`netstat -l --inet`	找出监听端口
`netstat -ln --tcp`	找出 TCP 监听端口（数字的）
`dlint example.com`	查询"`example.com`"的 DNS zone 信息

Table 5.16: 底层网络命令列表

提示
部分底层网络配置工具放在"`/sbin/`"目录。你可以像"`/sbin/ifconfig`"这样使用完整命令路径，或把"`/sbin`"加到"`~/.bashrc`"文件列出的"`$PATH`"环境变量里。

5.9 网络优化

通用的网络优化超出了本文的范围。我提及消费等级连接相关的主题。

5.9.1 找出最佳 MTU

最大传输单元 Maximum Transmission Unit (MTU) 的值能够通过加"`-M do`"选项的 ping(8) 实验来确定，它发送从 1500 字节（对于 IP+ICMP 包头，有 28 字节的偏移）大小开始的 ICMP 包，来找出 IP 不分片的最大包大小。

软件包	流行度	大小	说明
iftop	V:8, I:112	97	显示一个网络接口上的带宽使用信息
iperf	V:4, I:54	197	互联网协议带宽测量工具
ifstat	V:0, I:9	56	接口统计监控
bmon	V:1, I:14	143	便携式带宽监视器和网速估计工具
ethstatus	V:0, I:5	40	快速测量网络设备吞吐的脚本
bing	V:0, I:1	71	实验性的随机带宽测试器
bwm-ng	V:1, I:17	89	小巧简单的控制台带宽监测器
ethstats	V:0, I:0	23	基于控制台的以太网统计监视器
ipfm	V:0, I:0	78	带宽分析工具

Table 5.17: 网络优化工具列表

尝试下列例子

```
$ ping -c 1 -s $((1500-28)) -M do www.debian.org
PING www.debian.org (194.109.137.218) 1472(1500) bytes of data.
From 192.168.11.2 icmp_seq=1 Frag needed and DF set (mtu = 1454)

--- www.debian.org ping statistics ---
0 packets transmitted, 0 received, +1 errors
```

尝试 1454 代替 1500

你看到用 1454 ping(8) 成功了。

这个过程是 路径 MTU (PMTU) 发现 (RFC1191)，tracepath(8) 命令能够自动完成这个。

提示
上面的列子，PMTU 的值是 1454，这是我先前的光纤到户提供商，使用了 异步传输模式 Asynchronous Transfer Mode (ATM) 作为他们的骨干网络，并使用 PPPoE 作为客户端。实际 PMTU 值依赖于你的环境，比如说，我新的光纤到户提供商是 1500。

网络环境	**MTU**	基本原理
拨号连接 (IP: PPP)	576	标准的
以太网连接 (IP: DHCP 或固定)	1500	默认标准值
以太网连接 (IP: PPPoE)	1492 (=1500-8)	PPP 头部 2 字节和 PPPOE 头部 6 字节
以太网连接 (ISP 骨干网: ATM, IP: DHCP 或固定 IP)	1462 (=48*31-18-8)	作者推断：18 字节的以太网头，8 字节 SAR 尾（译注：SAR 为 ATM 技术名词）
以太网连接 (ISP 骨干: ATM, IP: PPPoE)	1454 (=48*31-8-18-8)	参见”Optimal MTU configuration for PPPoE ADSL Connections” 来了解基本原理

Table 5.18: 最佳 MTU 值的基本指引方法

除了这些基本的指引方法外，你还应当知道下面的信息。

- 使用任何隧道方式 (VPN 等.) 的最佳 MTU 需要进一步减去它们上面的头部。
- MTU 值不应当超过通过实验验证的 PMTU 值。
- 当遇到其它限制的时候，较大的 MTU 值通常比较好。

5.9.2 设置 MTU

这里示例设置 MTU 值，从默认的 1500 设置到 1454.

对于 DHCP (参见第 5.6.4 节), 你能够使用下面的方式替换”/etc/network/interfaces” 文件里 **iface** 节相关的行.

```
iface eth0 inet dhcp
 pre-up /sbin/ifconfig $IFACE mtu 1454
```

对于静态 IP (参见第 5.6.5 节), 你能够使用下面的方式替换”/etc/network/interfaces” 文件里 **iface** 节相关的行.

```
iface eth0 inet static
 address 192.168.11.100
 netmask 255.255.255.0
 gateway 192.168.11.1
 mtu 1454
 dns-domain example.com
 dns-nameservers 192.168.11.1
```

对于直接 PPPoE (参见第 5.5.5 节), 你能够使用下面的方式替换”/etc/ppp/peers/dsl-provider” 里”mtu” 相关的行.

```
mtu 1454
```

最大分片大小 (MSS) 是另外一种衡量包大小的方法。MSS 和 MTU 的关系如下.

- 对于 IPv4，MSS = MTU - 40
- 对于 IPv6，MSS = MTU - 60

注意
基于 iptables(8) (参见第 5.10 节) 的优化，能够通过 MSS 来压缩包大小，路由器会用到 MMS 。参见 iptables(8) 中的”TCPMSS” .

5.9.3 WAN TCP 优化

TCP 吞吐量能够通过调整 TCP 缓冲大小的参数来最大化，对现代大带宽和高延时的 WAN，在”TCP Tuning Guide” 和”TCP tuning” 里有描述. 到目前为止，当前 Debian 默认设置能够很好的服务好我的 1G bps 光纤到户 LAN 连接。

5.10 Netfilter 网络过滤框架

Netfilter 使用 Linux 内核 模块 (参见第 3.3.1 节) 提供 状态防火墙 和 网络地址转换 (NAT) 框架。

netfilter 主要的用户层程序是 iptables(8). 你能从 shell 手工交付式的配置 netfilter，使用 iptables-save(8) 保存当前状态, 当系统重启时，通过 init 脚本调用 iptables-restore(8) 来恢复。

像 shorewall 这样的配置帮助脚本能够使这个过程变得更简单。

参见 http://www.netfilter.org/documentation/ 上的文档 (或在”/usr/share/doc/iptables/html/” 里面的文档).

软件包	流行度	大小	说明
iptables	V:270, I:995	2569	netfilter 管理工具 (iptables(8) 用于 IPv4, ip6tables(8) 用于 IPv6)
arptables	V:0, I:2	95	netfilter 管理工具 (arptables(8) 用于 ARP)
ebtables	V:36, I:66	265	netfilter 管理工具 (ebtables(8) 用于以太网桥)
iptstate	V:0, I:4	116	持续性监控 netfilter 状态 (和 top(1) 相似)
shorewall-init	V:0, I:0	68	Shoreline 防火墙 初始化
shorewall	V:6, I:14	2456	Shoreline 防火墙, netfilter 配置文件生成器
shorewall-lite	V:0, I:0	65	Shoreline 防火墙, netfilter 配置文件生成器 (精简版)
shorewall6	V:1, I:2	779	Shoreline 防火墙, netfilter 配置文件生成器 (IPv6 版本)
shorewall6-lite	V:0, I:0	64	Shoreline 防火墙, netfilter 配置文件生成器 (IPv6, 精简版)

Table 5.19: 防火墙工具列表

- Linux Networking-concepts HOWTO
- Linux 2.4 Packet Filtering HOWTO
- Linux 2.4 NAT HOWTO

提示
虽然这些是为 Linux **2.4** 写的,iptables(8) 命令和 netfilter 内核功能都能够在 Linux**2.6** 和 **3.x** 内核系列实现.

Chapter 6

网络应用

建立网络连接后（参加第 5 章），你可以运行各种网络应用。

提示
对于现代的 Debian 网络基础设施的具体说明，阅读 Debian 管理员手册——网络基础设施。

提示
在某些 ISP 下，如果你启用“两步验证”，你可能需要获取一个应用密码以从你的程序访问 POP 和 SMTP 服务。你也可能需要事先允许你的主机 IP 进行访问。

6.1 网页浏览器

有许多网页浏览器软件包，使用超文本传输协议（HTTP）访问远程内容。

软件包	流行度	大小	类型	网络浏览器说明
chromium	V:58, I:148	183777	X	Chromium，（来自 Google 的开源浏览器）
firefox	V:15, I:26	167408	同上	Firefox，（来自 Mozilla 的开源浏览器，仅在 Debian Unstable 中可用）
firefox-esr	V:265, I:461	162529	同上	Firefox ESR（Firefox 延长支持版本）
epiphany-browser	V:6, I:27	3089	同上	GNOME，兼容 HIG，Epiphany
konqueror	V:21, I:108	21034	同上	KDE，Konqueror
dillo	V:1, I:5	1540	同上	Dillo，（基于 FLTK 的轻量级浏览器）
w3m	V:80, I:433	2323	文本	w3m
lynx	V:20, I:103	1924	同上	Lynx
elinks	V:10, I:29	1752	同上	ELinks
links	V:12, I:42	2207	同上	Links（纯文本）
links2	V:2, I:16	5486	图像	Links（没有 X 的控制台图像）

Table 6.1: 网页浏览器列表

6.1.1 浏览器配置

在某些浏览器中，你可以使用下列特殊的 URL 来确认它们的设置。

- ”about:”
- ”about:config”
- ”about:plugins”

Debian 提供了在 main 档案库中提供了许多自由的浏览器插件软件包，不仅可以处理 Java（软件平台）和 Flash，也可以处理 MPEG、MPEG2、MPEG4、DivX、Windows Media Video (.wmv)、QuickTime (.mov)、MP3 (.mp3)、Ogg/Vorbis 文件、DVD、VCD 等等。Debian 也提供相关辅助程序，可以用来安装来自 contrib 或 non-free 的 non-free 浏览器插件软件包。

软件包	流行度	大小	区域	说明
icedtea-plugin	I:18	19	main	基于 OpenJDK 和 IcedTea 的 Java 插件
flashplugin-nonfree	V:2, I:58	71	contrib	安装 Adobe Flash Player 的 Flash 插件辅助程序（仅适用 i386、amd64）

Table 6.2: 浏览器插件软件包列表

提示
尽管使用上述的 Debian 软件包会更容易，但你依旧可以手动启用插件，你需要将“*.so”文件安装到插件目录中（例如“/usr/lib/iceweasel/plugins/”）并重启浏览器。

有些网站拒绝基于你所使用浏览器的用户代理字符串的连接。你可以通过 伪装用户代理字符串 来解决这个问题。例如，你可以添加下面这行到用户配置文件中（例如“~/.gnome2/epiphany/mozilla/epiphany/user.js”或“~/.mozilla/firefox/*.default/user.js”）。

```
user_pref("general.useragent.override","Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0) ↩
    ");
```

或者，你也可以通过输入“about:config”到 URL，并右击它所显示的内容，来添加并重置这个变量。

小心
伪装的用户代理字符串可以会导致 来自 Java 的不良副作用。

6.2 邮件系统

小心
如果你想设置邮件服务器来直接通过互联网交换邮件，你应该最好阅读一下这个基本文档。

邮件系统涉及到运行在多个主机上的许多服务器程序和客户端程序。从功能来说，有 3 种类型的邮件代理程序：

- 邮件传输代理（MTA，参见第 6.3 节），是不同主机之间传送邮件的程序。
- 邮件投递代理（MDA，参见第 6.6 节），是传递信息到一台主机内的用户邮箱的程序。
- 邮件用户代理（MUA，也被称为电子邮件客户端，参见第 6.4 节），是生成信息和访问传递的信息的程序。

注意
对于那些消费者级网络连接的典型移动工作站，以下的配置例子是有效的。

6.2.1 电子邮件基础

电子邮件 由三个部分组成，消息的信封，邮件标头及邮件正文。

SMTP 用电子邮件信封上的”To” 和”From” 信息来投递邮件。(信封上的”From” 信息也被叫做退回地址, 例如 From_ 等等)。

电子邮件头的”To” 和”From” 信息，显示在 电子邮件客户端上. (在大部分情况下，这些信息是跟电子邮件信封一致，但并不全是这样。)

为了处理正文数据类型及其编码，电子邮件客户端 (MUA) 需要用多用途互联网邮件扩展 (MIME)来解释邮件标头和邮件正文。

6.2.2 现代邮件服务基础

为了尽可能减少垃圾邮件 (不想要的和未经请求的电子邮件) 的问题，许多提供消费者级互联网连接的 ISP 服务商正在采取应对措施。

- 智能主机服务于 ISP 的客户，使用rfc4409里面规定的 (587) 端口发送邮件，并使用在rfc4954里面规定的密码 (SMTP 认证服务).
- 内部的网络主机 (除了 ISP 自己的发送邮件服务器) 连接到互联网的 SMTP 25 端口已经被封锁了。
- 从一些可疑的外部网络主机到 ISP 接收邮件服务器SMTP 25 端口的连接会被阻隔.(连接来自用于拨号和其它消费等级互联网连接的动态 IP 地址范围, 首先被阻隔.)
- 像域名密钥识别邮件 (DKIM)、发信者策略框架 (SPF) 和 基于域名的消息认证、报告和反应 (DMARC) 这样的反垃圾邮件技术广泛用于电子邮件过滤。
- 域名密钥识别邮件服务可能会用于你的通过 smarthost 的电子邮件发送。
- 智能主机可以在上面重写源电子邮件地址为你的邮件账户。

当配置电子邮件系统或解决邮递问题时，你必须考虑这些新的限制。

小心
在消费者级的网络上运行 SMTP 服务器来直接发送邮件到远端可信赖主机是不现实的。

小心
期望单个智能主机可靠的发送不相关的源邮件地址到远程主机，这是不现实的。

小心
一个邮件能够被任何主机静悄悄的拒绝，即使路由到了目的地。发送一个邮件到远程主机的可靠方法，就是使你的邮件尽可能的看起来是经过认证的。

鉴于这些不利的互联网情况和限制，像 Yahoo.com 和 Gmail.com 这样的独立互联网邮件 ISP 提供了安全的邮件服务，使用传输层安全协议 (TLS) 和它的前身，安全套接层协议 (SSL) 就可以在任何地方通过网络连接到这些邮件服务。

- 智能主机上的 465 端口服务, 是过时的在 SSL 上的 SMTP (SMTPS 协议).
- 智能主机上的 587 端口服务使用 STARTTLS 协议。
- TLS/POP3 端口 (995) 是用 POP3 协议来接受邮件的。

为了简便起见，在接下来的文本中，我假定 smarthost 是”`smtp.hostname.dom`”, 需要 SMTP 认证并且使用带有STARTTLS 协议的信息发送端口 (587) 。

6.2.3 工作站的邮件配置策略

最简单的电子邮件配置是使用 MUA 发送邮件到 ISP 的 smarthost，然后从 ISP 的 POP3 服务器接收邮件 (参见第 6.4 节)。这种类型的配置流行使用全功能的基于 GUI 的 MUA，例如 icedove(1)，evolution(1) 等等。如果需要通过邮件的类型来过滤它们，你应该使用 MUA 的过滤功能。对于这种情况，本地 MTA (参见第 6.3 节) 只需在本地投递 (当发送者和接收者在同一主机上)。

请注意 Debian 是多用户系统。即使你是唯一的用户，这里仍然有许多以 root 用户运行的程序并且它们会给你发送电子邮件。

另外可选的邮件配置是通过本地 MTA 发送邮件到 ISP 的 smarthost，通过邮件检索 (参见第 6.5 节) 从 ISP 的 POP3 服务器接受邮件，并把邮件保存到本地邮箱。如果需要通过邮件的类型来过滤它们，你应该使用 MDA 的过滤功能 (参见第 6.6 节) 来过滤邮件到单独的邮箱。这种类型的配置流行使用基于终端的简单 MUA，例如 mutt(1)，mew(1) 等等，尽管使用任何 MUA 都是可以的 (参见第 6.4 节)。对于这种情况，本地 MTA (参见第 6.3 节) 需要做 smarthost 投递和本地投递。因为移动工作站没有有效的 FQDN，你必须配置本地 MTA 来隐藏和伪装外发邮件中的真实本地邮件名称，来避免邮件投递错误 (参见第 6.3.3 节)。

提示
你可能想要配置 MUA/MDA 来使用 Maildir，以便存储邮件到你用户目录的某个位置。

6.3 邮件传输代理 (MTA)

对于一般的工作站而言，邮件传输代理 (MTA) 的主流选择是 `exim4-*` 或者 `postfix` 软件包，这由你决定。

软件包	流行度	大小	说明
`exim4-daemon-light`	V:408, I:443	1332	Exim4 邮件传输代理 (MTA : Debian 默认的)
`exim4-base`	V:417, I:454	1621	Exim4 文档 (文本) 和通用文件
`exim4-doc-html`	I:1	3575	Exim4 文档 (html)
`exim4-doc-info`	I:1	611	Exim4 文档 (info)
`postfix`	V:146, I:162	4324	Postfix 邮件传输代理 (MTA : 替代品)
`postfix-doc`	I:10	4237	Postfix 文档 (html+text)
`sasl2-bin`	V:6, I:21	428	Cyrus SASL API 实现 (实现 postfix SMTP 认证)
`cyrus-sasl2-doc`	I:1	575	Cyrus SASL - 文档

Table 6.3: 用于工作站的基础的邮件传输代理相关的软件包列表

尽管在流行度投票数上，`exim4-*` 某些时候看起来要比 `postfix` 流行，但这并不意味着 `postfix` 在 Debian 开发者中不流行。Debian 服务器系统使用 `exim4` 和 `postfix`。著名的 Debian 开发者发到邮件列表的帖子的邮件标头分析的结果也表明这两种 MTA 一样受欢迎。

`exim4-*` 软件包最为人所知的是，有着非常小的内存消耗和非常灵活的配置。`postfix` 软件包最为人所知的是，它的简洁、快速、简单和安全的特性。这两种工具都带有充足的文档，在质量和许可证上都同样是不错的。

在 Debian 档案库里，有许多不同性能和不同关注点的邮件传输代理 (MTA) 软件包可供选择。

6.3.1 exim4 的配置

小心
配置 `exim4` 来发送互联网邮件，多个源电子邮件地址使用多个相应的智能主机，这是不寻常的。对于 `popcon` 和 `cron` 这样的系统程序，配置 `exim4` 仅仅只使用一个电子邮件地址；对于 `mutt` 这样的用户程序，配置 `msmtp` 来使用多个源电子邮件地址。

软件包	流行度	大小	性能和关注点
exim4-daemon-light	V:408, I:443	1332	全功能
postfix	V:146, I:162	4324	全功能 (安全)
exim4-daemon-heavy	V:8, I:9	1473	全功能 (灵活)
sendmail-bin	V:12, I:13	1863	全功能 (如果你已经对它熟悉)
nullmailer	V:5, I:8	479	部分功能, 没有本地邮件
ssmtp	V:12, I:19	2	部分功能, 没有本地邮件
courier-mta	V:0, I:0	2328	非常全功能 (web 接口等.)
masqmail	V:0, I:0	337	轻量
esmtp	V:0, I:0	128	轻量
esmtp-run	V:0, I:0	32	轻量 (sendmail 兼容扩展到 esmtp)
msmtp	V:2, I:6	434	轻量
msmtp-mta	V:1, I:1	60	轻量 (sendmail 兼容扩展到 msmtp)

Table 6.4: Debian 档案库中可供选择的邮件传输代理 (MTA) 软件包的列表

对于那些通过 smarthost 的网络邮件，你应该按如下所示的 (重新) 配置 exim4-* 软件包。

```
$ sudo /etc/init.d/exim4 stop
$ sudo dpkg-reconfigure exim4-config
```

配置”General type of mail configuration” 时，选择”mail sent by smarthost; received via SMTP or fetchmail”。

设置”System mail name:” 为默认的 FQDN (参见第 5.1.1 节)。

设置”IP-addresses to listen on for incoming SMTP connections:” 为默认的”127.0.0.1; ::1”。

”Other destinations for which mail is accepted:” 选项留空。

”Machines to relay mail for:” 选项留空。

设置”IP address or host name of the outgoing smarthost:” 为”smtp.hostname.dom:587”。

设置”Hide local mail name in outgoing mail?” 选项为”<No>”。(或者像第 6.3.3 节描述的那样使用 /etc/email-addresses” 代替)

选择如下所示的其中一个来回答”Keep number of DNS-queries minimal (Dial-on-Demand)?”。

- ”No” 如果启动的时候，系统就连上了互联网。
- ”Yes” 如果启动的时候，系统没有连上互联网。

设置”Delivery method for local mail:” 选项为”mbox format in /var/mail/”。

”Split configuration into small files?:” 选项设为”<Yes>”。

通过修改”/etc/exim4/passwd.client” 文件，来创建用于 smarthost 的密码条目。

```
$ sudo vim /etc/exim4/passwd.client
 ...
$ cat /etc/exim4/passwd.client
^smtp.*\.hostname\.dom:username@hostname.dom:password
```

通过如下所示的启动 exim4。

```
$ sudo /etc/init.d/exim4 start
```

”/etc/exim4/passwd.client” 文件中的主机名不应该是别名，你应该按如下所示的检查真正的主机名。

```
$ host smtp.hostname.dom
smtp.hostname.dom is an alias for smtp99.hostname.dom.
smtp99.hostname.dom has address 123.234.123.89
```

我在”/etc/exim4/passwd.client” 文件中使用正则表达式来绕过别名问题。即使 ISP 更改了别名所指向的主机名，SMTP AUTH 还是可能工作的。

你能够通过如下所示的手动更新 exim4 配置:

- 更新”/etc/exim4/” 目录下的 exim4 配置文件。
  - 创建”/etc/exim4/exim4.conf.localmacros” 来设置宏命令和修改”/etc/exim4/exim4.conf.template” 文件。(没有分割的配置)
  - 在” /etc/exim4/exim4.conf.d” 子目录中创建新文件或编辑已存在的文件。(分割的配置)
- 运行”invoke-rc.d exim4 reload” 命令。

请阅读”/usr/share/doc/exim4-base/README.Debian.gz” 官方指导和 update-exim4.conf(8)。

小心
如果 debconf 询问”Keep number of DNS-queries minimal (Dial-on-Demand)?” 这个问题时，选择了”No” (默认值)，那么启动 exim4 会花很长时间并且系统在启动的时候不会连接到互联网。

警告
虽然你的 ISP 允许，但是使用没有加密的明文密码是不安全的。

提示
尽管推荐在 587 端口上使用 STARTTLS 的 SMTP 协议，但是有些 ISP 仍然使用废弃的 SMTPS 协议 (在 465 端口上的 SSL)。4.77 版本以后的 Exim4 支持在客户端和服务器上的废弃 SMTPS 协议。

提示
如果你正在为笔记本电脑寻找一个遵守”/etc/aliases” 规则的轻量 MTA，你应该考虑配置 exim4(8)，在”/etc/default/exim4” 文件中写入”QUEUERUNNER='queueonly'”，”QUEUERUNNER='nodaemon'” 等等。

6.3.2 带有 SASL 的 postfix 配置

对于通过 smarthost 的网络邮件，你应该首先阅读 postfix 文档和关键的手册页。

你应该按如下所示的 (重新) 配置 postfix 和 sasl2-bin 软件包。

命令	功能
postfix(1)	Postfix 控制程序
postconf(1)	Postfix 配置工具
postconf(5)	Postfix 配置参数
postmap(1)	Postfix 查找表维护
postalias(1)	Postfix 别名数据库维护

Table 6.5: 重要的 postfix 手册页列表

```
$ sudo /etc/init.d/postfix stop
$ sudo dpkg-reconfigure postfix
```

选择”Internet with smarthost”。

设置”SMTP relay host (blank for none):” 为”`[smtp.hostname.dom]:587`” 并按如下所示配置。

```
$ sudo postconf -e 'smtp_sender_dependent_authentication = yes'
$ sudo postconf -e 'smtp_sasl_auth_enable = yes'
$ sudo postconf -e 'smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd'
$ sudo postconf -e 'smtp_sasl_type = cyrus'
$ sudo vim /etc/postfix/sasl_passwd
```

为 smarthost 创建密码条目。

```
$ cat /etc/postfix/sasl_passwd
[smtp.hostname.dom]:587     username:password
$ sudo postmap hush:/etc/postfix/sasl_passwd
```

通过如下所示的启动 `postfix`。

```
$ sudo /etc/init.d/postfix start
```

`dpkg-reconfigure` 会话中使用的”[” 和”]” 和”`/etc/postfix/sasl_passwd`” 确保不去检查 MX 记录而是直接使用指定的明确主机名。参见”`/usr/share/doc/postfix/html/SASL_README.html`” 里面的”Enabling SASL authentication in the Postfix SMTP client” 条目。

6.3.3 邮件地址配置

这里有一些用于邮件传输、投递和用户代理的邮件地址配置文件。

文件	功能	应用
`/etc/mailname`	用于 (外发) 邮件的默认主机名	Debian 专用的，mailname(5)
`/etc/email-addresses`	用于外发邮件的主机名伪装	exim(8) 专用的，exim4-config_files(5)
`/etc/postfix/generic`	用于外发邮件的主机名伪装	postfix(1) 专用的，postmap(1) 命令执行后激活。
`/etc/aliases`	用于接收邮件的账户别名	通用的，newaliases(1) 命令执行后激活。

Table 6.6: 与邮件地址相关的配置文件列表

”`/etc/mailname`” 文件中的 **mailname** 通常是全称域名 (FQDN)，这个全程域名将会被解析成主机的 IP 地址。对于没有可解析成 IP 地址的主机名的移动工作站，设置 **mailname** 为”`hostname -f`” 的值。(这对于 `exim4-*` 和 `postfix` 都是安全有效的选择。)

提示
”/etc/mailname” 中的内容被许多非 MTA 程序用作它们的默认行为。对于 mutt，在 ~/muttrc 文件中设置”hostname” 和”from” 变量来覆盖 **mailname** 值。对于 devscripts 软件包的程序，例如 bts(1) 和 dch(1)，导出环境变量”$DEBFULLNAME” 和”$DEBEMAIL” 的值来覆盖它。

提示
popularity-contest 软件包一般以 FQDN 形式的 root 账户发送邮件。你需要像 /usr/share/popularity-contest/default.conf 文件中描述的那样去设置 /etc/popularity-contest.conf 文件中的 MAILFROM 值。否则，你的邮件会被 smarthost SMTP 服务器拒绝。尽管这些过程很乏味，这种方法比为所有通过 MTA 并且是以 root 用户发送的邮件重写源地址更安全。这也可以被其他守护进程或者是 cron 脚本使用。

当设置 **mailname** 为”hostname -f” 的值时，通过 MTA 的源邮件地址的伪装可以通过如下所示的来实现。

- 用于 exim4(8) 的”/etc/email-addresses” 文件，exim4-config_files(5) 手册页中有关于它的解释
- 用于 postfix(1) 的”/etc/postfix/generic” 文件，generic(5) 手册页中有关于它的解释

对于 postfix，接下来的额外步骤需要执行。

```
# postmap hash:/etc/postfix/generic
# postconf -e 'smtp_generic_maps = hash:/etc/postfix/generic'
# postfix reload
```

你能够通过如下所示的来测试邮件地址配置。

- exim(8) 用 -brw, -bf, -bF, -bV, ... 选项
- postmap(1) 用 -q 选项。

提示
Exim 带有一些有用的程序，例如 exiqgrep(8) 和 exipick(8)。参见”dpkg -L exim4-base|grep man8/” 来获得可用的命令。

6.3.4 基础 MTA 操作

这里有一些基础的 MTA 操作。有一些可能会通过 sendmail(1) 的兼容性接口来实现。

提示
往”/etc/ppp/ip-up.d/*” 里写一个刷新所有邮件的脚本会是个不错的主意。

6.4 邮件用户代理 (MUA)

如果你订阅了 Debian 相关的邮件列表，使用像 mutt 和 mew 这样的 MUA 会是个不错主意，同时对用户来说，它们也是事实上的标准并且可以像预期的那样工作良好。

exim 命令	**postfix** 命令	说明
`sendmail`	`sendmail`	从标准输入读取邮件并且安排投递 (`-bm`)
`mailq`	`mailq`	列出带有状态和队列 ID 的邮件队列 (`-bq`)
`newaliases`	`newaliases`	初始化别名数据库 (`-I`)
`exim4 -q`	`postqueue -f`	刷新等待邮件 (`-q`)
`exim4 -qf`	`postsuper -r ALL deferred; postqueue -f`	刷新所有邮件
`exim4 -qff`	`postsuper -r ALL; postqueue -f`	刷新甚至已经冻结的邮件
`exim4 -Mg queue_id`	`postsuper -h queue_id`	通过邮件的队列 ID 来冻结它
`exim4 -Mrm queue_id`	`postsuper -d queue_id`	通过邮件的队列 ID 来移除它
N/A	`postsuper -d ALL`	移除所有邮件

Table 6.7: 基础 MTA 操作列表

软件包	流行度	大小	类型
`evolution`	V:40, I:265	449	X GUI 程序 (GNOME3, groupware 套件)
`thunderbird`	V:62, I:142	136737	X GUI 程序 (GNOME2, 无品牌的 Mozilla Thunderbird)
`kmail`	V:44, I:95	17630	X GUI 程序 (KDE)
`mutt`	V:99, I:488	6010	很有可能与 `vim` 一起使用的字符终端程序
`mew`	V:0, I:0	2322	`(x)emacs` 下的字符终端程序

Table 6.8: 邮件用户代理列表 (MUA)

6.4.1 基础 MUA —Mutt

按如下所示的自定义"`~/.muttrc`"，与 `vim` 结合使用邮件用户代理 (MUA) 软件 `mutt`。

```
#
# User configuration file to override /etc/Muttrc
#
# spoof source mail address
set use_from
set hostname=example.dom
set from="Name Surname <username@example.dom>"
set signature="~/.signature"

# vim: "gq" to reformat quotes
set editor="vim -c 'set tw=72 et ft=mail'"

# "mutt" goes to Inbox, while "mutt -y" lists mailboxes
set mbox_type=Maildir           # use qmail Maildir format for creating mbox
set mbox=~/Mail                 # keep all mail boxes in $HOME/Mail/
set spoolfile=+Inbox            # mail delivered to $HOME/Mail/Inbox
set record=+Outbox              # save fcc mail to $HOME/Mail/Outbox
set postponed=+Postponed        # keep postponed in $HOME/Mail/postponed
set move=no                     # do not move Inbox items to mbox
set quit=ask-yes                # do not quit by "q" only
set delete=yes                  # always delete w/o asking while exiting
set fcc_clear                   # store fcc as non encrypted

# Mailboxes in Maildir (automatic update)
mailboxes `cd ~/Mail; /bin/ls -1|sed -e 's/^/+/' | tr "\n" " "`
unmailboxes Maillog *.ev-summary

## Default
```

```
#set index_format="%4C %Z %{%b %d} %-15.15L (%4l) %s"
## Thread index with senders (collapse)
set index_format="%4C %Z %{%b %d} %-15.15n %?M?(#%03M)&(%4l)? %s"

## Default
#set folder_format="%2C %t %N %F %2l %-8.8u %-8.8g %8s %d %f"
## just folder names
set folder_format="%2C %t %N %f"
```

增加下面的内容到"/etc/mailcap" 或"~/.mailcap" 来内镶显示 HTML 邮件和微软 Word 附件.

```
text/html; lynx -force_html %s; needsterminal;
application/msword; /usr/bin/antiword '%s'; copiousoutput; description="Microsoft Word Text ↩
    "; nametemplate=%s.doc
```

提示
Mutt 能够作为 IMAP 客户端和 mailbox 格式转换器. 你可以使用"t", "T" 等标识邮件. 这些标识的邮件能够使用";C" 在不同的邮箱之间拷贝, 并可以使用";d" 来一次性删除.

6.4.2 高级 MUA —Mutt + msmtp

Mutt 能够使用 msmtp 来配置多个源电子邮件地址使用多个相应的智能主机。

提示
Msmtp 是一个 sendmail 模拟器，它允许和其它提供 /usr/sbin/sendmail 命令的 sendmail 模拟器一起安装。所以你可以保留你系统上的邮件系统为 exim4 或 postfix。

让我们考虑支持 3 个电子邮件地址作为例子：

- "My Name1 <myaccount1@gmail.com>"
- "My Name2 <myaccount2@gmail.com>"
- "My Name3 <myaccount3@example.org>"

一个定制的 ~/.muttrc 例子，支持 3 个智能主机用于 3 个不同的源电子邮件地址。

```
set use_from
set from="My Name3 <myaccount3@example.org>"
set reverse_name
alternates myaccount1@gmail\.com|myaccount1@gmail\.com|myaccount3@example\.org

# ...

# MACRO
macro compose "1" "<edit-from>^UMy Name1 \<myaccount1@gmail.com\>\n"
macro compose "2" "<edit-from>^UMy Name2 \<myaccount2@gmail.com\>\n"
macro compose "3" "<edit-from>^UMy Name3 \<myaccount3@example.org\>\n"

send2-hook '~f myaccount1@gmail.com' "set sendmail = '/usr/bin/msmtp --read-envelope-from'"
send2-hook '~f myaccount2@gmail.com' "set sendmail = '/usr/bin/msmtp --read-envelope-from'"
send2-hook '~f myaccount3@example.org' "set sendmail = '/usr/bin/msmtp --read-envelope-from ↩
    '"

# ...
```

让我们来安装 msmtp-gnome 并按下面的方式设置 ~/.msmtprc。

```
defaults
logfile ~/.msmtp.log
domain myhostname.example.org
tls on
tls_starttls on
tls_certcheck on
tls_trust_file /etc/ssl/certs/ca-certificates.crt
auth on
port 587
auto_from

account myaccount1@gmail.com
host smtp.gmail.com
from  myaccount1@gmail.com
user  myaccount1@gmail.com

account myaccount2@gmail.com
host smtp.gmail.com
from  myaccount2@gmail.com
user  myaccount2@gmail.com

account myaccount3@example.org
host mail.example.org
from  myaccount3@example.org
user  myaccount3@example.org

account default : myaccount3@example.org
```

然后，增加密码数据到 Gnome 钥匙环。例如：

```
$ secret-tool store --label=msmtp \
    host smtp.gmail.com \
    service smtp \
    user myaccount1@gmail.com
...
```

提示
如果你不想使用 Gnome 钥匙环，你可以通过安装 msmtp 软件包来代替，在 ~/.msmtprc 文件里面，给每一个账号增加一个类似”password secret123” 的条目。更多信息请参见 memtp 文档。

6.5 远程邮件检索和转发实用工具

而不是手动运行 MUA 去访问远程邮件并去处理它们，你可能希望自动化这些过程，然后把所有邮件都投递到本地。远程邮件检索和转发实用工具很适合你使用。

尽管 fetchmail(1) 已经成为 GNU/Linux 用于远程邮件检索的事实上的标准，作者现在还是喜欢 getmail(1)。如果你想要在下载邮件之前拒绝邮件来达到节省带宽的目的，mailfilter 或 mpop 工具可能是很有用的。不管使用哪种邮件检索实用程序，配置系统使之能够投递已检索的邮件到 MDA 会是个不错的主意，例如通过管道的 maildrop。

软件包	流行度	大小	说明
fetchmail	V:6, I:19	2376	邮件检索 (POP3, APOP, IMAP) (旧的)
getmail	V:0, I:0	661	邮件检索 (POP3, IMAP4 和 SDPS) (简单、安全和可靠)
mailfilter	V:0, I:0	271	有正则表达式过滤功能的邮件检索 (POP3)
mpop	V:0, I:0	373	邮件检索 (POP3) 和带有过滤功能的 MDA

Table 6.9: 远程邮件检索和转发实用程序列表

6.5.1 getmail 配置

getmail(1) 的配置在getmail documentation里描述. 这里是我作为用户搭建访问多个 POP3 帐号.

按如下所示的创建"/usr/local/bin/getmails"。

```
#!/bin/sh
set -e
if [ -f $HOME/.getmail/running ]; then
  echo "getmail is already running ... (if not, remove $HOME/.getmail/running)" >&2
  pgrep -l "getmai[l]"
  exit 1
else
  echo "getmail has not been running ... " >&2
fi
if [ -f $HOME/.getmail/stop ]; then
  echo "do not run getmail ... (if not, remove $HOME/.getmail/stop)" >&2
  exit
fi
if [ "x$1" = "x-l" ]; then
  exit
fi
rcfiles="/usr/bin/getmail"
for file in $HOME/.getmail/config/* ; do
  rcfiles="$rcfiles --rcfile $file"
done
date -u > $HOME/.getmail/running
eval "$rcfiles $@"
rm $HOME/.getmail/running
```

按如下所示的配置它。

```
$ sudo chmod 755 /usr/local/bin/getmails
$ mkdir -m 0700 $HOME/.getmail
$ mkdir -m 0700 $HOME/.getmail/config
$ mkdir -m 0700 $HOME/.getmail/log
```

按如下所示的为每个 POP3 账户创建"$HOME/.getmail/config/pop3_name" 配置文件。

```
[retriever]
type = SimplePOP3SSLRetriever
server = pop.example.com
username =  pop3_name@example.com
password = <your-password>

[destination]
type = MDA_external
```

```
path = /usr/bin/maildrop
unixfrom = True

[options]
verbose = 0
delete = True
delivered_to = False
message_log = ~/.getmail/log/pop3_name.log
```

按如下所示的配置它。

```
$ chmod 0600 $HOME/.getmail/config/*
```

计划使用 cron(8) 每 15 分钟运行一次”/usr/local/bin/getmails”,通过执行”sudo crontab -e -u <user_name>” 并把如下所示的命令添加到用户的 cron 条目中。

```
5,20,35,50 * * * * /usr/local/bin/getmails --quiet
```

提示
POP3 访问的问题可能并不来自于 getmail。一些主流的免费 POP3 服务可能违反了 POP3 协议并且它们的垃圾邮件过滤机制可能不是非常完美。例如，它们可能在刚刚接收到 RETR 命令并且没有接收到 DELE 命令就可能删除了邮件并且可能隔离邮件到垃圾邮件信箱。你应该尽可能的减少损害，通过配置它们使之成为可访问的归档文件并且不要删除它们。参见 ”Some mail was not downloaded”。

6.5.2 fetchmail 配置

”/etc/default/fetchmail”,”/etc/fetchmailrc” 和”$HOME/.fetchmailrc” 可以配置 fetchmail(1)。参见”/usr/sh 配置例子。

6.6 带有过滤器的邮件投递代理 (MDA)

大多数 MTA 程序，例如 postfix 和 exim4，兼任 MDA (邮件投递代理)。这里有专门的带有过滤功能的 MDA。

尽管 procmail(1) 已经成为 GUN/Linux 上关于带有过滤器的 MDA 的事实标准，作者现在还是喜欢 maildrop(1)。不管使用哪种过滤程序，配置系统使之能投递已过滤的邮件到 qmail 风格的 Maildir 都是一个好主意。

软件包	流行度	大小	说明
procmail	V:54, I:432	300	有过滤器的 MDA (旧的)
mailagent	V:0, I:8	1283	带有 Perl 过滤器的 MDA
maildrop	V:1, I:3	1141	有结构化过滤语言的 MDA

Table 6.10: 有过滤器的 MDA 列表

6.6.1 maildrop 配置

maildrop(1) 配置在 maildropfilter documentation 中有说明。这里有一个关于”$HOME/.mailfilter 文件的配置例子。

```
# Local configuration
MAILROOT="$HOME/Mail"
# set this to /etc/mailname contents
MAILHOST="example.dom"
logfile $HOME/.maildroplog

# rules are made to override the earlier value by the later one.

# mailing list mails ?
if (     /^Precedence:.*list/:h || /^Precedence:.*bulk/:h )
{
    # rules for mailing list mails
    # default mailbox for mails from mailing list
    MAILBOX="Inbox-list"
    # default mailbox for mails from debian.org
    if ( /^(Sender|Resent-From|Resent-Sender): .*debian.org/:h )
    {
        MAILBOX="service.debian.org"
    }
    # default mailbox for mails from bugs.debian.org (BTS)
    if ( /^(Sender|Resent-From|Resent-sender): .*@bugs.debian.org/:h )
    {
        MAILBOX="bugs.debian.org"
    }
    # mailbox for each properly maintained mailing list with "List-Id: foo" or "List-Id:  ↩
        ...<foo.bar>"
    if ( /^List-Id: ([^<]*<)?([^<>]*)>?/:h )
    {
        MAILBOX="$MATCH2"
    }
}
else
{
    # rules for non-mailing list mails
    # default incoming box
    MAILBOX="Inbox-unusual"
    # local mails
    if ( /Envelope-to: .*@$MAILHOST/:h )
    {
        MAILBOX="Inbox-local"
    }
    # html mails (99% spams)
    if ( /DOCTYPE html/:b ||\
         /^Content-Type: text\/html/ )
    {
        MAILBOX="Inbox-html"
    }
    # blacklist rule for spams
    if ( /^X-Advertisement/:h ||\
         /^Subject:.*BUSINESS PROPOSAL/:h ||\
         /^Subject:.*URGENT.*ASISSTANCE/:h ||\
         /^Subject: *I NEED YOUR ASSISTANCE/:h )
    {
        MAILBOX="Inbox-trash"
    }
    # whitelist rule for normal mails
    if ( /^From: .*@debian.org/:h ||\
         /^(Sender|Resent-From|Resent-Sender): .*debian.org/:h ||\
         /^Subject: .*(debian|bug|PATCH)/:h )
    {
        MAILBOX="Inbox"
```

```
    }
    # whiltelist rule for BTS related mails
    if ( /^Subject: .*Bug#.*/:h ||\
         /^(To|Cc): .*@bugs.debian.org/:h )
    {
        MAILBOX="bugs.debian.org"
    }
    # whitelist rule for getmails cron mails
    if ( /^Subject: Cron .*getmails/:h )
    {
        MAILBOX="Inbox-getmails"
    }
}

# check existance of $MAILBOX
`test -d $MAILROOT/$MAILBOX`
if ( $RETURNCODE == 1 )
{
    # create maildir mailbox for $MAILBOX
    `maildirmake $MAILROOT/$MAILBOX`
}
# deliver to maildir $MAILBOX
to "$MAILROOT/$MAILBOX/"
exit
```

警告
不像 procmail，maildrop 不会自动创建不存在的 maildir 目录。你必须提前使用 maildirmake(1) 手动创建它们，正如"$HOME/.mailfilter" 例子里的那样。

6.6.2 procmail 配置

这里有一个 procmail(1) 的"$HOME/.procmailrc" 文件的类似配置例子。

```
MAILDIR=$HOME/Maildir
DEFAULT=$MAILDIR/Inbox/
LOGFILE=$MAILDIR/Maillog
# clearly bad looking mails: drop them into X-trash and exit
:0
* 1^0 ^X-Advertisement
* 1^0 ^Subject:.*BUSINESS PROPOSAL
* 1^0 ^Subject:.*URGENT.*ASISSTANCE
* 1^0 ^Subject: *I NEED YOUR ASSISTANCE
X-trash/

# Delivering mailinglist messages
:0
* 1^0 ^Precedence:.*list
* 1^0 ^Precedence:.*bulk
* 1^0 ^List-
* 1^0 ^X-Distribution:.*bulk
{
:0
* 1^0 ^Return-path:.*debian-devel-admin@debian.or.jp
jp-debian-devel/
```

```
:0
* ^Resent-Sender.*debian-user-request@lists.debian.org
debian-user/

:0
* ^Resent-Sender.*debian-devel-request@lists.debian.org
debian-devel/

:0
* ^Resent-Sender.*debian-announce-request@lists.debian.org
debian-announce

:0
mailing-list/
}

:0
Inbox/
```

6.6.3 重新投递 mbox 内容

如果你的家目录已经满了并且 procmail(1) 失败了，你需要从"/var/mail/<username>" 目录手动投递邮件到家目录下的已分类好的邮箱中。家目录有空闲空间以后，运行如下命令。

```
# /etc/init.d/${MAILDAEMON} stop
# formail -s procmail </var/mail/<username>
# /etc/init.d/${MAILDAEMON} start
```

6.7 POP3/IMAP4 服务器

如果将要在局域网上运行一个私有服务器，你应该考虑运行 POP3 / IMAP4 服务器，用来投递邮件到局域网客户端。

软件包	流行度	大小	类型	说明
courier-pop	V:3, I:4	288	POP3	Courier 邮件服务器 - POP3 服务器 (只有 maildir 格式)
cyrus-pop3d	V:0, I:0	153	POP3	Cyrus 邮件系统 (支持 POP3)
courier-imap	V:4, I:6	564	IMAP	Courier 邮件服务器 - IMAP 服务器 (只支持 maildir 格式)
cyrus-imapd	V:1, I:1	466	IMAP	Cyrus 邮件系统 (支持 IMAP)

Table 6.11: POP3/IMAP4 服务器列表

6.8 打印服务和工具

在老的类 Unix 系统，BSD Line printer daemon 行打印机后台守护 是标准。因此，在类 Unix 系统中，自由软件的标准打印输出格式是 PostScript，为了能够打印到非 PostScript 打印机，需要将一些过滤器系统和 Ghostscript 一道使用。

近来, Common UNIX Printing System 通用 UNIX 打印系统 (CUPS) 是新的事实标准。CUPS 使用 Internet Printing Protocol 互联网打印协议 (IPP). IPP 现在已经被其它操作系统，如 Windows XP 和 Mac OS X，支持。它已经变成新的具备双向通信能力的跨平台远程打印的事实标准。

Debian 系统上的应用程序的标准打印数据格式是 PostScript (PS)，它是一个页描述语言。PS 格式的数据被送到 Ghostscript PostScript 解释器来生成特定的打印机可打印的数据。参见第 11.4.1 节.

幸亏有 CUPS 系统的文件格式依赖自动转化特征，简单的发送任何数据到 `lpr` 命令，都将产生期望的打印输出。(在 CUPS 里, `lpr` 能够通过安装 `cups-bsd` 软件包来获取.)

Debian 系统有一些不错的软件包用于打印服务和作为打印工具。

软件包	流行度	大小	端口	说明
`lpr`	V:5, I:6	362	printer (515)	BSD lpr/lpd (线性打印机后台守护进程 daemon)
`lprng`	V:1, I:1	3852	同上	, , (增强)
`cups`	V:252, I:432	1127	IPP (631)	互联网打印 CUPS 服务器
`cups-client`	V:60, I:493	523	同上	用于 CUPS 的 System V 打印机命令: lp(1), lpstat(1), lpoptions(1), cancel(1), lpmove(8), lpinfo(8), lpadmin(8), ⋯
`cups-bsd`	V:38, I:423	127	同上	用于 CUPS 的 BSD 打印机命令: lpr(1), lpq(1), lprm(1), lpc(8)
`printer-driver-gutenprint`	V:133, I:420	930	没有使用	CUPS 打印机驱动

Table 6.12: 打印服务和工具列表

提示
你可以让你的 web 浏览器访问"http://localhost:631/" 来配置 CUPS 系统。

6.9 服务器远程访问和工具 (SSH)

Secure SHell (SSH) 是因特网上的安全连接方式。在 Debian 里面，有一个叫 OpenSSH 的免费 SSH 版本，在 `openssh-client` 和 `openssh-server` 包里。

软件包	流行度	大小	工具	说明
`openssh-client`	V:811, I:994	3545	ssh(1)	SSH 客户端
`openssh-server`	V:686, I:813	1449	sshd(8)	SSH 服务端
`ssh-askpass-fullscreen`	V:0, I:1	42	ssh-askpass-fullscreen(1)	请求用户输入密码的 ssh-add (GNOME2)
`ssh-askpass`	V:3, I:40	101	ssh-askpass(1)	请求用户输入密码的 ssh-add (plain X)

Table 6.13: 服务器远程访问和工具列表

小心
如果你的 SSH 是从因特网来访问，参见第 4.7.3 节。

提示
请使用 screen(1) 程序来让远程 shell 在中断的连接上存活 (参见第 9.1 节).

6.9.1 SSH 基础

警告
如果想要运行 OpenSSH 服务，"/etc/ssh/sshd_not_to_be_run" 必须不存在。

SSH 有两个认证协议。

SSH 协议	**SSH** 方式	说明
SSH-1	"RSAAuthentication"	基于 RSA 身份秘钥的用户认证
同上	"RhostsAuthentication"	".rhosts" 基于主机的认证（不安全，禁用）
同上	"RhostsRSAAuthentication"	".rhosts" 使用 RSA 主机秘钥的主机认证（禁用）
同上	"ChallengeResponseAuthentication"	RSA 质疑-应答认证
同上	"PasswordAuthentication"	基于密码的认证
SSH-2	"PubkeyAuthentication"	基于公钥的用户认证
同上	"HostbasedAuthentication"	"~/.rhosts" or "/etc/hosts.equiv" 使用客户端主机公钥的主机认证（禁用）
同上	"ChallengeResponseAuthentication"	质疑-应答认证
同上	"PasswordAuthentication"	基于密码的认证

Table 6.14: SSH 认证协议和方式列表

小心
如果你使用一个非 Debian 的系统，请小心注意这些不同。

细节参见"/usr/share/doc/ssh/README.Debian.gz", ssh(1), sshd(8), ssh-agent(1), and ssh-keygen(1).

下面是秘钥配置文件。

配置文件	配置文件描述
/etc/ssh/ssh_config	SSH 客户端默认, 参见 ssh_config(5)
/etc/ssh/sshd_config	SSH 服务端默认, 参见 sshd_config(5)
~/.ssh/authorized_keys	该账户连接到这个服务器上的客户端使用的默认 SSH 公钥
~/.ssh/identity	用户的 SSH-1 RSA 私钥
~/.ssh/id_rsa	用户的 SSH-2 RSA 私钥
~/.ssh/id_dsa	用户的 SSH-2 DSA 私钥

Table 6.15: SSH 配置文件列表

提示
参见 ssh-keygen(1), ssh-add(1) 和 ssh-agent(1) 来了解怎样使用 SSH 公钥和私钥。

提示
一定要通过连接测试来确认设置。有任何问题的连接，使用"ssh -v".

提示
稍后可以使用"ssh-keygen -p" 改变密码来加密本地 SSH 私钥.

提示
你可以在"~/.ssh/authorized_keys" 里给条目增加选项来限制主机和运行特定的命令。细节请参见 sshd(8).

从客户端启动一个 ssh(1) 连接.

命令	说明
ssh username@hostname.domain.ext	使用默认模式连接
ssh -v username@hostname.domain.ext	有详细信息的默认连接模式
ssh -1 username@hostname.domain.ext	强制使用 SSH 1 版本连接
ssh -1 -o RSAAuthentication=no -l username hostname.domain.ext	SSH 1 版本，强制使用密码
ssh -o PreferredAuthentications=password -l username hostname.domain.ext	SSH 2 版本，强制使用密码

Table 6.16: SSH 客户端启动例子列表

如果本地和远程主机，使用同样的用户名，你可以省略输入"username@". 即使在本地和远程主机使用不同的用户名，你可以使用"~/.ssh/config" 来省略输入用户名. 对于 Debian Salsa 服务器，使用账户名"foo-guest", 你可以设置"~/.ssh/config" 包含下面的内容。

```
Host salsa.debian.org people.debian.org
    User foo-guest
```

对于用户来讲, ssh(1) 功能比 telnet(1) 更加智能和安全. 不像 telnet 命令, ssh 命令不会在遇到 telnet 的退出字符 (初始默认是 CTRL-]) 时停止.

6.9.2 SMTP/POP3 隧道的端口转发

通过 ssh 建立一个这样的管道连接，从 localhost 的 4025 端口到 remote-server 的 25 端口，并从 localhost 的 4110 端口到 remote-server 的 110 端口，请在本机执行如下命令.

```
# ssh -q -L 4025:remote-server:25 4110:remote-server:110 username@remote-server
```

这是跨越因特网建立 SMTP/POP3 服务连接的安全方法。在远程主机"/etc/ssh/sshd_config" 里设置"AllowTcpForwarc
条目为"yes".

6.9.3 免密码远程连接

使用"RSAAuthentication" (SSH-1 协议) 或"PubkeyAuthentication" (SSH-2 协议)，人们可以避免记住远程系统的密码.

在远程系统的"/etc/ssh/sshd_config" 里,设置相应的条目, "RSAAuthentication yes" 或"PubkeyAuthenticatio
yes"。

在本地生成授权秘钥对，并安装公钥到远程系统。

- ”RSAAuthentication”: SSH-1 的 RSA key (不建议使用，因为已被废弃.)

```
$ ssh-keygen
$ cat .ssh/identity.pub | ssh user1@remote ”cat - >>.ssh/authorized_keys”
```

- ”PubkeyAuthentication”: SSH-2 的 RSA key

```
$ ssh-keygen -t rsa
$ cat .ssh/id_rsa.pub | ssh user1@remote ”cat - >>.ssh/authorized_keys”
```

- ”PubkeyAuthentication”: SSH-2 的 DSA key(不建议，因为慢.)

```
$ ssh-keygen -t dsa
$ cat .ssh/id_dsa.pub | ssh user1@remote ”cat - >>.ssh/authorized_keys”
```

提示
使用 SSH-2 的 DSA key 是不建议的，应为 key 较小并且慢。由于 RSA 专利已经过期，没有理由使用 DSA 来作为规避 RSA 专利的临时措施。DSA 表示 Digital Signature Algorithm，速度慢。同时参见 DSA-1571-1.

注意
为了让”HostbasedAuthentication” 在 SSH-2 下运行，你必须同时调整服务端主机”/etc/ssh/sshd_config” 和客户机”/etc/ssh/ssh_config” 或”~/.ssh/config” 的”HostbasedAuthentication” 配置为”yes” 。

6.9.4 处理其它 SSH 客户端

其它平台上有一些免费的 SSH 客户端。

环境	免费 SSH 程序
Windows	puTTY (http://www.chiark.greenend.org.uk/~sgtatham/putty/) (GPL)
Windows (cygwin)	cygwin 里的 SSH (http://www.cygwin.com/) (GPL)
Macintosh 类	macSSH (http://www.macssh.com/) (GPL)
Mac OS X	OpenSSH; 在终端应用中使用 ssh (GPL)

Table 6.17: 其它平台上免费 SSH 客户端列表

6.9.5 建立 ssh 代理

用密码来保护你的 SSH 认证私钥是安全的。如果密码没有设置，使用”ssh-keygen -p” 来设置。

把你的公钥 (比如: ”~/.ssh/id_rsa.pub”) 放到远程主机的”~/.ssh/authorized_keys”，这个远程主机使用上面描述的基于密码的连接方式。

```
$ ssh-agent bash
$ ssh-add ~/.ssh/id_rsa
Enter passphrase for /home/<username>/.ssh/id_rsa:
Identity added: /home/<username>/.ssh/id_rsa (/home/<username>/.ssh/id_rsa)
```

从这里执行接下来的命令，就不再需要密码。

```
$ scp foo <username>@remote.host:foo
```

按 ^D 来终结 ssh 代理会话。

对于 X 服务端，通常的 Debian 启动脚本会作为父进程执行 `ssh-agent`。所以你只需要执行一次 `ssh-add`。进一步的信息，请阅读 ssh-agent(1) 和 ssh-add(1).

6.9.6 怎样通过 SSH 关闭远程系统

你可以使用 at(1) 命令 (参见第 9.3.13 节) 来从 SSH 终端里保护"`shutdown -h now`" (参见第 1.1.8 节) 操作过程。

```
# echo "shutdown -h now" | at now
```

在 screen(1) (参见第 9.1 节) 会话里运行"`shutdown -h now`"，是另外一个方法来做这同样的事情。

6.9.7 SSH 故障排查

如果你遇到问题，检查配置文件的权限并用"`-v`" 选项运行 `ssh`。

如果你是 root 账户，并有使用防火墙，使用"`-p`" 选项; 这可以避免使用 1 —1023 之间的服务端口.

如果 `ssh` 连接到远程站点突然停止工作，这也许是系统管理员胡乱操作的结果，可能是在系统维护时改变了"`host_key`". 在确认这个情况后,并且没有人试图用聪明的黑客技术来篡改远程主机,你可以在本机"`~/.ssh/known_hos`里删除"`host_key`" 条目来重新获得连接。

6.10 其它网络应用服务

这里是其它网络应用服务。

通用互联网文件系统协议 (CIFS) 和服务消息块 (SMB) 协议一样，被微软 Windows 广泛应用。

提示
参见第 4.5.2 节服务系统集成。

提示
主机名解析通常由 DNS 服务提供. 对于由 DHCP 动态分配的主机 IP 地址, 动态 DNS 能够使用 `bind9` 和 `isc-dhcp-server` 建立主机名解析，Debian wiki 的 DDNS 页 有说明.

提示
使用 `squid` 之类的代理服务器，和使用 Debian 文档库的完全本地镜像服务器相比，能够大量节省带宽。

软件包	流行度	大小	协议	说明
telnetd	V:1, I:3	115	TELNET	TELNET 服务
telnetd-ssl	V:0, I:0	165	同上	TELNET 服务 (支持 SSL)
nfs-kernel-server	V:35, I:83	345	NFS	Unix 文件共享
samba	V:104, I:171	15967	SMB	Windows 文件和打印共享
netatalk	V:2, I:4	1805	ATP	Apple/Mac 文件和打印共享 (AppleTalk)
proftpd-basic	V:30, I:38	5189	FTP	通用文件下载
apache2	V:250, I:340	598	HTTP	通用 web 服务器
squid	V:13, I:16	8374	同上	通用 web 代理服务器
squid3	V:7, I:14	239	同上	同上
bind9	V:54, I:73	2144	DNS	其它主机的 IP 地址
isc-dhcp-server	V:23, I:60	1434	DHCP	客户端自身的 IP 地址

Table 6.18: 其它网络应用服务列表

6.11 其它网络应用客户端

这里是其它网络应用客户端。

软件包	流行度	大小	协议	说明
netcat	I:35	16	TCP/IP	TCP/IP 瑞士军刀
openssl	V:808, I:992	1452	SSL	安全套接字层 (SSL) 二进制和相关的加密工具
stunnel4	V:4, I:18	487	同上	通用 SSL 封装
telnet	V:64, I:907	163	TELNET	TELNET 客户端
telnet-ssl	V:1, I:6	209	同上	TELNET 服务 (支持 SSL)
nfs-common	V:248, I:454	758	NFS	Unix 文件共享
smbclient	V:17, I:187	1798	SMB	微软 Windows 文件和打印共享客户端
cifs-utils	V:35, I:120	231	同上	远程微软 Windows 文件系统挂载和卸载命令
ftp	V:24, I:430	137	FTP	FTP 客户端
lftp	V:6, I:39	2254	同上	同上
ncftp	V:4, I:26	1486	同上	全屏 FTP 客户端
wget	V:383, I:998	3257	HTTP 和 FTP	web 下载工具
curl	V:136, I:490	400	同上	同上
axel	V:0, I:5	190	同上	下载加速器
aria2	V:2, I:18	1848	同上	BitTorrent 和 Metalink 支持的下载加速器
bind9-host	V:442, I:952	358	DNS	来自 bind9 的 host(1), "Priority: standard"
dnsutils	V:71, I:591	719	同上	来自 bind 的 dig(1), "Priority: standard"
isc-dhcp-client	V:255, I:973	673	DHCP	获得 IP 地址
ldap-utils	V:17, I:76	709	LDAP	从 LDAP 服务获取数据

Table 6.19: 网络应用客户端列表

6.12 系统后台守护进程 (daemon) 诊断

`telnet` 程序能够手工连接到系统后台守护进程 (daemon)，并进行诊断。

测试纯 POP3 服务，尝试用下面的操作

```
$ telnet mail.ispname.net pop3
```

部分 ISP 提供 TLS/SSL 加密的POP3 服务，为了测试它，你需要用到 `telnet-ssl` 包里支持 TLS/SSL 的 `telnet` 客户端，或 `openssl` 软件包。

```
$ telnet -z ssl pop.gmail.com 995
```

```
$ openssl s_client -connect pop.gmail.com:995
```

下面的 RFCs 提供每一个系统后台守护进程（daemon）所需要的知识。

RFC	说明
rfc1939 和 rfc2449	POP3 服务
rfc3501	IMAP4 服务
rfc2821 (rfc821)	SMTP 服务
rfc2822 (rfc822)	邮件文件格式
rfc2045	多用途互联网邮件扩展 (MIME)
rfc819	DNS 服务
rfc2616	HTTP 服务
rfc2396	URI 定义

Table 6.20: 常用 RFC 列表

在”`/etc/services`” 里，描述了端口用途.

Chapter 7

X 窗口系统

警告
本章是基于 2013 年发布的 Debian 7.0 (wheezy) 编写的，所以其内容正在变得过时。

Debian 上的 X 窗口系统基于 X.Org 的源代码。

7.1 关键软件包

这里有一些用于简易安装的（元）软件包。

（元）软件包	流行度	大小	说明
xorg	I:500	52	X 库、一个 X 服务器、一系列字体以及一组基础的 X 客户端和工具（元软件包）
xserver-xorg	V:83, I:538	238	X 服务器的全部套件及其配置
xbase-clients	I:38	46	各种 X 客户端（元软件包）
x11-common	V:424, I:792	305	X 窗口系统的文件系统的基础设施
xorg-docs	I:7	2032	X.Org 软件套件的各种文档
menu	V:73, I:246	1435	为理解菜单的应用生成 Debian 菜单
menu-xdg	V:35, I:129	27	将 Debian 菜单结构转换为 freedesktop.org xdg 菜单结构
xdg-utils	V:260, I:559	327	freedesktop.org 提供的集成桌面环境的工具
task-gnome-desktop	I:202	6	标准 GNOME 桌面环境（元软件包）
task-kde-desktop	I:69	6	核心 KDE 桌面环境（元软件包）
task-xfce-desktop	I:111	6	Xfce 轻量级桌面环境（元软件包）
task-lxde-desktop	I:44	6	LXDE 轻量级桌面环境（元软件包）
fluxbox	V:2, I:11	3860	Fluxbox：可配置度高且资源耗费低的 X 窗口管理器

Table 7.1: X 窗口的关键（元）软件包列表

关于 X 基础知识，请参考 X(7) 和 the LDP XWindow-User-HOWTO。

7.2 设置桌面环境

一个 桌面环境 通常是一个 X 窗口管理器、一个文件管理器和一套兼容的实用程序组合而成。

你能够在 aptitude 的任务菜单下安装全部的桌面环境，例如 GNOME，KDE，Xfce 或者 LXDE。

提示
在 Debian unstable/testing 下，任务菜单可能与最新的软件包过渡状态不同步。在这种情况下，您需要取消选择 aptitude(8) 任务菜单下列出的一些（元) 包。当取消选择（元）软件包时，您必须选择那些提供依赖项手动操作的软件包，避免它们被自动删除。

你也可以手动只建立一个 X 窗口管理器，例如 Fluxbox。

关于 X 窗口管理器和桌面环境的介绍，参见 Window Managers for X。

7.2.1 Debian 菜单

Debian 菜单系统从 menu 软件包中为带有 update-menus(1) 的文本和 X 程序提供了一个通用接口。每个软件包都会将它的菜单数据安装到“/usr/share/menu/”目录。参见“/usr/share/menu/README”。

7.2.2 Freedesktop.org 菜单

每个遵从 Freedesktop.org xdg 菜单的软件包都会将由“*.desktop”提供的菜单数据安装到“/usr/share/applications/”下。符合 Freedesktop.org 标准的现代桌面环境会利用它们的数据使用 xdg-utils 软件包生成菜单。参见“/usr/share/doc/x

7.2.3 从 Freedesktop.org 菜单到 Debian 菜单

为了从遵从 Freedesktop.org 菜单的窗口管理器环境（例如 GNOME 和 KDE）访问传统的 Debian 菜单，你必须安装 menu-xdg 软件包。

7.3 服务器/客户端关系

X Window 系统作为服务和客户端程序的组合被启动. 在这里，**server** 和 **client** 的意义需要和 **local** 和 **remote** 区分开来.

类型	说明
X 服务器	一个运行在本地主机上的程序，连接了使用者的显示和输入设备。
X 客户端	一个运行在远程主机上的程序，它会与 X 服务器进行数据的处理和传输。
应用程序服务器	一个运行在远程主机上的程序，会与应用程序客户端进行数据处理和传输。
应用程序客户端	一个运行在本地主机上的程序，连接了使用者的显示和输入设备。

Table 7.2: 服务器/客户端术语表

现代 X 服务器具有MIT 共享内存扩展 ，他们和本地 X 客户端通过本地共享内存进行通讯. 这就绕过了网络透明的 Xlib 进程间通讯通道，提升了大型图像的处理性能.

7.4 X 服务器

关于 X 服务器的信息，参见 xorg(1)。

7.4.1 X 服务器的（重新）配置

使用下面命令（重新）配置一个 X 服务器。

```
# dpkg-reconfigure --priority=low x11-common
```

> 注意
> 新的 Linux 内核使用 DRM、KMS 和 udev，对图形和输入设备进行了良好的支持。X 服务器被重写以使用它们。因此“/etc/X11/xorg.conf”通常不存在于你的系统中。这些参数由内核进行配置。参见 Linux 内核文档中的“fb/modedb.txt”。

对于高分辨率的 CRT 大显示器，最好将将刷新率设置为显示器支持的最高值（85 Hz 不错，75 Hz 也行）以减少闪烁。对于 LCD 显示器，较慢的标准刷新率（60 Hz）就可以了，因为它的反应较慢。

> 注意
> 当心，别使用过高的刷新率，这可能会导致你的显示器系统发生重大的硬件故障。

7.4.2 连接到 X 服务器的方式

这里有一些方式，可以使“X 服务器”（显示端）接受来自“X 客户端”（应用端）的连接。

软件包	流行度	大小	用户	加密	方式	相关使用
xbase-clients	I:38	46	不检查	无	xhost 命令	弃用
xbase-clients	I:38	46	检查	无	xauth 命令	通过管道进行本地连接
openssh-client	V:811, I:994	3545	检查	有	ssh -X 命令	远程网络连接
gdm3	V:189, I:263	4791	检查	无（XDMCP）	GNOME 显示管理器	通过管道进行本地连接
sddm	V:46, I:82	1830	检查	无（XDMCP）	KDE 显示管理器	通过管道进行本地连接
xdm	V:3, I:7	665	检查	无（XDMCP）	X 显示管理器	通过管道进行本地连接
wdm	V:80, I:433	2323	检查	无（XDMCP）	WindowMaker 显示管理器	通过管道进行本地连接
ldm	V:0, I:1	414	检查	有	LTSP 显示管理器	远程 SSH 网络连接（瘦客户端）

Table 7.3: 连接到 X 服务器的方式

警告
不要在不安全的网络中使用远程 TCP/IP 进行 X 连接，除非你有非常好的理由，例如使用了加密。未加密的远程 TCP/IP socket 连接容易遭到窃听攻击并且 Debian 系统默认禁用了它。请使用“`ssh -X`”。

警告
也不要在不安全的网络中使用 XDMCP 连接。它通过未加密的 UDP/IP 传输数据，很容易遭到窃听攻击。

提示
LTSP 代表 Linux 终端服务器项目（Linux Terminal Server Project）。

7.5 启动 X 窗口系统

X 窗口系统通常是作为 X 会话启动的，X 会话是由 X 服务器和连接客户端组成的。对于常规的桌面系统，它们两个都是在一个工作站上运行的。

X 会话 由以下方式之一启动。

- 从命令行用 `startx` 命令启动
- 基于”`graphical.target`” 的依赖关系，一个 X 图形显示管理器 的后台守护程序 `*dm` 通过 `systemd` 启动。

提示
显示管理器后台守护进程（daemon）的启动脚本在实际执行它们自己时会检查“`/etc/X11/default-display-manager`”文件的内容。这可以确保只有一个 X 显示管理器 后台守护进程（daemon）程序被激活。

提示
关于 X 显示管理器的初始环境变量，参见第 8.4.5 节。

本质上，所有的这些程序都是执行“`/etc/X11/Xsession`”脚本。之后“`/etc/X11/Xsession`”脚本通过 run-parts(8) 执行“`/etc/X11/Xsession.d/`”目录中的脚本。这本质上是通过内建的 `exec` 命令执行按下面顺序第一个被找到的程序。

1. X 显示管理器调用”`/etc/X11/Xsession`” 的参数中指定的脚本, 如果他被定义了。
2. “`~/.xsession`”或“`~/.Xsession`”脚本，如果它被定义。
3. “`/usr/bin/x-session-manager`”命令，如果它被定义。
4. “`/usr/bin/x-window-manager`”命令，如果它被定义。
5. “`/usr/bin/x-terminal-emulator`”命令，如果它被定义。

这个过程会受“`/etc/X11/Xsession.options`”的影响。“`/usr/bin/x-*`”命令所指向的准确程序，是由 Debian 选择系统决定的，并且可以通过“`update-alternatives --config x-session-manager`”修改。

细节参见 Xsession(5).

7.5.1 使用 gdm3 启动 X 会话

gdm3(1) 让你从它的菜单中选择会话类型（或桌面环境：第 7.2 节），还有 X 会话的语言（或语言环境：第 8.4 节）。它在“~/.dmrc”中像下面那样设定选择的默认值。

```
[Desktop]
Session=default
Language=ja_JP.UTF-8
```

7.5.2 自定义 X 会话（经典方式）

系统中的“/etc/X11/Xsession.options”包含没有前置“#”字符的“allow-user-xsession”行，则定义了“~/.xsession”或“~/.Xsession”的任何用户都能够通过完全覆盖系统代码来自定义“/etc/X11/Xsession”的动作。在“~/.xsession”文件中的最后一个命令应该使用“exec some-window/session-manager”这样的形式来启动你最喜欢的 X 窗口/会话管理器。

如果使用了这个特性，系统实用程序选择的显示 (或登录) 管理器 (DM), 会话管理器或者窗口管理器 (WM) 会被忽略。

7.5.3 自定义 X 会话（新方式）

以下是自定义 X 会话的新方式，它不会像上面那样完全覆盖系统代码。

- 显示管理器 gdm3 可以选择一个特定的会话并将它设置为“/etc/X11/Xsession”的参数。
  - ”/etc/profile”, ”~/.profile”, ”/etc/xprofile”, 和”~/.xprofile” 文件会被作为 gdm3 启动进程的一部分来执行。
- “~/.xsessionrc”文件作为启动进程的一部分被执行。（独立桌面）
  - ”/etc/X11/Xsession.options” 中的”#allow-user-xsession” 不会限制”~/.xsessionrc” 文件的执行。
- “~/.gnomerc”文件作为启动进程的一部分被执行。（仅 GNOME 桌面）

系统实用程序选择的显示 (或登录) 管理器 (DM), 会话管理器或者窗口管理器 (WM) 是相当不错的。

在这些配置文件里面，不应当有”exec …” 或”exit”.

7.5.4 通过 SSH 连接一个远程 X 客户端

使用“ssh -X”可以启用一个来自本地 X 服务器到远程应用程序服务器的安全连接。

如果你想避免命令行选项“-X”，你需要将远程主机的“/etc/ssh/sshd_config”文件中的“X11Forwarding”设置为“yes”。

在本地主机上启动 X 服务器。

在本地主机中打开一个 xterm。

通过下列命令，运行 ssh(1) 建立与远程站点的连接。

```
localname @ localhost $ ssh -q -X loginname@remotehost.domain
Password:
```

通过下列命令，在远程站点运行一个 X 应用程序，例如“gimp”。

```
loginname @ remotehost $ gimp &
```

这个方法可以显示来自远程 X 客户端的输出，相当于它是通过一个本地 UNIX 域名 socket 进行本地的连接。

7.5.5 连接互联网的安全 X 终端

连接互联网的 X 安全终端，并且会显示在远程运行的整个 X 桌面环境，这可以通过使用特定软件包（例如 `ldm`）轻松地做到。通过 SSH，你的本地机器会变成连接到远程应用程序服务器的一个安全瘦客户端。

7.6 X 窗口中的字体

在 2002 年，人们创建了发行版独立的库Fontconfig 2.0用于配置和定制字体访问. `squeeze` 以后的 Debian 使用 Fontconfig 2.0 进行字体配置.

X 窗口系统中的字体支持可以概括如下。

- 传统的 X 服务器端的字体支持系统
  - 原始的核心 X11 字体系统为旧版本的 X 客户端应用提供了向后兼容。
  - 原始的核心 X11 字体被安装到了 X 服务器上。
- 现代 X 客户端的字体支持系统
  - 现代 X 系统支持下列带有高级功能（例如抗锯齿）的所有字体（第 7.6.1 节，第 7.6.2 节和第 7.6.3 节）。
  - Xft 2.0 连接现代 X 应用，例如来自 GNOME、KDE 和带有 FreeType 2.0 库的LibreOffice 的应用。
  - FreeType 2.0 提供字体栅格化的库。
  - Fontconfig 为 Xft 2.0 解决了字体规范的问题。配置参见 fonts.conf(5)。
  - 所有使用 Xft 2.0 的现代 X 应用都可以与使用 X 渲染扩展 的现代 X 服务器进行交流。
  - X 渲染扩展 将字体访问和 glyph 图像生成从 X 服务器移动到了 X 客户端.

软件包	流行度	大小	说明
`xfonts-utils`	V:67, I:593	415	X 窗口系统字体实用程序
`libxft2`	V:102, I:712	122	Xft 的，一个库，连接到了带有 FreeType 字体栅格化库的 X 应用
`libfreetype6`	V:476, I:995	841	FreeType 2.0 字体栅格化库
`fontconfig`	V:395, I:832	575	Fontconfig 的，一个通用的字体配置库——支持二进制
`fontconfig-config`	V:406, I:902	441	Fontconfig 的，一个通用的字体配置库——配置数据

Table 7.4: 支持 X 窗口字体系统的软件包

你可以通过下列方法查看字体配置信息。

- 使用“`xset q`”查看核心 X11 字体路径
- 使用“`fc-match`”查看 fontconfig 的字体默认
- 使用“`fc-list`”查看所有可用的 fontconfig 字体

提示
“The Penguin and Unicode”很好地概述了现代的 X 窗口系统。在 http://unifont.org/ 中的其它文档提供了关于 Unicode 字体、支持 Unicode 的软件以及国际化的相关信息，还有 免费/自由/开源（free/libre/open source，FLOSS）操作系统中的 Unicode 可用性问题。

7.6.1 基础字体

计算机字体有两种主要的类型。

- 点阵字体（低分辨率栅格化下表现良好）
- 轮廓/笔画字体（高分辨率栅格化下表现良好）

缩放点阵字体会导致图像产生锯齿，而缩放轮廓/笔画字体则生成平滑的图像。

Debian 系统中的点阵字体通常由压缩的 X11 pcf 点阵字体文件 提供，它带有“`.pcf.gz`”文件后缀名。

Debian 系统中的轮廓字体由以下提供。

- PostScript Type 1 字体文件，它带有“`.pfb`”（二进制字体文件）和“`.afm`”（字体规格文件）文件后缀名。
- TrueType（或 OpenType）字体文件，通常带有“`.ttf` 文件后缀名。

> 提示
> OpenType 是为了取代 TrueType 和 PostScript Type 1。

字体软件包	流行度	大小	无衬线字体	衬线字体	等宽字体	字体来源
PostScript	N/A	N/A	Helvetica	Times	Courier	Adobe
gsfonts	I:677	4439	Nimbus Sans L	Nimbus Roman No9 L	Nimbus Mono L	URW（Adobe 兼容的大小）
gsfonts-x11	I:109	95	Nimbus Sans L	Nimbus Roman No9 L	Nimbus Mono L	PostScript Type 1 字体支持的 X 字体。
t1-cyrillic	I:22	4884	Free Helvetian	Free Times	Free Courier	URW 扩展（Adobe 兼容的大小）
lmodern	I:130	33270	LMSans*	LMRoman*	LMTypewriter*	基于现代计算机的可缩放 PostScript 和 OpenType 字体（来自 Tex）

Table 7.5: 相应的 PostScript Type 1 字体

> 提示
> DejaVu 字体基于 Bitstream Vera 字体，并对其进行了扩充。

7.6.2 其它字体

aptitude(8) 可以帮你轻松找到其它的字体。

- 简单的软件包列表位于“软件集” → “本地化”
- 平面软件包列表可以使用 debtag 正则表达式来过滤字体数据：“`~Gmade-of::data:font`”
- 在软件包名称里采用下面的正则表达式来过滤得到 BDF（位图）字体包列表: ”`~nxfonts-`”
- 在软件包名称里采用下面的正则表达式来得到 TrueType 字体包列表: ”`~nttf-|~nfonts-`”

字体软件包	流行度	大小	无衬线字体	衬线字体	等宽字体	字体来源
ttf-mscorefonts-installer	V:1, I:77	92	Arial	Times New Roman	Courier New	微软（Adobe 兼容的大小）（这会安装 non-free 数据）
fonts-liberation	I:559	2093	Liberation Sans	Liberation Serif	Liberation Mono	Liberation 字体项目（微软兼容的大小）
fonts-freefont-ttf	V:44, I:345	10750	FreeSans	FreeSerif	FreeMono	GNU 自由字体（微软兼容的大小）
fonts-dejavu	I:513	39	DejaVu Sans	DejaVu Serif	DejaVu Sans Mono	覆盖了 Unicode 的 DejaVu、Bitstream Vera
fonts-dejavu-core	V:306, I:796	2954	DejaVu Sans	DejaVu Serif	DejaVu Sans Mono	覆盖了 Unicode 的 DejaVu、Bitstream Vera（sans、sans-bold、serif、serif-bold、mono、mono-bold）
fonts-dejavu-extra	I:543	7217	N/A	N/A	N/A	覆盖了 Unicode 的 DejaVu、Bitstream Vera（oblique、italic、bold-oblique、bold-italic、condensed）
ttf-unifont	I:23	17654	N/A	N/A	unifont	GNU Unifont，带有 Unicode 5.1 基本多文种平面（BMP）中所有的可打印字符

Table 7.6: 对应的 TrueType 字体

因为自由字体有时会受限，因此对于 Debian 用户而言，可以选择安装或分享一些商业的 TrueType 字体。为了简化该过程，建立一些方便的软件包。

- `mathematica-fonts`
- `fonts-mscorefonts-installer`

当你付出使用非自由字体污染你自由系统的代价后，你会得到一些相当不错的 trueType 字体。

7.6.3 CJK 字体

下面是一些主要关注 CJK 字符 的字体。

字体类型	日文字体名称	中文字体名称	韩文字体名称
无衬线	gothic、ゴチック	hei、gothic	dodum、gulim、gothic
衬线	mincho、明朝	song、ming	batang

Table 7.7: CJK 字体名称中所使用的暗示字体类型的关键词

像“VL PGothic”这样带有“P”的字体名称是对应的“VL Gothic”字体修改宽度后的比例字体。

举个例子，Shift_JIS 的码表包含了 7070 个字符。它们可以像下面那样分类。

- JIS X 0201 单字节字符（191 个字符，又名半角字符）
- JIS X 0208 双字节字符（6879 个字符，又名全角字符）

使用修复宽度的 CJK 字体的双字节字符占用控制台终端的双倍宽度。为了应对这种情况，带有“.hbf”文件后缀名的 汉字点阵字体（Hanzi Bitmap Font，HBF）文件 被设计为包含了单字节和双字节字符的字体。

为了节省 TrueType 字体文件的空间，可以使用带有“.ttc”文件后缀名的 TrueType 字体集合文件。

为了覆盖复杂的编码字体空间, CID 采用”%!PS-Adobe-3.0 Resource-CMap” 开头的 CMap 文件作为PostScript类型 1 字体的关键字。这个几乎很少用在普通的 X 显示上，更多用于 PDF 等格式的文件渲染上。(参考第 7.7.2 节).

提示
对于Han unification，一些Unicode编码点阵期望能够有多样化的glyphs。其中最令人讨厌的在中日韩三个国家“U + 3001 顿号”和“U + 3002 表意的句号”的字符位置各不相同。配置日文中心字体的优先级，中文用的是”7~/.fonts.conf 8” 能够让日本感到有所平衡。

7.7 X 应用

7.7.1 X 办公应用

下面是一些基础的办公应用（LO 是 LibreOffice）。

软件包	流行度	软件包大小	类型	说明
libreoffice-writer	V:318, I:478	31712	LO	文字处理软件
libreoffice-calc	V:315, I:473	29409	LO	电子表格
libreoffice-impress	V:312, I:469	4175	LO	演示文稿
libreoffice-base	V:297, I:445	9211	LO	数据库管理
libreoffice-draw	V:313, I:470	9960	LO	矢量图形编辑器（绘图）
libreoffice-math	V:315, I:475	1469	LO	数学方程/公式编辑器
abiword	V:3, I:14	5074	GNOME	文字处理软件
gnumeric	V:13, I:25	9758	GNOME	电子表格
gimp	V:85, I:489	19016	GTK	位图图形编辑器 (绘图)
inkscape	V:129, I:332	78502	GNOME	矢量图形编辑器（绘图）
dia	V:18, I:37	3824	GTK	流程图和示意图编辑器
planner	V:3, I:7	1151	GNOME	项目管理
calligrawords	V:1, I:8	5837	KDE	文字处理软件
calligrasheets	V:0, I:6	11248	KDE	电子表格
calligrastage	V:0, I:6	5064	KDE	演示文稿
calligraplan	V:0, I:6	15402	KDE	项目管理
kexi	V:0, I:6	7547	KDE	数据库管理
karbon	V:1, I:7	4318	KDE	矢量图形编辑器（绘图）

Table 7.8: 基础的 X 办公应用

7.7.2 X 实用应用

下面是一些引起我注意的基础实用应用。

软件包	流行度	软件包大小	类型	说明
evince	V:170, I:405	936	GNOME	文档（pdf）阅读器
okular	V:69, I:122	13136	KDE	文档（pdf）阅读器
calibre	V:8, I:39	51670	KDE	电子书转换器和库管理
fbreader	V:2, I:18	3030	GTK	电子书阅读器
evolution	V:40, I:265	449	GNOME	个人信息管理（群组软体和电子邮件）
kontact	V:2, I:19	2071	KDE	个人信息管理（群组软体和电子邮件）
scribus	V:3, I:28	19995	KDE	桌面页面布局编辑器
glabels	V:0, I:4	1417	GNOME	标签编辑器
gnucash	V:3, I:13	22199	GNOME	个人会计
homebank	V:0, I:3	949	GTK	个人会计
kmymoney	V:0, I:3	12975	KDE	个人会计
shotwell	V:20, I:224	6096	GTK	数码照片管理器
xsane	V:19, I:190	935	GTK	扫描仪前段

Table 7.9: 基础的实用应用

小心
为了让 evince 和 okular 使用 Cmap 数据（第 7.6.3 节）来显示 CJK PDF 文档，必须要安装 poppler-data 软件包（之前为 non-free，参见第 11.4.1 节）。

注意
如果在 GNOME 桌面环境中没有相应功能的软件，那么安装例如 scribus（KDE）这样的软件包是完全可以接受的。但别安装过多功能重复的软件包，这会使你的菜单凌乱。

7.8 X 琐事

7.8.1 剪贴板

使用鼠标的 3 个按键来进行 X 选择是 X 原生的剪贴板（参见第 1.4.4 节）。

提示
Shift-Insert 等同于单击鼠标中键。

软件包	流行度	软件包大小	类型	说明
xsel	V:11, I:45	51	X	X 选择的命令行接口
xclip	V:9, I:40	64	X	X 选择的命令行接口

Table 7.10: 基础的 X 选择程序

现代的桌面环境（GNOME、KDE……）提供了不同的剪贴板系统用于剪切、复制和粘贴，它们使用鼠标左键和关键键（CTRL-X、CRTL-C 和 CTRL-V）。

7.8.2 X 中的键盘和鼠标按钮映射

xmodmap(1) 是 X 窗口系统中用来修改键盘和鼠标按钮映射的工具。为了获得 **keycode**，你需要在 X 中运行 xev(1) 并按下对应按键。为了获得 **keysym** 的含义，你需要查看“`/usr/include/X11/keysymdef.h`”文件（`x11proto-core-dev` 软件包）中 MACRO 的定义。文件中所有的“`#define`”语句后面都是以“`XK_`”开头，后接 **keysym** 名称。

7.8.3 典型的 X 客户端

大多数传统的 X 客户端程序，例如 xterm(1)，可以用一组标准的命令行选项来启动，从而指定大小、字体和显示。

它们还使用 X 资源数据库来配置它们的外观。系统默认的 X 资源保存在“`/etc/X11/Xresources/*`”中，应用默认的 X 资源保存在“`/etc/X11/app-defaults/*`”中。使用这些设置作为起点。

“`~/.Xresources`”文件被用来保存用户资源设定。这个文件在登陆时会自动合并到默认的 X 资源。要更改这些设置并使其立即生效，使用下列命令将它们合并到数据库中。

```
$ xrdb -merge ~/.Xresources
```

参见 x(7) 和 xrdb(1).

7.8.4 X 终端模拟器——xterm

在 http://dickey.his.com/xterm/xterm.faq.html 可以了解到关于 xterm(1) 的一切信息。

7.8.5 以 root 运行 X 客户端

警告
不要通过在显示管理器（例如 `gdm3`）的提示符后输入 `root` 来以 root 身份启动 X 显示/会话管理器，因为这是不安全的，即使你打算进行管理员操作。以 root 运行整个 X 架构被认为是不安全的。你必须总是使用尽可能低权限的账号，例如普通的用户账号。

运行一个特殊的 X 客户端（例如使“`foo`”取得 root 权限）的简单方法是像下面那样使用 sudo(8) 等。

```
$ sudo foo &
```

```
$ sudo -s
# foo &
```

```
$ ssh -X root@localhost
# foo &
```

小心
为了该目的像上面那样使用 ssh(1) 会浪费资源。

为了使 X 客户端链接到 X 服务器，请注意以下几点。

- 旧用户“$XAUTHORITY”和“$DISPLAY”环境变量的值必须复制给新用户。
- “$XAUTHORITY”环境变量的值所指向的文件必须对新用户可读。

Chapter 8

国际化和本地化

一个应用软件的 多语言化 (M17N) 或本地语言支持，通过 2 个步骤完成。

- 国际化 (I18N): 使一个软件能够处理多个语言环境。
- 本地化 (L10N)：使一个软件处理一个特定的语言环境。

提示
在 multilingualization (多语言化)、internationalization (国际化) 和 localization (本地化) 中，有 17, 18, 或 10 个字母在”m” 和”n”, ”i” 和”n”, 或”l” 和”n” 中间，它们相应表示为 M17N, I18N 和 L10N.

GNOME 和 KDE 等现代软件是多语言的。他们通过处理 UTF-8 数据来实现国际化，并通过 gettext(1) 架构提供翻译信息来本地化。翻译信息可以由独立的本地化软件包来提供。翻译信息易于选择使用，通过给相关的环境变量设置适当的语言环境即可。

最简单的文本数据表示方法是 **ASCII**，使用少于 127 个字符 (用 7 位表示)，这对英语足够了。为了支持用于国际化的更多字符，人们发明了许多字符编码系统。现代知名的编码系统是 **UTF-8**，它可以处理人类所知的几乎所有字符 (参见第 8.4.1 节).

细节请参见 i18n 介绍.

有本地化硬件配置数据便能支持国际化硬件。

警告
本章是基于 2013 年发布的 Debian 7.0 (wheezy) 编写的，所以其内容正在变得过时。

8.1 键盘输入

Debian 系统可以使用 `keyboard-configuration` 和 `console-setup` 软件包配置多个国际化键盘布局。

```
# dpkg-reconfigure keyboard-configuration
# dpkg-reconfigure console-setup
```

这将配置 Linux 控制台和 X 窗口的键盘,并更新“`/etc/default/keyboard`”和“`/etc/default/console-setup`”中的配置参数。这也可以用来配置 Linux 控制台的字体。

许多非 ASCII 字符，包括许多欧洲语言使用的重音字符，可以使用 死键、AltGr 键 和 组合键 来输入它们。

对于亚洲语言，你需要更复杂的输入法支持，例如下面将要讨论的 IBus。

8.1.1 IBus 支持的输入法

输入多种语言到应用程序的处理流程如下：

```
键盘                                    应用程序
   |                                        ^
   |                                        |
   +-> Linux 内核 -> 输入法 -> Gtk、Qt、或 X
```

通过 `im-config` 使用 IBus 家族的软件包可以简单地为 Debian 系统建立多语种的输入。下面列出了一些 IBus 软件包。

软件包	流行度	大小	支持的语言环境
ibus	V:8, I:12	45417	使用 dbus 的输入方式框架
ibus-mozc	V:1, I:2	897	日文
ibus-anthy	V:0, I:1	8500	同上
ibus-kkc	V:0, I:0	214	同上
ibus-skk	V:0, I:0	244	同上
ibus-pinyin	V:0, I:2	1425	中文 (zh_CN)
ibus-chewing	V:0, I:0	415	中文 (zh_TW)
ibus-hangul	V:0, I:1	218	韩文
ibus-table	V:0, I:1	979	IBus 表引擎
ibus-table-thai	I:0	46	泰文
ibus-unikey	V:0, I:0	256	越南语
ibus-m17n	V:0, I:0	154	多语言：印度语、阿拉伯语和其它

Table 8.1: IBus 支持的输入法

kinput2 方式和其它本地独立的亚洲经典输入法依旧存在，但不推荐在现代的 UTF-8 X 环境中使用。SCIM 和 uim 工具链是用于现代的 UTF-8 X 环境下的国际化输入法的较旧的方法。

8.1.2 一个日语的例子

我发现在英语环境 (”`en_US.UTF-8`”) 下启动日文输入法非常有用. 下面是在 GNOME3 下使用 IBus 的做法:

1. 安装日文输入法软件包 `ibus-anthy`，以及 `im-config` 等推荐的软件包.
2. 从用户 Shell 中执行”`im-config`”，然后选择”`ibus`” 作为输入法.
3. 选择”Settings” → ”Keyboard” → ”Input Sources” → 在”Input Sources” 中单击”+” → ”Japanese” → ”Japanese (anthy)”，然后单击”Add”.
4. 选择” 日语” 并” 添加” 到支持日语键盘，就不需要字符转换。(你可能会选择更多的输入源)
5. 重新登录用户账户。
6. 使用”`im-config`” 验证设置.
7. 右键单击 GUI 工具条图标，设置输入源。
8. 使用 SUPER-SPACE 在安装的输入法之间进行切换. (SUPER 键通常是 Windows 键.)

请注意以下几点。

- im-config(8) 如果命令是从 root 账户执行的表现会有所不同。
- im-config(8) 让最佳的输入法作为系统默认而不需要用户干预。
- 用户界面菜单入口 im-config(8) 默认会被禁用，以免造成混乱。

8.1.3 禁用输入法

如果你不想通过 XIM（X 所使用的机制）来进行输入，你可以在启动程序时将“$XMODIFIERS”的值设置为“none”。这可能会是这种情况，你想在 emacs(1) 中使用日文输入基础设施 egg 同时禁用 ibus。你可以从 shell 中执行如下命令。

```
$ XMODIFIERS=none emacs
```

为了调整 Debian 菜单执行的命令，请根据“/usr/share/doc/menu/html”中描述的方法定制“/etc/menu/”中的配置。

8.2 显示输出

Linux 控制台只能显示有限的字符。（你需要使用特殊的终端程序，例如 jfbterm(1)，从而在非 X 控制台中显示非欧洲语言。）

只要需要的字库数据存在 X 窗口可以通过 UTF-8 编码显示任意字符。（X 窗口系统能够维护好原始字体数据编码，这对用户来说是透明的）

8.3 东亚环境下宽度有歧义的字符

在东亚语言环境下，方框绘制、希腊字符和西里尔字符可能会显示得比你预期的样子更宽，这样会导致终端输出排列不再整齐（参见 Unicode 标准附录 #11）。

您可以绕过这个问题：

- gnome-terminal：编辑 → 首选项 → 配置文件 → 编辑 → 兼容性 → 宽度有歧义的字符 → 窄
- ncurses：设置环境变量 export NCURSES_NO_UTF8_ACS=0。

8.4 语言环境

下面重点介绍在从 gdm3(1) 启动的 X 窗口环境下运行的应用程序的语言设置。

8.4.1 编码的基础知识

环境变量“LANG=xx_YY.ZZZZ”将语言环境设置为语言代码“xx”、国家代码“YY”和编码“ZZZZ”（参见第 1.5.2 节）。

现在的 Debian 系统一般将语言环境设置为“LANG=xx_YY.UTF-8”。这将会使用带有 Unicode 字符集的 UTF-8 编码。UTF-8 编码系统是多字节的代码系统并且码点的使用更加智能。ASCII 数据（只包含了 7 位二进制代码）总是合法的 UTF-8 数据（每个字符使用 1 个字节）。

之前的 Debian 系统曾经将语言环境设置为“LANG=C”或“LANG=xx_YY”（没有“.UTF-8”）。

- “LANG=C”或“LANG=POSIX”使用 ASCII 字符集。
- “LANG=xx_YY”使用 Unix 的传统编码系统。

“LANG=xx_YY”所使用的确切传统编码系统可以通过“/usr/share/i18n/SUPPORTED”来确认。例如，“en_US”使用“ISO-8859-1”编码，“fr_FR@euro”使用“ISO-8859-15”编码。

提示
编码值的含义，参见表 11.2。

8.4.2 UTF-8 语言环境的基本原理

Unicode 字符集可以用从 0 到 10FFFF（十六进制）范围的码点来显示几乎所有人类已知的字符。它的存储至少需要 21 位。

文本编码系统 UTF-8 将 Unicode 码点适配到一个合理的 8 位数据流，并兼容 ASCII 数据处理系统。**UTF** 表示 Unicode 转换格式（Unicode Transformation Format）。

我建议在你的桌面使用 UTF-8 语言环境，例如“`LANG=zh_CN.UTF-8`”。语言环境的第一部分决定了应用程序中显示的信息。例如，“`LANG=fr_FR.UTF-8`”语言环境下的 gedit(1)（GNOME 桌面的文本编辑器），菜单是用法语显示的，但只要安装所需的字体和输入法就可以显示和编辑中文字符文本数据。

我还建议只使用“`$LANG`”环境变量来设置语言环境。我没有看到在 UTF-8 语言环境下设置复杂的“`LC_*`”变量组合有什么好处（参见 locale(1)）。

即使纯英文文本也可能包含非 ASCII 字符，例如微微卷曲的左右引号在 ASCII 中是不可用的。

```
“双引号的文本” 并非 “双引号的 ASCII”
‘单引号的文本’ 并非 ‘单引号的 ASCII’
```

当纯 ASCII 文本数据转换为 UTF-8 后，它会具有与原本完全相同的内容和大小。因此使用 UTF-8 语言环境并不会使你损失什么。

一些程序在支持 I8N 后会消耗更多的内存。这是因为它们为了速度优化，而在内部使用 UTF-32(UCS4) 来支持 Unicode，并且每个独立于语言环境所选的 ASCII 字符数据都会消耗 4 个字节。再一次地，使用 UTF-8 语言环境并不会使你损失什么。

供应商指定的旧的非 UTF-8 编码系统在一些字符上往往有较小但恼人的不同，例如许多国家使用的字形。而使用了 UTF-8 系统的现代操作系统基本上能解决这行编码冲突问题。

8.4.3 语言环境的重新配置

为了使系统访问特定的语言环境，必须从语言环境数据库编译相应语言环境数据。（Debian 系统不带有所有提前编译的可用语言环境，除非你安装 `locales-all` 软件包。）所支持的可编译语言环境的完整列表位于“`/usr/share/i18n/SUPPORTED`”，它列出了所有准确的语言环境名称。下列命令列出已编译成二进制形式的所有可用的 UTF-8 语言环境。

```
$ locale -a | grep utf8
```

下列的命令会重新配置 `locales` 软件包。

```
# dpkg-reconfigure locales
```

该过程包含 3 个步骤。

1. 更新可用的语言环境列表
2. 将它们编译为二进制形式
3. 在“`/etc/default/locale`”设置系统默认的语言环境值给 PAM 使用（参见第 4.5 节）

可用的语言环境列表应该包含“`en_US.UTF-8`”和所有你感兴趣的带有“`UTF-8`”的语言。

对于美式英语，推荐默认的语言环境为“`en_US.UTF-8`”。对于其它语言，请确保所选的语言环境带有“`UTF-8`”。这些设置中的任何一个都能够处理任何国际字符。

注意
虽然将语言环境设置为“`C`”会使用美式英语，但它只处理 ASCII 字符。

8.4.4 “$LANG”环境变量的值

“$LANG”环境变量的值由许多应用程序设置和改变。

- login(1) 的 PAM 机制为本地 Linux 控制台程序进行了最初的设置
- 显示管理器的 PAM 机制为所有的 X 程序进行了最初的设置
- ssh(1) 的 PAM 机制为远程控制台程序进行了最初的设置
- 一些显示管理器，例如 gdm3(1) 会为所有 X 程序改变设置
- 通过“~/.xsessionrc”，X 会话启动码会为所有 X 程序改变设置
- shell 启动码，例如“~/.bashrc”，会为所有控制台程序改变设置

提示
将系统默认语言环境设置为“en_US.UTF-8”能够获得最大的兼容性。

8.4.5 只用于 X 窗口的特定语言环境

你可以像下面那样选择只用于 X 窗口的特定语言环境，而不管你的系统使用 PAM 定制（参见第 4.5 节）的默认语言环境。

这个环境能够给你提供最好的桌面体验，并保持稳定。即使 X 窗口系统不工作，你也可以访问带有可读信息的多功能字符终端。这对于使用非罗马字符（如中文，日语和韩语）的语言来说是必不可少的。

注意
改善 X 会话管理软件包可能会使另一种可用的方法，但请阅读下面的内容作为设置语言环境的通用和基础的方法。对于 gdm3(1)，我们知道你能够通过它的菜单来选择 X 会话的语言环境。

在 PAM 配置文件中的下面这行定义了语言环境的文件位置，例如“/etc/pam.d/gdm3”。

```
auth    required        pam_env.so read_env=1 envfile=/etc/default/locale
```

将这行改成下面那样。

```
auth    required        pam_env.so read_env=1 envfile=/etc/default/locale-x
```

对于中文，建立一个带有“-rw-r--r-- 1 root root”权限的“/etc/default/locale-x”文件，并包含下面这行。

```
LANG="zh_CN.UTF-8"
```

保持用于其它程序的默认“/etc/default/locale”文件有下面这行。

```
LANG="en_US.UTF-8"
```

这是定制语言环境最通用的技术，并且会使 gdm3(1) 本身的菜单选择对话框被本地化。

对于该情况的另一种解决方法是使用“~/.xsessionrc”文件来改变语言环境。

8.4.6 文件名编码

对于跨平台的数据交换 (参见第 10.1.7 节), 你需要使用特殊的编码挂载文件系统. 举个例子，不使用选项时，mount(8) 假设 vfat 文件系统 使用 CP437. 你需要给文件名提供明确的挂载选项来使用UTF-8 或 CP932.

注意
在 GNOME 这类的现代桌面环境下，当自动挂载一个热拔插 U 盘时，你可以提供这样的挂载选项。右击桌面上的图标，点击”Drive” , ”Setting”, 输入”utf8” 到”Mount options:”. 当这个 U 盘下次挂载时，UTF-8 就可以了。

注意
如果你在升级一个系统，或者从老的非 UTF-8 系统迁移磁盘，非 ASCII 字符的文件名也许是使用老旧的 ISO-8859-1 或 eucJP 来编码. 请寻求文本转换工具把他们转换到 UTF-8. 参见第 11.1 节.

在默认情况下，Samba 对新的客户端 (Windows NT, 200x, XP) 使用 Unicode，但对老的客户端 (DOS 和 Windows 9x/Me) 使用 CP850. 可以在”`/etc/samba/smb.conf`” 文件里面，使用”`dos charset`” 来改变老客户端的这个默认编码。比如说，CP932 表示为日语。

8.4.7 本地化信息和翻译文档

在 Debian 系统中显示的许多文档和文本信息有翻译存在，比如错误信息、标准程序输出、菜单和手册页。GNU gettext(1) 命令工具链是大部分翻译活动的后端工具。

aptitude(8) 里，”Tasks” → ”Localization” 提供一个有用的二进制包扩展列表，给应用程序增加本地化信息和提供翻译文档。

举个例子，你可以安装 `manpages-<LANG>` 包来获得本地化 man 手册页信息。从”`/usr/share/man/it/`” 来读取 <programname> 意大利语的 man 手册页, 执行下面的操作。

```
LANG=it_IT.UTF-8 man <programname>
```

8.4.8 语言环境的影响

sort(1) 的字符排序，受语言环境的影响. 西班牙语和英语语言环境排序是不一样的。

ls(1) 的日期格式受语言环境影响. ”`LANG=C ls -l`” 和”`LANG=en_US.UTF-8`” 的日期格式是不一样的 (参见第 9.2.5 节).

不同语言环境的数字标点不一样. 比如说，英语语言环境中, 一千点一显示为”`1,000.1`”，而在德语语言环境中, 它显示为”`1.000,1`”. 你可以在电子表格程序里面看到这个不同。

Chapter 9

系统技巧

这里，描述配置和管理系统的基本技巧，大部分在控制台操作。

9.1 screen 程序

对通过不可靠或断断续续的连接访问远程主机的人们而言，screen(1) 是一个非常有用的工具，因为它支持可中断的网络连接。

软件包	流行度	大小	说明
screen	V:195, I:283	995	VT100/ANSI 终端模拟器混合复用的终端
tmux	V:31, I:118	681	终端复用的备选方案（使用“Control-B”代替）

Table 9.1: 支持可中断网络连接的程序列表

9.1.1 screen(1) 的使用场景

screen(1) 不但允许一个终端窗口运行多个进程，还允许远程 **shell** 进程支持中断的连接. 这里是一个典型的 screen(1) 使用场景.

1. 登录到一个远程机器。
2. 在单个控制台上启动 `screen`。
3. 使用 `^A c` (”Control-A” 接着”c”) 在 `screen` 中创建的窗口执行多个程序.
4. 按 `^A n` (”Control-A” 接着”n”) 来在多个 `screen` 窗口间转换.
5. 突然，你需要离开你的终端，但你不想丢掉正在做的工作，而这些工作需要连接来保持。
6. 你可以通过任何方式分离 `screen` 会话。
   - 残忍地拔掉你的网络连接
   - 输入 `^A d` (”Control-A” 接着”d”) 并手工从远程连接退出登录
   - 输入 `^A DD` (”Control-A” 接着”DD”) 分离 `screen` 并退出登录
7. 你重新登录到同一个远处主机（即使从不同的终端）。
8. 使用”`screen -r`” 启动 `screen`.

9. screen 魔术般的重新附上先前所有的 screen 窗口和所有在活动运行的程序.

提示
对于拨号或者按包计费的网络连接，你可以通过 screen 节省连接费用，应为你可以在断开连接时让一个进程继续运行，当你稍后再次连接时重新附上它。

9.1.2 screen 命令的键绑定

在 screen 会话里, 除了命令按键外的所有键盘输入都会被发送到当前窗口。screen 所有命令按键是通过 ^A (”Control-A”) 加单个键 [加任何参数] 来输入. 这里有一些重要的命令按键需要记住。

键绑定功能	说明
^A ?	显示帮助屏幕（显示键绑定）
^A c	创建一个新的窗口并切换到该窗口
^A n	到下一个窗口
^A p	到前一个窗口
^A 0	到 0 号窗口
^A 1	到 1 号窗口
^A w	显示窗口列表
^A a	作为键盘输入发送 Ctrl-A 到当前窗口
^A h	把当前窗口的硬拷贝写到一个文件
^A H	开始/结束当前窗口到文件的记录
^A ^X	锁定终端 (密码保护)
^A d	从终端分离 screen 会话
^A DD	分离 screen 会话并退出登录

Table 9.2: screen 键绑定列表

细节参见 screen(1).

9.2 数据记录和展示

9.2.1 日志后台守护进程（daemon）

许多程序在”/var/log/” 目录下记录它们的活动.

- 系统日志后台守护进程（daemon）：rsyslogd(8)

参见第 3.2.5 节和第 3.2.4 节.

9.2.2 日志分析

这里是主要的日志分析软件 (”~Gsecurity::log-analyzer” 在 aptitude(8) 中).

注意
CRM114 提供语言架构来写模糊过滤器，使用了 TRE 正则表达式库 。它主要在垃圾邮件过滤器中使用，但也能够用于日志分析。

软件包	流行度	大小	说明
logwatch	V:17, I:19	2251	用 Perl 写的日志分析软件，有好的输出
fail2ban	V:103, I:114	1735	禁用造成多个认证错误的 IP
analog	V:4, I:123	3529	web 服务器日志分析
awstats	V:11, I:17	6799	强大和特性全面的 web 服务器日志分析
sarg	V:4, I:4	429	生成 squid 分析报告
pflogsumm	V:1, I:4	111	Postfix 日志条目概要
syslog-summary	V:0, I:3	30	总结 syslog 日志文件内容
fwlogwatch	V:0, I:0	474	防火墙日志分析软件
squidview	V:0, I:1	189	监控和分析 squid access.log 文件
swatch	V:0, I:0	101	有正则表达式、高亮和曲线的日志文件查看器
crm114	V:0, I:0	1119	Controllable Regex Mutilator 和垃圾邮件过滤 (CRM114)
icmpinfo	V:0, I:0	39	解释 ICMP 信息

Table 9.3: 系统日志分析软件列表

9.2.3 清晰的记录 shell 活动

简单地使用 script(1)（参见第 1.4.9 节）记录 shell 活动会产生一个有控制字符的文件。这些控制字符可以按下面的方式，使用 col(1) 去掉。

```
$ script
Script started, file is typescript
```

做些操作……按 Ctrl-D 退出 script.

```
$ col -bx <typescript >cleanedfile
$ vim cleanedfile
```

如果你没有 script (例如：在 initramfs 里的启动过程中)，你可以使用下面的方式代替。

```
$ sh -i 2>&1 | tee typescript
```

提示
像 gnome-terminal 之类的 x-terminal-emulator 也能够记录。你也许需要增加行缓冲来用滚动条查看。

提示
你可以使用 screen(1) 和"^A H" (参见第 9.1.2 节) 来进行控制台记录。

提示
你可以使用 emacs(1) 和"M-x shell", "M-x eshell", 或"M-x term" 来进行控制台记录。你稍后可以使用"C-x C-w" 将缓冲写到文件。

9.2.4 定制文本数据的显示

尽管例如 more(1) 和 less(1) 这样的分页程序（参见第 1.4.5 节）和用于高亮和格式的自定义工具（参见第 11.1.8 节）可以漂亮地显示文本数据，但通用的编辑器（参见第 1.4.6 节）是用途最广的，且可定制性最高。

提示
对于 vim(1) 和它的分页模式别名 view(1)，“:set hls”可以启用高亮搜索。

9.2.5 定制时间和日期的显示

“ls -l”命令默认的时间和日期显示格式取决于语言环境（相关的值参见第 1.2.6 节）。“$LANG”变量将被首先考虑，但它会被“$LC_TIME”变量覆盖。

每个语言环境实际的默认显示格式取决于所使用的 C 标准库的版本（libc6 软件包），也就是说，不同的 Debian 发行版有不同的默认情况。

如果你真的想自定义超出语言环境的时间和日期显示格式，你应该通过“--time-style”参数或“$TIME_STYLE”的值来设置时间样式值（参见 ls(1)、date(1)、“info coreutils 'ls invocation'”）。

时间样式值	语言环境	时间和日期显示
iso	任何值	01-19 00:15
long-iso	任何值	2009-01-19 00:15
full-iso	任何值	2009-01-19 00:15:16.000000000 +0900
语言环境	C	Jan 19 00:15
语言环境	en_US.UTF-8	Jan 19 00:15
语言环境	es_ES.UTF-8	ene 19 00:15
+%d.%m.%y %H:%M	任何值	19.01.09 00:15
+%d.%b.%y %H:%M	C 或 en_US.UTF-8	19.Jan.09 00:15
+%d.%b.%y %H:%M	es_ES.UTF-8	19.ene.09 00:15

Table 9.4: wheezy 中“ls -l”命令时间和日期的显示案例

提示
你可以使用命令别名以避免在命令行中输入长的选项，例如“alias ls='ls --time-style=+%d.%m.%y\n%H:%M'”（参见第 1.5.9 节）。

提示
ISO 8601 遵循这些 iso 格式。

9.2.6 shell 中 echo 的颜色

大部分现代终端的 shell 中 echo 能够使用 ANSI 转义字符来显示颜色（参见“/usr/share/doc/xterm/ctlseqs.txt.gz”

尝试下列例子

```
$ RED=$(printf "\x1b[31m")
$ NORMAL=$(printf "\x1b[0m")
$ REVERSE=$(printf "\x1b[7m")
$ echo "${RED}RED-TEXT${NORMAL} ${REVERSE}REVERSE-TEXT${NORMAL}"
```

9.2.7 有颜色输出的命令

在交互式的环境下，命令的输出带颜色，能够给检查命令的输出带来便利。我在我的”~/.bashrc” 里加入了下面内容.

```
if [ ”$TERM” != ”dumb” ]; then
    eval ”‘dircolors -b‘”
    alias ls=’ls --color=always’
    alias ll=’ls --color=always -l’
    alias la=’ls --color=always -A’
    alias less=’less -R’
    alias ls=’ls --color=always’
    alias grep=’grep --color=always’
    alias egrep=’egrep --color=always’
    alias fgrep=’fgrep --color=always’
    alias zgrep=’zgrep --color=always’
else
    alias ll=’ls -l’
    alias la=’ls -A’
fi
```

在交互式命令中，使用别名来限制颜色的影响范围。导出环境变量”export GREP_OPTIONS=’--color=auto’” 也有好处，这样能够让 less(1) 之类的页面程序看到颜色。当使用管道到其它命令时，你想去掉颜色，上面列子”~/.bashrc” 中的内容，可以使用”--color=auto” 代替.

提示
在交互式的环境中，通过”TERM=dumb bash” 调用 shell ，你能够关闭这些颜色别名。

9.2.8 记录编辑器复杂的重复操作动作

你能够记录编辑器复杂的重复操作动作。

对于 Vim, 请按下面操作。

- ”qa”: 开始记录输入字符到有名字的寄存器”a”.
- …编辑器操作
- ”q”: 结束记录输入的字符。
- ”@a”: 执行寄存器”a 的内容”.

对于 Emacs, 请按下面操作。

- ”C-x (”: 开始定义一个键盘宏.
- …编辑器操作
- ”C-x)”: 结束定义一个键盘宏.
- ”C-x e”: 执行一个键盘宏.

9.2.9 记录 X 应用程序的图形

有少量方法可以记录 X 应用程序的图像，包括 xterm 显示。

软件包	流行度	大小	命令
xbase-clients	I:38	46	xwd(1)
gimp	V:85, I:489	19016	GUI 菜单
imagemagick	V:43, I:549	209	import(1)
scrot	V:8, I:92	54	scrot(1)

Table 9.5: 图形图像处理工具列表

软件包	流行度	大小	说明
etckeeper	V:26, I:31	158	使用 Git（默认）、Mercurial 或 Bazaar（新）来保存配置文件和它们的元数据
changetrack	V:0, I:0	63	使用 RCS（旧）保存配置文件

Table 9.6: 在 VCS 中记录配置历史的软件包

9.2.10 记录配置文件的变更

有特定的工具可以通过 DVCS 系统的帮助来记录配置文件的变更。

我建议使用带有 git(1) 的 etckeeper 软件包，它将整个“/etc”置于 VCS 控制之下。它的安装指南和教程参见“/usr/share/doc/etckeeper/README.gz”。

从本质上讲，运行“sudo etckeeper init”来为“/etc”初始化 git 仓库，与第 10.6.5 节中所解释的过程相似，但需要特殊的 hook 脚本来进行更全面的设置。

当你改变你的配置时，你可以使用 git(1) 来正常地记录它们。你每次运行软件包管理命令时，它也会自动记录变更。

> 提示
> 你可以通过执行“sudo GIT_DIR=/etc/.git gitk”来浏览“/etc”的变更记录，你可以清晰地看到新的已安装软件包、已移除软件包和软件包版本的变更。

9.3 监控、控制和启动程序活动

程序活动能够使用特殊的工具监控和控制。

> 提示
> procps 包提供了非常基础的监控、控制程序活动功能和启动程序功能。你应当把他们全部学会。

9.3.1 进程耗时

显示命令调用进程的时间消耗。

```
# time some_command >/dev/null
real    0m0.035s        # 执行时间 (占用的真实时间)
user    0m0.000s       # 用户模式时间
sys     0m0.020s        # 内核模式时间
```

软件包	流行度	大小	说明
coreutils	V:888, I:999	15719	nice(1): 用指定的调度优先权运行一个程序
bsdutils	V:866, I:999	293	renice(1): 调整一个目前在运行的进程的调度优先权值
procps	V:793, I:999	729	"/proc" 文件系统工具: ps(1), top(1), kill(1), watch(1), ⋯
psmisc	V:473, I:895	637	"/proc" 文件系统工具: killall(1), fuser(1), peekfd(1), pstree(1)
time	V:22, I:428	82	time(1): 运行一个程序，并从时间消耗方面来报告系统资源的使用
sysstat	V:144, I:165	1684	sar(1), iostat(1), mpstat(1), ⋯: linux 系统性能工具
isag	V:0, I:4	111	sysstat 的交互式的系统活动图
lsof	V:464, I:946	454	lsof(8): 使用"-p" 选项列出被一个系统进程打开的文件
strace	V:19, I:159	2051	strace(1): 跟踪系统调用和信号
ltrace	V:1, I:21	363	ltrace(1): 跟踪库调用
xtrace	V:0, I:0	352	xtrace(1): 跟踪 X11 客户端和服务器端之间的通信
powertop	V:6, I:231	604	powertop(1): 系统能耗使用信息
cron	V:878, I:997	263	根据 cron(8) 后台守护进程（daemon）的调度运行一个进程
anacron	V:447, I:521	99	用于非整天 24 小时运行系统的命令计划，类 cron
at	V:260, I:453	157	at(1) 或 batch(1): 在一个特定的时间运行任务或在某一系统负载下运行

Table 9.7: 监控和控制程序活动工具列表

进程优先级值	调度优先级
19	最低优先级进程
0	非常高的普通用户优先级进程
-20	root 用户非常高的优先级进程

Table 9.8: 调度优先级值列表

9.3.2 调度优先级

进程的调度优先级是被一个进程优先级值控制。

```
# nice  -19 top                                        # 非常优先
# nice --20 wodim -v -eject speed=2 dev=0,0 disk.img # 非常快
```

在某些情况下，极端的进程优先级值会对系统造成伤害。小心使用这个命令。

9.3.3 ps 命令

在 Debian 系统上的 ps(1) 命令同时支持 BSD 和 SystemV 特征，有助于识别静态的进程活动。

样式	典型的命令	特征
BSD	ps aux	显示%CPU %MEM
System V	ps -efH	显示 PPID

Table 9.9: ps 命令样式列表

对于僵尸（死了的）子进程，你能够通过"PPID" 字段的父进程 ID 来杀死它们。

pstree(1) 命令显示进程树。

9.3.4 top 命令

Debian 系统上的 top(1) 拥有丰富的特征，有助于识别进程有趣的动态行为。

它是一个交互式的全屏程序。你可以通过按”h” 键来得到它的使用帮助，按”q” 键来终止该程序。

9.3.5 列出被一个进程打开的文件

你能够通过一个进程 ID(PID) 来列出该进程所有打开的文件，例如，PID 为 1 的进程，使用下面的方式。

```
$ sudo lsof -p 1
```

PID=1 通常用于 `init` 程序.

9.3.6 跟踪程序活动

你能够跟踪程序活动，使用 strace(1), ltrace(1), xtrace(1) 来跟踪系统调用和信号、库调用、X11 客户端和服务器端之间的通信。

跟踪 `ls` 命令的系统调用。

```
$ sudo strace ls
```

9.3.7 识别使用文件和套接字的进程

你可以通过 fuser(1) 来识别出使用文件的进程，例如，用下面的方式识别出”`/var/log/mail.log`” 由哪个进程打开。

```
$ sudo fuser -v /var/log/mail.log
                     USER        PID ACCESS COMMAND
/var/log/mail.log:   root       2946 F.... rsyslogd
```

你可以看到”`/var/log/mail.log`” 是由 rsyslogd(8) 命令打开并写入。

你可以通过 fuser(1) 来识别出使用套接字的进程，例如，用下面的方式识别出”`smtp/tcp`” 由哪个进程打开。

```
$ sudo fuser -v smtp/tcp
                     USER        PID ACCESS COMMAND
smtp/tcp:            Debian-exim   3379 F.... exim4
```

现在你知道你的系统运行 exim4(8) 来处理连接到 SMTP 端口 (25) 的 TCP 连接.

9.3.8 使用固定间隔重复一个命令

watch(1) 使用固定间隔重新执行一个命令，并全屏显示输出。

```
$ watch w
```

显示哪些人登录到系统，每 2 秒钟更新一次。

9.3.9 使用文件循环来重复一个命令

通过匹配某些条件的文件来循环重复一个命令，有几种方法，例如，匹配全局模式"`*.ext`".

- Shell 循环方式 (参见第 12.1.4 节):

```
for x in *.ext; do if [ -f "$x"]; then command "$x" ; fi; done
```

- find(1) 和 xargs(1) 联合:

```
find . -type f -maxdepth 1 -name '*.ext' -print0 | xargs -0 -n 1 command
```

- find(1) 使用"`-exec`" 选项并执行命令:

```
find . -type f -maxdepth 1 -name '*.ext' -exec command '{}' \;
```

- find(1) 使用"`-exec`" 选项并执行一个短的 shell 脚本:

```
find . -type f -maxdepth 1 -name '*.ext' -exec sh -c "command '{}' && echo 'successful'" \;
```

上面的列子确保适当处理怪异的文件名 (如包含空格)。find(1) 更多高级的用法，参见第 10.1.5 节.

9.3.10 从 GUI 启动一个程序

对于 命令行界面 (command-line interface，CLI)，`$PATH` 环境变量所指定的目录中第一个匹配相应名称的程序会被执行。参见第 1.5.3 节。

对于遵从 freedesktop.org 标准的 图形用户界面 (graphical user interface，GUI)，`/usr/share/applications/` 目录中的 `*.desktop` 文件给每个程序的 GUI 菜单显示提供了必要的属性。参见第 7.2.2 节。

举个例子，`chromium.desktop` 文件中为“Chromium 网络浏览器”定义了相关属性，例如程序名“Name”，程序执行路径和参数“Exec”，所使用的图标“Icon”等等 (参见 桌面配置项规范)。文件内容如下：

```
[Desktop Entry]
Version=1.0
Name=Chromium Web Browser
GenericName=Web Browser
Comment=Access the Internet
Comment[fr]=Explorer le Web
Exec=/usr/bin/chromium %U
Terminal=false
X-MultipleArgs=false
Type=Application
Icon=chromium
Categories=Network;WebBrowser;
MimeType=text/html;text/xml;application/xhtml_xml;x-scheme-handler/http;x-scheme-handler/ ↩
    https;
StartupWMClass=Chromium
StartupNotify=true
```

这是一个较为简单的说明。*.desktop 文件像下面那样被搜寻。

桌面环境设置 $XDG_DATA_HOME 和 $XDG_DATA_DIR 环境变量。举个例子，在 GNOME 3 中：

- 未设置 $XDG_DATA_HOME。(将使用默认值 $HOME/.local/share。)
- $XDG_DATA_DIRS 被设置为 /usr/share/gnome:/usr/local/share/:/usr/share/。

基准目录 (参见 XDG Base Directory Specification) 和应用程序目录如下所示。

- $HOME/.local/share/ → $HOME/.local/share/applications/
- /usr/share/gnome/ → /usr/share/gnome/applications/
- /usr/local/share/ → /usr/local/share/applications/
- /usr/share/ → /usr/share/applications/

*.desktop 文件将按照这个顺序在这些 applications 目录中进行搜寻。

提示
要建立一个用户自定义的 GUI 菜单项，需要在 $HOME/.local/share/applications/ 目录中添加一个 *.desktop 文件。

提示
相似地，如果在这些基准目录下的 autostart 目录中建立了一个 *.desktop 文件，则 *.desktop 文件中指定的程序会在桌面环境启动时自动执行。参见 Desktop Application Autostart Specification。

提示
相似地，如果在 $HOME/Desktop 目录中建立了一个 *.desktop 文件并且桌面环境被配置为支持桌面图标启动器功能，则点击图标时指定的程序会被执行。请注意，$HOME/Desktop 目录的实际名称与语言环境有关。参见 xdg-user-dirs-update(1)。

9.3.11 自定义被启动的程序

一些程序会被另一个程序自动启动。下面是自定义该过程的方法。

- 应用程序配置菜单：
  - GNOME3 桌面：“设置” → “系统” → “详细信息” → “默认应用程序”
  - KDE 桌面: ”K” → ”Control Center 控制中心” → ”KDE Components 组件” → ”Component Chooser 组件选择器”
  - Iceweasel 浏览器：“编辑” → “首选项” → “应用程序”
  - mc(1)：“/etc/mc/mc.ext”
- 例如“$BROWSER”、“$EDITOR”、“$VISUAL”和“$PAGER”这样的环境变量 (参见 eviron(7))
- 用于例如“editor”、“view”、“x-www-browser”、“gnome-www-browser”和“www-browser”这样的程序的 update-alternatives(1) 系统 (参见第 1.4.7 节)
- “~/.mailcap”和“/etc/mailcap”文件的内容关联了程序的 MIME 类型 (参见 mailcap(5))
- “~/.mime.types”和“/etc/mime.types”文件的内容关联了 MIME 类型的文件扩展名 (参见 run-mailcap(1))

提示
update-mime(8) 会更新”/etc/mailcap” 文件，期间会用到”/etc/mailcap.order” 文件 (参见 mailcap.order(5)).

提示
debianutils 软件包提供 sensible-browser(1)、sensible-editor(1) 和 sensible-pager(1)，它们可以分别对要调用的编辑器、分页程序和网络浏览器作出明智的选择。我建议你阅读那些 shell 脚本。

提示
为了在 X 下运行例如 mutt 这样的控制台应用程序来作为你的首选应用程序，你应该像下面那样建立一个 X 应用程序并设置“/usr/local/bin/mutt-term”为你想要启动的首选应用程序。

```
# cat /usr/local/bin/mutt-term <<EOF
#!/bin/sh
gnome-terminal -e ”mutt \$@”
EOF
chmod 755 /usr/local/bin/mutt-term
```

9.3.12 杀死一个进程

使用 kill(1) 通过进程 ID 来杀死（发送一个信号）一个进程。

使用 killall(1) 或 pkill(1) 通过进程命令的名字或其它属性来做同样的事情。

信号值	信号名	功能
1	HUP	重启后台守护进程（daemon）
15	TERM	普通 kill
9	KILL	硬 kill

Table 9.10: kill 命令常用信号列表

9.3.13 单次任务时间安排

运行 at(1) 命令来安排一次性的工作。

```
$ echo ’command -args’| at 3:40 monday
```

9.3.14 定时任务安排

使用 cron(8) 来进行定时任务安排。参见 crontab(1) 和 crontab(5).

你能够作为一个普通用户定时运行一个进程，比如，foo 使用”crontab -e” 命令创建一个 crontab(5) 的文件”/var/spool/c

这里是一个 crontab(5) 文件的列子。

```
# use /bin/sh to run commands, no matter what /etc/passwd says
SHELL=/bin/sh
# mail any output to paul, no matter whose crontab this is
MAILTO=paul
# Min Hour DayOfMonth Month DayOfWeek command (Day... are OR'ed)
# run at 00:05, every day
5  0  *  * *   $HOME/bin/daily.job >> $HOME/tmp/out 2>&1
# run at 14:15 on the first of every month -- output mailed to paul
15 14 1  * *   $HOME/bin/monthly
# run at 22:00 on weekdays(1-5), annoy Joe. % for newline, last % for cc:
0 22 *   * 1-5 mail -s "It's 10pm" joe%Joe,%%Where are your kids?%.%%
23 */2 1 2 *   echo "run 23 minutes after 0am, 2am, 4am ..., on Feb 1"
5  4 *   * sun echo "run at 04:05 every Sunday"
# run at 03:40 on the first Monday of each month
40 3 1-7 * *   [ "$(date +%a)" == "Mon" ] && command -args
```

提示
对那些非连续运行的系统，安装 anacron 软件包来定时执行周期性的命令，命令在接近机器启动的时间运行，并允许有特定的时间间隔。参见 anacron(8) 和 anacrontab(5).

提示
对于定时系统维护脚本，你能够以 root 账户定时运行，把这类脚本放入"/etc/cron.hourly/", "/etc/cron.daily/", "/etc/cron.weekly/", 或"/etc/cron.monthly/". 这些脚本的执行时间，可以通过"/etc/crontab" 和"/etc/anacrontab" 来定制。

9.3.15 Alt-SysRq 键

内核编译选项"Magic SysRq key" (SAK 键) 提供预防系统故障的措施，该选项现在是 Debian 内核的默认值。按 Alt-SysRq 键，接着按下面键中的一个键，会做拯救系统的神奇事情。

Alt-SysRq 之后的键	行为描述
r	在 X 崩溃后，从 **r**aw 模式恢复键盘
0	把控制台日志级别改变到 **0** 来减少错误信息
k	**k**ill 在当前虚拟控制台上的所有进程
e	发送 SIGT**E**RM 到所有进程，除开 init(8)
i	发送 SIGK**I**LL 到所有进程，除开 init(8)
s	**s**ync 所有已经挂载的文件系统来避免数据损坏
u	重新以只读方式挂载所有已挂载的文件系统 (**u**mount)
b	re**b**oot 系统，不同步或卸载

Table 9.11: SAK 命令键列表

提示
阅读 signal(7), kill(1) 和 sync(1) 手册页来理解上面的描述。

"Alt-SysRq s", "Alt-SysRq u" 和"Alt-SysRq r" 组合，有助于跳出真正坏的情形，并可以在不停止系统的情况下获得可用的键盘。

参见"/usr/share/doc/linux-doc-3.*/Documentation/sysrq.txt.gz".

小心
由于允许用户访问 root 权限的功能，Alt-SysRq 特性可能被认为是安全风险。在”/etc/rc.local” 里面放入”echo 0 >/proc/sys/kernel/sysrq” 或在”/etc/sysctl.conf” 里放入”kernel.sysrq = 0” 来禁用 Alt-SysRq 特性。

提示
从 SSH 终端等，你能够通过向”/proc/sysrq-trigger” 写入内容来使用 Alt-SysRq 特性。例如，从 root shell 提示符运行”echo s > /proc/sysrq-trigger; echo u > /proc/sysrq-trigger” 来 **s**yncs 和 **u**mounts 所有已挂载的文件系统。

9.4 系统维护技巧

9.4.1 谁在系统里？

你可以通过下面的方法检查谁登录在系统里。

- who(1) 显示谁登录在系统里面。
- w(1) 显示谁登录在系统里面，他们在做什么。
- last(1) 显示用户最后登录的列表。
- lastb(1) 显示用户最后错误登录的列表。

提示
”/var/run/utmp” 和”/var/log/wtmp” 存储这样的用户信息。参见 login(1) 和 utmp(5).

9.4.2 警告所有人

你可以通过下面的方式使用 wall(1) 给登录系统的每一个人发送信息。

```
$ echo ”We are shutting down in 1 hour” | wall
```

9.4.3 硬件识别

对于 PCI 类设备 (AGP, PCI-Express, CardBus, ExpressCard 等), 一开始就使用 lspci(8) (也许加上”-nn” 选项) 进行硬件识别比较好。

此外，你可以通过阅读”/proc/bus/pci/devices” 里面的内容或浏览”/sys/bus/pci” 下面的目录树来进行硬件识别 (参见第 1.2.12 节).

9.4.4 硬件配置

像 GNOME 和 KDE 这类现代图形桌面系统，虽然大部分硬件的配置都能够通过相应的图形配置工具来管理，但知道一些配置它们的基础方式，也是一个好的主意。

这里, ACPI 是一个比 APM 新的电源管理系统框架。

提示
现代系统的 CPU 频率调整功能，是由内核模块 acpi_cpufreq 管理的。

软件包	流行度	大小	说明
pciutils	V:158, I:994	1220	Linux PCI 工具: lspci(8)
usbutils	V:94, I:866	312	Linux USB 工具: lsusb(8)
pcmciautils	V:18, I:30	105	Linux PCMCIA 工具: pccardctl(8)
scsitools	V:0, I:3	365	SCSI 硬件管理工具集: lsscsi(8)
procinfo	V:0, I:15	123	从”/proc”: lsdev(8) 获得系统信息
lshw	V:10, I:89	763	硬件配置信息: lshw(1)
discover	V:39, I:945	90	硬件识别系统: discover(8)

Table 9.12: 硬件识别工具列表

软件包	流行度	大小	说明
console-setup	V:280, I:951	417	Linux 控制台字体和键盘表工具
x11-xserver-utils	V:354, I:575	511	X 服务端工具: xset(1), xmodmap(1)
acpid	V:262, I:484	146	管理高级可配置和电源接口 (ACPI) 事件分发的后台守护进程 (daemon)
acpi	V:23, I:463	45	显示 ACPI 设备信息的工具
sleepd	V:0, I:0	86	在笔记本空闲时，使其进入休眠状态的后台守护进程 (daemon)
hdparm	V:435, I:653	255	硬盘访问优化 (参见第 9.5.9 节)
smartmontools	V:127, I:191	1846	使用 S.M.A.R.T. 控制和监控存储系统
setserial	V:5, I:10	117	串口管理工具集
memtest86+	V:1, I:33	2391	内存硬件管理工具集
scsitools	V:0, I:3	365	SCSI 硬件管理工具集
setcd	V:0, I:1	35	光驱访问优化
big-cursor	I:1	27	X 系统的大鼠标光标

Table 9.13: 硬件配置工具列表

9.4.5 系统时间和硬件时间

下面设置系统的硬件时间为：MM/DD hh:mm, CCYY.

```
# date MMDDhhmmCCYY
# hwclock --utc --systohc
# hwclock --show
```

Debian 系统的时间通常显示为本地时间，但硬件时间通常使用 UTC(GMT) 时间。

如果硬件（BIOS）时间设置为 UTC 时间，请在“/etc/default/rcS”里面设置“UTC=yes”。

下面是重新配置 Debian 系统使用的时区。

```
# dpkg-reconfigure tzdata
```

如果你希望通过网络来更新系统时间，考虑使用 ntp, ntpdate 和 chrony 这类包提供的 NTP 服务。

提示
在 systemd 下，是使用 systemd-timesyncd 来替代进行网络时间同步。参见 systemd-timesyncd(8).

参见下面内容。

- 精确时间和日期管理 HOWTO
- NTP 公共服务项目
- ntp-doc 包

提示
在 ntp 包里面的 ntptrace(8) 能够跟踪 NTP 服务链至原始源。

9.4.6 终端配置

有几个组件可以用来配置字符控制台和 ncurses(3) 系统功能。

- “/etc/terminfo/*/*”文件（terminfo(5)）
- “$TERM”环境变量（term(7)）
- setterm(1)、stty(1)、tic(1) 和 toe(1)

如果 xterm 的 terminfo 对非 Debian 的 xterm 不起作用，则当你从远程登陆到 Debian 系统时，你需要改变你的终端类型“$TERM”，从“xterm”更改为功能受限的版本（例如“xterm-r6”）。更多内容参见“/usr/share/doc/libncurses5/F
是“$TERM”中最通用的。

9.4.7 声音基础设施

用于现在的 Linux 的声卡设备驱动程序由 高级 Linux 声音体系（Advanced Linux Sound Architecture，ALSA）提供。ALSA 提供了兼容之前的 开放声音系统（Open Sound System，OSS）的模拟模式。

提示
使用“`cat /dev/urandom > /dev/audio`”或 speaker-test(1) 来测试扬声器（^C 停止）。

提示
如果你无法听到声音，那你的扬声器可能连接到了一个静音输出。现代的声音系统有许多输出。`alsa-utils` 软件包中的 alsamixer(1) 可以很好地配置声音和静音设置。

应用软件可被配置为不仅直接访问声音设备，也可以通过一些标准化声音服务器系统来访问它们。

软件包	流行度	大小	说明
`alsa-utils`	V:393, I:519	2253	配置和使用 ALSA 的工具
`oss-compat`	V:2, I:35	20	在 ALSA 下兼容 OSS，预防“`/dev/dsp not found`”错误
`jackd`	V:3, I:27	9	JACK Audio Connection Kit. (JACK) 服务器（低延迟）
`libjack0`	V:0, I:14	337	JACK Audio Connection Kit. (JACK) 库（低延迟）
`nas`	V:0, I:0	239	网络音频系统（Network Audio System，NAS）服务器
`libaudio2`	I:541	161	网络音频系统（Network Audio System，NAS）库
`pulseaudio`	V:387, I:504	6411	PulseAudio 服务器，替代 ESD
`libpulse0`	V:278, I:646	968	PulseAudio 客户端库，替代 ESD
`libgstreamer1.0-0`	V:256, I:549	5063	GStreamer：GNOME 声音引擎
`libphonon4`	I:192	680	Phonon：KDE 声音引擎

Table 9.14: 声音软件包

每个流行的桌面环境通常都有一个通用的声音引擎。每个被应用程序使用的声音引擎都可以选择连接到不同的声音服务器。

9.4.8 关闭屏幕保护

关闭屏幕保护，使用下面的命令。

环境	命令
Linux 控制台	`setterm -powersave off`
X 窗口（关闭屏幕保护）	`xset s off`
X 窗口（关闭 dpms）	`xset -dpms`
X 窗口（屏幕保护 GUI 配置）	`xscreensaver-command -prefs`

Table 9.15: 关闭屏幕保护命令列表

9.4.9 关闭蜂鸣声

可以把电脑的扬声器拔掉来关闭蜂鸣声。把 `pcspkr` 内核模块删除，也可以做到这点。

bash(1) 用到的 readline(3) 程序，当遇到告警字符 (ASCII=7) 时，将会发生。下面的操作将阻止发生。

```
$ echo "set bell-style none">> ~/.inputrc
```

9.4.10 内存使用

对你来说，这里有两种可用的方法来得到内存的使用情况。

- “/var/log/dmesg”中的内核启动信息包含了可用内存的精确总大小。
- free(1) 和 top(1) 显示正在运行的系统中内存资源的相关信息。

下面是一个例子。

```
# grep '\] Memory' /var/log/dmesg
[    0.004000] Memory: 990528k/1016784k available (1975k kernel code, 25868k reserved, 931k ↩
    data, 296k init)
$ free -k
             total       used       free     shared    buffers     cached
Mem:        997184     976928      20256          0     129592     171932
-/+ buffers/cache:     675404     321780
Swap:      4545576          4    4545572
```

你可能会觉得奇怪：“dmesg 告诉你 free 为 990 MB，而 free -k 告诉你 free 为 320 MB。这丢失了超过 600 MB ……”。

别担心“Mem:”这行中“used”较大的值以及“free”较小的值，放轻松，你需要查看的是下面的那个（在上面的例子中它们是 675404 和 321780 ）。

对于我的 MacBook，有 1GB=1048576k 内存（显卡系统用掉一些），我看到的如下。

报告	大小
dmesg 中 total 的大小	1016784k = 1GB - 31792k
dmesg 中的 free	990528k
shell 下的 total	997184k
shell 下的 free	20256k（但有效的为 321780k ）

Table 9.16: 报告的内存大小

9.4.11 系统安全性和完整性检查

糟糕的系统维护可能会暴露你的系统，导致它被外部非法使用。

对于系统安全性和完整性的检查，你需要从下面这些方面开始。

- debsums 软件包，参见 debsums(1) 和第 2.5.2 节。
- chkrootkit 软件包，参见 chkrootkit(1)。
- clamav 软件包家族，参见 clamscan(1) 和 freshclam(1)。
- Debian security FAQ。
- Securing Debian Manual。

下面是一个简单的脚本，用来检测典型的所有人可写的错误文件权限。

```
# find / -perm 777 -a \! -type s -a \! -type l -a \! \( -type d -a -perm 1777 \)
```

小心
由于 debsums 软件包使用本地存储的 MD5 校验码，因此面对恶意攻击，也不能完全相信系统安全性检测工具。

软件包	流行度	大小	说明
logcheck	V:10, I:12	102	后台守护进程（daemon），将系统日志文件中的异常通过邮件发送给管理员
debsums	V:6, I:40	120	实用程序，使用 MD5 校验码对已安装软件包的文件进行校验
chkrootkit	V:6, I:26	934	rootkit 检测软件
clamav	V:14, I:65	727	Unix 的反病毒实用程序——命令行界面
tiger	V:3, I:3	2485	报告系统安全漏洞
tripwire	V:2, I:3	12055	文件和目录完整性检测软件
john	V:2, I:13	449	先进的密码破解工具
aide	V:2, I:2	2063	高级入侵环境检测——静态二进制
integrit	V:0, I:0	329	文件完整性验证程序
crack	V:0, I:1	128	密码猜测程序

Table 9.17: 用于系统安全性和完整性检查的工具

9.5 数据存储技巧

使用 live CD 或 debian-installer CD 以救援模式启动你的系统，可以让你简单地重新配置你的启动设备的数据存储。

9.5.1 硬盘空间使用情况

硬盘空间的使用情况可以通过 `mount`、`coreutils` 和 `xdu` 软件包提供的程序来评估：

- mount(8) 显示所有挂载的文件系统 (= 磁盘).
- df(1) 报告文件系统使用的硬盘空间。
- du(1) 报告目录树使用的硬盘空间。

提示
你可以将 du(8) 的输出传输给 xdu(1x)，来使用它的图形交互式演示，例如“`du -k . |xdu`”、“`sudo du -k -x / |xdu`”等等。

9.5.2 硬盘分区配置

对于硬盘分区配置，尽管 fdisk(8) 被认为是标准的配置，但是 parted(8) 工具还是值得注意的。

大多数 PC 使用经典的主引导记录（Master Boot Record，MBR）方案，将硬盘分区数据保存在第一个扇区，即 LBA 扇区 0（512 字节）。

注意
一些带有可扩展固件接口（Extensible Firmware Interface，EFI）的新 PC，包括基于 Intel 的 Mac，使用 全局唯一标识分区表（GUID Partition Table，GPT）方案，硬盘分区数据不保存在第一个扇区。

尽管 fdisk(8) 一直是硬盘分区的标准工具，但现在 parted(8) 替代了它。

小心
尽管 parted(8) 声称也能用来创建和调整文件系统，但使用维护最好的专门工具来做这些事会更为安全，例如 mkfs(8)（mkfs.msdos(8)、mkfs.ext2(8)、mkfs.ext3(8)、mkfs.ext4(8)……）和 resize2fs(8)。

软件包	流行度	大小	GPT	说明
util-linux	V:891, I:999	4327	不支持	多种系统工具，包含 fdisk(8) 和 cfdisk(8)
parted	V:391, I:579	286	支持	GNU Parted，硬盘分区调整程序
gparted	V:22, I:144	7537	支持	基于 libparted 的 GNOME 分区编辑程序
gdisk	V:71, I:515	811	支持	用于 GPT 硬盘的分区编辑程序
kpartx	V:15, I:26	89	支持	为分区建立设备映射的程序

Table 9.18: 硬盘分区管理软件包

注意
为了在 GPT 和 MBR 之间切换，你需要直接删除开头的几个块中的内容（参见第 9.7.6 节）并使用“parted /dev/sdx mklabel gpt”或“parted /dev/sdx mklabel msdos”来设置它。请注意，这里使用的“msdos”是用于 MBR。

9.5.3 使用 UUID 访问分区

尽管重新配置你的分区或可移动存储介质的激活顺序可能会给分区产生不同的名字，但你可以使用同一个 UUID 来访问它们。如果你有多个硬盘并且你的 BIOS 没有给它们一致的设备名的话，使用 UUID 是不错的选择。

- mount(8) 命令带有“-U”选项可以使用UUID 来挂载一个块设备，而不必使用他的文件名称，例如“/dev/sda3”。
- “/etc/fstab”（参见 fstab(5)）可以使用 UUID。
- 引载加载程序（第 3.1.2 节）也可以使用 UUID。

提示
你可以使用 blkid(8) 来查看一个特定块设备的 UUID。

提示
如果需要的话，设备（例如可移动存储介质）的设备节点可以通过 udev 规则 使其变为静态。参见第 3.3 节。

9.5.4 LVM2

LVM2 是一个用于 Linux 内核的逻辑卷管理器。使用 LVM2 的话，硬盘分区可以创建在逻辑卷上来替代物理硬盘。LVM 有下列需求。

- Linux 内核中的设备映射支持（Debian 内核默认支持）
- 用户自定义设备映射支持库（libdevmapper* 软件包）
- 用户自定义 LVM2 工具（lvm2 软件包）

请从下面的 man 手册开始了解 LVM2。

- lvm(8)：LVM2 机制的基础知识（列出了所有 LVM2 命令）
- lvm.conf(5)：LVM2 的配置文件
- lvs(8)：报告逻辑卷的相关信息
- vgs(8)：报告卷组的相关信息
- pvs(8)：报告物理卷的相关信息

9.5.5 文件系统配置

对于 ext4 文件系统, `e2fsprogs` 包提供下面的工具。

- mkfs.ext4(8) 创建新的 ext4 文件系统
- fsck.ext4(8) 检查和修复现有 ext4 文件系统
- tune2fs(8) 配置 ext4 文件系统的超级块
- debugfs(8) 交互式的调试 ext4 文件系统. (它有 `undel` 命令来恢复已经删除的文件.)

mkfs(8) 和 fsck(8) 命令是由 `e2fsprogs` 包提供的,是各种文件系统相关程序的前端。(`mkfs.fstype` 和 `fsck.fstype`). 对于 ext4 文件系统, 它们是 mkfs.ext4(8) 和 fsck.ext4(8) (它们被符号链接到 mke2fs(8) 和 e2fsck(8)).

Linux 支持的每一个文件系统，有相似的命令。

软件包	流行度	大小	说明
`e2fsprogs`	V:598, I:999	1419	ext2/ext3/ext4 文件系统工具
`reiserfsprogs`	V:13, I:26	923	Reiserfs 文件系统工具
`dosfstools`	V:114, I:556	235	FAT 文件系统工具. (Microsoft: MS-DOS, Windows)
`xfsprogs`	V:18, I:92	3349	XFS 文件系统工具. (SGI: IRIX)
`ntfs-3g`	V:273, I:548	1471	NTFS 文件系统工具. (Microsoft: Windows NT, ⋯)
`jfsutils`	V:1, I:13	1561	JFS 文件系统工具. (IBM: AIX, OS/2)
`reiser4progs`	V:0, I:4	1325	Reiser4 文件系统工具
`hfsprogs`	V:0, I:8	303	HFS 和 HFS Plus 文件系统工具. (Apple: Mac OS)
`btrfs-progs`	V:32, I:48	3314	Btrfs 文件系统工具
`zerofree`	V:2, I:74	25	把 ext2/3/4 文件系统上空闲块设置为零的程序

Table 9.19: 文件系统管理包列表

提示
Ext4 文件系统是 Linux 系统上默认的文件系统，强烈推荐使用这个文件系统，除非你有特殊的理由不使用。

提示
Btrfs 文件系统在 Linux 内核 3.2(Debian `wheezy`) 上存在。它被期望作为 ext4 文件系统之后的下一个默认文件系统。

警告
在得到当前内核空间的 fsck(8) 特征和引导管理器支持前，你的关键数据不应当使用 Btrfs 文件系统。

提示
一些工具可以在没有 Linux 内核支持的情况下访问文件系统 (参见第 9.7.2 节).

9.5.6 文件系统创建和完整性检查

mkfs(8) 在 Linux 系统上创建文件系统。fsck(8) 命令在 Linux 系统上提供文件系统完整性检查和修复功能。

在文件系统创建后，Debian 现在默认不周期性的运行 `fsck`。

小心
在已经挂载的文件系统上运行 fsck ，一般是不安全的.

提示
在"/etc/mke2fs.conf" 里设置"enable_periodic_fsck" 并使用"tune2fs -c0 /dev/<partition_name>" 设置最大挂载数为 0，便可以在重启时，让 root 文件系统包括在内的所有文件系统上，安全的运行 fsck(8) 命令. 参见 mke2fs.conf(5) 和 tune2fs(8).

提示
从启动脚本里面运行的 fsck(8) 命令结果，可以在"/var/log/fsck/" 目录下查看。

9.5.7 通过挂载选项优化文件系统

“/etc/fstab”中包含了基础的静态文件系统配置。例如，

```
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
proc            /proc           proc    defaults        0       0
UUID=709cbe4c-80c1-56db-8ab1-dbce3146d2f7 / ext4 noatime,errors=remount-ro 0 1
UUID=817bae6b-45d2-5aca-4d2a-1267ab46ac23 none swap sw  0       0
/dev/scd0       /media/cdrom0   udf,iso9660 user,noauto 0       0
```

提示
UUID（参见第 9.5.3 节）可以替代一般的块设备名称（例如“/dev/sda1”、“/dev/sda2”……）来识别一个块设备。

一个文件系统的性能和特性可以通过所用的挂载选项来进行优化（参见 fstab(5) 和 mount(8) ）。值得注意的有以下几点。

- “defaults”选项隐含的默认选项为:“rw,suid,dev,exec,auto,nouser,async”。（通常）
- “noatime”或“relatime”选项对于加速读取访问非常有效。（通常）
- “user”选项允许一个普通用户挂载文件系统。这个选项是“noexec,nosuid,nodev”选项的组合。（通常，用于 CD 或 usb 存储设备）
- “noexec,nodev,nosuid”选项组合被用来增强安全性。（通常）
- "noauto" 选项限制挂载，只有明确进行挂载操作才进行挂载（通常）
- 用于 ext3fs 的“data=journal”选项可以增强电源故障时数据的完整性，但会损失一些写入速度。

提示
配置 root 文件系统非默认的日志模式，你需要向内核提供启动参数 (参见第 3.1.2 节), 比如说"rootflags=data=journal"。对于 lenny 版本, 默认的日志模式是"rootflags=data=ordered". 对于 squeeze 版本，是"rootflags=data=writeback".

9.5.8 通过超级块（superblock）优化文件系统

一个文件系统的特性可以使用 tune2fs(8) 命令通过超级块来优化。

- 执行“`sudo tune2fs -l /dev/hda1`”可以显示“`/dev/hda1`”上的文件系统超级块内容。
- 执行”`sudo tune2fs -c 50 /dev/hda1`” 改变”`/dev/hda1`” 文件系统的检查 (在启动时执行 `fsck`) 频率为每 50 次启动.
- 执行“`sudo tune2fs -j /dev/hda1`”会给文件系统添加日志功能，即“`/dev/hda1`”的文件系统从 ext2 转换为 ext3。（对未挂载的文件系统这么做。）
- 执行“`sudo tune2fs -O extents,uninit_bg,dir_index /dev/hda1 && fsck -pf /dev/hda1`”在“`/dev/hda1`”上将它从 ext3 转换为 ext4。（对未挂载的系统这么做。）

提示
尽管 tune2fs(8) 的名字是这样的，但它不仅能用于 ext2 文件系统，也能用于 ext3 和 ext4 文件系统。

9.5.9 硬盘优化

警告
在你折腾硬盘配置之前，请检查你的硬件并阅读 hdparam(8) 的 man 手册页，因为这可能会对数据完整性造成相当大的危害。

你可以通过“`hdparm -tT /dev/hda`”来测试“`/dev/hda`”硬盘的访问速度。对于一些使用 (E)IDE 连接的硬盘，你可以使用“`hdparm -q -c3 -d1 -u1 -m16 /dev/hda`”来启用“(E)IDE 32 位支持”、启用“using_dma flag”、设置“interrupt-unmask flag”并设置“multiple 16 sector I/O”（危险！），从而加速硬盘访问速度。

你可以通过“`hdparm -W /dev/sda`”来测试“`/dev/sda`”硬盘的写入缓存功能。你可以使用“`hdparm -W 0 /dev/sda`”关闭写入缓存功能。

现代高速 CD-ROM 光驱，你可以使用”`setcd -x 2`” 降低速度，来读取不当压缩的 CDROM 光盘。

9.5.10 固态硬盘优化

固态硬盘（solid state drive，SSD）的性能和硬盘磨损可以通过下列方式优化。

- 使用最新的 Linux 内核。（>= 3.2）
- 减少读取硬盘访问的硬盘写入。
  - 在 `/etc/fstab` 中设置“`noatime`”或“`relatime`”挂载选项。
- 启用 TRIM 命令。
  - 在 `/etc/fstab` 中为 ext4 文件系统、swap 分区、Btrfs 等设置 `discard` 挂载选项。参见 fstab(5)。
  - 在 `/etc/lvm/lvm.conf` 中为 LVM 设置“`discard`”选项。参见 lvm.conf(5)。
  - 在 `/etc/crypttab` 中为 dm-crypt 设置“`discard`”选项。参见 crypttab(5)。
- 启用 SSD 硬盘空间分配优化方案。
  - 在 `/etc/fstab` 中为 Brtfs 设置“`ssd`”挂载选项。

- 对于笔记本电脑，使系统每 10 分钟刷新数据到硬盘。
  - 在 /etc/fstab 中设置“commit=600”挂载选项。参见 fstab(5)。
  - 设置 pm-utils 使用笔记本模式，即使在 AC 电源供电下。参见 Debian BTS #659260.

警告
将刷新间隔从一般的 5 秒改为 10 分钟会导致遇到电源故障时数据容易丢失。

9.5.11 使用 SMART 预测硬盘故障

你可以使用兼容 SMART 的 smartd(8) 后台守护进程（daemon）来监控和记录你的硬盘。

1. 在 BIOS 中启用 SMART 功能。
2. 安装 smartmontools 软件包。
3. 通过 df(1) 列出硬盘驱动并识别它们。
   - 让我们假设要监控的硬盘为“/dev/hda”。
4. 检查“smartctl -a /dev/hda”的输出，看 SMART 功能是否已启用。
   - 如果没有，通过“smartctl -s on -a /dev/hda”启用它。
5. 通过下列方式运行 smartd(8) 后台守护进程（daemon）。
   - 消除 /etc/default/smartmontools”文件中“start_smartd=yes”的注释。
   - 通过“sudo /etc/init.d/smartmontools restart”重新启动 smartd(8) 后台守护进程（daemon）。

提示
smartd(8) 后台守护进程（daemon）可以使用 /etc/smartd.conf 文件进行自定义，文件中包含了相关的警告。

9.5.12 通过 $TMPDIR 指定临时存储目录

应用程序一般在临时存储目录“/tmp”下建立临时文件。如果“/tmp”没有足够的空间，你可以通过 $TMPDIR 变量来为程序指定临时存储目录。

9.5.13 通过 LVM 扩展可用存储空间

在安装时创建在 Logical Volume Manager 逻辑卷管理 (LVM) (Linux 特性) 上的分区，它们可以容易的通过合并扩展或删除扩展的方式改变大小，而不需要在多个存储设备上进行大量的重新配置。

9.5.14 通过挂载另一个分区来扩展可用存储空间

如果你有一个空的分区（例如“/dev/sdx”），你可以使用 mkfs.ext4(1) 将它格式化，并使用 mount(8) 将它挂载到你需要更多空间的目录。（你需要复制原始数据内容。）

```
$ sudo mv work-dir old-dir
$ sudo mkfs.ext4 /dev/sdx
$ sudo mount -t ext4 /dev/sdx work-dir
$ sudo cp -a old-dir/* work-dir
$ sudo rm -rf old-dir
```

提示
你也可以选择挂载一个空硬盘映像文件（参见第 9.6.5 节）作为一个循环设备（参见第 9.6.3 节）。实际的硬盘使用量会随着实际存储数据的增加而增加。

9.5.15 通过“mount --bind”挂载另一个目录来扩展可用存储空间

如果你在另一个分区里有一个带有可用空间的空目录（例如“/path/to/emp-dir”），你可以通过带有“--bind”选项的 mount(8)，将它挂载到一个你需要更多空间的目录（例如“work-dir”）。

```
$ sudo mount --bind /path/to/emp-dir work-dir
```

9.5.16 通过 overlay 挂载（overlay-mounting）另一个目录来扩展可用存储空间

如果你在另一个分区表中有可用的空间（例如，“/path/to/empty”和“/path/to/work”），你可以在其中建立一个目录并堆栈到你需要空间的那个旧的目录（例如，“/path/to/old”），要这样做，你需要用于 Linux 3.18 版内核或更新版本（对应 Debian Stetch 9.0 或更新版本）的 OverlayFS。

```
$ sudo mount -t overlay overlay \
  -olowerdir=/path/to/old-dir,upperdir=/path/to/empty,workdir=/path/to/work
```

“/path/to/empty”和“/path/to/work”应该位于可读写的分区，从而能够写入“/path/to/old”。

9.5.17 使用符号链接扩展可用存储空间

小心
这是一个已弃用的做法。某些软件在遇到“软链接目录”时可能不会正常工作。请优先使用上文所述的“挂载”的途径。

如果你在另一个分区里有一个带有可用空间的空目录（例如“/path/to/emp-dir”），你可以使用 ln(8) 建立目录的一个符号链接。

```
$ sudo mv work-dir old-dir
$ sudo mkdir -p /path/to/emp-dir
$ sudo ln -sf /path/to/emp-dir work-dir
$ sudo cp -a old-dir/* work-dir
$ sudo rm -rf old-dir
```

警告
别对由系统管理的目录（例如“/opt”）使用“链接到目录”，这样的链接在系统升级时可能会被覆盖。

9.6 磁盘映像

我们在这里讨论磁盘影响的操作。

9.6.1 制作磁盘映像文件

一个未挂载设备（例如，第二个 SCSI 或串行 ATA 设备“/dev/sdb”）的磁盘映像文件“disk.img”可以使用 cp(1) 或 dd(1) 通过下列方式建立。

```
# cp /dev/sdb disk.img
# dd if=/dev/sdb of=disk.img
```

传统 PC 中位于主 IDE 硬盘第一扇区的主引导记录（MBR）（参见第 9.5.2 节）的磁盘映像可以使用 dd(1) 通过下列方式建立。

```
# dd if=/dev/hda of=mbr.img bs=512 count=1
# dd if=/dev/hda of=mbr-nopart.img bs=446 count=1
# dd if=/dev/hda of=mbr-part.img skip=446 bs=1 count=66
```

- “mbr.img”：带有分区表的 MBR
- “mbr-nopart.img”：不带分区表的 MBR
- “mbr-part.img”：仅 MBR 的分区表

如果你使用 SCSI 或串行 ATA 设备作为启动硬盘，你需要使用“/dev/sda”替代“/dev/hda”。

如果你要建立原始硬盘的一个硬盘分区的映像，你需要使用“/dev/hda1”等替代“/dev/hda”。

9.6.2 直接写入硬盘

磁盘映像文件“disk.img”可以通过下列方式写入到一个匹配大小的未挂载设备（例如，第二个 SCSI 设备“/dev/sdb”。

```
# dd if=disk.img of=/dev/sdb
```

相似地，硬盘分区映像文件“partition.img”可以通过下列方式写入到匹配大小的未挂载分区（例如，第二个 SCSI 设备的第一个分区“/dev/sdb1”）。

```
# dd if=partition.img of=/dev/sdb1
```

9.6.3 挂载磁盘映像文件

可以使用循环设备通过下列方式挂载和卸载包含单个分区映像的磁盘映像“partition.img”。

```
# losetup -v -f partition.img
Loop device is /dev/loop0
# mkdir -p /mnt/loop0
# mount -t auto /dev/loop0 /mnt/loop0
...hack...hack...hack
# umount /dev/loop0
# losetup -d /dev/loop0
```

可以简化为如下步骤。

```
# mkdir -p /mnt/loop0
# mount -t auto -o loop partition.img /mnt/loop0
...hack...hack...hack
# umount partition.img
```

可以使用 循环设备 挂载包含多个分区的磁盘映像“disk.img”的每个分区。因为循环设备默认不管理分区，因此我们需要通过下列方式重新设置它。

```
# modinfo -p loop # verify kernel capability
max_part:Maximum number of partitions per loop device
max_loop:Maximum number of loop devices
# losetup -a # verify nothing using the loop device
# rmmod loop
# modprobe loop max_part=16
```

现在循环设备可以管理多达 16 个分区。

```
# losetup -v -f disk.img
Loop device is /dev/loop0
# fdisk -l /dev/loop0

Disk /dev/loop0: 5368 MB, 5368709120 bytes
255 heads, 63 sectors/track, 652 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x452b6464

      Device Boot      Start         End      Blocks   Id  System
/dev/loop0p1               1         600     4819468+  83  Linux
/dev/loop0p2             601         652      417690   83  Linux
# mkdir -p /mnt/loop0p1
# mount -t ext4 /dev/loop0p1 /mnt/loop0p1
# mkdir -p /mnt/loop0p2
# mount -t ext4 /dev/loop0p2 /mnt/loop0p2
```

```
...hack...hack...hack
# umount /dev/loop0p1
# umount /dev/loop0p2
# losetup -d /dev/loop0
```

或者，你也可以使用 kpartx 软件包中的 kpartx(8) 建立 设备映射设备来达到相同的效果。

```
# kpartx -a -v disk.img
...
# mkdir -p /mnt/loop0p2
# mount -t ext4 /dev/mapper/loop0p2 /mnt/loop0p2
...
...hack...hack...hack
# umount /dev/mapper/loop0p2
...
# kpartx -d /mnt/loop0
```

注意
你也可以使用循环设备利用偏移量来跳过 MBR 等，来挂载此类磁盘映像的单个分区。但这更加容易出错。

9.6.4 清理磁盘映像文件

使用下面的方式，一个磁盘映像文件"disk.img" 能够清理掉所有已经删除的文件，成为一个干净的稀疏映像"new.img"。

```
# mkdir old; mkdir new
# mount -t auto -o loop disk.img old
# dd bs=1 count=0 if=/dev/zero of=new.img seek=5G
# mount -t auto -o loop new.img new
# cd old
# cp -a --sparse=always ./ ../new/
# cd ..
# umount new.img
# umount disk.img
```

如果“disk.img”位于 ext2、ext3 或 ext4，你也可以像下面那样使用 zerofree 软件包中的 zerofree(8)。

```
# losetup -f -v disk.img
Loop device is /dev/loop3
# zerofree /dev/loop3
# cp --sparse=always disk.img new.img
```

9.6.5 制作空的磁盘映像文件

按下面的方式使用 dd(1)，可以制作一个大小为 5GiB 的空磁盘映像文件。

```
$ dd bs=1 count=0 if=/dev/zero of=disk.img seek=5G
```

按下面的方式使用环回设备，你能够在这个磁盘映像”disk.img” 上创建 ext4 文件系统。

```
# losetup -f -v disk.img
Loop device is /dev/loop1
# mkfs.ext4 /dev/loop1
...hack...hack...hack
# losetup -d /dev/loop1
$ du  --apparent-size -h disk.img
5.0G  disk.img
$ du -h disk.img
83M disk.img
```

对于”disk.img”, 它的文件大小是 5.0 GiB，而它实际磁盘使用仅仅是 83MiB. 这个差距可能是由于 ext4 里面有稀疏文件.

提示
稀疏文件的实际磁盘使用会随着数据的写入而增加。

回环设备 或 设备映射 设备上使用类似的操作，在这些设备按第 9.6.3 节挂载后, 你能够使用 parted(8) 或 fdisk(8) 对这个磁盘映像”disk.img” 进行分区，能够使用 mkfs.ext4(8), mkswap(8) 在上面创建文件系统等.

9.6.6 制作 ISO9660 镜像文件

” 源目录” 下的目录树可以通过如下所示的 cdrkit 提供的 genisoimage(1) 命令来制作 ISO9660 镜像文件，”cd.iso”。

```
#  genisoimage -r -J -T -V volume_id -o cd.iso source_directory
```

类似的,可引导的 ISO9660 镜像文件,”cdboot.iso”, 能够从 debian-installer 类似目录树”source_directory” 制作，方式如下。

```
#  genisoimage -r -o cdboot.iso -V volume_id \
   -b isolinux/isolinux.bin -c isolinux/boot.cat \
   -no-emul-boot -boot-load-size 4 -boot-info-table source_directory
```

这里的 Isolinux boot loader (参见第 3.1.2 节) 是用于启动的.

按下面的方式，你可以直接从光驱设备计算 md5sum 值，并制作 ISO9660 镜像。

```
$ isoinfo -d -i /dev/cdrom
CD-ROM is in ISO 9660 format
...
Logical block size is: 2048
Volume size is: 23150592
...
# dd if=/dev/cdrom bs=2048 count=23150592 conv=notrunc,noerror | md5sum
# dd if=/dev/cdrom bs=2048 count=23150592 conv=notrunc,noerror > cd.iso
```

警告
为了得到正确结果，你必须小心避免 Linux ISO9600 文件系统预读 bug。

9.6.7 直接写入文件到 CD/DVD-R/RW

提示
对于由 cdrkit 提供的 wodim(1) 来讲，DVD 仅仅是一个大的 CD。

你能够通过如下所示的命令找到可用的设备。

```
# wodim --devices
```

然后将空的 CD-R 插入 CD 驱动器并且把 ISO9660 镜像文件，”cd.iso” 写入到设备中，例如用如下所示的 wodim(1) 将数据写入到”/dev/hda” 设备。

```
# wodim -v -eject dev=/dev/hda cd.iso
```

如果用 CD-RW 代替 CD-R，用如下所示的命令来替代。

```
# wodim -v -eject blank=fast dev=/dev/hda cd.iso
```

提示
如果你的桌面系统自动挂载 CDs，在使用 wodim(1) 之前在终端里面用”sudo umount /dev/hda” 卸载它。

9.6.8 挂载 ISO9660 镜像文件

如果”cd.iso” 包含一个 ISO9660 镜像, 下面的命令手工挂载这个文件到”/cdrom”.

```
# mount -t iso9660 -o ro,loop cd.iso /cdrom
```

提示
现代桌面系统能够自动挂载可移动介质，如按 ISO9660 格式化的 CD(参见第 10.1.7 节).

9.7 二进制数据

这里，我们讨论直接操作存储介质上的二进制数据。

9.7.1 查看和编辑二进制数据

最基础的查看二进制数据的方法是使用”od -t x1” 命令。

提示
HEX 是十六进制英文hexadecimal首字母缩略词，基数 radix 是 16。OCTAL 是八进制英文octal 首字母缩略词，基数 radix是 8。ASCII 是美国信息交换标准代码 American Standard Code for Information Interchange 的英文缩写，即正常的英语文本代码。EBCDIC 是扩展二进制编码十进制交换码 Extended Binary Coded Decimal Interchange Code 的英文缩写，在 IBM 大型机 操作系统上使用。

软件包	流行度	大小	说明
coreutils	V:888, I:999	15719	基础软件包，有 od(1) 来导出文件 (HEX, ASCII, OCTAL, ⋯)
bsdmainutils	V:861, I:999	587	工具软件包，有 hd(1) 来导出文件 (HEX, ASCII, OCTAL, ⋯)
hexedit	V:1, I:12	71	二进制浏览和编辑器 (HEX, ASCII)
bless	V:0, I:5	1028	全功能的十六进制编辑器 (GNOME)
okteta	V:1, I:18	1446	全功能的十六进制编辑器 (KDE4)
ncurses-hexedit	V:0, I:2	132	二进制浏览和编辑器 (HEX, ASCII, EBCDIC)
beav	V:0, I:0	133	二进制浏览和编辑器 (HEX, ASCII, EBCDIC, OCTAL, ...)

Table 9.20: 查看和修改二进制数据的软件包列表

9.7.2 不挂载磁盘操作文件

有工具可以在没有挂载磁盘的情况下读写文件。

软件包	流行度	大小	说明
mtools	V:10, I:93	384	不挂载磁盘的 MSDOS 文件工具
hfsutils	V:0, I:7	1771	不挂载磁盘的 HFS 和 HFS+ 文件工具

Table 9.21: 不挂载磁盘操作文件的软件包列表

9.7.3 数据冗余

Linux 内核所提供的RAID软件系统提供内核文件系统级别的数据冗余来实现高水平的存储可靠性。

有在应用程序级别增加数据冗余来实现高水平存储可靠性的工具。

软件包	流行度	大小	说明
par2	V:1, I:9	246	奇偶校验档案卷设置，用于检查和修复文件
dvdisaster	V:0, I:2	1737	CD/DVD 媒体数据损失/划伤/老化的保护
dvbackup	V:0, I:0	412	使用 MiniDV 便携式摄像机的备份工具 (提供 rsbep(1))
vdmfec	V:0, I:0	97	使用前向纠错恢复丢失的块

Table 9.22: 向文件添加数据冗余的工具列表

9.7.4 数据文件恢复和诊断分析

有用于数据文件恢复和诊断分析的工具。

提示
在 e2fsprogs 软件包里有 debugfs(8) 命令，使用该命令里的 list_deleted_inodes 和 undel 指令，你能够恢复 ext2 文件系统上删除的文件。

9.7.5 把大文件分成多个小文件

当一个文件太大而不能备份的时候，你应该在备份之前先把它分割为多个小于 2000MiB 的小文件，稍后再把这些小文件合并为初始的文件。

软件包	流行度	大小	说明
testdisk	V:3, I:39	1339	分区扫描和磁盘恢复的实用程序
magicrescue	V:0, I:3	254	通过查找幻数 magic 字节来恢复文件的工具（译注：请 man file 来了解幻数）
scalpel	V:0, I:4	87	简洁、高性能的文件提取
myrescue	V:0, I:3	82	恢复损坏硬盘中的数据
extundelete	V:1, I:11	148	恢复删除 ext3/4 文件系统上的文件的实用程序
ext4magic	V:0, I:4	233	恢复删除 ext3/4 文件系统上的文件的实用程序
ext3grep	V:0, I:3	281	帮助恢复 ext3 文件系统上删除的文件的工具
scrounge-ntfs	V:0, I:3	49	NTFS 文件系统的数据恢复程序
gzrt	V:0, I:0	57	gzip 恢复工具包
sleuthkit	V:2, I:18	1212	诊断分析工具 (Sleuthkit)
autopsy	V:0, I:2	1021	SleuthKit 的图形化界面
foremost	V:0, I:7	100	恢复数据的诊断程序
guymager	V:0, I:1	1067	基于 Qt 的诊断图像工具
dcfldd	V:0, I:5	95	增强版的 dd，用于诊断和安全

Table 9.23: 数据文件恢复和诊断分析软件包列表

```
$ split -b 2000m large_file
$ cat x* >large_file
```

小心
为了防止文件名冲突，请确保没有任何以”x” 开头的文件。

9.7.6 清空文件内容

为了清除诸如日志文件之类的文件的内容，不要用 rm(1) 命令去删除文件然后创建新的空文件，因为这个文件可能在命令执行的期间还在被使用。以下是清除文件内容的正确方法。

```
$ :>file_to_be_cleared
```

9.7.7 样子文件

下面的命令创建样子文件或空文件。

```
$ dd if=/dev/zero    of=5kb.file bs=1k count=5
$ dd if=/dev/urandom of=7mb.file bs=1M count=7
$ touch zero.file
$ : > alwayszero.file
```

你将发现下列文件。

- ”5kb.file” 是 5KB 的全零数据。
- ”7mb.file” 是 7MB 随机数据。

- ”zero.file” 也许是一个 0 字节的文件。如果这个文件之前就存在，则它的 mtime 会被更新，而它的内容和长度保持不变。
- ”alwayszero.file” 一定是一个 0 字节文件。如果这个文件之前存在，则它的 mtime 会被更新，而它的内容会被清零。

9.7.8 擦除整块硬盘

有几种方法来完全擦除设备上整个硬盘上数据，比如说，在”/dev/sda” 上的 USB 内存盘。

小心
在执行这里的命令之前，你应该用 mount(8) 命令来查看 USB 记忆棒的挂载位置。”/dev/sda” 指向的设备可能是装有整个系统的 SCSI 硬盘或者 serial-ATA 硬盘。

如下所示是通过数据归 0 的方式来擦除硬盘上所有数据的。

```
# dd if=/dev/zero of=/dev/sda
```

如下是用随机数据重写的方式来擦除所有数据的。

```
# dd if=/dev/urandom of=/dev/sda
```

如下是用随机数据重写的方式来高效擦除所有数据。

```
# shred -v -n 1 /dev/sda
```

因为 dd(1) 命令在许多可引导的 Linux CDs (例如 Debian 安装光盘) 上的 shell 环境下都是可用的，你能够在装有系统的硬盘上，例如”/dev/hda”, ”/dev/sda” 等等设备上运行擦除命令来完全清除已经安装的系统。

9.7.9 擦除硬盘上的未使用的区域

硬盘（或 USB 记忆棒）上未使用的区域，例如“/dev/sdb1”可能仍然包含可被擦除的数据，因为他们本身只是解除了从文件系统的链接，这些可以通过重写来清除。

```
# mount -t auto /dev/sdb1 /mnt/foo
# cd /mnt/foo
# dd if=/dev/zero of=junk
dd: writing to ‘junk’: No space left on device
...
# sync
# umount /dev/sdb1
```

警告
这对您的 USB 记忆棒来说通常已经足够好了，但这还不完美。大部分已擦除的文件名和它们的属性可能隐藏并留在文件系统中。

9.7.10 恢复已经删除但仍然被打开的文件

即使你不小心删除了某个文件，只要这个文件仍然被一些应用程序所使用（读或者写），恢复此文件是可能的。

尝试下列例子

```
$ echo foo > bar
$ less bar
$ ps aux | grep 'less[ ]'
bozo    4775  0.0  0.0  92200   884 pts/8    S+   00:18   0:00 less bar
$ rm bar
$ ls -l /proc/4775/fd | grep bar
lr-x------ 1 bozo bozo 64 2008-05-09 00:19 4 -> 2 /home/bozo/bar (deleted)
$ cat /proc/4775/fd/4 > 3bar
$ ls -l
-rw-r--r-- 1 bozo bozo 4 2008-05-09 00:25 bar
$ cat bar
foo
```

当你安装了 lsof 软件包的时候，在另外一个终端执行如下命令。

```
$ ls -li bar
2228329 -rw-r--r-- 1 bozo bozo 4 2008-05-11 11:02 bar
$ lsof |grep bar|grep less
less 4775 bozo 4r REG 8,3 4 2228329 /home/bozo/bar
$ rm bar
$ lsof |grep bar|grep less
less 4775 bozo 4r REG 8,3 4 2228329 /home/bozo/bar (deleted)
$ cat /proc/4775/fd/4 > bar
$ ls -li bar
2228302 -rw-r--r-- 1 bozo bozo 4 2008-05-11 11:05 bar
$ cat bar
foo
```

9.7.11 查找所有硬链接

有硬链接的文件，能够使用"ls -li" 确认。

```
$ ls -li
total 0
2738405 -rw-r--r-- 1 root root 0 2008-09-15 20:21 bar
2738404 -rw-r--r-- 2 root root 0 2008-09-15 20:21 baz
2738404 -rw-r--r-- 2 root root 0 2008-09-15 20:21 foo
```

"baz" 和"foo" 的链接数为"2" (>1)，表示他们有硬链接。它们的 inode 号都是"2738404". 这表示它们是同样的硬链接文件。如果你不想偶然碰巧发现硬链接文件，你可以通过 inode 号来查找它。比如说, 按下面的方式查找"2738404" 。

```
# find /path/to/mount/point -xdev -inum 2738404
```

9.7.12 不可见磁盘空间消耗

所有打开的文件被删除后，仍然消耗磁盘空间，尽管他们不能够被普通的 du(1) 所看见。这些被删除的文件和他们的大小，可以通过下面的方式列出。

```
# lsof -s -X / |grep deleted
```

9.8 数据加密提示

在可以物理访问您的 PC 的情况下，任何人都可以轻易获得 root 权限，访问您的 PC 上的所有文件 (见第 4.7.4 节)。这意味着登录密码系统在您的 PC 被偷盗时并不能保证您私人和敏感数据的安全。您必须部署数据加密技术来实现。尽管 GNU 隐私守护 (见第 10.3 节) 可以对文件进行加密，但它需要一些用户端的工作。

dm-crypt 和 eCryptfs通过 Linux 内核模块与很少的用户操作实现本地自动数据加密。

软件包	流行度	大小	说明
cryptsetup	V:32, I:80	67	可用于加密的块设备的实用程序 (dm-crypt / 3LUKS)
cryptmount	V:3, I:5	228	可用于加密的块设备着重于正常用户挂载/卸载的实用程序 (dm-crypt / LUKS)
ecryptfs-utils	V:5, I:8	396	可用于堆叠加密文件系统的实用程序 (eCryptfs)

Table 9.24: 数据加密工具列表

Dm-crypt 是一个使用 device-mapper 加密的文件系统. Device-mapper 映射一个块设备到另外一个。

eCryptfs 是另外一个加密文件系统，使用了堆叠文件系统。堆叠文件系统把它自己堆叠在已挂载文件系统的一个已有目录之上。

小心
数据加密会消耗 CPU 时间等资源，请权衡其利弊。

注意
通过 debian-installer (lenny 或更新版)，整个 Debian 系统能够被安装到一个加密的磁盘上，使用 dm-crypt/LUKS 和 initramfs.

提示
请参阅第 10.3 节用户空间加密实用程序：GNU Privacy Guard。

9.8.1 使用 dm-crypt/LUKS 加密移动磁盘

您可以用dm-crypt/LUKS加密大容量可移动设备上数据，例如挂载在“/dev/sdx”上的 USB 记忆棒。你只需按如下步骤简单地把它格式化。

```
# badblocks -c 1024 -s -w -t random -v /dev/sdx
# fdisk /dev/sdx
... "n" "p" "1" "return" "return" "w"
# cryptsetup luksFormat /dev/sdx1
...
# cryptsetup open --type luks /dev/sdx1 sdx1
...
# ls -l /dev/mapper/
total 0
crw-rw---- 1 root root  10, 60 2008-10-04 18:44 control
brw-rw---- 1 root disk 254,  0 2008-10-04 23:55 sdx1
# mkfs.vfat /dev/mapper/sdx1
...
# cryptsetup luksClose sdx1
```

然后，它就可以正常的在现代桌面环境下，例如 GNOME 桌面可以使用 gnome-mount(1)，挂载到"/media/<disk_label"。只不过它会要求输入密码 (参见第 10.1.7 节)。不同的是写入的数据都是加密的。你可以把它格式化成其他格式的文件系统，例如用"mkfs.ext4 /dev/mapper/sdx1" 把它格式化为 ext4。

注意
如果您对数据的安全性要求很高，您可能需要重写多次 (在上述示例中的"badblocks" 命令)。虽然这个操作非常耗费时间。

9.8.2 用 dm-crypt 加密的交换分区

让我们假设你原先的"/etc/fstab" 包含以下内容。

```
/dev/sda7 swap sw 0 0
```

您可以使用 dm-crypt 通过如下步骤启用加密的交换分区。

```
# aptitude install cryptsetup
# swapoff -a
# echo "cswap /dev/sda7 /dev/urandom swap" >> /etc/crypttab
# perl -i -p -e "s/\/dev\/sda7/\/dev\/mapper\/cswap/" /etc/fstab
# /etc/init.d/cryptdisks restart
 ...
# swapon -a
```

9.8.3 使用 dm-crypt/LUKS 挂载加密的磁盘

用 dm-crypt/LUKS 在"/dev/sdc5" 上创建的加密磁盘可以用如下步骤挂载到"/mnt"：

```
$ sudo cryptsetup open /dev/sdc5 ninja --type luks
Enter passphrase for /dev/sdc5: ****
$ sudo lvm
lvm> lvscan
  inactive          '/dev/ninja-vg/root' [13.52 GiB] inherit
  inactive          '/dev/ninja-vg/swap_1' [640.00 MiB] inherit
  ACTIVE            '/dev/goofy/root' [180.00 GiB] inherit
  ACTIVE            '/dev/goofy/swap' [9.70 GiB] inherit
```

```
lvm> lvchange -a y /dev/ninja-vg/root
lvm> exit
  Exiting.
$ sudo mount /dev/ninja-vg/root /mnt
```

9.8.4 用 eCryptfs 自动加密文件

您可以用eCryptfs和 `ecryptfs-utils` 包对 `~/Private/`下的创建的文件自动加密。

- 根据下面的提示运行 ecryptfs-setup-private(1) 并设置 `~/Private/`。
- 通过运行 ecryptfs-mount-private(1) 激活“`~/Private/`”。
- 将敏感数据文件移动到”`~/Private/`” 并根据要求创建符号链接。
  - 候选：”`~/.fetchmailrc`”、”`~/.ssh/identity`”，”`~/.ssh/id_rsa`”，”`~/.ssh/id_dsa`” 和”`go-rwx`” 的其他文件
- 将敏感数据目录移动到”`~/Private/`” 的子目录中并按要求创建符号链接。
  - 候选：”`~/.gnupg`” 和”`go-rwx` 的其他目录
- 创建从“`~/Desktop/Private/`”到“`~/Private/`的符号链接，实现更方便的桌面操作。
- 通过运行 ecryptfs-umount-private(1) 停用”`~/Private/`”。
- 在你需要加密文件时，使用“`ecryptfs-mount-private`”命令激活“`~/Private/`”目录。

提示
因为 eCryptfs 只是选择性的加密敏感数据，它的花费比使用 dm-crypt 在 root 或”`/home`” 设备加密的花费少的多。它不需要任何特殊的磁盘上的存储分配，但是其不能保证文件系统所有元数据的秘密性。

9.8.5 自动挂载 eCryptfs

如果您使用您的登录密码为环绕加密密钥，您可以通过 PAM（可插拔身份验证模块）自动化安装 eCryptfs。

在”`/etc/pam.d/common-auth`” 文件中的”`pam_permit.so`” 前插入下面的行。

```
auth required pam_ecryptfs.so unwrap
```

在”`/etc/pam.d/common-session`” 文件中插入下面的行作为最后一行。

```
session optional pam_ecryptfs.so unwrap
```

在”`/etc/pam.d/common-password`” 中的第一个活动行插入下面的行。

```
password required pam_ecryptfs.so
```

这相当方便。

警告
PAM的配置错误可能会把您锁在自己的系统外。请参阅第 4 章。

小心
如果你使用你自己的登录密码作为环绕加密密钥，加密的数据和用户登录密码的安全性一样 (参见第 4.3 节)。除非你已认真设置了一个强密码，否则你的数据仍然处在危险中，当别人偷了笔记本以后，然后运行密码破解 软件 (参见第 4.7.4 节)。

9.9 内核

对于支持的架构，Debian 使用软件包来分发模块化的 Linux 内核.

9.9.1 Linux 内核 2.6/3.x

相对于 2.4 版来说，Linux 内核的 2.6/3.x 版有一些值得注意的特征。

- 设备由 udev 系统创建 (参见第 3.3 节).
- 读写访问 IDE CD/DVD 设备，不再使用 `ide-scsi` 模块.
- 网络包过滤功能使用 `iptables` 内核模块.

Linux 版本从 2.6.39 跳到 3.0，不仅仅是一个主要的技术改变，也是第 20 个周年纪念日。

9.9.2 内核参数

许多 Linux 特性可以按下面的方式，通过内核参数来配置。

- 内核参数通过 bootloader 初始化 (参见第 3.1.2 节)
- 对通过 sysfs 访问的内核参数，在运行时通过 sysctl(8) 修改 (参见第 1.2.12 节)
- 当一个模块被激活时，通过 modprobe(8) 参数来设置模块参数。(参见第 9.6.3 节)

参见”`kernel-parameters.txt(.gz)`”和 `linux-doc-3.*` 软件包提供的其它相关文档(”`/usr/share/doc/linux-d`

9.9.3 内核头文件

大部分普通程序编译时不需要内核头文件，如果你直接使用它们来编译，甚至会导致编译中断。在 Debian 系统上，普通程序编译依赖 `libc6-dev` 软件包 (由 `glibc` 源代码包创建) 提供的，在”`/usr/include/linux`”和”`/usr/include/asm`” 里的头文件。

注意
对于编译一些内核相关的程序，比如说从外部源代码编译的内核模块和 automounter 后台守护 (daemon) 程序 (amd), 你必须包含相应的内核头文件到路径里，比如”`-I/usr/src/linux-particular-version/include/`”, 到你的命令行。module-assistant(8) (它的简称 `m-a`) 帮助我们更简单的为一个或者多个个性化内核编译和安装模块软件包。

9.9.4 编译内核和相关模块

Debian 有它自己的方式来编译内核和相关模块。

软件包	流行度	大小	说明
build-essential	I:454	20	创建 Debian 软件包所必须的软件包: make, gcc, ···
bzip2	V:178, I:953	196	bz2 文件压缩和解压缩工具
libncurses5-dev	V:13, I:139	6	ncurses 开发者库和文档
git	V:301, I:458	35266	git：Linux 内核使用的分布式版本控制系统
fakeroot	V:31, I:501	215	为非 root 用户创建软件包提供一个伪造的 root 环境
initramfs-tools	V:367, I:990	111	创建 initramfs 的工具（Debian 规范）
dkms	V:89, I:214	278	动态内核模块支持 dynamic kernel module support (DKMS) (通用)
devscripts	V:8, I:62	2485	Debian Package maintainer Debian 包维护者的帮助脚本（Debian 规范）

Table 9.25: Debian 系统内核编译需要安装的主要软件包列表

如果你在第 3.1.2 节使用 initrd，请一定阅读 initramfs-tools(8), update-initramfs(8), mkinitramfs(8) 和 initramfs.conf(5) 里的相关信息。

警告
在编译 Linux 内核源代码时，请不要放置从"/usr/include/linux" 和"/usr/include/asm" 到源代码树 (比如："/usr/src/linux*") 里目录的符号链接。(一些过期的文档建议这样做.)

注意
当在 Debian stable 版里编译最新的 Linux 内核时，可能需要使用一些从 Debian unstable 版里 backported 向后移植的最新版本的工具。

注意
dynamic kernel module support (DKMS) 动态内核模块支持 是一个新的分布式独立框架，被设计用来允许单个的内核模块在不改变整个内核的情况下升级。这可以用于维护内核代码树外部的模块。这也使你升级内核时，重新编译模块变得非常简单。

9.9.5 编译内核源代码：Debian 内核团队推荐

从上游内核源代码编译个性化的内核二进制包，你应当使用由它提供的"deb-pkg" 对象。

```
$ sudo apt-get build-dep linux
$ cd /usr/src
$ wget http://www.kernel.org/pub/linux/kernel/v3.11/linux-<version>.tar.bz2
$ tar -xjvf linux-<version>.tar.bz2
$ cd linux-<version>
$ cp /boot/config-<version> .config
$ make menuconfig
 ...
$ make deb-pkg
```

提示
linux-source-<version> 软件包使用”/usr/src/linux-<version>.tar.bz2” 提供有 Debian 补丁的 Linux 内核源代码。

从 Debian 内核源代码软件包编译特定的二进制包，你应当使用”debian/rules.gen” 里的”binary-arch_<architectur
对象。

```
$ sudo apt-get build-dep linux
$ apt-get source linux
$ cd linux-3.*
$ fakeroot make -f debian/rules.gen binary-arch_i386_none_686
```

进阶信息参见：

- Debian Wiki: KernelFAQ
- Debian Wiki: DebianKernel
- Debian Linux 内核手册：https://kernel-handbook.debian.net

9.9.6 硬件驱动和固件

硬件驱动是运行在目标系统上的代码。大部分硬件驱动现在是自由软件，已经包含在普通的 Debian 内核软件包里，放在 main 区域。

- GPU 驱动
  - Intel GPU 驱动 (main)
  - AMD/ATI GPU 驱动 (main) 和/
  - NVIDIA GPU 驱动 (nouveau 驱动放在 main , 由厂家支持的二进制驱动，放在 non-free.)
- Softmodem 驱动
  - martian-modem 和 sl-modem-dkms 软件包 (non-free)

固件是加载在设备上的代码 (比如说，CPU microcode, GPU 运行的渲染代码, 或 FPGA / CPLD 数据……) 部分固件包是作为自由软件存在，但是很多固件包由于包含有没有源代码的数据，二进制不是作为自由软件存在。

- firmware-linux-free (main)
- firmware-linux-nonfree (non-free)
- firmware-linux-* (non-free)
- *-firmware (non-free)
- intel-microcode (non-free)
- amd64-microcode (non-free)

请注意 non-free 和 contrib 的软件包不是 Debian 系统的一部分。启用和禁用 non-free 和 contrib 区域的配置，在第 2.1.4 节里描述。你应当注意到第 2.1.5 节里的描述，使用 non-free 和 contrib 软件包会有负面影响。

9.10 虚拟化系统

通过使用虚拟系统，我们能在单个机器上同时运行多个系统。

提示
参见 http://wiki.debian.org/SystemVirtualization.

9.10.1 虚拟化工具

除了简单的 chroot 工具外，Debian 上还有一些有关系统虚拟化及仿真的软件包。这些软件包能够帮你创建虚拟系统。

软件包	流行度	大小	说明
schroot	V:7, I:10	2728	在 chroot 下执行 Debian 二进制包的特异工具
sbuild	V:1, I:4	298	从 Debian 源码构建 Debian 二进制包的工具
pbuilder	V:1, I:16	966	Debian 软件包的打包软件
debootstrap	V:6, I:63	283	搭建一个基本的 Debian 系统 (用 sh 写的)
cdebootstrap	V:0, I:3	116	搭建一个 Debian 系统 (用 C 写的)
virt-manager	V:9, I:34	6770	虚拟机管理器: 用于管理虚拟机的桌面应用
libvirt-clients	V:30, I:51	2139	libvirt 的库程序
bochs	V:0, I:1	6706	Bochs: IA-32 PC 仿真器
qemu	I:35	95	QEMU: 快速的通用处理器仿真器
qemu-system	I:32	96	QEMU: 全功能系统的模拟二进制
qemu-user	V:2, I:30	84481	QEMU: 用户模式的模拟二进制
qemu-utils	V:9, I:97	5536	QEMU: 工具集
qemu-kvm	V:16, I:70	105	KVM: x86 硬件上有 硬件辅助虚拟化的全虚拟化
virtualbox	V:33, I:41	127956	VirtualBox:i386 和 amd64 上 x86 的虚拟化解决方案
xen-tools	V:0, I:5	704	用于管理 debian XEN 虚拟服务器的工具
wine	V:21, I:95	189	Wine: Windows 应用程序编程接口实现（标准套件）
dosbox	V:2, I:19	2778	DOSBox: 有 Tandy/Herc/CGA/EGA/VGA/SVGA 显卡, 声音和 DOS 的 x86 模拟器
dosemu	V:0, I:3	4891	DOSEMU: Linux DOS 模拟器
vzctl	V:1, I:2	1112	OpenVZ 服务器虚拟化解决方案 - 控制工具
vzquota	V:1, I:2	236	OpenVZ 服务器虚拟化解决方案 - 份额工具
lxc	V:9, I:14	2412	Linux 容器 用户层工具

Table 9.26: 虚拟化工具列表

参见维基百科 Comparison of platform virtual machines 来获得不同平台的虚拟化解决方案的详细比较信息。

9.10.2 虚拟化工作流

注意
这里所描述的功能只在 squeeze 或以后的版本中是可用的。

注意
自从 lenny 之后，默认的 Debian 内核就是支持 KVM 的。

典型的虚拟化工作流涉及以下几个步骤。

- 创建空文件系统 (目录树或磁盘映像)。
  - 目录树可以通过”`mkdir -p /path/to/chroot`” 创建。
  - 原始的磁盘映像文件能够使用 dd(1) 创建 (参见第 9.6.1 节和第 9.6.5 节).
  - qemu-img(1) 能够创建和转化 QEMU 支持的磁盘映像文件.
  - 原始的格式和 VMDK 文件格式, 能够作为虚拟化工具的通用格式.
- 使用 mount(8) 挂载磁盘映像到文件系统 (可选).
  - 对于原始磁盘映像文件, 把它作为回环设备 或 设备映射 设备挂载. (参见第 9.6.3 节).
  - 对于 QEMU 支持的磁盘映像, 把它们作为 network block device 网络块设备挂载 (参见第 9.10.3 节).
- 在目标文件系统上部署需要的系统数据。
  - 使用 `debootstrap` 和 `cdebootstrap` 之类的程序来协助处理这个过程 (参见第 9.10.4 节).
  - 在全功能系统模拟器下使用操作系统安装器。
- 在虚拟化环境下运行一个程序。
  - chroot 提供基本的虚拟化环境, 足够能在里面编译程序, 运行控制台应用, 运行后台守护程序 daemon.
  - QEMU 提供跨平台的 CPU 模拟器.
  - QEMU 和 KVM 通过 hardware-assisted virtualization 硬件辅助虚拟化来提供全功能系统的模拟.
  - VirtualBox 可以在 i386 和 amd64 上, 使用或者不使用 hardware-assisted virtualization 硬件辅助虚拟化来提供全功能系统模拟.

9.10.3 挂载虚拟磁盘映像文件

对于原始磁盘映像文件, 参见第 9.6 节.

对于其它虚拟磁盘映像文件, 你能够用使用 network block device 网络块设备 协议的 qemu-nbd(8) 来导出他们, 并使用内核模块 `nbd` 来挂载它们.

qemu-nbd(8) 支持 QEMU 所支持的磁盘格式: QEMU 支持下列磁盘格式: raw, qcow2, qcow, vmdk, vdi, bochs, cow (user-mode Linux copy-on-write), parallels, dmg, cloop, vpc, vvfat (virtual VFAT) 和主机设备.

网络块设备 能够用和回环设备一样的方式支持分区 (参见第 9.6.3 节). 你能够按下面的方式挂载”`disk.img`” 的第一个分区.

```
# modprobe nbd max_part=16
# qemu-nbd -v -c /dev/nbd0 disk.img
...
# mkdir /mnt/part1
# mount /dev/nbd0p1 /mnt/part1
```

提示
你可以给 qemu-nbd(8) 使用”`-P 1`” 选项来导出”`disk.img`” 的第一个分区.

9.10.4 Chroot 系统

chroot(8) 提供最基本的方式来运行一个不同的 GNU/Linux 系统实例，并且不需要重启原有的系统。

小心
下面的列子假设根源系统和 chroot 系统都共享相同的 CPU 架构。

你可以按下面的方式学会怎样建立和使用 chroot(8)，通过在 script(1) 下运行 pbuilder(8) 程序。

```
$ sudo mkdir /sid-root
$ sudo pbuilder --create --no-targz --debug --buildplace /sid-root
```

你能够看到 debootstrap(8) 或 cdebootstrap(1) 是如何在"`/sid-root`" 下部署 `sid` 环境的系统数据.

提示
这些 debootstrap(8) 或 cdebootstrap(1) 是 Debian 安装器用来安装 Debian 的. 这些也可以用来在不使用 Debian 安装盘的情况下，给一个系统安装 Debian，也可以替代安装其它 GNU/Linux 发行版。

```
$ sudo pbuilder --login --no-targz  --debug --buildplace /sid-root
```

你可以看到一个 `sid` 环境的系统 shell 是如何按下面的方式创建的。

1. 拷贝本地配置 ("`/etc/hosts`", "`/etc/hostname`", "`/etc/resolv.conf`")
2. 挂载"`/proc`" 文件系统
3. 挂载"`/dev/pts`" 文件系统
4. 创建"`/usr/sbin/policy-rc.d`" 的过程，总是 101 退出
5. 运行"`chroot /sid-root bin/bash -c 'exec -a -bash bin/bash'`"

注意
一些在 chroot 下的程序，需要访问比根源系统上的 `pbuilder` 能够提供的文件之外更多的文件. 例如，"`/sys`", "`/etc/passwd`", "`/etc/group`", "`/var/run/utmp`", "`/var/log/wtmp`" 等等. 也许需要使用 bind-mounted 或拷贝.

注意
"`/usr/sbin/policy-rc.d`" 文件阻止在 Debian 系统上自动启动后台守护程序。参见"`/usr/share/doc/sysv-rc/README.policy-rc.d.gz`".

提示
专用的 chroot 软件包 `pbuilder` 的原始用途, 是构建一个 chroot 系统，并在 chroot 里面打包软件包。它是一个理想的系统，可以用来检查软件包的安装依赖性是否正确，确保不需要的和错误的安装依赖在最终的软件包中不存在。

提示
类似的 `schroot` 软件包可以给你一个这样的主意，在 `amd64` 根源系统上运行 `i386` chroot 系统。

9.10.5 多桌面系统

我建议在 Debian 稳定版上使用 QEMU 或者 VirtualBox，这些软件应用虚拟化技术安全的运行多桌面系统。这能让你运行 Debian 不稳定版和测试版上的桌面应用并且没有与之相关的通常意义上的风险。

因为单纯的 QEMU 工具是非常慢的，当主机系统支持 KVM 的时候，建议使用它来加速。

按下面的方法，能够创建一个可以用于QEMU 的包含有 Debian 系统的虚拟磁盘映像"`virtdisk.qcow2`"，这个 Debian 系统使用 debian 安装器: 小 CD安装。

```
$ wget http://cdimage.debian.org/debian-cd/5.0.3/amd64/iso-cd/debian-503-amd64-netinst.iso
$ qemu-img create -f qcow2 virtdisk.qcow2 5G
$ qemu -hda virtdisk.qcow2 -cdrom debian-503-amd64-netinst.iso -boot d -m 256
...
```

在 Debian wiki: QEWU 可以查看更多信息。

VirtualBox 自带的 Qt 图形界面工具是相当直观的。关于它的图形界面和命令行工具的解释可以在 VisualBox 用户手册 和 VirtualBox 用户手册 (PDF) 中查看。

提示
在虚拟化下运行 Ubuntu 和 Fedora 之类的其它 GNU/Linux 发行版，是一个不错的学习其配置技巧的方法。其它专有操作系统也可以在这个 GNU/Linux 虚拟化下很好的运行。

Chapter 10

数据管理

以下是关于在 Debian 系统上管理二进制和文本数据的工具及其相关提示。

10.1 共享，拷贝和存档

警告
为避免竞争情况，不应当对正在进行写操作的设备和文件，多个进程进行不协调的写操作。采用 flock(1) 的文件锁定 机制可用于避免这种情况。

数据的安全和它的受控共享有如下几个方面。

- 存档文件的建立
- 远程存储访问
- 复制
- 跟踪修改历史
- 促进数据共享
- 防止未经授权的文件访问
- 检测未经授权的文件修改

这些可以通过使用工具集来实现。

- 存档和压缩工具
- 复制和同步工具
- 网络文件系统
- 移动存储媒介
- 安全 shell
- 认证体系
- 版本控制系统工具
- 哈希算法和加密工具

10.1.1 存档和压缩工具

以下是 Debian 系统上可用的存档和压缩工具的预览。

软件包	流行度	大小	扩展名	命令	描述
tar	V:916, I:999	2880	.tar	tar(1)	标准的归档工具（默认）
cpio	V:464, I:999	989	.cpio	cpio(1)	Unix System V 风格的归档器，与 find(1) 一起使用
binutils	V:186, I:694	93	.ar	ar(1)	创建静态库的归档工具
fastjar	V:4, I:45	172	.jar	fastjar(1)	Java 归档工具（类似 zip）
pax	V:14, I:36	164	.pax	pax(1)	新的 POSIX 归档工具，介于 tar 和 cpio 之间
gzip	V:888, I:999	243	.gz	gzip(1), zcat(1), ⋯	GNU LZ77 压缩工具（默认）
bzip2	V:178, I:953	196	.bz2	bzip2(1), bzcat(1), ⋯	Burrows-Wheeler block-sorting 压缩工具有着比 gzip(1) 更高的压缩率 (跟 gzip 有着相似的语法但速度比它慢)
lzma	V:3, I:39	141	.lzma	lzma(1)	LZMA 压缩工具有着比 gzip(1) 更高的压缩率（不推荐)
xz-utils	V:434, I:964	442	.xz	xz(1), xzdec(1), ⋯	XZ 压缩工具有着比 bzip2(1) 更高的压缩率（压缩速度慢于 gzip 但是比 bzip2 快; LZMA 压缩工具的替代品)
p7zip	V:88, I:439	986	.7z	7zr(1), p7zip(1)	有着更高压缩率的 7-zip 文件归档器（LZMA 压缩）
p7zip-full	V:131, I:521	4659	.7z	7z(1), 7za(1)	有着更高压缩率的 7-Zip 文件归档器（LZMA 压缩和其他）
lzop	V:6, I:51	97	.lzo	lzop(1)	LZO 压缩工具有着比 gzip(1) 更高的压缩和解压缩速度 (跟 gzip 有着相似的语法但压缩率比它低)
zip	V:50, I:442	608	.zip	zip(1)	InfoZip：DOS 归档器和压缩工具
unzip	V:250, I:804	554	.zip	unzip(1)	InfoZIP：DOS 解档器和解压缩工具

Table 10.1: 存档和压缩工具列表

警告
除非你知道将会发生什么，否则不要设置”$TAPE” 变量。它会改变 tar(1) 的行为。

注意
gzipped tar(1) 归档器用于扩展名是”.tgz” 或者”.tar.gz” 的文件。

注意
xz-compressed tar(1) 归档器用于扩展名是”.txz” 或者”.tar.xz” 的文件。

注意
FOSS 工具，例如 tar(1)，中的主流压缩方法已经按如下所示的迁移: gzip → bzip2 → xz

注意
cp(1),scp(1) 和 tar(1) 工具可能并不适用于一些特殊的文件。cpio(1) 工具的适用范围是最广的。

注意
cpio(1) 是被设计为与 find(1) 和其它命令一起使用，适合于创建备份脚本的场景，因此，脚本的文件选择部分能够被独立测试。

注意
Libreoffice 数据文件的内部结构是".jar" 文件，它也可以使用 unzip 工具来打开。

注意
事实上跨平台支持最好的存档工具是 zip。按照“zip -rX”的方式调用可以获得最大的兼容性。如果最大文件大小需要纳入考虑范围，请同时配合“-s”选项使用。

10.1.2 复制和同步工具

以下是 Debian 系统上的可用的简单复制和备份工具的预览。

软件包	流行度	大小	工具	功能
coreutils	V:888, I:999	15719	GNU cp	复制本地文件和目录 ("-a" 参数实现递归)
openssh-client	V:811, I:994	3545	scp	复制远端文件和目录 (客户端，"-r" 参数实现递归)
openssh-server	V:686, I:813	1449	sshd	复制远端文件和目录 (远程服务器)
rsync	V:365, I:628	729	-	单向远程同步和备份
unison	V:3, I:18	3457	-	双向远程同步和备份

Table 10.2: 复制和同步工具列表

在复制文件的时候，rsync(8) 比其他工具提供了更多的特性。

- 差分传输算法只会发送源文件与已存在的目标文件之间的差异部分
- 快速检查算法 (默认) 会查找大小或者最后的修改时间有变化的文件
- "--exclude" 和"--exclude-from" 选项类似于 tar(1)
- 在源目录中添加反斜杠的语法能够避免在目标文件中创建额外的目录级别。

提示
在 cron(8) 下使用"-gl" 选项执行在第 10.2.3 节里提到的 bkup 脚本，将提供一个和 Plan9 (译注：Plan9 也是一种文件系统，又称 v9fs, 可以用 apt-cache show 9mount 命令获取相关信息) 的 dumpfs 静态数据归档非常相似的功能。

提示
在表 10.11 中的版本控制系统 (VCS) 可以被认为是多路拷贝和同步工具。

10.1.3 归档语法

以下是用不同的工具压缩和解压缩整个"./source" 目录中的内容。

GNU tar(1):

```
$ tar -cvJf archive.tar.xz ./source
$ tar -xvJf archive.tar.xz
```

或者，如下所示。

```
$ find ./source -xdev -print0 | tar -cvJf archive.tar.xz --null -F -
```

cpio(1):

```
$ find ./source -xdev -print0 | cpio -ov --null > archive.cpio; xz archive.cpio
$ zcat archive.cpio.xz | cpio -i
```

10.1.4 复制语法

如下是用不同的工具复制整个"./source"目录中的内容。

- 本地复制: "./source"目录 → "/dest"目录
- 远程复制：本地主机上的"./source"目录 → "user@host.dom"主机上的"/dest"目录

rsync(8):

```
# cd ./source; rsync -aHAXSv . /dest
# cd ./source; rsync -aHAXSv . user@host.dom:/dest
```

你能够选择使用“源目录上的反斜杠”语法。

```
# rsync -aHAXSv ./source/ /dest
# rsync -aHAXSv ./source/ user@host.dom:/dest
```

或者，如下所示。

```
# cd ./source; find . -print0 | rsync -aHAXSv0 --files-from=- . /dest
# cd ./source; find . -print0 | rsync -aHAXSv0 --files-from=- . user@host.dom:/dest
```

GNU cp(1) 和 openSSH scp(1):

```
# cd ./source; cp -a . /dest
# cd ./source; scp -pr . user@host.dom:/dest
```

GNU tar(1):

```
# (cd ./source && tar cf - . ) | (cd /dest && tar xvfp - )
# (cd ./source && tar cf - . ) | ssh user@host.dom '(cd /dest && tar xvfp - )'
```

cpio(1):

```
# cd ./source; find . -print0 | cpio -pvdm --null --sparse /dest
```

你能够在所有包含”.” 的例子里用”foo” 替代”.”，这样就可以从”./source/foo” 目录复制文件到”/dest/foo” 目录。

在所有包含”.” 的列子里,你能够使用绝对路径”/path/to/source/foo” 来代替”.”,这样可以去掉”cd ./source;”. 如下所示，这些文件会根据工具的不同，拷贝到不同的位置。

- ”/dest/foo”: rsync(8), GNU cp(1), 和 scp(1)
- ”/dest/path/to/source/foo”: GNU tar(1), 和 cpio(1)

提示
rsync(8) 和 GNU cp(1) 可以用”-u” 选项来忽略接受端上更新的文件。

10.1.5 查找文件的语法

find(1) 被用作从归档中筛选文件也被用作拷贝命令 (参见第 10.1.3 节和第 10.1.4 节) 或者用于 xargs(1) (参见第 9.3.9 节)。通过 find 的命令行参数能够使其功能得到加强。

以下是 find(1) 基本语法的总结。

- find 条件参数的运算规则是从左到右。
- 一旦输出是确定的，那么运算就会停止。
- “逻辑 **OR**”（由条件之间的”-o” 参数指定的）优先级低于” 逻辑 **AND**”（由”-a” 参数指定或者条件之间没有任何参数）。
- ” 逻辑 **NOT**” (由条件前面的”!” 指定) 优先级高于“逻辑 **AND**”。
- ”-prune” 总是返回逻辑 **TRUE** 并且如果这个目录是存在的，将会搜索除这个目录以外的文件。
- ”-name” 选项匹配带有 shell 通配符 (参见第 1.5.6 节) 的文件名但也匹配带有类似”*” 和”?” 元字符的 .”。(新的 POSIX 特性)
- ”-regex” 匹配整个文件路径，默认采用 emacs 风格的 **BRE** (参见第 1.6.2 节)。
- ”-size” 根据文件大小来匹配 (值前面带有”+” 号匹配更大的文件，值前面带有”-” 号匹配更小的文件)
- ”-newer” 参数匹配比参数名中指定的文件还要新的文件。
- ”-print0” 参数总是返回逻辑 **TRUE** 并将完整文件名 (null terminated) 打印到标准输出设备上。

如下是 find(1) 语法格式。

```
# find /path/to \
    -xdev -regextype posix-extended \
    -type f -regex ”.*\.cpio|.*~” -prune -o \
    -type d -regex ”.*/\.git” -prune -o \
    -type f -size +99M -prune -o \
    -type f -newer /path/to/timestamp -print0
```

这些命令会执行如下动作。

1. 查找”/path/to” 下的所有文件

2. 限定全局查找的文件系统并且使用的是 **ERE** (参见第 1.6.2 节)
3. 通过停止处理的方式来排除匹配”`.*\.cpio`” 或”`.*~`” 正则表达式的文件
4. 通过停止处理的方式来排除匹配”`.*/\.git`” 正则表达式的目录
5. 通过停止处理的方式来排除比 99MB (1048576 字节单元) 更大的文件
6. 显示文件名，满足以上搜索条件并且比”`/path/to/timestamp`” 新的文件

请留心以上例子中的”`-prune -o`” 排除文件的习惯用法。

注意
对于非 Debian 系的 Unix-like 系统，有些参数可能不被 find(1) 命令所支持。在这种情况下，应该考虑调整匹配方法并用”`-print`” 替代”`-print0`”。你可能同样需要更改其他相关的命令。

10.1.6 归档媒体

为重要的数据存档寻找 存储设备 时，你应该注意它们的局限性。对于小型的个人数据备份，我使用品牌公司的 CD-R 和 DVD-R 然后把它放在阴凉、干燥、清洁的地方。(专业的一般使用磁带存档介质)

注意
防火安全是对于纸质文档来说的，大多数的计算机数据存储媒介耐热性比纸差。我经常依赖存储在多个安全地点的加密拷贝。

网上（主要是来源于供应商信息）可以查看存储介质的最大使用寿命。

- 大于 100 年：用墨水的无酸纸
- 100 年：光盘存储（CD/DVD，CD/DVD-R）
- 30 年：磁带存储（磁带，软盘)
- 20 年：相变光盘存储（CD-RW）

这不包括由于人为导致的机械故障等等。

网上（主要来源于供应商信息）可以查看存储介质的最大的写次数。

- 大于 250,000 次：硬盘驱动器
- 大于 10,000 次：闪存
- 1,000 次：CD/DVD-RW
- 1 次：CD/DVD-R，纸

小心
这里的存储寿命和写次数的数据不应该被用来决定任何用于关键数据的存储媒介，请翻阅制造商提供的特定产品的说明。

提示
因为 CD/DVD-R 和纸只能写一次，它们从根本上阻止了因为重写导致的数据意外丢失。这是优点！

提示
如果你需要更快更频繁的进行大数据备份，那么通过高速网络连接的远端主机上的硬盘来实现备份，可能是唯一可行的方法。

10.1.7 可移动存储设备

可移动存储设备可能是以下的任何一种。

- USB 闪存盘
- 硬盘驱动器
- 光盘驱动器
- 数码相机
- 数字音乐播放器

它们可以通过以下的方式来进行连接。

- USB
- IEEE 1394 / FireWire
- PC 卡

像 GNOME 和 KDE 这样的现代桌面环境能够在"`/etc/fstab`" 文件中没有匹配条目的时候，自动挂载这些可移动设备。

- `udisks` 包提供了守护进程和相关的实用程序来挂载和卸载这些设备。
- D-bus 创建事件来触发自动处理。
- PolicyKit 提供了所需的特权。

提示
umount(8) 在自动挂载设备的时候可能会带有"`uhelper=`" 参数。

提示
只有当这些可移动设备没有在"`/etc/fstab`" 文件中列出时，桌面环境下才会自动挂载。

现代桌面环境下的挂载点被选为"`/media/<disk_label>`"，它可以被如下所示的来定制。

- FAT 格式的文件系统使用 mlabel(1) 命令
- ISO9660 文件系统使用带有"`-V`" 选项的 genisoimage(1) 命令
- ext2/ext3/ext4 文件系统使用带有"`-L`" 选项的 tune2fs(1) 命令

提示
挂载时可能需要提供编码选项（参见第 8.4.6 节）。

提示
在图形界面菜单上移除文件系统，可能会移除它的动态设备节点例如"`/dev/sdc`"。如果你想要保留它的设备节点，你应该在命令行提示符上输入 umount(8) 命令来卸载它。

文件系统	典型使用场景描述
FAT12	软盘（<32MiB）上跨平台的数据分享
FAT16	在小硬盘 (<2GiB) 上的跨平台的数据分享
FAT32	在大硬盘（<8TiB，被 MS Windows95 OSR2 以上的操作系统所支持) 上的跨平台的数据分享
NTFS	在大硬盘类设备上的跨平台共享数据 (在 MS Windows NT 和后续版本原生支持；在 Linux 上，通过使用 FUSE 的 NTFS-3G 支持。)
ISO9660	在 CD-R 和 DVD+/-R 上的跨平台的静态数据分享
UDF	CD-R 和 DVD+/-R (新) 上的增量数据写入
MINIX 文件系统	软盘上磁盘空间高利用率的 unix 文件数据存储
ext2 文件系统	在装有老旧 linux 系统的硬盘上的数据分享
ext3 文件系统	在装有老旧 linux 系统的硬盘上的数据分享
ext4 文件系统	在装有较新的 linux 系统的硬盘上的数据分享

Table 10.3: 典型使用场景下可移动存储设备可选择的文件系统列表

10.1.8 选择用于分享数据的文件系统

当你通过可移动存储设备与其他系统分享数据的时候，你应该先把它格式化为被两种操作系统都支持的通用的 文件系统。下面是文件系统的列表。

提示
查看第 9.8.1 节来获得关于使用设备级加密的跨平台的数据共享的信息。

FAT 文件系统被绝大多数的现代操作系统支持，它对于通过可移动硬盘进行的数据交换是非常有用的。

当格式化像装有 FAT 文件系统的跨平台数据共享的可移动设备时，以下应该是保险的选择。

- 用 fdisk(8)，cfdisk(8) 或者 parted(8) 命令（参见第 9.5.2 节）把它们格式化为单个的主分区并对把它做如下标记。
  - 标记小于 2GB 的 FAT 设备为字符”6”。
  - 标记更大的 FAT32 设备为字符”c”。
- 如下所示是用 mkfs.vfat(8) 命令格式化主分区的。
  - 它的设备名字，例如”`/dev/sda1`” 用于 FAT16 设备
  - 明确的选项和它的设备名，例如”`-F 32 /dev/sda1`” 用于 FAT32 设备

当使用 FAT 或 ISO9660 文件系统分享数据时，如下是需要注意的安全事项。

- 用 tar(1)，或 cpio(1) 命令压缩文件，目地是为了保留文件名，符号链接，原始的文件权限和文件所有者信息。
- 用 split(1) 命令把压缩文件分解成若干小于 2GiB 的小文件，使其免受文件大小限制。
- 加密压缩文件保护其内容免受未经授权的访问。

注意
因为 FAT 文件系统的设计，最大的文件大小为 `(2^32 - 1) bytes = (4GiB -1 byte)`。对于一些老旧的 32 位系统上的应用程序而言，最大的文件大小甚至更小 `(2^31 -1) bytes = (2GiB -1 byte)`。Debian 没有遇到后者的问题。

注意
微软系统本身并不建议在超过 200MB 的分区或者驱动器上使用 FAT。他们的"Overview of FAT, HPFS, and NTFS File Systems 这篇文章突出显示了微软系统的缺点，例如低效的磁盘空间利用。当然了，我们在 Linux 系统上还是应该使用 ext4 文件系统。

提示
有关文件系统和访问文件系统的更多信息，请参考"Filesystems HOWTO"。

10.1.9 网络上的数据分享

当使用网络来分享数据的时候，你应该使用通用的服务。这里有一些提示。

网络服务	典型使用场景描述
SMB/CIFS 用 Samba 挂载网络文件系统	通过“Microsoft Windows 网络”分享文件，参见 smb.conf(5) 和 官方 Samba 3.x.x 指导和参考手册（The Official Samba 3.x.x HOWTO and Reference Guide） 或 samba-doc 软件包
NFS 用 Linux 内核挂载网络文件系统	通过“Unix/Linux 网络”分享文件，参见 exports(5) 和 Linux NFS-HOWTO
HTTP 服务	在 web 服务器/客户端之间分享文件
HTTPS 服务	在有加密的安全套接层 (SSL) 或者安全传输层 (TLS) 的网络服务器/客户端中分享文件
FTP 服务	在 FTP 服务器/客户端之间分享文件

Table 10.4: 典型使用场景下可选择的网络服务列表

尽管对于文件分享来说，通过网络挂载文件系统和传输文件是相当方便的，但这可能是不安全的。它们的网络连接必须通过如下所示的加强安全性。

- 用 SSL/TLS 加密
- 建立 SSH 通道
- 建立 VPN 通道
- 网络之间需要有安全的防火墙

参见第 6.10 节和第 6.11 节。

10.2 备份和恢复

我们都熟知计算机有时会出问题，或者由于人为的错误导致系统和数据损坏。备份和恢复操作是成功的系统管理中非常重要的一部分。可能有一天你的电脑就会出问题。

提示
保持你的备份系统简洁并且经常备份你的系统，有备份数据比你采用的备份方法的技术先进要重要的多。

有 3 个关键的因素决定实际的备份和恢复策略。

1. 知道要备份和恢复什么。

- 你自己创建的数据文件：在”~/” 下的数据
- 你使用的应用程序创建的数据文件:在”/var/” 下的数据 (除了”/var/cache/”,”/var/run/” 和”/var/tmp/”)
- 系统配置文件：在”/etc/” 下的数据
- 本地软件：在”/usr/local/” 或”/opt/” 下的数据
- 系统安装信息：关键步骤 (分区,...) 的纯文本备忘录
- 验证数据结果：通过实验性的恢复操作来预先验证

2. 知道怎样去备份和恢复。
   - 安全的数据存储：保护其免于覆盖和系统故障
   - 经常备份：有计划的备份
   - 冗余备份：数据镜像
   - 傻瓜式操作：单个简单命令备份
3. 评估涉及的风险和成本。
   - 评估数据丢失的损失
   - 备份所需的资源：人力，硬件，软件，…
   - 数据丢失的方式及其可能性

注意
除非你知道自己做的是什么，否则不要备份 /proc, /sys, /tmp, 和 /run 目录下的伪文件系统 (参见第 1.2.12 节和第 1.2.13 节)。它们是庞大且无用的数据。

至于安全的数据存储，数据至少是应该在不同的磁盘分区上最好是在不同的磁盘和机器上，来承受文件系统发生的损坏。重要的数据最好存储在只能写一次的媒介上例如 CD/DVD-R 来防止覆盖事故。(参见第 9.7 节怎样在 shell 命令行写入存储媒介。GNOME 桌面图形环境可以让你轻松的通过菜单：“位置 → CD/DVD 刻录”来实现写入操作。)

注意
当备份数据的时候，你可能希望停止一些应用程序的守护进程例如 MTA (参见第 6.3 节)。

注意
你应该格外小心地备份和恢复身份认证相关的数据文件例如”/etc/ssh/ssh_host_dsa_key”, ”/etc/ssh/ssh_host_rsa_key”, ”~/.gnupg/*”, ”~/.ssh/*”, /etc/passwd”, ”/etc/shadow”, ”/etc/fetchmailrc”, ”popularity-contest.conf”, ”/etc/ppp/pap-secrets” 和”/etc/exim4/passwd.client/”。这些数据中的有一些文件是不能通过向系统输入同样的字符串来再生的。

注意
如果你以用户进程的方式执行 cron job,你必须存储文件到”/var/spool/cron/crontabs” 目录并且重启 cron(8)。参见第 9.3.14 节来获得关于 cron(8) 和 crontab(1) 的信息。

软件包	流行度	大小	说明
dump	V:1, I:6	340	4.4 BSD dump(8) 和 restore(8) 命令用于 ext2/ext3/ext4 文件系统
xfsdump	V:0, I:10	834	在 GNU/Linux 和 IRIX 上用 xfsdump(8) 和 xfsrestore(8) 命令来备份和恢复 XFS 文件系统
backupninja	V:4, I:4	355	轻量的可扩展的 **meta-backup** 系统
bacula-common	V:10, I:17	2369	Bacula: 网络数据备份，恢复和核查-常见的支持文件
bacula-client	I:4	175	Bacula: 网络数据备份，恢复和核查-客户端元软件包
bacula-console	V:1, I:6	75	Bacula: 网络数据备份，恢复和核查-文本终端
bacula-server	I:1	175	Bacula: 网络数据备份，恢复和核查-服务器端元软件包
amanda-common	V:1, I:2	9890	Amanda: 马里兰大学开发的高级自动化网络磁盘归档器（库）
amanda-client	V:1, I:2	1133	Amanda: 马里兰大学开发的高级自动化网络磁盘归档器（客户端）
amanda-server	V:0, I:0	1089	Amanda: 马里兰大学开发的高级自动化网络磁盘归档器（服务器端）
backup-manager	V:1, I:2	571	命令行备份工具
backup2l	V:0, I:1	114	用于可挂载媒介 (基于磁盘的) 的低维护的备份/恢复工具
backuppc	V:4, I:4	2284	BackupPC 是用于备份 PC 机数据（基于磁盘）的高性能的企业级工具
duplicity	V:8, I:15	1609	(远程) 增量备份
flexbackup	V:0, I:0	243	(远程) 增量备份
rdiff-backup	V:8, I:16	704	(远程) 增量备份
restic	V:0, I:1	22182	(远程) 增量备份
rsnapshot	V:6, I:11	452	(远程) 增量备份
slbackup	V:0, I:0	152	(远程) 增量备份

Table 10.5: 实用备份程序套件列表

10.2.1 实用备份套件

以下是 Debian 系统上值得注意的实用备份程序套件的列表。

备份工具有各自的专用的用途。

- Mondo Rescue 是一个备份系统，它能够方便的从备份 CD/DVD 等设备中快速恢复整个系统，而不需要经过常规的系统安装过程。
- 定期备份用户数据, 可以通过一个简单的脚本（第 10.2.2 节）和 cron(8) 来实现。
- Bacula，Amanda 和 BackupPC 是全功能的备份实用套件，主要用于联网的定期备份。

第 10.1.1 节和第 10.1.2 节描述的基础工具能够通过自定义脚本来帮助系统备份。这些脚本的功能可以通过如下的工具来增强。

- `restic` 软件包能够增量备份（远程）。
- `rdiff-backup` 软件包能够增量备份（远程）。
- `dump` 软件包用于高效增量的归档和恢复整个文件系统。

提示
参见"`/usr/share/doc/dump/`" 和 "Is dump really deprecated?" 来了解 `dump` 程序。

10.2.2 一个系统备份的脚本例子

对于运行 unstable 套件的个人 Debian 桌面系统来说，只需要保护个人数据和关键数据。我不管怎样每年都会重新安装一次系统。因此没理由去备份整个系统或者安装全功能的备份实用程序。

我使用简单的脚本来制作用于备份的压缩文件并用 GUI 界面把它烧写到 CD/DVD 里。以下是关于这个的脚本例子。

```
#!/bin/sh -e
# Copyright (C) 2007-2008 Osamu Aoki <osamu@debian.org>, Public Domain
BUUID=1000; USER=osamu # UID and name of a user who accesses backup files
BUDIR="/var/backups"
XDIR0=".+/Mail|.+/Desktop"
XDIR1=".+/\.thumbnails|.+/\.?Trash|.+/\.?[cC]ache|.+/\.gvfs|.+/sessions"
XDIR2=".+/CVS|.+/\.git|.+/\.svn|.+/Downloads|.+/Archive|.+/Checkout|.+/tmp"
XSFX=".+\.iso|.+\.tgz|.+\.tar\.gz|.+\.tar\.bz2|.+\.cpio|.+\.tmp|.+\.swp|.+~"
SIZE="+99M"
DATE=$(date --utc +"%Y%m%d-%H%M")
[ -d "$BUDIR" ] || mkdir -p "BUDIR"
umask 077
dpkg --get-selections \* > /var/lib/dpkg/dpkg-selections.list
debconf-get-selections > /var/cache/debconf/debconf-selections

{
find /etc /usr/local /opt /var/lib/dpkg/dpkg-selections.list \
     /var/cache/debconf/debconf-selections -xdev -print0
find /home/$USER /root -xdev -regextype posix-extended \
  -type d -regex "$XDIR0|$XDIR1" -prune -o -type f -regex "$XSFX" -prune -o \
  -type f -size  "$SIZE" -prune -o -print0
find /home/$USER/Mail/Inbox /home/$USER/Mail/Outbox -print0
find /home/$USER/Desktop  -xdev -regextype posix-extended \
  -type d -regex "$XDIR2" -prune -o -type f -regex "$XSFX" -prune -o \
  -type f -size  "$SIZE" -prune -o -print0
} | cpio -ov --null -O $BUDIR/BU$DATE.cpio
chown $BUUID $BUDIR/BU$DATE.cpio
touch $BUDIR/backup.stamp
```

这是一个用 root 权限执行的脚本例子。

我建议你按照如下所示的去更改和执行这个脚本。

- 编辑这个脚本使其能够覆盖到你所有的重要数据（参见第 10.1.5 节和第 10.2 节）。
- 用"`find ...-newer $BUDIR/backup.stamp -print0`"替代"`find ...-print0`"来实现增量备份。
- 为保险起见，使用 scp(1) 或 rsync(1) 命令来备份文件到远端或者把它们烧写到 CD/DVD 里。（我使用 GNOME 桌面 GUI 来烧写 CD/DVD。参见第 12.1.8 节来获得更多的信息。）

把事情简单化！

提示
你能够用"`debconf-set-selections debconf-selections`"命令恢复 debconf 配置数据，可以用"`dpkg --set-selection <dpkg-selections.list`"命令恢复 dpkg 筛选数据。

10.2.3 用于备份数据的复制脚本

对于目录树下面的数据集，"`cp -a`"命令可以实现常规备份。

对于类似”/var/cache/apt/packages/” 目录下面的大量不可覆盖的静态数据集, 使用”cp -al” 命令来创建硬链接是一种替代常规备份的方式，这样可以高效的利用磁盘空间。

以下是一个用于数据备份的名为 bkup 的复制脚本。它把当前目录下的所有 (non-VCS) 文件复制到父目录下的指定目录中或者远程主机上。

```
#!/bin/sh -e
# Copyright (C) 2007-2008 Osamu Aoki <osamu@debian.org>, Public Domain
fdot(){ find . -type d \( -iname ".?*" -o -iname "CVS" \) -prune -o -print0;}
fall(){ find . -print0;}
mkdircd(){ mkdir -p "$1";chmod 700 "$1";cd "$1">/dev/null;}
FIND="fdot";OPT="-a";MODE="CPIOP";HOST="localhost";EXTP="$(hostname -f)"
BKUP="$(basename $(pwd)).bkup";TIME="$(date  +%Y%m%d-%H%M%S)";BU="$BKUP/$TIME"
while getopts gcCsStrlLaAxe:h:T f; do case $f in
g)  MODE="GNUCP";; # cp (GNU)
c)  MODE="CPIOP";; # cpio -p
C)  MODE="CPIOI";; # cpio -i
s)  MODE="CPIOSSH";; # cpio/ssh
t)  MODE="TARSSH";; # tar/ssh
r)  MODE="RSYNCSSH";; # rsync/ssh
l)  OPT="-alv";; # hardlink (GNU cp)
L)  OPT="-av";;  # copy (GNU cp)
a)  FIND="fall";; # find all
A)  FIND="fdot";; # find non CVS/ .???/
x)  set -x;; # trace
e)  EXTP="${OPTARG}";; # hostname -f
h)  HOST="${OPTARG}";; # user@remotehost.example.com
T)  MODE="TEST";; # test find mode
\?) echo "use -x for trace."
esac; done
shift $(expr $OPTIND - 1)
if [ $# -gt 0 ]; then
  for x in $@; do cp $OPT $x $x.$TIME; done
elif [ $MODE = GNUCP ]; then
  mkdir -p "../$BU";chmod 700 "../$BU";cp $OPT . "../$BU/"
elif [ $MODE = CPIOP ]; then
  mkdir -p "../$BU";chmod 700 "../$BU"
  $FIND|cpio --null --sparse -pvd ../$BU
elif [ $MODE = CPIOI ]; then
  $FIND|cpio -ov --null | ( mkdircd "../$BU"&&cpio -i )
elif [ $MODE = CPIOSSH ]; then
  $FIND|cpio -ov --null|ssh -C $HOST "( mkdircd \"$EXTP/$BU\"&&cpio -i )"
elif [ $MODE = TARSSH ]; then
  (tar cvf - . )|ssh -C $HOST "( mkdircd \"$EXTP/$BU\"&& tar xvfp - )"
elif [ $MODE = RSYNCSSH ]; then
  rsync -aHAXSv ./ "${HOST}:${EXTP}-${BKUP}-${TIME}"
else
  echo "Any other idea to backup?"
  $FIND |xargs -0 -n 1 echo
fi
```

如上只是一个范例。在你自己使用脚本之前，请阅读此脚本并且修改它。

提示
我把 bkup 保存在我的”/usr/local/bin” 目录。我假定当需要临时快照备份的时候，能够在工作目录不带任何参数运行 bkup 命令。

提示
如果是要制作源文件树或者配置文件树的快照历史的话，使用 git(7) (参见第 10.6.5 节) 是更简便并且也是空间高效的。

10.3 数据安全基础

数据安全基础设施是数据加密，讯息摘要和签名工具的结合。

软件包	流行度	大小	命令	说明
gnupg	V:737, I:996	727	gpg(1)	GNU 隐私卫士 - OpenPGP 加密和签名工具
gpgv	V:880, I:999	840	gpgv(1)	GNU 隐私卫士 - 签名验证工具
paperkey	V:0, I:6	58	paperkey(1)	从 OpenPGP 私钥里面，仅仅导出私密信息
cryptsetup	V:32, I:80	67	cryptsetup(8), …	dm-crypto 块设备加密支持 LUKS 工具
ecryptfs-utils	V:5, I:8	396	ecryptfs(7), …	ecryptfs 堆叠文件系统加密工具
coreutils	V:888, I:999	15719	md5sum(1)	计算与校验 MD5 讯息摘要
coreutils	V:888, I:999	15719	sha1sum(1)	计算与校验 SHA1 讯息摘要
openssl	V:808, I:992	1452	openssl(1ssl)	使用”openssl dgst” (OpenSSL) 计算信息摘要

Table 10.6: 数据安全基础工具列表

参见第 9.8 节的 dm-crypto 和 ecryptfs，它们通过 Linux 内核模块实现了自动数据加密架构。

10.3.1 GnuPG 密钥管理

如下是 GNU 隐私卫士 基本的密钥管理命令。

命令	说明
gpg --gen-key	生成一副新的密钥对
gpg --gen-revoke my_user_ID	生成 my_user_ID 的一份吊销证书
gpg --edit-key user_ID	交互式的编辑密钥，输入”help” 来获得帮助信息
gpg -o file --export	把所有的密钥输出到文件
gpg --import file	从文件导入密钥
gpg --send-keys user_ID	发送 user_ID 的公钥到公钥服务器
gpg --recv-keys user_ID	从公钥服务器下载 user_ID 的公钥
gpg --list-keys user_ID	列出 user_ID 的所有密钥
gpg --list-sigs user_ID	列出 user_ID 的签字
gpg --check-sigs user_ID	检查 user_ID 密钥签字
gpg --fingerprint user_ID	检查 user_ID 的指纹
gpg --refresh-keys	更新本地密钥

Table 10.7: GNU 隐私卫士密钥管理命令的列表

信任码含义.

如下命令上传我的”1DD8D791” 公钥到主流的公钥服务器”hkp://keys.gnupg.net”。

```
$ gpg --keyserver hkp://keys.gnupg.net --send-keys 1DD8D791
```

代码	信任描述
-	没有所有者信任签名/没有计算
e	信任计算失败
q	没有足够的信息用于计算
n	从不信任这个键
m	最低限度的信任
f	完全信任
u	最终信任

Table 10.8: 信任码含义列表

默认良好的公钥服务器在"~/.gnupg/gpg.conf"（旧的位置在"~/.gnupg/options"）文件中设置，此文件包含了以下信息。

```
keyserver hkp://keys.gnupg.net
```

从钥匙服务器获取无名钥匙。

```
$ gpg --list-sigs --with-colons | grep '^sig.*\[User ID not found\]' |\
  cut -d ':' -f 5| sort | uniq | xargs gpg --recv-keys
```

有一个错误在 OpenPGP 公钥服务器 (先前的版本 0.9.6)，会将键中断为 2 个以上的子键。新的 gnupg (>1.2.1-2) 软件包能够处理这些中断的子键。参见 gpg(1) 下的"--repair-pks-subkey-bug" 选项.

10.3.2 在文件上使用 GnuPG

这里有一些在文件上使用 GNU 隐私卫士 命令的例子。

10.3.3 在 Mutt 中使用 GnuPG

增加下面内容到"~/.muttrc"，在自动启动时，避免一个慢的 GnuPG，在索引菜单中按"S" 来允许它使用。

```
macro index S ":toggle pgp_verify_sig\n"
set pgp_verify_sig=no
```

10.3.4 在 Vim 中使用 GnuPG

gnupg 插件可以让你对扩展名为".gpg", ".asc", 和".ppg" 的文件可靠的运行 GnuPG.

```
# aptitude install vim-scripts vim-addon-manager
$ vim-addons install gnupg
```

命令	说明
`gpg -a -s file`	ASCII 封装的签名文件 file.asc
`gpg --armor --sign file`	同上
`gpg --clearsign file`	生成明文签字信息
`gpg --clearsign file|mail foo@example.org`	发送一份明文签字到 `foo@example.org`
`gpg --clearsign --not-dash-escaped patchfile`	明文签名的补丁文件
`gpg --verify file`	验证明文文件
`gpg -o file.sig -b file`	生成一份分离的签字
`gpg -o file.sig --detach-sig file`	同上
`gpg --verify file.sig file`	使用 file.sig 验证文件
`gpg -o crypt_file.gpg -r name -e file`	公钥加密，从文件里面获取名字，生成二进制的 crypt_file.gpg
`gpg -o crypt_file.gpg --recipient name --encrypt file`	同上
`gpg -o crypt_file.asc -a -r name -e file`	公钥加密，从文件中获取名字，生成 ASCII 封装的 crypt_file.asc
`gpg -o crypt_file.gpg -c file`	将文件对称加密到 crypt_file.gpg
`gpg -o crypt_file.gpg --symmetric file`	同上
`gpg -o crypt_file.asc -a -c file`	对称加密，从文件到 ASCII 封装的 crypt_file.asc
`gpg -o file -d crypt_file.gpg -r name`	解密
`gpg -o file --decrypt crypt_file.gpg`	同上

Table 10.9: 在文件上使用的 GNU 隐私卫士的命令列表

10.3.5 MD5 校验和

md5sum(1) 提供了制作摘要文件的一个工具, 它使用 rfc1321 里的方式制作摘要文件.

```
$ md5sum foo bar >baz.md5
$ cat baz.md5
d3b07384d113edec49eaa6238ad5ff00  foo
c157a79031e1c40f85931829bc5fc552  bar
$ md5sum -c baz.md5
foo: OK
bar: OK
```

注意
MD5 校验和的 CPU 计算强度是比 GNU Privacy Guard (GnuPG) 加密签名要少的. 在通常情况下, 只有顶级的摘要文件才需要加密签名来确保数据完整性.

10.4 源代码合并工具

这里有许多源代码合并工具。如下的是我感兴趣的工具。

软件包	流行度	大小	命令	说明
diffutils	V:874, I:987	1574	diff(1)	逐行比较两个文件
diffutils	V:874, I:987	1574	diff3(1)	逐行比较和合并三个文件
vim	V:119, I:395	2799	vimdiff(1)	在 vim 中并排比较两个文件
patch	V:115, I:779	243	patch(1)	给原文件打补丁
dpatch	V:1, I:13	191	dpatch(1)	管理 Debian 软件包的系列补丁
diffstat	V:17, I:179	69	diffstat(1)	通过 diff 生成一个改变柱状图
patchutils	V:19, I:173	223	combinediff(1)	从两个增量补丁创建一个积累补丁
patchutils	V:19, I:173	223	dehtmldiff(1)	从一个 HTML 页面提取出一个 diff
patchutils	V:19, I:173	223	filterdiff(1)	从一个 diff 文件里面提取或者排除 diff 文件
patchutils	V:19, I:173	223	fixcvsdiff(1)	修复由 CVS patch(1) 错误创建的 diff 文件
patchutils	V:19, I:173	223	flipdiff(1)	交换两个补丁的顺序
patchutils	V:19, I:173	223	grepdiff(1)	显示哪些文件是由匹配正则表达式的补丁修改
patchutils	V:19, I:173	223	interdiff(1)	显示在两个统一格式 diff 文件（基于同一个文件的两个不同 diff 文件）之间的差异
patchutils	V:19, I:173	223	lsdiff(1)	显示哪些文件由补丁修改
patchutils	V:19, I:173	223	recountdiff(1)	重新计算通用内容 diff 文件的数量和偏移
patchutils	V:19, I:173	223	rediff(1)	修复手工编辑 diff 文件的数量和偏移
patchutils	V:19, I:173	223	splitdiff(1)	隔离出增量补丁
patchutils	V:19, I:173	223	unwrapdiff(1)	识别已经被分词的补丁
wiggle	V:0, I:0	174	wiggle(1)	应用被拒绝的补丁
quilt	V:3, I:38	785	quilt(1)	管理系列补丁
meld	V:17, I:42	2942	meld(1)	比较和移植文件 (GTK)
dirdiff	V:0, I:2	161	dirdiff(1)	显示目录树之间的不同并移植改变
docdiff	V:0, I:0	573	docdiff(1)	逐词逐字地比较两个文件
imediff	V:0, I:0	157	imediff(1)	全屏交互式两路/三路合并工具
makepatch	V:0, I:0	102	makepatch(1)	生成扩展补丁文件
makepatch	V:0, I:0	102	applypatch(1)	应用扩展补丁文件
wdiff	V:8, I:77	644	wdiff(1)	在文本文件中，显示单词的不同

Table 10.10: 源代码合并工具列表

10.4.1 从源代码文件导出差异

下面的操作，导出两个源文件的不同，并根据文件的位置，创建通用 diff 文件"`file.patch0`" 或"`file.patch1`".

```
$ diff -u file.old file.new > file.patch0
$ diff -u old/file new/file > file.patch1
```

10.4.2 源代码文件移植更新

diff 文件 (通常被叫作 patch 补丁文件)，用于发送一个程序更新。通过下面的方式，接收到的部分，应用这个更新到其它文件。

```
$ patch -p0 file < file.patch0
$ patch -p1 file < file.patch1
```

10.4.3 通过三方移植进行更新

如果一个源代码，你有三个版本，你可以通过下面的方式，使用 diff3(1) 高效执行三方移植。

```
$ diff3 -m file.mine file.old file.yours > file
```

10.5 版本控制系统

如下是 Debian 系统上可用的版本控制系统 (VCS) 的摘要。

注意
如果是刚接触版本控制系统，你应该从 **git** 入门，git 人气日益高涨。

VCS 有时被认为是修订控制系统 (RCS), 或者是软件配置管理程序 (SCM)。

像 Git 这样的分布式 VCS 是现在正在使用的工具。参加那些已经存在的开源软件的开发活动，掌握 CVS 和 Subversion 仍然是有用的。

通过 Debian Salsa 服务，Debian 能够提供免费的 Git 服务。在 https://wiki.debian.org/Salsa 能找到它的说明文档。

小心
Debian 已经停止了其旧有的 alioth 服务，旧的 alioth 服务数据可以在 alioth-archive 站点上以 tar 压缩包的形式获取。

这里有一些关于创建共享访问 VCS 归档的基础知识。

- 使用"`umask 002`" (参见第 1.2.4 节)
- 使得所有的 VCS 归档文件属于一个相关的组
- 能够在所有的 VCS 归档目录设置组 ID (类似 BSD 的文件创建方案，参见第 1.2.3 节)
- 使得属于这个组的用户能够共享 VCS 归档

软件包	流行度	大小	工具	VCS 类型	描述
cssc	V:0, I:2	2035	CSSC	本地	Unix SCCS (过时) 的克隆
rcs	V:3, I:21	555	RCS	本地	” 比 Unix SCCS 做的好”
cvs	V:5, I:49	4596	CVS	远程	以前的远程 VCS 标准
subversion	V:27, I:131	4809	Subversion	远程	” 比 CVS 做的好 “，远程 VCS 的新标准
git	V:301, I:458	35266	Git	分布式	用 C 写的快速 DVCS (被 Linux 内核和其他项目使用)
mercurial	V:11, I:56	913	Mercurial	分布式	mercurial 主要是用 Python 写的还有一部分是 C 写的
bzr	V:3, I:20	74	Bazaar	分布式	受 tla 启发并且是用 Python 写的 DVCS (被 Ubuntu 使用)
darcs	V:0, I:8	27950	Darcs	分布式	有智能代数补丁的 DVCS (慢)
tla	V:0, I:6	1011	GNU arch	分布式	主要由 Tom Lord 写的 DVCS (成为历史的)
monotone	V:0, I:0	5815	Monotone	分布式	用 C++ 写的 DVCS
tkcvs	V:0, I:1	1498	CVS, …	远程	VCS (CVS,Subversion,RCS) 存储库树的图形界面显示
gitk	V:8, I:47	1539	Git	分布式	VCS (Git) 存储库树的图形界面显示

Table 10.11: 版本控制系统工具列表

10.5.1 VCS 命令的比较

这里有原生 VCS 命令的简单比较来提供大图概要。典型的命令序列需要选项和参数。

小心
从命令行通过”git-xyz” 直接调用 git 子命令的方式，从 2006 年早期开始就被取消。

提示
如果有一个可执行文件 git-foo 在路径环境变量 $PATH 里面, 在命令行输入没有中划线的”git foo”，则将调用 git-foo. 这是 git 命令的一个特性.

提示
例如 tkcvs(1) 和 gitk(1) 这样的图形界面工具有助于追踪文件的修改历史。许多公共的归档提供的用于浏览它们的存储库的 web 界面同样是很有用的。

提示
Git 能够直接在不同的 VCS 仓库上工作，比如说 CVS 和 Subversion 提供的仓库, 通过 git-cvs 和 git-svn 软件包提供本地仓库的本地改变. 参见 用于 CVS 用户的 git 和第 10.6.4 节.

提示
Git 中的有些命令在 CVS 和 Subversion 中并没有对应的命令：”fetch”,”rebase”,”cherry-pick”, …

Git	CVS	Subversion	功能
`git init`	`cvs init`	`svn create`	创建 (本地) 存储库
-	`cvs login`	-	登录远程存储库
`git clone`	`cvs co`	`svn co`	签出远程存储库到本地工作目录树
`git pull`	`cvs up`	`svn up`	通过合并远程存储库来更新工作目录树
`git add .`	`cvs add`	`svn add`	把工作目录树中的文件添加到 VCS
`git rm`	`cvs rm`	`svn rm`	从 VCS 中移除工作目录树中的文件
-	`cvs ci`	`svn ci`	提交改变到远程存储库
`git commit -a`	-	-	提交改变到本地存储库
`git push`	-	-	通过本地存储库来更新远程存储库
`git status`	`cvs status`	`svn status`	从 VCS 中显示工作目录树的状态
`git diff`	`cvs diff`	`svn diff`	比较 < 参考存储库 > 和 < 工作目录树 > 的差异
`git repack -a -d; git prune`	-	-	重新打包本地仓库到一个单独的包
`gitk`	`tkcvs`	`tkcvs`	VCS 存储库树的图形界面显示

Table 10.12: 本地 VCS 命令比较

10.6 Git

Git 可以用来做本地和远程源代码管理的任何事情。这意味着，你能够在本地记录源代码修改，而不是必须要和远程仓库有网络连接。

10.6.1 配置 Git 客户端

你可以在”`~/.gitconfig`” 里面设置几个 Git 接下来需要使用的全局配置，比如说你的名字和电子邮件地址。

```
$ git config --global user.name ”姓名”
$ git config --global user.email 电子邮件地址
```

如果你习惯使用 CVS 或 Subversion 命令，你也许希望设置如下几个命令别名。

```
$ git config --global alias.ci ”commit -a”
$ git config --global alias.co checkout
```

你能够通过如下方式检查你的全局配置：

```
$ git config --global --list
```

10.6.2 Git 参考

参见下面内容。

- man 手册：git(1) (/usr/share/doc/git-doc/git.html)
- Git 用户手册 (/usr/share/doc/git-doc/user-manual.html)
- git 介绍教程 (/usr/share/doc/git-doc/gittutorial.html)
- git 介绍教程：第二部 (/usr/share/doc/git-doc/gittutorial-2.html)
- GIT 每一天 20 个左右的命令 (/usr/share/doc/git-doc/everyday.html)
- CVS 用户用 git (/usr/share/doc/git-doc/gitcvs-migration.html)
  - 描述了怎样搭建服务，以及如何把老的数据从 CVS 迁移到 Git。
- 其它在互联网上存在的 git 资源
  - Git - SVN 碰撞课程
  - Git 魔术 (/usr/share/doc/gitmagic/html/index.html)

git-gui(1) 和 gitk(1) 命令使 Git 变得非常容易使用。

警告
不要使用带空格的标签字符串。即使一些工具，如 gitk(1) 允许你使用它，但会阻碍其它 git 命令。

10.6.3 Git 命令

即使你的上游使用不同的版本控制系统，使用 git(1) 作为本地活动的版本控制系统，仍然是一个好的主意，因为 git 可以让你在没有上游网络连接的情况下，管理你的本地源代码树拷贝。这里有一些 git(1) 使用的包和命令。

提示
在 git(1) 下, 你在本地分支下进行了许多提交，稍后你可以使用"git rebase -i master" 之类的命令来重新组织改变历史。这可以使你制作一个干净的改变历史。参见 git-rebase(1) 和 git-cherry-pick(1).

提示
当你想要回到一个干净的工作目录，并且不丢失工作目录当前的状态，你可以使用"git stash". 参见 git-stash(1).

10.6.4 用于 Subversion 仓库的 Git

你可以把一个在"svn+ssh://svn.example.org/project/module/trunk" 的 Subversion 仓库检出到一个本地的 Git 仓库，使用"./dest" 目录，并把修改提交回 Subversion 仓库。例如：

```
$ git svn clone -s -rHEAD svn+ssh://svn.example.org/project dest
$ cd dest
... 进行修改
$ git commit -a
... 继续在本地用 git 工作
$ git svn dcommit
```

软件包	流行度	大小	命令	说明
git-doc	I:18	11118	N/A	Git 官方文档
gitmagic	I:1	719	N/A	”Git 魔术”，易于理解的 Git 手册
git	V:301, I:458	35266	git(7)	Git 快速、可扩展、分布式的版本控制系统
gitk	V:8, I:47	1539	gitk(1)	有历史功能的 Git 图形仓库浏览器
git-gui	V:2, I:27	2266	git-gui(1)	Git 图形界面（无历史功能）
git-svn	V:2, I:26	1037	git-svnimport(1)	从 Subversion 导出数据，导入到 Git
git-svn	V:2, I:26	1037	git-svn(1)	在 Subversion 和 Git 之间提供双向操作
git-cvs	V:0, I:12	1172	git-cvsimport(1)	从 CVS 导出数据，导入到 Git
git-cvs	V:0, I:12	1172	git-cvsexportcommit(1)	从 Git 中检出一个 CVS 的提交
git-cvs	V:0, I:12	1172	git-cvsserver(1)	Git 的 CVS 服务模拟器
git-email	V:0, I:13	860	git-send-email(1)	从 Git 用电子邮件发送收集到的补丁
stgit	V:0, I:0	1535	stg(1)	封装的 git (Python)
git-buildpackage	V:2, I:12	3928	git-buildpackage(1)	用 Git 自动制作 Debian 包
guilt	V:0, I:0	146	guilt(7)	封装的 git (SH/AWK/SED/…)

Table 10.13: git 相关包和命令列表

提示
使用”`-rHEAD`” 能够避免克隆从 Subversion 仓库来的整个历史内容。

10.6.5 记录配置历史的 Git

你可以使用 Git 工具来手工记录按时间先后顺序的配置历史。这里是一个例子，让你练习记录”`/etc/apt/`” 内容。

```
$ cd /etc/apt/
$ sudo git init
$ sudo chmod 700 .git
$ sudo git add .
$ sudo git commit -a
```

提交配置，描述此次提交。

对配置文件进行修改。

```
$ cd /etc/apt/
$ sudo git commit -a
```

提交配置，说明提交，继续你的工作。

```
$ cd /etc/apt/
$ sudo gitk --all
```

你有全部的配置历史。

注意
sudo(8) 是需要用于配置数据文件，任意文件权限的情况。对于普通用户的配置数据，你需要省略 sudo。

注意
在上面例子里的"chmod 700 .git" 命令，是用来保护文档数据不被未经授权的读访问。

提示
要更加完整的建立配置历史记录，请参阅 etckeeper 包：第 9.2.10 节。

10.7 CVS

CVS 是一个古老的版本控制系统，它的出现早于 Subversion 和 Git。

小心
下面例子里给出的 CVS 相关的链接许多已不存在。

参见下面内容。

- cvs(1)
- "/usr/share/doc/cvs/html-cvsclient"
- "/usr/share/doc/cvs/html-info"
- "/usr/share/doc/cvsbook"
- "info cvs"

10.7.1 CVS 存储库的配置

如下的配置将只允许"src" 组的成员向 CVS 存储库提交修改，只允许"staff" 组的成员管理 CVS，这样可以减少出错的机会。

```
# cd /var/lib; umask 002; mkdir cvs
# export CVSROOT=/srv/cvs/project
# cd $CVSROOT
# chown root:src .
# chmod 2775 .
# cvs -d $CVSROOT init
# cd CVSROOT
# chown -R root:staff .
# chmod 2775 .
# touch val-tags
# chmod 664 history val-tags
# chown root:src history val-tags
```

提示
你可以改变"$CVSROOT" 目录为"root:staff" 并把它的权限设置为"3775"，这样就可以限制创建新的项目。

10.7.2 本地访问 CVS

默认的 CVS 存储库由”$CVSROOT” 指定。如下将建立用于本地访问的”$CVSROOT”。

```
$ export CVSROOT=/srv/cvs/project
```

10.7.3 使用 pserver 远程访问 CVS

许多公共 CVS 服务器可以通过 pserver 服务用”anonymous” 账户远程只读访问。例如，Debian 网站的内容曾经使用名为 webwml project 的仓库经由 Debian alioth 服务的 CVS 服务进行维护。如下命令曾被用来建立用于远程访问该旧 CVS 仓库的”$CVSROOT”。

```
$ export CVSROOT=:pserver:anonymous@anonscm.debian.org:/cvs/webwml
$ cvs login
```

注意
因为 pserver 容易被窃听攻击并且是不安全的，所以写访问通常是被服务器管理员禁用的。

10.7.4 使用 ssh 远程访问 CVS

如下所示的命令曾被用来配置“$CVS_RSH”和“$CVSROOT”变量，以此实现使用 SSH 远程访问旧的 Debian webwml 项目所使用的 CVS 仓库。

```
$ export CVS_RSH=ssh
$ export CVSROOT=:ext:account@cvs.alioth.debian.org:/cvs/webwml
```

你也可以使用 SSH 的公钥认证，这能够去除远程密码提示。

10.7.5 往 CVS 导入新的源

按如下所示创建”~/path/to/module1” 路径下的新的本地源目录树。

```
$ mkdir -p ~/path/to/module1; cd ~/path/to/module1
```

把文件添加到”~/path/to/module1” 下的新的本地源目录树。
使用如下的参数把文件导入到 CVS。

- 模块名: ”module1”
- 提供商标签: ”Main-branch”（用于整个分支的标签）
- 发布标签: ”Release-initial”（用于特定发布版本的标签）

```
$ cd ~/path/to/module1
$ cvs import -m ”Start module1” module1 Main-branch Release-initial
$ rm -Rf . # optional
```

10.7.6 CVS 存储库中的文件权限

CVS 不会覆盖当前的存储库文件，只是用另外的文件来替代它。因此，存储库目录的写权限是很重要的。存储库位于”/srv/cvs/project” 的”module1”，对于其下的每一个新模块而言，如果需要的话运行如下所示的来确保这种情况。

```
# cd /srv/cvs/project
# chown -R root:src module1
# chmod -R ug+rwX   module1
# chmod    2775     module1
```

10.7.7 CVS 工作流

这里有一个 CVS 典型工作流的例子。

按如下所示查看”$CVSROOT” 所指的 CVS 项目上所有可用的模块。

```
$ cvs rls
CVSROOT
module1
module2
...
```

按如下所示签出”module1” 到默认的目录”./module1”。

```
$ cd ~/path/to
$ cvs co module1
$ cd module1
```

按需修改里面的内容。

通过如下所示的命令来检查改变，其作用相当于使用”diff -u [repository] [local]”。

```
$ cvs diff -u
```

你发现自己改坏了”file_to_undo” 文件，而其他的文件都是好的。

按如下所示用 CVS 中的原始副本覆盖”file_to_undo” 文件。

```
$ cvs up -C file_to_undo
```

按如下所示把更新了的本地源目录树保存到 CVS。

```
$ cvs ci -m ”Describe change”
```

按如下创建并添加”file_to_add” 文件到 CVS。

```
$ vi file_to_add
$ cvs add file_to_add
$ cvs ci -m ”Added file_to_add”
```

按如下所示合并 CVS 中的最新版本。

```
$ cvs up -d
```

当心以”C filename” 开头的行，这意味着冲突的改变。

查看”.#filename.version” 中未经修改的代码。

查找文件中的”<<<<<<<” 和”>>>>>>>” 来获得冲突的改变的信息。

按需更改文件来解决冲突。

按如下所示添加一个发布标签”Release-1”。

```
$ cvs ci -m ”last commit for Release-1”
$ cvs tag Release-1
```

继续编辑文件。

按如下所示移除发布分支”Release-1”。

```
$ cvs tag -d Release-1
```

按如下所示把改变签入到 CVS。

```
$ cvs ci -m ”real last commit for Release-1”
```

按如下所示给已经更新了的 CVS 主干中的 HEAD 重新添加”Release-1” 发布标签。

```
$ cvs tag Release-1
```

按如下所示从”Release-initial” 标签指向的初始版本中创建一个带有粘性标签的”Release-initial-bugfixes” 分支，并把它签出到”~/path/to/old” 目录。

```
$ cvs rtag -b -r Release-initial Release-initial-bugfixes module1
$ cd ~/path/to
$ cvs co -r Release-initial-bugfixes -d old module1
$ cd old
```

提示
使用”-D 2005-12-20” (ISO 8601 日期格式) 而不是”-r Release-initial” 来指定某个特定日期作为分支点。

在基于原始版本的有”Release-initial-bugfixes” 粘性标签的本地源目录树上工作。

独自在”Release-initial-bugfixes” 分支上工作... 直到有其他人加入到此分支。

当要创建新的目录时，按如下所示同步其他人在此分支上对文件所做的修改。

```
$ cvs up -d
```

按需更改文件来解决冲突。

按如下所示把改变签入到 CVS。

```
$ cvs ci -m "checked into this branch"
```

按如下所示更新本地目录树为主干的最新版本，同时移除粘性标签 ("-A") 并且不使用关键字扩展 ("-kk")。

```
$ cvs up -d -kk -A
```

按如下所示通过合并"Release-initial-bugfixes" 分支并且不使用关键字扩展的方式来更新本地目录树 (内容为主干中的最新版本)。

```
$ cvs up -d -kk -j Release-initial-bugfixes
```

用编辑器来解决冲突。

按如下所示把改变签入到 CVS。

```
$ cvs ci -m "merged Release-initial-bugfixes"
```

按如下所示创建归档。

```
$ cd ..
$ mv old old-module1-bugfixes
$ tar -cvzf old-module1-bugfixes.tar.gz old-module1-bugfixes
$ rm -rf old-module1-bugfixes
```

提示
"cvs up" 命令能够使用"-d" 选项来创建新目录并且可以使用"-P" 选项来删除空目录。

提示
你可以通过形如"cvs co module1/subdir" 这样的列出其名的方式，来签出"module1" 的一个子目录。

选项	说明
-n	测试，没有影响
-t	显示 cvs 活动步骤的信息

Table 10.14: 值得注意的 CVS 命令选项 (用作 cvs(1) 的第一个选项)

10.7.8 CVS 中最新的文件

按如下所示使用"tomorrow" 选项，就能得到 CVS 中的最新文件。

```
$ cvs ex -D tomorrow module_name
```

10.7.9 CVS 的管理

按如下所示往 CVS 项目 (本地服务器) 里添加”mx” 模块别名。

```
$ export CVSROOT=/srv/cvs/project
$ cvs co CVSROOT/modules
$ cd CVSROOT
$ echo ”mx -a module1” >>modules
$ cvs ci -m ”Now mx is an alias for module1”
$ cvs release -d .
```

按如下所示，你可以从 CVS 中签出”module1” (别名为: ”mx”) 到”new” 目录。

```
$ cvs co -d new mx
$ cd new
```

注意
为了执行上述步骤，你应当有合适的文件权限。

10.7.10 用于 CVS 签出时的可执行位

当你从 CVS 中签出文件时，它们的可执行权限是保留的。

当你发现在检出的形如”filename” 这样的文件中，可执行权限有问题时，按如下所示在相应的 CVS 存储库中改变文件的权限来解决这个问题。

```
# chmod ugo-x filename
```

10.8 Subversion

Subversion 是在 Git 之前出现的旧的版本控制系统，但它出现在 CVS 之后。它缺少 CVS 和 Git 中的标签和分支功能。

你需要安装 subversion，libapache2-mod-svn 和 subversion-tools 软件包来搭建 Subversion 服务器。

10.8.1 Subversion 存储库的配置

subversion 软件包通常不会自动建立存储库,所以你必须手动搭建它。存储库可能的位置是在”/srv/svn/project”。

按如下所示建立目录。

```
# mkdir -p        /srv/svn/project
```

按如下所示建立存储库数据库。

```
# svnadmin create /srv/svn/project
```

10.8.2 通过 Apache2 服务器访问 Subversion

如果只是用 Apache2 服务器访问 Subversion 存储库，你只需按如下所示的使存储库只是对于 WWW 服务器是可写的。

```
# chown -R www-data:www-data /srv/svn/project
```

在”/etc/apache2/mods-available/dav_svn.conf” 中添加 (或取消注释) 如下所示的来允许通过用户认证访问存储库。

```
<Location /project>
  DAV svn
  SVNPath /srv/svn/project
  AuthType Basic
  AuthName ”Subversion repository”
  AuthUserFile /etc/subversion/passwd
<LimitExcept GET PROPFIND OPTIONS REPORT>
    Require valid-user
</LimitExcept>
</Location>
```

用如下所示的命令创建用户认证文件。

```
# htpasswd2 -c /etc/subversion/passwd some-username
```

重启 Apache2。

通过”http://localhost/project” 和”http://example.com/project” URL 来访问 svn(1) 中的 Subversion 存储库 (假设你的 web 服务器的 URL 为”http://example.com/”)。

10.8.3 按组本地访问 Subversion

如下所示将建立用户组，例如 project，可以本地访问的 Subversion 存储库。

```
# chmod  2775     /srv/svn/project
# chown -R root:src /srv/svn/project
# chmod -R ug+rwX   /srv/svn/project
```

属于 project 组的本地用户可以访问在”file:///localhost/srv/svn/project” 或 file:///srv/svn/project” 下 svn(1) 中的新 Subversion 存储库。你必须在”umask 002” 下运行诸如 svn, svnserve, svnlook 和 svnadmin 命令，来确保用户组可以访问。

10.8.4 通过 SSH 远程访问 Subversion

用户组可以访问的 Subversion 存储库的 URL 为”example.com:/srv/svn/project”。至于 SSH 访问，你能够在 svn(1) 中的”svn+ssh://example.com:/srv/svn/project” URL 访问它。

10.8.5 Subversion 目录结构

对于 Subversion 来说，许多项目使用类似如下的目录树来弥补它的分支和标签的不足。

```
----- module1
  |   |-- branches
  |   |-- tags
  |   |   |-- release-1.0
  |   |   `-- release-2.0
  |   |
  |   `-- trunk
  |       |-- file1
  |       |-- file2
  |       `-- file3
  |
  `-- module2
```

提示
你必须使用"`svn copy …`" 命令来标记分支和标签。这确保 Subversion 正确的记录文件的修改历史同时节省存储空间。

10.8.6 往 Subversion 里导入一个新的源

按如下所示创建"`~/path/to/module1`" 路径下的新的本地源目录树。

```
$ mkdir -p ~/path/to/module1; cd ~/path/to/module1
```

把文件添加到"`~/path/to/module1`" 下的新的本地源目录树。
把它导入到 Subversion 的时候带有以下的参数。

- 模块名: "`module1`"
- Subversion 位置 URL: "`file:///srv/svn/project`"
- Subversion 目录: "`module1/trunk`"
- Subversion 标签: "`module1/tags/Release-initial`"

```
$ cd ~/path/to/module1
$ svn import file:///srv/svn/project/module1/trunk -m "Start module1"
$ svn cp file:///srv/svn/project/module1/trunk file:///srv/svn/project/module1/tags/Release ↩
    -initial
```

或者，如下所示。

```
$ svn import ~/path/to/module1 file:///srv/svn/project/module1/trunk -m "Start module1"
$ svn cp file:///srv/svn/project/module1/trunk file:///srv/svn/project/module1/tags/Release ↩
    -initial
```

提示
你能够用像"`http://…`" 和"`svn+ssh://...`" 这样格式的 URL 来替代"`file:///…`" URL。

10.8.7 Subversion 工作流

这里给出使用 Subversion 及其原生客户端的典型工作流示例。

> 提示
> `git-svn` 软件包提供的客户端命令，可以作为使用 `git` 命令的 Subversion 工作流的一个另外选择。参见第 10.6.4 节.

查看如下所示的 URL "`file:///srv/svn/project`" 指向的 Subversion 项目上所有可用的模块。

```
$ svn list file:///srv/svn/project
module1
module2
...
```

按如下所示的检出"`module1/trunk`" 到"`module1`" 目录。

```
$ cd ~/path/to
$ svn co file:///srv/svn/project/module1/trunk module1
$ cd module1
```

按需修改里面的内容。

通过如下所示的命令来检查改变，其作用相当于使用"`diff -u [repository] [local]`"。

```
$ svn diff
```

你发现自己改坏了"`file_to_undo`" 文件，而其他的文件都是好的。

按如下所示的用 Subversion 中的干净副本来覆盖"`file_to_undo`" 文件。

```
$ svn revert file_to_undo
```

按如下所示的把已经更新了的本地源目录树保存到 Subversion。

```
$ svn ci -m "Describe change"
```

按如下所示的创建"`file_to_add`" 文件并把它添加到 Subversion。

```
$ vi file_to_add
$ svn add file_to_add
$ svn ci -m "Added file_to_add"
```

按如下所示更新工作拷贝到 Subversion 中的最新版本。

```
$ svn up
```

当心以”C filename” 开头的行，这意味着冲突的改变。

查看文件中未经修改的代码，例如”filename.r6”, ”filename.r9” 和”filename.mine” 文件。

查找文件中的”<<<<<<<” 和”>>>>>>>” 来获得冲突的改变的信息。

按需更改文件来解决冲突。

按如下所示添加一个发布标签”Release-1”。

```
$ svn ci -m ”last commit for Release-1”
$ svn cp file:///srv/svn/project/module1/trunk file:///srv/svn/project/module1/tags/Release ↩
    -1
```

继续编辑文件。

按如下所示移除发布分支”Release-1”。

```
$ svn rm file:///srv/svn/project/module1/tags/Release-1
```

按如下所示把改变签入到 Subversion。

```
$ svn ci -m ”real last commit for Release-1”
```

按如下所示在最新的 Subversion 主干的基础上重新添加发布分支”Release-1”。

```
$ svn cp file:///srv/svn/project/module1/trunk file:///srv/svn/project/module1/tags/Release ↩
    -1
```

按如下所示在”module1/tags/Release-initial” 路径指定的最初版本的基础上再创建一个路径为”module1/branche
的分支，并把它签出到”~/path/to/old” 目录。

```
$ svn cp file:///srv/svn/project/module1/tags/Release-initial file:///srv/svn/project/ ↩
    module1/branches/Release-initial-bugfixes
$ cd ~/path/to
$ svn co file:///srv/svn/project/module1/branches/Release-initial-bugfixes old
$ cd old
```

提示
使用”module1/trunk@{2005-12-20}” (ISO 8601 日期格式) 而不是”module1/tags/Release-initial” 来指定分支创建时候的日期。

在基于原始版本的”Release-initial-bugfixes” 分支的本地源目录树上工作。

独自在”Release-initial-bugfixes” 分支上工作... 直到有其他人加入到此分支。

按如下所示同步其他人在此分支上改动的文件。

```
$ svn up
```

按需更改文件来解决冲突。

按如下所示把改变签入到 Subversion。

```
$ svn ci -m "checked into this branch"
```

按如下所示更新本地目录树为主干的最新版本。

```
$ svn switch file:///srv/svn/project/module1/trunk
```

按如下所示通过合并"Release-initial-bugfixes" 分支的方式来更新本地目录树 (内容为主干的最新版本)。

```
$ svn merge file:///srv/svn/project/module1/branches/Release-initial-bugfixes
```

用编辑器来解决冲突。

按如下所示把改变签入到 Subversion。

```
$ svn ci -m "merged Release-initial-bugfixes"
```

按如下所示创建归档。

```
$ cd ..
$ mv old old-module1-bugfixes
$ tar -cvzf old-module1-bugfixes.tar.gz old-module1-bugfixes
$ rm -rf old-module1-bugfixes
```

提示
你能够用像"http://…" 和"svn+ssh://..." 这样格式的 URL 来替代"file:///…" URL。

提示
通过"svn co file:///srv/svn/project/module1/trunk/subdir module1/subdir" 等命令，你可以只签出"module1" 的一个子目录。

选项	说明
--dry-run	测试，没有影响
-v	显示 svn 活动的详细信息

Table 10.15: 值得注意的 Subversion 命令选项 (使用时作为 svn(1) 的第一个参数)

Chapter 11

数据转换

下面是关于 Debian 系统上可用的格式转化工具及其相关提示的信息。

基于标准的工具，是非常好用的，但支持的专有数据格式有限.

11.1 文本数据转换工具

如下是文本数据转换工具。

软件包	流行度	大小	关键词	说明
libc6	V:937, I:999	12333	字符集	使用 iconv(1) 的不同语言环境（locale）之间的文本编码转换器 (基础的)
recode	V:4, I:28	608	字符集 + 换行	不同语言环境（locale）之间的文本编码转换器（多功能的，更多别名和特性）
konwert	V:1, I:57	123	字符集	不同语言环境（locale）之间的文本编码转换器（高档的）
nkf	V:0, I:11	357	字符集	日语字符集翻译
tcs	V:0, I:0	518	字符集	字符集翻译
unaccent	V:0, I:0	29	字符集	代替重音字符，使用和它们相当的非重音字符
tofrodos	V:2, I:30	55	换行	在 DOS 和 Unix 之间的文本格式转换: fromdos(1) 和 todos(1)
macutils	V:0, I:1	298	换行	在 Macintosh 和 Unix 之间的文本格式转换: frommac(1) 和 tomac(1)

Table 11.1: 文本数据转化工具列表

11.1.1 用 iconv 命令来转换文本文件

提示
iconv(1) 是 libc6 软件包的一部分并且它可以在类 Unix 的系统上转换字符的编码。

你能够通过如下的命令用 iconv(1) 来转换文本文件的编码。

```
$ iconv -f encoding1 -t encoding2 input.txt >output.txt
```

编码值	用法
ASCII	美国信息交换标准代码，7 位代码不带重音符号
UTF-8	用于所有现代操作系统的多语言标准
ISO-8859-1	旧的西欧语言标准，ASCII + 重音符号
ISO-8859-2	旧的东欧语言标准，ASCII + 重音符号
ISO-8859-15	旧的带有欧元符号的西欧语言标准 (ISO-8859-1)
CP850	code page 850，用于西欧语言的微软 DOS 的带有图形的字符，ISO-8859-1 的变体
CP932	code page 932, 日语 Microsoft Windows 的 Shift-JIS 变体
CP936	code page 936，用于简体中文的微软操作系统风格的 GB2312，GBK 或者 GB18030 的变体
CP949	code page 949，用于韩语的微软操作系统风格的 EUC-KR 或者 Unified Hangul Code 的变体
CP950	code page 950，用于繁体中文的微软操作系统风格的 Big5 的变体
CP1251	code page 1251，用于西里尔字母的微软操作系统风格的编码
CP1252	code page 1252，用于西欧语言的微软操作系统风格的 ISO-8859-15 的变体
KOI8-R	用于西里尔字母的旧俄语 UNIX 标准
ISO-2022-JP	日文邮件的标准编码，只使用 7 位字节
eucJP	老的日文 UNIX 标准的 8 位字节，和 Shift-JIS 完全不同
Shift-JIS	日文 JIS X 0208 附录 1 标准 (参见 CP932)

Table 11.2: 编码值和用法的列表

编码值是大小写不敏感的，且会在匹配时忽略“-”和“_”。可以使用“`iconv -l`”命令检查支持的编码。

注意
一些编码只支持数据转换，它不能作为语言环境的值 (第 8.4.1 节)。

像 ASCII 和 ISO-8859 这样适用于单字节的字符集，字符编码和字符集几乎指的是同一件事情。

对于多字符的字符集，比如说，用于日文的 JIS X 0213，或用于差不多所有语言的 Universal Character Set (UCS, Unicode, ISO-10646-1) , 有多种编码方案来序列化它们的字节数据。

- 日文的 EUC 和 ISO/IEC 2022 (也被称为 JIS X 0202)
- Unicode 的 UTF-8、UTF-16/UCS-2 和 UTF-32/UCS-4 编码

对于以上这些，字符集和字符编码之间有着明显的区别。

对某些计算机厂家而言，code page 是作为字符编码表的同义词来使用。

注意
请注意，大部分编码系统共享 ASCII 的 7 位字符的同样编码，但也有一些列外。如果你从通常所说的 shift-JIS 编码格式，转化老的日文 C 语言程序和 URL 数据，到 UTF-8 格式, 你需要使用”`CP932`” 作为编码名来代替”`shift-JIS`” 来得到期望的结果：`0x5C` → ”\” 和 `0x7E` → ”~”. 否则，这些将被转化为错误的字符。

提示
recode(1) 也可能被使用并且不仅仅是 iconv(1)，fromdos(1)，todos(1)，frommac(1) 和 tomac(1) 功能的结合。想要获得更多信息，请参见”`info recode`”。

11.1.2 用 iconv 检查文件是不是 UTF-8 编码

你能够通过如下命令用 iconv(1) 来检查一个文本文件是不是用 UTF-8 编码的。

```
$ iconv -f utf8 -t utf8 input.txt >/dev/null || echo "non-UTF-8 found"
```

提示
在上面的例子中使用"`--verbose`" 参数来找到第一个 non-UTF-8 字符。

11.1.3 使用 iconv 转换文件名

这里是一个示例脚步，在同一目录下，将在老的操作系统系统下创建的文件名编码，转换为现代 UTF-8.

```
#!/bin/sh
ENCDN=iso-8859-1
for x in *;
 do
 mv "$x" "$(echo "$x" | iconv -f $ENCDN -t utf-8)"
done
```

"`$ENCDN`" 变量定义了在老的操作系统下，文件名使用的原始编码，见表 11.2.

对于更加复杂的情况，请使用适当的编码作为 mount(8) 的选项 (参见第 8.4.6 节) 来挂载包含有这样文件名的文件系统 (比如说，磁盘上的一个分区)，使用"`cp -a`" 命令来拷贝它的整个内容到另外一个使用 UTF-8 挂载的文件系统上。

11.1.4 换行符转换

文本文件的格式，特别是行尾 (换行符) 编码，有平台独立性。

平台	换行符编码	控制码	十进制	**16** 进制
Debian (unix)	LF	`^J`	10	0A
MSDOS 和 Windows	CR-LF	`^M^J`	13 10	0D 0A
苹果的 Macintosh	CR	`^M`	13	0D

Table 11.3: 不同平台的换行符样式列表

换行符转换程序, fromdos(1), todos(1), frommac(1), 和 tomac(1), 是相当方便. recode(1) 也是有用的。

注意
在 Debian 系统上的一些数据，如 `python-moinmoin` 软件包的 wiki 页面数据，使用 MSDOS 式样的 CR-LF 作为换行符编码。所以，上面的规则仅仅是一个通用规则。

注意
大部分编辑器 (比如：`vim`, `emacs`, `gedit`, ···) 能够透明处理 MSDOS 式样的换行符文件。

提示
对于混合 MSDOS 和 Unix 式样的文件，统一到 MSDOS 换行符式样，使用"`sed -e '/\r$/!s/$/\r/'`" 代替 todos(1) 比较好。(例如，在使用 diff3(1) 移植两个 MSDOS 式样的文件后.) 这是因为 `todos` 给所有的行增加 CR.

11.1.5 TAB 转换

这里有一些转换 TAB 代码的专业工具。

功能	**bsdmainutils**	**coreutils**
把制表符扩展成空格	"col -x"	expand
将空格转换为制表符 (unexpand)	"col -h"	unexpand

Table 11.4: bsdmainutils 和 coreutils 包中的用于转换 TAB 的命令列表

indent 包中的 indent(1) 命令能够重新格式化 C 程序中的空格。

例如 vim 和 emacs 这样的编辑软件可以被用来扩展 TAB。就拿 vim 来说，你能够按顺序输入":set expandtab" 和":%retab" 命令来扩展 TAB。你也可以按顺序输入 :%set noexpandtab" 和":%retab" 命令来复原。

11.1.6 带有自动转换功能的编辑器

像 vim 这样的现代智能编辑器软件是相当聪明的并且能够处理任何编码系统以及任何文件格式。你应该在支持 UTF-8 编码的控制台上并在 UTF-8 环境下使用这些编辑器来获得最好的兼容性。

以 latin1 (iso-8859-1) 编码存储的旧西欧语言的 Unix 文本文件，"u-file.txt"，能通过如下所示的用 vim 轻易的编辑。

```
$ vim u-file.txt
```

这是可能的因为 vim 的文件编码自动检测机制先假定文件是 UTF-8 编码，如果失败了，则假定它是 latin1 编码。

以 latin2 (iso-8859-2) 编码存储的旧波兰语的 Unix 文本文件，"pu-file.txt"，能通过如下所示的用 vim 编辑。

```
$ vim '+e ++enc=latin2 pu-file.txt'
```

以 eucJP 编码存储的旧日语的 Unix 文本文件，"ju-file.txt"，能通过如下所示的用 vim 编辑。

```
$ vim '+e ++enc=eucJP ju-file.txt'
```

以所谓的 shift-JIS 编码 (更确切的说法是：CP932) 存储的旧日语 MS-Windows 文本文件，"jw-file.txt"，能通过如下所示的用 vim 编辑。

```
$ vim '+e ++enc=CP932 ++ff=dos jw-file.txt'
```

当一个文件用 vim 打开的时候带有"++enc" 和"++ff" 选项，在 Vim 命令行输入":w" 命令会以原格式存储文件并且会覆盖原文件。你也可以在 Vim 命令行指定存储文件名及其格式，例如，":w ++enc=utf8 new.txt"。

请查阅 vim 在线帮助中的 mbyte.txt，" 多字节文本支持" 和表 11.2来获得"++enc" 使用的本地值的信息。

emacs 家族的程序能够实现同样的功能。

11.1.7 提取纯文本

如下所示读入 web 页面并把它转化成文本文件。当从 Web 中拷贝配置或者是在 web 页面中应用类似 grep(1) 的基础 Unix 文本工具时，以下命令是非常有用的。

```
$ w3m -dump http://www.remote-site.com/help-info.html >textfile
```

同样，你可以使用如下所示的工具从其他格式提取纯文本数据。

软件包	流行度	大小	关键词	功能
w3m	V:80, I:433	2323	html→text	用”w3m -dump” 命令把 HTML 转化为文本的转换器
html2text	V:10, I:46	269	html→text	高级的 HTML 到文本文件的转换器 (ISO8859-1)
lynx	V:20, I:103	1924	html→text	用”lynx -dump” 命令把 HTML 转化为文本的转化器
elinks	V:10, I:29	1752	html→text	用”elinks -dump” 命令把 HTML 转化为文本的转换器
links	V:12, I:42	2207	html→text	用”links -dump” 命令把 HTML 转化为文本的转换器
links2	V:2, I:16	5486	html→text	用”links2 -dump” 命令把 HTML 转化为文本的转换器
antiword	V:4, I:12	618	MSWord→text,ps	转化 MSWord 文件到纯文本或 ps 文件
catdoc	V:54, I:114	675	MSWord→text,TeX	转化 MSWord 文件到纯文本或 TeX 文件
pstotext	V:2, I:4	126	ps/pdf→text	从 PostScript 和 PDF 文件里导出文本
unhtml	V:0, I:0	42	html→text	从一个 HTML 文件里面删除标记标签
odt2txt	V:2, I:7	53	odt→text	从开放文档格式到文本格式的转化器

Table 11.5: 用于提取纯文本数据的工具列表

11.1.8 高亮并格式化纯文本数据

你可以通过如下所示的来高亮并格式化纯文本数据。

11.2 XML 数据

扩展标记语言 Extensible Markup Language (XML) 是一种标记语言，用于含有结构化信息的文档。

在 XML.COM 查看介绍信息.

- ” 什么是 XML?”
- ” 什么是 XSLT?”
- ” 什么是 XSL-FO?”
- ” 什么是 XLink?”

11.2.1 XML 的基本提示

XML 文本看起来有些像 HTML. 它能够使我们管理一个文档的多个格式。一个简单的 XML 系统是 `docbook-xsl` 软件包，在这里使用。

每一个 XML 文件使用下面的标准 XML 声明开始。

软件包	流行度	大小	关键词	说明
vim-runtime	V:19, I:434	29624	高亮	用”:source $VIMRUNTIME/syntax/html.vim” Vim 宏命令转化源代码到 HTML
cxref	V:0, I:0	1182	c→html	从 C 程序到 latex 和 HTML 的转换器（C 语言）
src2tex	V:0, I:0	622	高亮	转换许多源代码到 TeX（C 语言）
source-highlight	V:0, I:7	2019	高亮	转换源代码到带有高亮显示的 HTML, XHTML, LaTeX, Texinfo, ANSI 颜色转义序列和 DocBook 文件 (C++)
highlight	V:1, I:15	1043	高亮	转化许多源代码到带有高亮显示的 HTML, XHTML, RTF, LaTeX, TeX or XSL-FO 文件。(C++)
grc	V:0, I:2	188	text→ 有颜色的	用于任何文本的通用颜色生成器 (Python)
txt2html	V:0, I:4	254	text→html	文本到 HTML 转换器（Perl）
markdown	V:0, I:6	57	text→html	markdown 文本文档到 (X)HTML (Perl)
asciidoc	I:13	80	text→any	AsciiDoc 文本文档到 XML/HTML (Python)
pandoc	V:6, I:42	113143	text→any	通用标记转化器 (Haskell)
python-docutils	V:32, I:241	1752	text→any	重构文本文档到 XML (Python)
txt2tags	V:0, I:1	813	text→any	转化文本到 HTML, SGML, LaTeX, man page, MoinMoin, Magic Point and PageMaker (Python)
udo	V:0, I:0	564	text→any	通用的文本文件转化工具 (C 语言)
stx2any	V:0, I:0	264	text→any	结构化纯文本到其他格式的文档转化器 (m4)
rest2web	V:0, I:0	527	text→html	重构文本到 html 的文档转化器 (Python)
aft	V:0, I:0	235	text→any	” 自由格式” 的文件准备系统 (Perl)
yodl	V:0, I:0	615	text→any	用预文档语言工具来处理文件 (C 语言)
sdf	V:0, I:0	1445	text→any	简单的文档剖析器 (Perl)
sisu	V:0, I:0	5341	text→any	文档组织、排版、搜索框架 (Ruby)

Table 11.6: 高亮纯文本数据的工具列表

```
<?xml version="1.0" encoding="UTF-8"?>
```

XML 元素的基本语法是按下面的方式标记。

```
<name attribute="value">content</name>
```

内容为空的 XML 元素，使用下面的短格式标记。

```
<name attribute="value"/>
```

上面列子中的"attribute="value"" 是可选的。

XML 里面的注释部分，是按下面的方式标记。

```
<!-- comment -->
```

不同于增加标记，XML 至少要求使用预定义实体里的内容来转化下列字符。

预定义实体	转化的字符
"	": 引号
'	': 撇号
<	<: 小于号
>	>: 大于号
&	&: & 号

Table 11.7: XML 预定义实体列表

小心
"<" 或 "&" 不能在属性（attributes）或元素（elements）中使用。

注意
当 SGML 式样的用户定义实体，比如"&some-tag:", 被使用的时候，第一个定义会覆盖其它的。实体定义在"<!ENTITY some-tag "entity value">" 里表示.

注意
只要 XML 标记是一致使用某一标签名集合（一些数据作为内容或属性值），使用 Extensible Stylesheet Language Transformations (XSLT) 来转换到另外一个 XML，是一个微不足道的任务。

11.2.2 XML 处理

有许多工具可以用于处理 XML 文件，比如说：可扩展样式表语言 Extensible Stylesheet Language (XSL).

一旦你创建了一个好的成形的 XML 文件，基本上来讲，你就可以使用 可扩展样式表语言转换 Extensible Stylesheet Language Transformations (XSLT)，将其转换成任何格式。

软件包	流行度	大小	关键词	说明
docbook-xml	I:488	2131	xml	DocBook 的 XML 文档类型定义 (DTD)
xsltproc	V:17, I:109	154	xslt	XSLT 命令行处理器 (XML → XML, HTML, 纯文本，等等)
docbook-xsl	V:13, I:208	14998	xml/xslt	使用 XSLT 将 DocBook XML 处理成各种输出格式的 XSL 样式表
xmlto	V:2, I:29	130	xml/xslt	使用 XSLT 将 XML 转换到任意格式的转换器
dbtoepub	V:0, I:0	71	xml/xslt	DocBook XML 到.epub 转换
dblatex	V:6, I:20	4648	xml/xslt	使用 XSLT 将 Docbook 文件转换为 DVI, PostScript, PDF 文档
fop	V:2, I:44	291	xml/xsl-fo	转换 Docbook XML 文件到 PDF

Table 11.8: XML 工具列表

格式化对象的可扩展样式表语言 Extensible Stylesheet Language for Formatting Objects (XSL-FO) 是用来作为格式化的解决方案. `fop` 软件包比 Debian `main` 档案库要新，因为它依赖 Java 编程语言. LaTeX 代码通常是从 XML 使用 XSLT 生成，LaTeX 系统是用来创建 DVI, PostScript 和 PDF 这类可打印的文件。

由于 XML 是 标准通用标记语言 Standard Generalized Markup Language (SGML)的一个子集, 用于处理 SGML 的扩展工具，也能够处理 XML，比如说 文档式样语言和规范语言 Document Style Semantics and Specification Language (DSSSL).

软件包	流行度	大小	关键词	说明
openjade	V:2, I:43	988	dsssl	ISO/IEC 10179:1996 标准 DSSSL 处理器 (最新的)
docbook-dsssl	V:1, I:28	2604	xml/dsssl	使用 DSSSL 处理 DocBook XML 到各种输出格式的 DSSSL 样式表
docbook-utils	V:1, I:20	281	xml/dsssl	DocBook 文件的工具包，包括使用 DSSSL 的转换成其它格式 (HTML, RTF, PS, man, PDF) 的 `docbook2*` 命令
sgml2x	V:0, I:0	90	SGML/dsssl	SGML 和 XML 使用 DSSSL 样式表的转换器

Table 11.9: DSSSL 工具列表

提示
GNOME 的 `yelp` 往往能够方便的直接读取 DocBook XML 文件，这是因为它可以从 X 获得适当的渲染。

11.2.3 XML 数据提取

使用下面的方法，你能够从其它格式提取 HTML 或 XML 数据。

对于非 XML 的 HTML 文件，你能够转换它们为 XHTML，XHTML 是一个相当成型的 XML 实例。XHTML 能够被 XML 工具处理.

一旦适当的 XML 生成，基于标记的内容等，你能够使用 XSLT 技术提取数据。

11.3 排版

Unix 上的 troff 程序最初是由 AT&T 公司开发的，可以被用做简单排版。现在被用来创建手册页。

Donald Knuth 发明的 Tex 是非常强大的排版工具也是实际上的标准。最初是由 Leslie Lamport 开发的 LaTex 使得用户可以更为方便的利用 Tex 的强大功能。

软件包	流行度	大小	关键词	说明
wv	V:4, I:8	717	MSWord → 任何格式	从微软 Word 格式到 HTML, LaTeX, 等格式的文件转换器。
texi2html	V:0, I:9	1832	texi → html	从 Texinfo 到 HTML 的转换器
man2html	V:0, I:3	141	man 手册页 → html	从 man 手册页到 HTML 的转换器 (支持 CGI)
unrtf	V:1, I:4	148	rtf → html	从 RTF 到 HTML 等的转换器
info2www	V:2, I:3	156	info → html	从 GNU info 到 HTML 的转换器 (支持 CGI)
ooo2dbk	V:0, I:0	217	sxw → xml	从 OpenOffice.org SXW 文档到 DocBook XML 的转换器
wp2x	V:0, I:0	215	WordPerfect → 任意格式	WordPerfect 5.0 和 5.1 文件到 TeX, LaTeX, troff, GML 和 HTML
doclifter	V:0, I:0	451	troff → xml	troff 到 DocBook XML 的转换器

Table 11.10: XML 数据提取工具列表

软件包	流行度	大小	关键词	说明
libxml2-utils	V:22, I:289	173	xml ↔ html ↔ xhtml	使用 xmllint(1) 的 XML 命令行工具 (语法检查，重新格式化，梳理, ⋯)
tidy	V:1, I:16	84	xml ↔ html ↔ xhtml	HTML 语法检查和重新格式化

Table 11.11: XML 美化打印工具列表

软件包	流行度	大小	关键词	说明
texlive	V:6, I:60	70	(La)TeX	用于排版、预览和打印的 TeX 系统
groff	V:4, I:91	11818	troff	GNU troff 文本格式化系统

Table 11.12: 排版工具的列表

11.3.1 roff 排版

传统意义上，roff 是 Unix 上主要的文本处理系统。参见 roff(7), groff(7), groff(1), grotty(1), troff(1), groff_mdoc(7), groff_man(7), groff_ms(7), groff_me(7), groff_mm(7) 和"`info groff`"。

安装好 `groff` 软件包以后，你输入"`-me`" 宏指令就能看到一份不错的指导手册，它的位置是"`/usr/share/doc/groff/`"。

提示
"`groff -Tascii -me -`" 输出带有 ANSI 转义码的纯文本。如果你想要 manpage 的输出带有许多"^H" 和"_"，那么使用替代命令"`GROFF_NO_SGR=1 groff -Tascii -me -`"。

提示
如果想要移除 `groff` 生成的文本文件中的"^H" 和"_"，使用"`col -b -x`" 来过滤它。

11.3.2 TeX/LaTeX

Tex Live 软件提供了全部的 TeX 系统。`texlive` 元包只是 TeX Live 中的一部分，但是它足够应付日常任务。

这里有许多可用的 TeX 和 LaTeX 的参考资料。

- The teTeX HOWTO: The Linux-teTeX Local Guide
- tex(1)
- latex(1)
- texdoc(1)
- texdoctk(1)
- "The TeXbook", 作者 Donald E. Knuth, (Addison-Wesley)
- "LaTeX - A Document Preparation System", 作者 Leslie Lamport, (Addison-Wesley)
- "The LaTeX Companion", 作者 Goossens, Mittelbach, Samarin, (Addison-Wesley)

这是最强大的排版环境。许多 SGML 处理器把它作为其后台字处理工具。`lyx` 软件包提供的 Lyx 和 `texmacs` 软件包提供的 GNU TeXmacs 都为 LaTeX 提供了非常不错的所见即所得的编辑环境，然而许多人使用 Emacs 和 Vim 作为其源代码编辑器。

有许多在线资源存在。

- TEX Live Guide - TEX Live 2007 ("`/usr/share/doc/texlive-doc-base/english/texlive-en/live.html`") (`texlive-doc-base` 包)
- Latex/Lyx 的一个简单指引
- 使用 LaTeX 进行文字处理
- teTeX/LaTeX 的本地用户指引

当文档变得更大时，TeX 有时会出错。你必须在"`/etc/texmf/texmf.cnf`" 中增加 pool 的大小 (更确切的说话是编辑是"`/etc/texmf/texmf.d/95NonPath`" 并且运行 update-texmf(8)) 来修复此问题。

注意
"The TeXbook" 的 TeX 源码可以从 http://tug.ctan.org/tex-archive/systems/knuth/dist/tex/texbook.tex 上下载。此文件包含了绝大多数所需的宏指令。我听说把文档中的第 7 到第 10 行注释了并且添加"`\input manmac \proofmodefalse`"，就可以用 tex(1) 来处理此文档。我强烈建议去购买这本书 (还有 Donald E. Knuth 写的其他书) 而不是使用在线版本，但是在线版本中的源码确实是学习 Tex 输入很好的例子！

11.3.3 漂亮的打印手册页

你能够用如下任意一个命令在打印机上漂亮的打印手册页。

```
$ man -Tps some_manpage | lpr
```

11.3.4 创建手册页

尽管用纯 troff 格式写手册页（manpage）是可能的，这里还是有一些辅助的程序包用于创建手册页。

软件包	流行度	大小	关键词	说明
docbook-to-man	V:0, I:15	187	SGML → man 手册页	从 DocBook SGML 到 roff 手册页宏指令的转换器
help2man	V:0, I:10	480	text → man 手册页	通过 --help 参数自动生成手册页的工具
info2man	V:0, I:0	134	info → man 手册页	转换 GNU info 到 POD 或手册页的转换器
txt2man	V:0, I:1	92	text → man 手册页	把纯粹的 ASCII 文本转化为手册页格式

Table 11.13: 创建手册页的工具列表

11.4 可印刷的数据

在 Debian 系统中，可打印的数据是 PostScript 格式的。对于非 PostScript 打印机，通用 Unix 打印系统 (CUPS) 使用 Ghostscript 作为其后台光栅处理程序。

11.4.1 Ghostscript

处理可印刷的数据的核心是 Ghostscript PostScript 解释器，它能够生成光栅图像。

来自 Artifex 的最新上游 Ghostscript 软件包的许可从 AFPL 变成 GPL，并且发布的是合并版本，其中合并了最新的 ESP 版本的改变，例如 CUPS 8.60 版本。

软件包	流行度	大小	说明
ghostscript	V:173, I:665	225	GPL Ghostscript PostScript/PDF 解释器
ghostscript-x	V:26, I:70	219	GPL Ghostscript PostScript/PDF 解释器-X 显示支持
libpoppler82	V:28, I:68	3652	PDF 渲染库 (xpdf PDF 浏览器的分支)
libpoppler-glib8	V:199, I:522	421	PDF 渲染库（基于 Glib 的共享库)
poppler-data	V:133, I:666	12219	用于 PDF 渲染库的 CMaps (CJK 支持：Adobe-*)

Table 11.14: Ghostscript PostScript 解释器列表

提示
”gs -h” 能够显示 Ghostscript 的配置信息。

11.4.2 合并两个 PS 或 PDF 文件

你能够使用 Ghostscript 中的 gs(1) 来合并两个 PostScript(PS) 或可移植文档格式（PDF）文件。

```
$ gs -q -dNOPAUSE -dBATCH -sDEVICE=pswrite -sOutputFile=bla.ps -f foo1.ps foo2.ps
$ gs -q -dNOPAUSE -dBATCH -sDEVICE=pdfwrite -sOutputFile=bla.pdf -f foo1.pdf foo2.pdf
```

注意
PDF 是用途很广的跨平台可印刷的数据格式，它本质上是带有一些额外特性和扩展的压缩了的 PS 格式。

提示
对于命令行来说，psmerge(1) 和 `psutils` 包中的其他命令在处理 PostScript 文档时是很有用的。`pdftk` 包中的 pdftk(1) 在处理 PDF 文档的时候同样是很好用的。

11.4.3 处理可印刷数据的工具

如下是处理可印刷数据的工具列表。

软件包	流行度	大小	关键词	说明
`poppler-utils`	V:44, I:470	665	pdf → ps,text, …	PDF 工具：`pdftops`, `pdfinfo`, `pdfimages`, `pdftotext`, `pdffonts`
`psutils`	V:8, I:139	219	ps → ps	PostScript 文件转换工具
`poster`	V:0, I:6	49	ps → ps	用 PostScript 页制作大型海报
`enscript`	V:1, I:22	2111	text → ps, html, rtf	转化 ASCII 文本到 PostScript, HTML, RTF 或 Pretty-Print
`a2ps`	V:1, I:19	3648	text → ps	’任何文本到 PostScript’ 的转换器并且也是相当不错的打印程序
`pdftk`	V:6, I:54	27	pdf → pdf	PDF 文档转换工具：`pdftk`
`html2ps`	V:0, I:4	249	html → ps	从 HTML 到 PostScript 的转换器
`gnuhtml2latex`	V:0, I:1	27	html → latex	从 html 到 latex 的转换器
`latex2rtf`	V:0, I:6	478	latex → rtf	转换 LaTeX 文档到能被 Microsoft Word 读取的 RTF 格式的文档
`ps2eps`	V:5, I:97	94	ps → eps	从 PostScript 到 EPS（Encapsulated PostScript）的转换器
`e2ps`	V:0, I:0	112	text → ps	带有日文编码支持的文本到 PostScript 转换器
`impose+`	V:0, I:1	180	ps → ps	PostScript 工具
`trueprint`	V:0, I:0	146	text → ps	漂亮的打印许多源程序（C, C++, Java, Pascal, Perl, Pike, Sh, 和 Verilog）到 PostScript。(C 语言)
`pdf2svg`	V:0, I:4	26	ps → svg	PDF 到可升级的向量图形格式的转换器
`pdftoipe`	V:0, I:0	67	ps → ipe	从 PDF 到 IPE ‘s XML 格式的转换器

Table 11.15: 处理可印刷数据的工具列表

11.4.4 用 CUPS 打印

Unix 通用打印系统（CUPS）中的 lp(1) 和 lpr(1) 命令都提供了自定义打印数据的选项。
你可以使用下列命令中的一个来打印 3 份有装订页码的文件。

```
$ lp -n 3 -o Collate=True filename
```

```
$ lpr -#3 -o Collate=True filename
```

你能够通过"-o number-up=2","-o page-set=even","-o page-set=odd","-o scaling=200","-o natural-sca
等等打印机选项来进一步定制打印机操作，详细的文档参见命令行打印和选项。

11.5 邮件数据转换

下列邮件数据转换软件包捕获了我的眼球。

软件包	流行度	大小	关键词	说明
sharutils	V:5, I:73	1405	邮件	shar(1), unshar(1), uuencode(1), uudecode(1)
mpack	V:1, I:21	91	MIME	编码和解码 MIME 信息: mpack(1) 和 munpack(1)
tnef	V:5, I:11	98	ms-tnef	解包 MIME 附件类型"application/ms-tnef"，该格式仅由微软使用
uudeview	V:0, I:5	109	邮件	下列格式的编码器和解码器: uuencode, xxencode, BASE64, quoted printable 和 BinHex

Table 11.16: 有助于邮件数据转换的软件包列表

提示
如果邮件客户端可以配置使用 IMAP4 服务器，互联网消息访问协议 版本 4 (IMAP4) 服务器 (参见第 6.7 节) 可以用来把邮件从专有邮件系统里面移出来。

11.5.1 邮件数据基础

邮件 (SMTP) 数据需要被限制为 7 位数据序列。二进制数据和 8 位文本数据使用 Multipurpose Internet Mail Extensions (MIME) 互联网多用途邮件扩展 和选择的字符集编码到 7 位格式。(参见第 8.4.1 节).

标准的邮件存储格式是 mbox，它是依据 RFC2822 (由 RFC822 更新) 来的格式. 参见 mbox(5) (由 mutt 软件包提供).

对于欧洲语言, 由于没有什么 8 位字符，"Content-Transfer-Encoding: quoted-printable" 加 ISO-8859-1 字符集通常被用于邮件。如果欧洲文本是被编码为 UTF-8, 由于几乎全是 7 位数据，使用"Content-Transfer-Encoding: quoted-printable" 也是合适的。

对于日语，传统的"Content-Type: text/plain; charset=ISO-2022-JP" 通常被用于邮件来保持文本在 7 位。但是老的微软系统会在没有声明的情况下使用 Shift-JIS 来发送邮件。如果日语文本是用 UTF-8 编码, 由于含有许多 8 位数据，使用 Base64 是合适的。其它亚洲语言也是类似情形。

注意
如果你的非 Unix 邮件数据可以通过一个具备和 IMAP4 服务通讯的非 Debian 客户端访问，你可以通过运行你的 IMAP4 服务来将邮件数据移出。(参见第 6.7 节).

注意
如果你使用其它邮件存储格式，第一步把它们移动到 mbox 格式比较好。像 mutt(1) 这样多功能的客户端程序可以便捷的完成这类操作。

你可以使用 procmail(1) 和 formail(1) 把邮箱内容分开成每一封邮件.

每一封邮件能够使用来自 mpack 软件包的 munpack(1) 命令（或其它特异的工具）来获得 MIME 编码内容。

11.6 图形数据工具

如下是关于图形数据转换、编辑和管理的工具包。

> 提示
> 在 aptitude(8)（参考第 2.2.6 节）中用正则表达式"~Gworks-with::image" 来查找更多的图像工具。

虽然像 gimp(1) 这样的图形界面程序是非常强大的，但像 imagemagick(1) 这样的命令行工具在用脚本自动化处理图像时是很有用的。

实际上的数码相机的图像是可交换的图像文件格式(EXIF)，这种格式是在 JPEG 图像文件格式上添加一些元数据标签。它能够保存诸如日期、时间和相机设置的信息。

The Lempel-Ziv-Welch (LZW) 无损数据压缩专利已经过期了。使用 LZW 压缩方式的 图形交互格式（GIF）工具现在可以在 Debian 系统上自由使用了。

> 提示
> 任何带有可移动记录介质的数码相机或扫描仪都可以在 Linux 上通过 USB 存储读取器来工作，因为它遵循相机文件系统设计规则并且使用 FAT 文件系统，参考第 10.1.7 节。

11.7 不同种类的数据转换工具

这里有许多其他用于数据转换的工具。在 aptitude(8)（参考第 2.2.6 节）里用正则表达式"~Guse::converting"" 来查找如下的软件包。

你能够通过如下的命令从 RPM 格式的包中提取数据。

```
$ rpm2cpio file.src.rpm | cpio --extract
```

软件包	流行度	大小	关键词	说明
gimp	V:85, I:489	19016	图形 (位图)	GNU 图形处理程序
imagemagick	V:43, I:549	209	图形 (位图)	图形处理程序
graphicsmagick	V:6, I:17	5252	图形 (位图)	图像处理程序 (imagemagick 派生出来的)
xsane	V:19, I:190	935	图形 (位图)	用于 SANE 的基于 GTK+ 的前端图形界面 (现在访问扫描仪就很简单了)
netpbm	V:35, I:552	4302	图形 (位图)	图形界面的转换工具
icoutils	V:15, I:153	220	png↔ico(bitmap)	MS Windows 符号和光标转化为 PNG 格式，或者从 PNG 格式转化为位图格式 (favicon.ico)
scribus	V:3, I:28	19995	ps/pdf/SVG/ …	Scribus DTP 编辑器
libreoffice-draw	V:313, I:470	9960	图形 (矢量)	LibreOffice 办公套件-绘画
inkscape	V:129, I:332	78502	图形 (矢量)	SVG (可升级矢量图形)编辑器
dia	V:18, I:37	3824	图形 (矢量)	图表编辑器 (Gtk)
xfig	V:10, I:18	1793	图形 (矢量)	在图形界面下，交互式的生成图像变得方便
pstoedit	V:6, I:159	992	ps/pdf→image(矢量)	PostScript 和 PDF 文件到可编辑的矢量图形的转换器 (SVG)
libwmf-bin	V:11, I:335	113	Windows/image(矢量)	Windows 元文件 (矢量图形数据) 转换工具
fig2sxd	V:0, I:0	149	fig→sxd(矢量)	转换 XFig 文件为 OpenOffice.org 绘画格式
unpaper	V:2, I:17	460	image→image	后处理 OCR 扫描页面的工具
tesseract-ocr	V:6, I:32	1119	image→text	基于惠普的商业 OCR 引擎的免费 OCR 软件
tesseract-ocr-eng	I:33	4032	image→text	OCR 引擎数据：用于英文文本的 tesseract-ocr 语言文件
gocr	V:1, I:19	527	image→text	免费 OCR 软件
ocrad	V:0, I:6	303	image→text	免费 OCR 软件
eog	V:87, I:301	11807	图像 (Exif)	Eye of GNOME 图像浏览程序
gthumb	V:12, I:23	3532	图像 (Exif)	图像浏览器 (GNOME)
geeqie	V:12, I:22	12814	图像 (Exif)	基于 GTK+ 的图像浏览器
shotwell	V:20, I:224	6096	图像 (Exif)	数码相片管理器 (GNOME)
gtkam	V:0, I:6	1154	图像 (Exif)	从数码照相机中检索多媒体数据的应用 (GTK+)
gphoto2	V:0, I:13	965	图像 (Exif)	gphoto2 软件是命令行方式的管理数码相机的工具
gwenview	V:31, I:104	11266	图像 (Exif)	图片浏览器 (KDE)
kamera	I:103	748	图像 (Exif)	KDE 上的支持数码相机的应用软件
digikam	V:3, I:15	3644	图像 (Exif)	用于 KDE 桌面环境的数字照片管理应用
exiv2	V:4, I:53	239	图像 (Exif)	EXIF/IPTC 元数据处理工具
exiftran	V:1, I:23	70	图像 (Exif)	改变数码照相机的 jpeg 图像格式
jhead	V:1, I:12	113	图像 (Exif)	处理兼容 JPEG 文件 (数码相机图片) 的 Exif 中的非图形部分
exif	V:1, I:12	238	图像 (Exif)	显示 JPEG 文件中的 EXIF 信息的命令行工具
exiftags	V:0, I:4	288	图像 (Exif)	从数码相机的 JPEG 文件读取 Exif 标签的实用工具
exifprobe	V:0, I:4	491	图像 (Exif)	从数码图片中读取元数据
dcraw	V:2, I:22	535	image(原始的)→ppm	解码原始的数码相机图片
findimagedupes	V:0, I:1	78	image→fingerprint	找到相似或重复的图像
ale	V:0, I:0	753	image→image	合并图像来增加保真度或者用于创建马赛克
imageindex	V:0, I:0	145	image(Exif)→html	从图形中创建静态 HTML 图库
outguess	V:0, I:2	260	jpeg,png	通用的 Steganographic 工具
librecad	V:9, I:19	8201	DXF	CAD 数据编辑器 (KDE)
blender	V:4, I:35	107121	blend, TIFF, VRML, …	用于动画的 3D 编辑器
mm3d	V:0, I:0	3900	ms3d, obj, dxf, …	基于 OpenGL 的 3D 模型编辑器
open-font-design-toolkit	I:0	10	ttf, ps, …	用于开放字型设计的元包
fontforge	V:0, I:8	91	ttf, ps, …	用于 PS，TrueType 和 OpenType 的字体编辑器
xgridfit	V:0, I:0	876	ttf	用于 TrueType 字体的 网格拟合和小字还原技术 的程序

软件包	流行度	大小	关键词	说明
alien	V:3, I:45	166	rpm/tgz → deb	把外来的软件包转换为 Debian 软件包
freepwing	V:0, I:0	421	EB → EPWING	把” 电子书” (在日本流行) 变成单一的 JIS X 4081 格式 (EPWING V1 的子集)
calibre	V:8, I:39	51670	any → EPUB	电子书转换器和库管理

Table 11.18: 不同种类的数据转换工具列表

Chapter 12

编程

这里我给出一些 Debian 系统中的信息，帮助学习编程的人找出打包的源代码。下面是值得关注的软件包和与之对应的文档。

软件包	流行度	大小	包
autoconf	V:38, I:269	1868	由 autoconf-doc 包提供的“info autoconf”
automake	V:37, I:265	1784	由 automake1.10-doc 包提供的“info automake”
bash	V:826, I:999	6462	由 bash-doc 包提供的“info bash”
bison	V:11, I:109	2253	由 bison-doc 包提供的“info bison”
cpp	V:389, I:790	42	由 cpp-doc 包提供的“info cpp”
ddd	V:0, I:12	3929	由 ddd-doc 包提供的“info ddd”
exuberant-ctags	V:7, I:42	333	exuberant-ctags(1)
flex	V:10, I:98	1225	由 flex-doc 包提供的“info flex”
gawk	V:443, I:535	2412	由 gawk-doc 包提供的“info gawk”
gcc	V:173, I:598	45	由 gcc-doc 包提供的“info gcc”
gdb	V:17, I:124	8989	由 gdb-doc 包提供的“info gdb”
gettext	V:52, I:345	6594	由 gettext-doc 包提供的“info gettext”
gfortran	V:8, I:79	16	由 gfortran-doc 包提供的“info gfortran”(Fortran 95)
fpc	I:4	120	fpc(1) 和由 fp-docs 包提供的 html 文档 (Pascal)
glade	V:1, I:9	2306	通过 UI Builder 菜单提供的文档
libc6	V:937, I:999	12333	通过 glibc-doc 和 glibc-doc-reference 提供的“info libc”
make	V:169, I:604	1296	通过 make-doc 包提供的“info make”
xutils-dev	V:1, I:14	1466	imake(1)，xmkmf(1) 等。
mawk	V:342, I:998	183	mawk(1)
perl	V:618, I:994	575	perl(1) 以及通过 perl-doc 和 perl-doc-html 提供的 html 文档
python	V:578, I:986	68	python(1) 以及通过 python-doc 包提供的 html 文档
tcl	V:30, I:442	22	tcl(3) 以及通过 tcl-doc 包提供的更详细的手册页文档
tk	V:31, I:433	22	tk(3) 以及通过 tk-doc 包提供的更详细的手册页文档
ruby	V:173, I:341	37	ruby(1) 以及通过 ri 包提供的交互式参考手册
vim	V:119, I:395	2799	通过 vim-doc 包提供的帮助 (F1) 菜单
susv2	I:0	16	通过“单一 UNIX 规范 (版本 2)”获取 (英语文档)
susv3	I:0	16	通过“单一 UNIX 规范 (版本 3)”获取 (英语文档)

Table 12.1: 帮助编程的软件包清单

安装 manpages 和 manpages-dev 包之后，可以通过运行“man 名称”查看手册页中的参考信息。安装了 GNU 工具的相关文档包之后，可以通过运行“info 程序名称”查看参考文档。某些 GFDL 协议的文档与 DFSG 并不兼容，

所以你可能需要在 main 仓库中包含 contrib 和 non-free 才能下载并安装它们。

警告
不要用“test”作为可执行的测试文件的名字，因为 shell 中内建有“test”命令。

小心
你可以把从源代码编译得到的程序直接放到“/usr/local”或“/opt”目录，这样可以避免与系统程序撞车。

提示
“歌曲：99 瓶啤酒”的代码示例可以给你提供实践各种语言的好范本。

12.1 Shell 脚本

Shell 脚本 是指包含有下面格式的可执行的文本文件。

```
#!/bin/sh
……命令行
```

第一行指明了读取并执行这个文件的 shell 解释器。

读懂 shell 脚本的最好办法是先理解类 UNIX 系统是如何工作的。这里有一些 shell 编程的提示。看看“Shell 错误”(http://www.greenend.org.uk/rjk/2001/04/shell.html)，可以从错误中学习。

不像 shell 交互模式 (参见第 1.5 节和第 1.6 节)，shell 脚本会频繁使用参数、条件和循环等。

12.1.1 POSIX shell 兼容性

系统中的许多脚本都可以通过任意 POSIX shell (参见表 1.13) 来执行。系统的默认 shell 是“/bin/sh”，它是某个实际 shell 程序的链接。

- 对 lenny 或更老的系统来说，它是 bash(1)
- 对 squeeze 或更新的系统来说，它是 dash(1)

避免编写具有 **bashisms** (bash 化) 或者 **zshisms** (zsh 化) 语法的 shell 脚本，确保脚本在所有 POSIX shell 之间具有可移植性。你可以使用 checkbashisms(1) 对其进行检查。

好的：**POSIX**	应该避免的：**bashism**
`if [ "$foo" = "$bar" ] ; then …`	`if [ "$foo" == "$bar" ] ; then …`
`diff -u file.c.orig file.c`	`diff -u file.c{.orig,}`
`mkdir /foobar /foobaz`	`mkdir /foo{bar,baz}`
`funcname() { …}`	`function funcname() { …}`
八进制格式：”\377”	十六进制格式：”\xff”

Table 12.2: 典型 bashism 语法列表

使用”echo” 命令的时候需要注意以下几个方面，因为根据内置 shell 和外部命令的不同，它的实现也有差别。

- 避免使用除“-n”以外的任何命令行选项。
- 避免在字符串中使用转义序列，因为根据 shell 不同，计算后的结果也不一样。

注意
尽管“-n”选项并不是 POSIX 语法，但它已被广泛接受。

提示
如果你想要在输出字符串中嵌入转义序列，用”printf” 命令替代”echo” 命令。

12.1.2 Shell 参数

特殊的 shell 参数经常在 shell 脚本里面被用到。

shell 参数	值
$0	shell 或 shell 脚本的名称
$1	第一个 shell 参数
$9	第 9 个 shell 参数
$#	位置参数数量
”$*”	”$1 $2 $3 $4 …”
”$@”	”$1” ”$2” ”$3” ”$4” …
$?	最近一次命令的退出状态码
$$	这个 shell 脚本的 PID
$!	最近开始的后台任务 PID

Table 12.3: shell 参数列表

如下所示是需要记忆的基本的参数展开。

参数表达式形式	如果 **var** 变量已设置那么值为	如果 **var** 变量没有被设置那么值为
${var:-string}	”$var”	”string”
${var:+string}	”string”	”null”
${var:=string}	”$var”	”string” (并运行”var=string”)
${var:?string}	”$var”	在 **stderr** 中显示”string” (出错退出)

Table 12.4: shell 参数展开列表

以上这些操作中”:” 实际上都是可选的。

- 有”:” 等于测试的 var 值是存在且非空
- 没有”:” 等于测试的 var 值只是存在的，可以为空

12.1.3 Shell 条件语句

每个命令都会返回 退出状态，这可以被条件语句使用。

- 成功：0 (”True”)
- 失败：非 0 (”False”)

参数替换形式	结果
`${var%suffix}`	删除位于 var 结尾的 suffix 最小匹配模式
`${var%%suffix}`	删除位于 var 结尾的 suffix 最大匹配模式
`${var#prefix}`	删除位于 var 开头的 prefix 最小匹配模式
`${var##prefix}`	删除位于 var 开头的 prefix 最大匹配模式

Table 12.5: 重要的 shell 参数替换列表

注意
”0” 在 shell 条件语句中的意思是”True”，然而”0” 在 C 条件语句中的含义为”False”。

注意
”[” 跟 `test` 命令是等价的，它评估到”]” 之间的参数来作为一个条件表达式.

如下所示是需要记忆的基础 条件语法。

- ”`<command> && <if_success_run_this_command_too> || true`”
- ”`<command> || <if_not_success_run_this_command_too> || true`”
- 如下所示是多行脚本片段

```
if [ <conditional_expression> ]; then
 <if_success_run_this_command>
else
 <if_not_success_run_this_command>
fi
```

这里末尾的“`|| true`”是需要的，它可以保证这个 shell 脚本在不小心使用了“`-e`”选项而被调用时不会在该行意外地退出。

表达式	返回逻辑真所需的条件
`-e <file>`	<file> 存在
`-d <file>`	<file> 存在并且是一个目录
`-f <file>`	<file> 存在并且是一个普通文件
`-w <file>`	<file> 存在并且可写
`-x <file>`	<file> 存在并且可执行
`<file1> -nt <file2>`	<file1> 是否比 <file2> 新
`<file1> -ot <file2>`	<file1> 是否比 <file2> 旧
`<file1> -ef <file2>`	<file1> 和 <file2> 位于相同的设备上并且有相同的 inode 编号

Table 12.6: 在条件表达式中进行文件比较

算术整数的比较在条件表达式中为”`-eq`”，”`-ne`”，”`-lt`”，”`-le`”，”`-gt`” 和”`-ge`”。

12.1.4 shell 循环

这里有几种可用于 POSIX shell 的循环形式。

- ”`for x in foo1 foo2 …; do command ; done`”,该循环会将”`foo1 foo2 …`” 赋予变量”x” 并执行”command”。

表达式	返回逻辑真所需的条件
`-z <str>`	<str> 的长度为零
`-n <str>`	<str> 的长度不为零
`<str1> = <str2>`	<str1> 和 <str2> 相等
`<str1> != <str2>`	<str1> 和 <str2> 不相等
`<str1> < <str2>`	<str1> 排列在 <str2> 之前（取决于语言环境）
`<str1> > <str2>`	<str1> 排列在 <str2> 之后（取决于语言环境）

Table 12.7: 在条件表达式中进行字符串比较

- "`while condition ; do command ; done`"，当"`condition`" 为真时，会重复执行"`command`"。
- "`until condition ; do command ; done`"，当"`condition`" 为假时，会重复执行"`command`"。
- "`break`" 可以用来退出循环。
- "`continue`" 可以用来重新开始下一次循环。

提示
C 语言中的数值迭代可以用 seq(1) 实现来生成"`foo1 foo2 ...`"。

提示
参见第 9.3.9 节。

12.1.5 shell 命令行的处理顺序

shell 大致以下列的顺序来处理一个脚本。

- shell 读取一行。
- 如果该行包含有"…" 或 '…'，shell 对该行各部分进行分组作为 一个标识（**one token**）（译注：one token 是指 shell 识别的一个结构单元）.
- shell 通过下列方式将行中的其它部分分隔进 标识（**tokens**）。
  - 空白字符: `< 空格 > <tab> < 换行符 >`
  - 元字符: `< > | ; & ( )`
- shell 会检查每一个不位于 "…" 或 '...' 的 token 中的 保留字来调整它的行为。
  - 保留字: `if then elif else fi for in while unless do done case esac`
- shell 展开不位于 "…" 或 '...' 中的 别名。
- shell 展开不位于 "…" 或 '...' 中的 波浪线。
  - "~" → 当前用户的家目录
  - "~<user>" → <user> 的家目录
- shell 将不位于 '...' 中的 变量展开为它的值。
  - 变量: "$PARAMETER" 或"${PARAMETER}"
- shell 展开不位于 '...' 中的 命令替换。

  - "`$( command )`" → "command" 的输出
  - "`` ` command ` ``" → "command" 的输出
- shell 将不位于 "…" 或 '...' 中的 **glob** 路径展开为匹配的文件名。
  - `*` → 任何字符
  - `?` → 一个字符
  - `[…]` → 任何位于"…" 中的字符
- shell 从下列几方面查找 命令并执行。
  - 函数定义
  - 内建命令
  - "`$PATH`" 中的可执行文件
- shell 前往下一行，并按照这个顺序从头再次进行处理。

双引号中的单引号是没有效果的。

在 shell 中执行 "`set -x`" 或使用 "`-x`" 选项启动 shell 可以让 shell 显示出所有执行的命令。这对调试来说是非常方便的。

12.1.6 用于 shell 脚本的应用程序

为了使你的 shell 程序在 Debian 系统上尽可能地具有可移植性，你应该只使用 必要的软件包所提供的应用程序。

- "`aptitude search ~E`"，列出 必要的软件包。
- "`dpkg -L <package_name> |grep '/man/man.*/'`"，列出 `<package_name>` 软件包所提供的 man 手册。

软件包	流行度	大小	说明
coreutils	V:888, I:999	15719	GNU 核心工具
debianutils	V:941, I:999	226	用于 Debian 的各种工具
bsdmainutils	V:861, I:999	587	来自 FreeBSD 更多的工具集合
bsdutils	V:866, I:999	293	来自 4.4BSD-Lite 的基础工具
moreutils	V:5, I:24	237	额外的 Unix 工具

Table 12.8: 包含用于 shell 脚本的小型应用程序的软件包

提示
尽管 `moreutils` 可能不存在 Debian 之外，但它提供了一些有趣的小程序。最值得注意的一个是 sponge(8)，当你项覆盖原来的文件时，它会非常好用。

12.1.7 shell 脚本对话框

一个简单的 shell 程序的用户界面中，`echo` 和 `read` 命令的交互性较为一般，你可以使用对话程序等来提升交互性。

软件包	流行度	大小	说明
x11-utils	V:375, I:635	631	xmessage(1)：在一个窗口中显示一条消息或疑问（X）
whiptail	V:434, I:996	70	从 shell 脚本中显示用户友好的对话框（newt）
dialog	V:17, I:125	1168	从 shell 脚本中显示用户友好的对话框（ncurses）
zenity	V:229, I:395	369	从 shell 脚本中显示图形对话框（gtk2.0）
ssft	V:0, I:0	75	Shell 脚本前端工具 (zenity, kdialog, and 带有 gettext 的 dialog 封装)
gettext	V:52, I:345	6594	“/usr/bin/gettext.sh”：翻译信息

Table 12.9: 用户界面程序列表

12.1.8 zenity 的 shell 脚本案例

下面是一个简单的脚本，它通过 dvdisaster(1) 创建了带有 RS02 补充数据的 ISO 映像。

```
#!/bin/sh -e
# gmkrs02 : Copyright (C) 2007 Osamu Aoki <osamu@debian.org>, Public Domain
#set -x
error_exit()
{
  echo "$1" >&2
  exit 1
}
# Initialize variables
DATA_ISO="$HOME/Desktop/iso-$$.img"
LABEL=$(date +%Y%m%d-%H%M%S-%Z)
if [ $# != 0 ] && [ -d "$1" ]; then
  DATA_SRC="$1"
else
  # Select directory for creating ISO image from folder on desktop
  DATA_SRC=$(zenity --file-selection --directory  \
    --title="Select the directory tree root to create ISO image") \
    || error_exit "Exit on directory selection"
fi
# Check size of archive
xterm -T "Check size $DATA_SRC" -e du -s $DATA_SRC/*
SIZE=$(($(du -s $DATA_SRC | awk '{print $1}')/1024))
if [ $SIZE -le 520 ] ; then
  zenity --info --title="Dvdisaster RS02" --width 640  --height 400 \
    --text="The data size is good for CD backup:\\n $SIZE MB"
elif [ $SIZE -le 3500 ]; then
  zenity --info --title="Dvdisaster RS02" --width 640  --height 400 \
    --text="The data size is good for DVD backup :\\n $SIZE MB"
else
  zenity --info --title="Dvdisaster RS02" --width 640  --height 400 \
    --text="The data size is too big to backup : $SIZE MB"
  error_exit "The data size is too big to backup :\\n $SIZE MB"
fi
# only xterm is sure to have working -e option
# Create raw ISO image
rm -f "$DATA_ISO" || true
xterm -T "genisoimage $DATA_ISO" \
  -e genisoimage -r -J -V "$LABEL" -o "$DATA_ISO" "$DATA_SRC"
# Create RS02 supplemental redundancy
xterm -T "dvdisaster $DATA_ISO" -e  dvdisaster -i "$DATA_ISO" -mRS02 -c
zenity --info --title="Dvdisaster RS02" --width 640  --height 400 \
  --text="ISO/RS02 data ($SIZE MB) \\n created at: $DATA_ISO"
# EOF
```

你可能想要在桌面创建一个启动器，其中的命令设置为类似“/usr/local/bin/gmkrs02 %d”的形式。

12.2 make

Make 是一个维护程序组的工具。一旦执行 make(1)，make 会读取规则文件 Makefile，自从上次目标文件被修改后，如果目标文件依赖的相关文件发生了改变，那么就会更新目标文件，或者目标文件不存在，那么这些文件更新可能会同时发生。

规则文件的语法如下所示。

```
目标: [相关文件 ...]
[TAB] 命令1
[TAB] -命令2 # 忽略错误
[TAB] @命令3 # 禁止回显
```

这里面的”[TAB]” 是一个 TAB 代码。每一行在进行变量替换以后会被 shell 解释。在行末使用”\” 来继续此脚本。使用”$$” 输入”$” 来获得 shell 脚本中的环境变量值。

目标跟相关文件也可以通过隐式规则给出，例如，如下所示。

```
%.o: %.c header.h
```

在这里，目标包含了”%” 字符 (只是它们中确切的某一个)。”%” 字符能够匹配实际的目标文件中任意一个非空的子串。相关文件同样使用”%” 来表明它们是怎样与目标文件建立联系的。

自动变量	值
$@	当前目标
$<	首个相关文件
$?	所有较新的相关文件
$^	所有相关文件
$*	目标模式中，$* 指代匹配符”%” 匹配的部分

Table 12.10: 自动变量的列表

变量扩展	说明
foo1 := bar	一次性扩展
foo2 = bar	递归扩展
foo3 += bar	增加

Table 12.11: 变量扩展的列表

运行”make -p -f/dev/null” 命令来查看内部自动化的规则。

12.3 C

你可以通过下列方法设置适当的环境来编译使用 C 编程语言编写的程序。

```
# apt-get install glibc-doc manpages-dev libc6-dev gcc build-essential
```

libc6-dev 软件包，即 GNU C 库，提供了 C 标准库，它包含了 C 编程语言所使用的头文件和库例程。

参考信息如下。

- “info libc”（C 库函数参考）
- gcc(1) 和 “info gcc”
- each_C_library_function_name(3)
- Kernighan & Ritchie，“C 程序设计语言”，第二版（Prentice Hall）

12.3.1 简单的 C 程序（gcc）

一个简单的例子 “example.c” 可以通过如下方式和 “libm” 库一起编译为可执行程序 “run_example”。

```
$ cat > example.c << EOF
#include <stdio.h>
#include <math.h>
#include <string.h>

int main(int argc, char **argv, char **envp){
        double x;
        char y[11];
        x=sqrt(argc+7.5);
        strncpy(y, argv[0], 10); /* prevent buffer overflow */
        y[10] = '\0'; /* fill to make sure string ends with '\0' */
        printf("%5i, %5.3f, %10s, %10s\n", argc, x, y, argv[1]);
        return 0;
}
EOF
$ gcc -Wall -g -o run_example example.c -lm
$ ./run_example
        1, 2.915, ./run_exam,     (null)
$ ./run_example 1234567890qwerty
        2, 3.082, ./run_exam, 1234567890qwerty
```

为了使用 sqrt(3)，必须使用 “-lm” 链接来自 libc6 软件包的库 “/usr/lib/libm.so”。实际的库文件位于 “/lib/”，文件名为 “libm.so.6”，它是指向 “libm-2.7.so” 的一个链接。

请看一下输出文本的最后一段。即使指定了 “%10s”，它依旧超出了 10 个字符。

使用没有边界检查的指针内存操作函数，比如 sprintf(3) 和 strcpy(3), 是不建议使用，是为防止缓存溢出泄露而导致上面的溢出问题。请使用 snprintf(3) 和 strncpy(3) 来替代.

12.4 调试

调试是程序中很重要的一部分。知道怎样去调试程序使得作为 Debian 使用者的你, 能够做出有意义的错误报告。

12.4.1 基本的 gdb 使用命令

Debian 上原始的调试器是 gdb(1), 它能让你在程序执行的时候检查程序。

让我们通过如下所示的命令来安装 gdb 及其相关程序。

```
# apt-get install gdb gdb-doc build-essential devscripts
```

gdb 的好的教程由”info gdb” 提供或者可以在网上的其他地方找到。如下是用 gdb(1) 在” 程序” 带有”-g” 选项编译的时候来产生调试信息。

```
$ gdb program
(gdb) b 1                # 在第一行设置断点
(gdb) run args           # 带参数运行程序
(gdb) next               # 执行下一步
...
(gdb) step               # 单步进入
...
(gdb) p parm             # 打印 parm 的值
...
(gdb) p parm=12          # 把值设为 12
...
(gdb) quit
```

提示
许多 gdb(1) 命令都能被缩写。Tab 扩展跟在 shell 一样都能工作。

12.4.2 调试 Debian 软件包

因为在 Debian 系统上默认所有已安装的二进制程序都是精简的，绝大多数的调试符号已经从常规的软件包中移除了。为了能用 gdb(1) 调试 Debian 软件包，相对应的 *-dbg 软件包或 *-dbgsym 软件包需要被安装 (例如 libc6 需要安装 libc6-dbg，coreutils 需要安装 coreutils-dbgsym)。

老式的软件包将提供相应的 *-dbg 软件包。它将和原始软件包一起，直接放在 Debian main 档案库。对于新的软件包，当它们编译时，将会自动产生 *-dbgsym 软件包，那些调试软件包将被独立放在 debian-debug 档案库. 更多信息请参阅 Debian Wiki 文档 .

如果一个需要被调试的软件包没有提供其 *-dbg 软件包或 *-dbgsym 软件包，你需要按如下所示的从源代码中重构并且安装它。

```
$ mkdir /path/new ; cd /path/new
$ sudo apt-get update
$ sudo apt-get dist-upgrade
$ sudo apt-get install fakeroot devscripts build-essential
$ apt-get source package_name
$ cd package_name*
$ sudo apt-get build-dep ./
```

按需修改 bug。

软件包调试版本跟它的官方 Debian 版本不冲突，例如当重新编译已存在的软件包版本产生的”+debug1” 后缀，如下所示是编译未发行的软件包版本产生的”~pre1” 后缀。

```
$ dch -i
```

如下所示编译并安装带有调试符号的软件包。

```
$ export DEB_BUILD_OPTIONS=nostrip noopt
$ debuild
$ cd ..
$ sudo debi package_name*.changes
```

你需要检查软件包的构建脚本并确保编译二进制的时候使用了”CFLAGS=-g -Wall” 选项。

12.4.3 获得栈帧

当你碰到程序崩溃的时候，报告 bug 时附上栈帧信息是个不错的注意。

如下所示的步骤就可以取得栈帧信息。

- 在 gdb(1) 中运行程序。
- 重现崩溃。
  - 它使得你重新回到 gdb 提示符。
- 在 gdb 提示符后输入”bt”。

程序在终端中的 gdb 环境运行时，如果它没反应，你可以按下 Ctrl-C 来中止程序来取得 gdb 提示符。

提示
通常，你会看到堆栈顶部有一行或者多行有”malloc()” 或”g_malloc()”. 当这个出现的时候，你的堆栈不是非常有用的。找到一些有用信息的一个简单方法是设置环境变量”$MALLOC_CHECK_” 的值为 2 (malloc(3)). 你可以通过下面的方式在运行 gdb 时设置。

```
$ MALLOC_CHECK_=2 gdb hello
```

12.4.4 高级 gdb 命令

命令	命令用途的描述
(gdb) thread apply all bt	得到多线程程序的所有线程栈帧
(gdb) bt full	查看函数调用栈中的参数信息
(gdb) thread apply all bt full	和前面的选项一起得到堆栈和参数
(gdb) thread apply all bt full 10	得到前 10 个调用的栈帧和参数信息，以此来去除不相关的输出
(gdb) set logging on	把 gdb 的日志输出到文件 (默认的是”gdb.txt”)

Table 12.12: 高级 gdb 命令列表

12.4.5 调试与 X 相关的错误

如果一个 GNOME 程序 preview1 收到了一个 X 错误，您应当看见一条下面这样的信息。

```
’preview1’ 程序出现 X 桌面系统错误。
```

如果就是这种情况，你可以尝试在运行程序的时候加上”--sync” 选项，并且在”gdk_x_error” 函数处设置中断来获得栈帧信息。

12.4.6 检查库依赖性

按如下所示使用 ldd(1) 来找出程序的库依赖性。

```
$ ldd /bin/ls
        librt.so.1 => /lib/librt.so.1 (0x4001e000)
        libc.so.6 => /lib/libc.so.6 (0x40030000)
        libpthread.so.0 => /lib/libpthread.so.0 (0x40153000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

因为 ls(1) 运行在 `chroot`ed 环境，以上的库在 `chroot`ed 环境也必须是可用的。
参见第 9.3.6 节。

12.4.7 内存泄漏检测工具

Debian 上有一些可用的内存泄漏检测工具。

软件包	流行度	大小	说明
libc6-dev	V:276, I:613	18939	mtrace(1)：调试 glibc 中的 malloc
valgrind	V:7, I:52	80468	内存调试器和分析器
electric-fence	V:0, I:5	70	malloc(e) 调试器
leaktracer	V:0, I:3	56	C++ 程序内存泄露跟踪器
libdmalloc5	V:0, I:3	393	内存分配库调试

Table 12.13: 内存泄漏检测工具的列表

12.4.8 静态代码分析工具

如下是类似 lint 的静态代码分析工具。

软件包	流行度	大小	说明
splint	V:0, I:4	2239	静态检查 C 程序 bug 的工具
flawfinder	V:0, I:0	175	检查 C/C++ 源代码和查找安全漏洞的工具
perl	V:618, I:994	575	带有内部静态代码检测的解释器：B::Lint(3perl)
pylint	V:6, I:18	2668	Python 代码静态检查器
weblint-perl	V:0, I:1	32	用于 HTML 的小巧的语法检查器
linklint	V:0, I:0	343	快速的网站维护工具及链接检查器
libxml2-utils	V:22, I:289	173	使用 xmllint(1) 来检查 XML 文件

Table 12.14: 静态代码分析工具的列表

12.4.9 反汇编二进制程序

你可以使用下面的方式通过 objdump(1) 反编译二进制代码。

```
$  objdump -m i386 -b binary -D /usr/lib/grub/x86_64-pc/stage1
```

注意
gdb(1) 可以用来交互式反汇编代码。

12.5 Flex —一个更好的 Lex

Flex 是兼容 Lex 的快速语法分析程序生成器。

可以使用“`info flex`”查看 flex(1) 的教程。

你需要提供你自己的”`main()`” 和”`yywrap()`”. 否则，你的 flex 程序，看起来像这样的，编译的时候将不会带库。这是因为”`yywrap`” 是一个宏，”`%option main`” 隐性打开了”`%option noyywrap`”.

```
%option main
%%
.|\n    ECHO ;
%%
```

另外一种方法，在你的 cc(1) 命令行结尾，你可以使用编译链接器选项，”`-lfl`”。(像使用”`-ll`” 的 AT&T-Lex). 在这种情况下，不需要”`%option`”.

12.6 Bison —一个更好的 Yacc

在 Debian 里，有几个软件包提供 Yacc兼容的前瞻性的 LR 解析 或 LALR 解析的生成器。

软件包	流行度	大小	说明
bison	V:11, I:109	2253	GNU LALR 解析器生成器
byacc	V:0, I:6	160	伯克利 (Berkeley) LALR 解析器生成器
btyacc	V:0, I:0	207	基于 byacc 的回溯解析生成器

Table 12.15: 兼容 Yacc 的 LALR 解析器生成器列表

可以使用“`info bison`”查看 bison(1) 的教程。

你需要提供你自己的的”`main()`” 和”`yyerror()`”. 通常，Flex 创建的”`main()`” 调用”`yyparse()`”，它又调用了”`yylex()`”.

```
%%

%%
```

12.7 Autoconf

autoconf 是一种用于自动生成软件源代码包配置 shell 脚本的工具，以适应使用完整 GNU 构建系统的各种类 Unix 系统。

autoconf(1) 生成配置脚本“`configure`”。“`configure`”使用“`Makefile.in`”模板自动生成一个自定义的“`Makefile`”。

12.7.1 编译并安装程序

警告
当你安装编译好的程序的时候，注意不要覆盖系统文件。

Debian 不会在”/usr/local” 或”/opt” 目录下创建文件。如果你想要源码编译程序，把它安装到”/usr/local/” 目录下，因为这并不会影响到 Debian。

```
$ cd src
$ ./configure --prefix=/usr/local
$ make
$ make install # 这一步是把文件安装到系统上
```

12.7.2 卸载程序

如果你有源码并且它使用 autoconf(1)/automake(1)，如果你能记得你是怎样配置它的话，执行如下的命令来卸载程序。

```
$ ./configure ”all-of-the-options-you-gave-it”
# make uninstall
```

或者，如果你十分确信安装进程把文件都放在了”/usr/local/” 下并且这里没什么重要的东西，你可以通过如下的命令来清除它所有的内容。

```
# find /usr/local -type f -print0 | xargs -0 rm -f
```

如果你不确定文件被安装到了哪里，你可以考虑使用 checkinstall 软件包中的 checkinstall(8)，它将会提供一个清晰的卸载路径。现在，它支持创建带有“-D”选项的 Debian 软件包。

12.8 Perl 短脚本的疯狂

虽然任何 AWK 脚本都可以通过 a2p(1) 转换成 Perl，但单行的 AWK 脚本最好还是手动转换为单行的 Perl 脚本。

让我们来看看下面这个 AWK 脚本片段。

```
awk ’($2==”1957”) { print $3 }’ |
```

这等价于下列的任意一行。

```
perl -ne ’@f=split; if ($f[1] eq ”1957”) { print ”$f[2]\n”}’ |
```

```
perl -ne ’if ((@f=split)[1] eq ”1957”) { print ”$f[2]\n”}’ |
```

```
perl -ne ’@f=split; print $f[2] if ( $f[1]==1957 )’ |
```

```
perl -lane ’print $F[2] if $F[1] eq ”1957”’ |
```

```
perl -lane ’print$F[2]if$F[1]eq+1957’ |
```

最后一个简直就是个迷。它用上了下面列出的这些 Perl 的特性。

- 空格为可选项。
- 存在从数字到字符串的自动转换。

更多的命令行选项参见 perlrun(1)。想要更疯狂的 Perl 脚本，可以使用 Perl Golf。

12.9 Web

基本的动态交互网页可由如下方法制作。

- 呈现给浏览器用户的是 HTML 形式。
- 填充并点击表单条目将会从浏览器向 web 服务器发送带有编码参数的下列 URL 字符串之一。
  - "`http://www.foo.dom/cgi-bin/program.pl?VAR1=VAL1&VAR2=VAL2&VAR3=VAL3`"
  - "`http://www.foo.dom/cgi-bin/program.py?VAR1=VAL1&VAR2=VAL2&VAR3=VAL3`"
  - "`http://www.foo.dom/program.php?VAR1=VAL1&VAR2=VAL2&VAR3=VAL3`"
- 在 URL 里面"%nn" 是使用一个 16 进制字符的 nn 值代替。
- 环境变量设置为: "`QUERY_STRING="VAR1=VAL1 VAR2=VAL2 VAR3=VAL3"`".
- Web 服务器上的CGI程序 (任何一个"`program.*`") 在执行时，都会使用"`$QUERY_STRING`" 环境变量.
- CGI 程序的 `stdout` 发送到浏览器，作为交互式的动态 web 页面展示。

出于安全考虑，最好不要自己从头编写解析 CGI 参数的手艺. 在 Perl 和 Python 中有现有的模块可以使用. PHP 中包含这些功能. 当需要客户端数据存储时, 可使用HTTP cookies . 当需要处理客户端数据时, 通常使用Javascript.

更多信息，参见 通用网关接口, Apache 软件基金会, 和 JavaScript.

直接在浏览器地址中输入 http://www.google.com/search?hl=en&ie=UTF-8&q=CGI+tutorial 就可以在 Google 上搜索“CGI tutorial”。这是在 Google 服务器上查看 CGI 脚本运行的好方法。

12.10 源代码转换

源代码转换程序。

软件包	流行度	大小	关键词	说明
`perl`	V:618, I:994	575	AWK→PERL	把源代码从 AWK 转换为 PERL: a2p(1)
`f2c`	V:0, I:8	430	FORTRAN→C	把源代码从 FORTRAN 77 转换成 C/C++: f2c(1)
`intel2gas`	V:0, I:0	174	intel→gas	从 NASM (Intel 格式) 转换成 GNU 汇编程序 (GAS)

Table 12.16: 源代码转换工具列表

12.11 制作 Debian 包

如果你想制作一个 Debian 包，阅读下面内容。

- 第 2 章理解基本的包管理系统
- 第 2.7.13 节理解基本的移植过程
- 第 9.10.4 节理解基本的 chroot 技术
- debuild(1), pbuilder(1) 和 pdebuild(1)
- 第 12.4.2 节编译和除错
- Debian 新维护者指引 作为一个教程 (`maint-guide` 包)
- Debian 开发者参考手册 (`developers-reference` 包)
- Debian 策略手册 (`debian-policy` 包)
- Debian 维护者指引 (`debmake-doc` 包)

`debmake`, `dh-make`, `dh-make-perl` 等软件包，对软件包打包过程，也有帮助。

Appendix A

附录

本文档背景

A.1 Debian 迷宫

Linux 系统是一个面向网络计算机的功能强大的计算平台。然而，学习使用它的全部功能并非易事。使用非 PostScript 的打印机配置 LPR 打印机队列，就是个好例子。(自从使用新 CUPS 系统的新的安装系统出现后，就不再会有问题。)

有一张完整而详尽的地图叫做“SOURCE CODE”，它非常准确但极难理解。还有一些参考书叫 HOWTO 和 mini-HOWTO，它们易于理解，但给出了太多细节反而让人忘记了大方向。为了使用某个命令，我有时得在长长的 HOWTO 中找上半天。

我希望这个“Debian 参考手册 (版本 2.76)”(2019-03-21 15:39:20 UTC) 能帮助在 Debian 迷宫里面徘徊的人们，为他们提供一个好的出发方向。

A.2 版权历史

Debian 参考手册由我 (Osamu Aoki<osamu at debian dot org>) 发起，将其作为一个个人系统管理笔记。其中的许多内容都是我从Debian 用户邮件列表和其他 Debian 相关资源获得的积累。

在采纳了来自 Josip Rodin 的建议之后 (Josip Rodin 在Debian 文档项目 (DDP) 中非常活跃)，“Debian 参考手册 (第一版，2001-2007)”成为了 DDP 文档的一员。

6 年后，我意识到原来的“Debian 参考手册 (第一版)”内容陈旧，便开始重新很多内容。新的“Debian 参考手册 (第二版)”在 2008 年发布。

教程的起源和灵感，可以通过下面的内容来追溯。

- ”Linux 用户手册” Larry Greenfield (1996 年 12 月)
  - 该文档被后来的《Debian 教程》取代
- ”Debian 教程” Havoc Pennington (1998 年 12 月 11 日)
  - 部分由 Oliver Elphick, Ole Tetlie, James Treacy, Craig Sawyer 和 Ivan E. Moore II 书写
  - 该文档被后来的《Debian GNU/Linux: 安装和使用手册》取代
- ”Debian GNU/Linux: 安装和使用手册” John Goerzen 和 Ossama Othman (1999)
  - 该文档被《Debian 参考手册 (第一版)》取代

软件包和文档描述的一些起源和灵感，能够通过下面的内容来追溯。

- ”Debian FAQ” (2002 年 3 月版本，当时是由 Josip Rodin 维护)

其它内容的一些起源和灵感，能够通过下面的内容来追溯。

- ”Debian 参考手册 (第一版)” Osamu Aoki (2001–2007)
  - 于 2008 年被这个新的”Debian 参考手册 (第二版)” 取代。

先前的“Debian 参考手册 (第一版)”由许多贡献者创建。

- Thomas Hood 是网络配置主题的主要内容贡献者
- Brian Nelson 突出贡献了关于 X 和 VCS 的相关主题
- Jens Seidel 对构建脚本和许多内容的更正提供了帮助
- David Sewell 进行了大量的校对
- 来自翻译者、贡献者和 bug 报告者的许多贡献

Debian 系统中许多的帮助页面和信息页面被用来作为这个文档的主要参考. 在一些修正的程序上也考虑了公平使用, 它们中的许多地方, 尤其是命令的定义, 被经过精心的编辑, 以适应这些样式和作为本文档的短语部分.

gdb 调试器的描述使用了扩展Debian 维基内容的回溯系统, 这是被 Ari Pollak, Loïc Minier, 和 Dafydd Harries 同意的.

除了上面提到的部分之外，“Debian 参考手册 (版本 2.76) (2019-03-21 15:39:20 UTC)”的大部分内容是我自己的工作。一些贡献者也会对内容进行更新。

作者 Osamu Aoki 在此感谢所有在文档写作过程中曾给予帮助的人。

A.3 文档格式

英文原始文档的源文件目前是使用 AsciiDoc 文本文件来书写。AsciiDoc 使用起来非常便利，因为它比起整齐的 XML，键入比较少，并且用非常直观的方式支持表格。你可以认为 XML 和 PO 文件是真正的源文件。通过构建脚本，他们被转换成 DocBook XML 格式，自动产生的数据会被插入，并形成一个最终的 Docbook XML 源。这个最终的 Docbook XML 源文件能够被转换成 HTML, epub, 纯文本, PostScript, 和 PDF. (发布的时候，会省略部分格式。)

A.4 简体中文翻译

该文档的简体中文翻译，通过 Debian 简体中文邮件列表召集讨论，具体翻译工作通过 weblate 进行。欢迎大家继续通过 weblate 参加翻译: `https://hosted.weblate.org/projects/debian-reference/translations/-zh_CN/`

该项目继续征集校对人员, 欢迎大家在 weblate 参与, 有翻译得不好的地方, 可以直接修改。

Debian 官方网站也及时同步了我们的最新翻译成果: `https://www.debian.org/doc/manuals/debian-reference/index.zh-cn.html`

该手册软件包名字为：debian-reference-zh-cn

官方网页为： `https://packages.debian.org/testing/doc/debian-reference-zh-cn`

提示
在 testing 版里面, 软件包更新比较及时，大家如果在 apt 源里面设置了 testing 源, 则可以直接用 apt-get install debian-reference-zh-cn 命令安装该软件包。安装软件包后，就可以在本机看 pdf 格式 (/usr/share/debian-reference/debian-reference.zh-cn.pdf) 的文档。

对该手册翻译的任何问题或建议, 欢迎大家在 Debian 简体中文邮件列表讨论:

- debian-chinese-gb dot lists.debian.org
- debian-l10n-chinese dot lists.debian.org

《Debian 参考手册》(第二版) 翻译相关数据统计如下:

统计信息截止到:2017-09-19

(一) 英文原版情况

手册英文有 7638 个字符串，82658 个词。英文版 pdf 文件，有 271 页。

(二) 翻译耗时

zh-cn.po 文件 git 初始提交日期为:

```
commit 20948e15ffa005b6b4b6c6dd36f2833f01368f09
Author: Faris Xiao
Date:   Wed Jun 29 18:59:03 2016 +0800

    Chines init version po copy from templates.pot
```

我们在近 15 个月的时间内，完成了全部翻译。

(三) git 提交数量

weblate 和手工，总共有 688 次 git 提交。

```
git log zh-cn.po|grep commit|wc -l
688
```

(四) 参与情况

从 weblate 和 git 日志统计，先后有 26 位翻译贡献者。

贡献者的名字是:

chimez Dongliang Mu John Zhang Liang Guo zlffcn Zunway 孤月蓝风李 ZQ Anthony Fok mao CGH Jiagang Xu rainysia Xie Yanbo Zhang Rui zhangmiao wenqin chen Zongren Zhang scmarxx Boyuan Yang zlf chiachen Philip Ye 吴昊昱 Lou Letian 肖盛文