目录

第 1 章 - 序言.....4
1.1 官方文档.....4
1.2 文档约定.....4
1.3 脚本样例.....5
1.4 基本设置.....5
1.5 Ubuntu 发行版（distributions）基本概念.....5
第 2 章 - Ubuntu 基础.....6
2.1 Ubuntu 文件.....6
2.2 Ubuntu 软件包管理系统.....9
2.3 Ubuntu 系统升级.....14
2.4 Ubuntu 系统引导进程.....16
2.5 多样性支持.....17
2.6 国际化.....17
2.7 Ubuntu 和系统内核.....17
第 3 章 - Ubuntu 系统安装提示.....19
3.1 常规 Linux 安装提示.....19
3.2 Bash 设置.....26
3.3 鼠标设置.....26
3.4 NFS 设置.....29
3.5 Samba 设置.....29
3.6 打印机设置.....30
3.7 桌面 PC 的 CRON.....31
3.8 其它主机安装提示.....31
第 4 章 - Ubuntu 指南.....34
4.1 开始了.....34
4.2 Midnight Commander (MC).....37
4.3 类 Unix 工作环境.....39
4.4 类 Unix 文本处理.....49
4.5 类 Unix 文件系统.....52
4.6 X 窗口系统.....59
4.7 进一步学习.....60
第 5 章 - 发行版升级到 Breezy、Dapper 或 Edgy.....60
5.1 升级到 Hoary.....61
5.2 准备升级工作.....61
5.3 升级.....61
第 6 章 - Ubuntu 软件包管理.....62
6.1 介绍.....63
6.2 Ubuntu 软件包管理基础.....64
6.3 Ubuntu 生存命令.....67
6.4 Ubuntu 必杀技.....70
6.5 其他 Ubuntu 的特性.....76
第 7 章 - Ubuntu 下的 Linux 内核.....78
7.1 内核编译.....78

7.2 模块化的 2.4 内核.....80
7.3 通过 proc 文件系统调整内核.....83
7.4 2.6 版内核和 udev.....84
第 8 章 - Ubuntu 小技巧.....84
8.1 启动系统.....84
8.2 活动记录.....87
8.3 拷贝及创建子目录.....88
8.4 差异备份与数据同步.....90
8.5 系统冻结恢复.....91
8.6 记住这些可爱的小命令.....91
8.7 需要注意的典型错误.....107
第 9 章 - Ubuntu 系统微调.....107
9.1 系统初始化.....107
9.2 访问限制（Restricting access）.....109
9.3 刻录机.....112
9.4 X.....116
9.5 SSH.....130
9.6 邮件.....133
9.7 本地化（localization）.....137
9.8 多语言化（Multilingualization, m17n）.....145
第 10 章 - 网络设置.....146
10.1 IP 网络设置基础.....146
10.2 底层网络设置.....148
10.3 命名主机.....153
10.4 域名服务(DNS).....153
10.5 使用 DHCP 来配置网络接口.....155
10.6 Debian 的高级网络设置.....156
10.7 处理内核对接口命名的不一致性.....161
10.8 启动(triggering)网络设置.....162
10.9 多阶段(Multi-stage)映射.....165
10.10 网络服务设置.....166
10.11 网络故障排除.....168
10.12 建立路由网关.....168
第 11 章 - 编辑器.....171
11.1 流行的编辑器.....171
11.2 应急的编辑器.....172
11.3 Emacs 和 Vim.....172
第 12 章 - 系统版本控制.....177
12.1 并行版本系统 (CVS).....177
12.2 Subversion.....181
第 13 章 - 编程.....183
13.1 从哪儿开始.....183
13.2 Shell.....183
13.3 Awk.....188

13.4 Perl.....189
13.5 Python.....190
13.6 Make.....191
13.7 C.....192
13.8 Web.....196
13.9 准备文档.....197
13.10 打包.....200
第 14 章 - GnuPG.....201
14.1 安装 GnuPG.....201
14.2 使用 GnuPG.....202
14.3 管理 GnuPG.....202
14.4 在应用程序中使用 GnuPG.....203
第 15 章 - Ubuntu 技术支持.....203
15.1 参考资料.....203
15.2 查词意.....208
15.3 查找流行的 Debian 软件包.....208
15.4 Debian bug 跟踪系统.....208
15.5 邮件列表.....208
15.6 Internet Relay Chat (IRC).....208
15.7 搜索引擎.....209
15.8 网站.....209
附录 A - 附录.....210
A.1 作者.....210
A.2 保证.....213
A.3 反馈.....213
A.4 文档格式.....213
A.5 Debian 迷宫.....213
A.6 Debian 引言.....214
来源.....214

第 1 章 - 序言

本书《Ubuntu 参考手册》的目的是对整个 Ubuntu 系统作一个全面的介绍，提供一本“安装之後的”用户使用手册。本手册的读者应该愿意阅读 shell 脚本（scripts）。我也假定读者在阅读之前已经具备了类 Unix 系统的基本操作技能。

我决定**不**在本书中解释所有的细节，因为你可以在 manual 页、info 页或 HOWTO 文档中获得这些信息。我希望能给读者提供实用的信息，而非全面的解释。因此我会在正文中给出实际的命令序列，或是在 examples/ 给出脚本示例作为参考。在按照这些示例下达命令之前，你必须要先理解其内容的含义。实际的命令序列可能会依你的系统的具体情况而有细微的差别。

书中的很多信息实际上是对在 参考资料，第 15.1 节 中列出的众多权威参考文献的引用和提示。

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

如果你是在寻找紧急情况下系统维护方面的帮助，请直接阅读 Ubuntu 生存命令，第 6.3 节。

1.1 官方文档

本书的最新官方版本可在 http://wiki.ubuntu.org.cn/UbuntuManual/ 访问到。

1.2 文档约定

文中许多信息通过简短的 bash 命令给出，以下是其排版格式约定：

```
# command in root account  root 用户命令
$ command in user account  普通用户命令
... description of action  命令动作描述
```

这些 shell 命令的例子使用 PS2=" "。更多信息参见 Bash - GNU 标准交互式 shell，第 13.2.1 节。

参考：

- bash(1) 表示 Unix 风格 **manual** 页。
- info libc 表示 **GNU TEXINFO** 信息。
- *The C Programming Language* 表示**参考书目**。
- http://www.debian.org/doc/manuals/debian-reference/ 表示 **URL**。
- /usr/share/doc/Debian/reference/ 表示系统的**文件**。

文中用到了下列缩写：

- **LDP**: Linux Documentation Project (http://www.tldp.org/)
- **DDP**: Debian Documentation Project (http://www.debian.org/doc/)

本文中的其它缩写会在使用前定义。

文中只提供了 LDP 文档的 URL，然而，LDP 文档已经被 Ubuntu 打包。 当这些包被安装後，LDP 文档会在 /usr/share/doc/HOWTO/ 里。

参阅 参考资料, 第 15.1 节。

1.3 脚本样例

本文档的 脚本样例 需要通过网页访问下载。

1.4 基本设置

如果你的系统是最小安装，但你希望充分使用本文档， 请执行下面的命令来安装其它软件包，这些软件包含有一些有用的文档。

```
# apt-get install info man-db doc-base dhelp apt apt-utils auto-apt \
dpkg less mc ssh nano-tiny elvis-tiny vim sash \
kernel-package \
manpages manpages-dev doc-debian doc-linux-text \
debian-policy developers-reference maint-guide \
apt-howto harden-doc install-doc \
libpam-doc glibc-doc samba-doc exim-doc cvsbook \
gnupg-doc
# apt-get install debian-reference # for Sarge, do this too :)
```

对于 Woody 版，把 exim-doc-html 增加到上面的列表中。 对于 Sarge 版，请使用 exim4-doc-html 和 exim4-doc-info 来代替 exim-doc 包。

1.5 Ubuntu 发行版（distributions）基本概念

Ubuntu 软件仓库包含上千个软件包，它们按照我们提供的基础等级和是否符合自由软件哲学，被分成四种组件，即“main”（主要）、“restricted”（受限）、“universe”（公共）和“multiverse”（多元化）。

Ubuntu 软件仓库被分成四组，“main”、“restricted”、“universe”和“multiverse”，按照我们所能提供的基础能力以及它们是否符合 自由软件哲学 来划分。

标准 Ubuntu 安装都采用 main 和 restricted 的软件。你可以通过 Synaptic 软件管理器和 Aptitude 软件包管理器 安装其他的软件。通过编辑/etc/apt/sources.list 文件，就可以得到它们。如果要编辑 sources.list，请使用“man sources.list”来获得更多信息。

Ubuntu 发行版有代码名称，具体描述在 Ubuntu 发布版代号, 第 2.1.7 节 中。 Warty 在 2004 年 10 月发布， 在 Warty 发布後，三个发行版对应为 Hoary、 Breezy 和 Dapper。

订阅低流量的邮件列表 ubuntu-zh@lists.ubuntu.com, 可以得到关于 ubuntu 的重要声明信息。 参阅 Debian 文件, 第 2.1 节。

如果你想使用比发行版自带软件包更新的软件包版本， 你可以按照 发行版升级到 Hoary、Breezy 或 Dapper, 第 5 章 的描述，升级到一个新的发行版； 或者你只升级选择的软件

包。如果该软件包不能够容易的升级，你可以按照 向 Breezy 系统引入软件包, 第 6.4.10 节. 的说明，把该软件包向後移植到你现在所使用的发行版上。

如果使用混合型发行版，例如在 Breezy 中加入 Dapper ， 或是在 Hoary 中加入 Breezy，会一不留神从 Breezy 或 Dapper 中下载像 libc6 一样的核心软件包。没有保证能够使这些软件包不含有 bug ，请你高度注意！

运行 Breezy 或 Dapper 版本的 Ubuntu 意味着可能会遇到严重软件错误。包含一个稳定版 Ubuntu 的多启动方案可有效控制风险，另一个技巧是使用 chroot，详情参阅 chroot, 第 8.6.35 节。後者可以实际在不同的终端同时运行不同版本的 Ubuntu。

在 Ubuntu 基础, 第 2 章 中我们将讲解有关 Ubuntu 发行版的一些基本概念，之後，我将向你介绍一些基本信息以帮助你与最新软件快乐相处，并从 restricted 和 universe 组件中获益。心急的读者可以直接翻到 Ubuntu 生存命令, 第 6.3 节。祝你升级愉快！

第 2 章 - Ubuntu 基础

本章讲述非开发人员需要掌握的 Ubuntu 系统基础知识。有关知识的权威参考，请参阅：

- Desktop Guide
- Server Guide
- Install Guide

列表见 参考资料, 第 15.1 节。

如果你想查阅简要的“how-to”解释文档，可直接跳到 Ubuntu 软件包管理, 第 6 章 或其它相关章节。

本章的内容取自“Ubuntu FAQ”，经过较大的改编，以适于普通 Ubuntu 系统管理者上手。

2.1 Ubuntu 文件

2.1.1 目录结构

Ubuntu 软件包位于 Ubuntu 镜像站点 的目录树中，可通过 FTP 或 HTTP 访问它们。

下列目录存在于任何 Ubuntu 镜像站点的 ubuntu 目录下：

dists/::: 本目录包含“发行版”（distributions），此处是获得 Ubuntu 发布版本（releases）和已发布版本（pre-releases）的软件包的正规途径。有些旧软件包及 Contens-*.gz Packages.gz 等文件仍在其中。

pool/::: 所有 Ubuntu 发布版及已发布版的软件包的新的物理地址。

indices/::: 维护人员文件和重载文件。

project/::: 大部分为开发人员的资源，如： project/experimental/::: 本目录包含了处于开发中的软件包和工具，它们均处于 alpha 测试阶段。用户不应使用这些软件，因为即使是经验丰富的用户也会被搞得一团糟。 project/orphaned/::: 已不再有人维护的软件包，它们已从发行版中孤立出来。

2.1.2 Ubuntu 发行版

现在在 dists 目录下有三个 Ubuntu 发行版。它们是“hoary”发行版，“breezy”发行版，和“dapper”发行版。有时还有一个“edgy”发行版。其中也包含了一个现不再支持"warty"发行版。

到 2007 年 8 月，Ubuntu 发行版本又增加了 feisty 和 gutsy。

2.1.3 hoary 发行版

hoary 2005 年 4 月发布，其发行版软件包入口：

- hoary/main/:“main”组件包括了自由软件、可以被自由发布的软件和被 Ubuntu 团队完全支持的软件。其中包括了大多数流行的和稳定的开源软件，当您安装 Ubuntu 时默认安装的就是这些软件。
- hoary/restricted/:“restricted”组件是专供普遍用途，而且没有自由软件版权，但依然被 Ubuntu 团队支持的软件。请注意，因为不能直接修改软件，因而我们可能不能提供完全的技术支持，即便如此，我们还能向实际作者反馈 Bug 报告。
- hoary/universe/:“universe”组件是整个自由、开源 Linux 世界的缩影。在“universe”组件中，你可以找到大多数开源 软件，以及在开源版权下的软件，所有这些都是在公共源的基础上建立的。这些软件都是使用“main”中的组件编写的，它们能与“main”组件相安无事地 共同运行，但它们没有安全升级的保障。“universe”组件包含了数以千计的软件。虽然是公共的，用户必须明白它们与稳定的 Ubuntu 核心的软件的 差异和不稳定。
- hoary/multiverse/: multiverse”组件包含了“不自由”的软件，这意味着这些软件不满足 Ubuntu 相对于“main”组件的各种版权政策。当您使用这些软件时，如何调整各种权力以及遵重版权所有者的问题，就完全依靠您自己把握了。 这些软件不被我们支持，而且通常不能被修改和更新，您将自己承担任何风险。

现在，作为以上位置的新增功能，实际上新的软件包都存储在 pool 目录中（ pool 目录, 第 2.1.10 节）。

2.1.4 breezy 发行版

breezy 2005 年 10 月发布。现在，除了上述目录，新上载的软件包的物理存储位置为 pool 目录（ pool 目录, 第 2.1.10 节）。在 breezy 下同样有 main、restricted、universe 和 multiverse 子目录，它们的作用与 hoary 中的一样。

2.1.5 dapper 发行版

dapper 2006 年 6 月发布。提供了长达 3 年的支持，新版本以 Ubuntu 6.06 LTS(Long Term Support)(长期支持),它针对一些大组织的需求对桌面版和服务器版都做个一定程度上的增强。新上载的软件包的物理存储位置为 pool 目录（ pool 目录, 第 2.1.10 节）。在 dapper 下同样有 main、restricted、universe 和 multiverse 子目录，它们的作用与 hoary 中的一样。

2.1.6 edgy 发行版

edgy 发行版反映了系统的最新开发进展。欢迎广大用户使用并测试这些软件包，同时也提醒你们这些软件包还不完善。使用 edgy 发行版的好处就是你可以获得 Ubuntu 项目的最新更新—不过新东西也会出新问题，你得好坏兼收:-)

2.1.7 Ubuntu 发布版其它仓库

存在于 dists 目录下的物理目录名，例如 breezy-updates 和 breezy-security，就是“附加的仓库”。当某个 Ubuntu 发行版处于维护阶段。将包的必要升级和安全更新放在附加的仓库里。其附加仓库通常有 ***-updates、***-security、***-proposed、***-backports。

2.1.8 已用过的发布版代号

已使用过的发行版代号有：Ubuntu 4.10 (The Warty Warthog)、Ubuntu 5.04 (The Hoary Hedgehog)、 Ubuntu 5.10 (The Breezy Badger)、 Ubuntu 6-06 (The Dapper Darke)。

2.1.9 发布版代号

- Warty Warthog 多疣的疣猪，
- Hoary Hedgehog 灰白的刺猬，
- Breezy Badger 活泼的獾，
- Dapper Darke 漂亮的鸭子，
- Edgy Eft 躁动的蜥蜴

2.1.10 pool 目录

过去，软件包均放在 dists 目录下相应发行版的子目录中。这种做法产生了许多问题，当镜像站点进行新版本发布时大量带宽被消耗。

现在软件包均放进一个巨大的“池子（pool)”，按照源码包名称分类存放。为了方便管理，pool 目录下按属性再分类（main、 restricted、universe 和 multiverse)，分类下面再按源码包名称的首字母归档。这些目录包含的文件有：运行于各种系统架构的二 进制软件包，生成这些二进制软件包的源码包。

你可以执行命令 apt-cache showsrc mypackagename，查看“Directory:”行获知每个软件包的存放位置。例如：apache 软件包存放在 pool/main/a/apache/软件包存放在 lib*软件包数量巨大，它们以特殊的方式归档：例如，libpaper 软件包存放在 pool/main/libp/libpaper/。

诸如 apt 等命令访问的索引文件仍位于 dists 目录中。

通常，你大可不必注意这些，新版的 apt 和旧版 dpkg-ftp 会自动处理它们。

2.1.11 发布源目录结构

在每个主目录树下（dists/dapper/main、dists/dapper/restricted、dists/dapper/universe、dists/dapper/multiverse/，等）按芯片架构又分了子目录，每个子目录中存放着在该 芯片架构下编译的二进制软件包。

- binary-amd64/ AMD64 或 EM64T 架构
- binary-i386/ 所有的使用 Intel/AMD/等 处理器的 PC，
- binary-powerpc/ Apple Macintosh G3, G4, 和 G5
- binary-sparc/ Sun UltraSPARC 系统

2.1.12 源代码

Ubuntu 系统中的 main,universe 组件有源代码，不仅如此，许可证条款**规定**系统中所有的程序必须和其源代码一起发行，或提供源代码出售。

通常源代码发布在 source 目录，该目录同时处于所有架构目录中，更新的源码则在 pool 目录中（参阅 pool 目录, 第 2.1.10 节）。对于不太熟悉 Ubuntu 归档目录结构的用户想获得源代码可以试试 apt-get source mypackagename 命令。

有些软件包，如着名的 pine，由于许可证限制，只提供源码包。（最近，pine-tracker 软件包提供了一个简易的安装版）安装源码包的方法可参阅 向系统引入软件包, 第 6.4.10 节, 打包, 第 13.10 节教你如何手工创建一个软件包。

restricted 和 multiverse 目录中的软件包可能不提供源代码，因为它们没有正式加入 Ubuntu 系统。

2.2 Ubuntu 软件包管理系统

2.2.1 Ubuntu 软件包概述

软件包通常包含了实现一系列相关命令或特性所必须的所有的文件。Ubuntu 软件包采用了和 Debian 相同的软件包格式，有两种类型的软件包：

- **Binary packages**（二进制软件包），它包含可执行文件、配置文件、man/info 页面、版权声明和 其它文档。这些软件包以 Ubuntu 特定的格式发布（参阅 Ubuntu 软件包格式, 第 2.2.2 节）；它们通常使用.deb 的扩展名以示区别。这种二进制软件包可使用 Ubuntu 工具 dpkg 解包，详情见有关帮助页面。
- **Source packages**（源码包），包含一个.dsc 文件它用于描述源码包（包括下列文件的名称），一个. orig.tar.gz 文件它是未经修改的原始源代码压缩文件，以及一个.diff.gz 文件它包含了该软件包 Ubuntu 化时所做的修改。dpkg-source 工具可用于打包/解包 Ubuntu 源码包，详情可参阅有关帮助页面。

软件包管理系统安装的软件包时需要使用“倚赖关系”，它由软件包维护者声明。这些信息记录在与每个软件包关联的 control 文件中。例如，包含 GNU C 编译器（gcc）的软件包依赖于包含链接器和汇编器的 binutils 软件包。如果用户试图在没有安装 binutils 的情况下安装 gcc，软件包管理 系统（dpkg）将会显示一条错误信息，告诉你需要安装 binutils, 并停止安装 gcc。（不过，倔强的用户可以对这个信息视而不见，参阅 dpkg (8)）。）更多信息，参阅下面的章节 软件包依赖关系, 第 2.2.8 节。

Ubuntu 软件包管理工具可用于：

- 操作和管理软件包或软件包的局部内容，
- 帮助那些使用有限容量载体如软盘传输的用户分割软件包，
- 帮助开发者将开发文件打包成软件包，
- 帮助用户从远程 Ubuntu 文档站点安装软件包。

2.2.2 Ubuntu 软件包格式

Ubuntu “软件包”，或称之为 Debian 包文件（ Debian archive file），包含了可执行文件、库文件、和相关程序的文档。通常 Ubuntu 文件的文件名以.deb 结尾。

Ubuntu 二进制软件包内部格式描述见 deb(5)帮助页面。所以要操作.deb 文件请参阅 dpkg-deb(8)。

2.2.3 Ubuntu 软件包命名约定

Ubuntu 软件包命名遵循下列约定：

foo_ver-rev_arch.deb
一般这里的 foo 是软件包的名称，ver 是软件本身的版本号，rev 是 Ubuntu 修订版本号，arch 是目标架构名称。当然，文件很容易被改名；不过，你也可以通过运行下面的命令来找出文件 filename 实际是那个软件包：

dpkg --info filename

Ubuntu 修订版本号由 Ubuntu 开发者或创建这个软件包的人指定。通常，包被修改过之後，会把修改版本号加一，有些也会加上 ubuntu 的标志以便和 Debian 区分开来。

2.2.4 保存本地配置

有可能被本机管理员修改的文件保存在/etc/目录中。 Ubuntu 策略中规定所有对本地配置文件的修改都可以在软件包升级过程中被保留下来。

在软件包的发布中，如果包含默认的本地配置文件，这个文件就被称为“conffile”（默认配置）。如果不得到管理员的允许，软件包管理 系统不会对上次 安装之後被修改过的默认配置进行升级；不过，如果管理员没有改动过默认配置，那么它就会被升级成最新软件包中的版本。这种策略几乎总是合理的，它有益于把 默认配置的改动减到最小。

下面的命令可以列出一个软件包中包含那些默认配置文件：

dpkg --status package

文件列表位于“Conffiles”的後面。

在《 Ubuntu 策略手册》的“配置文件”一节，可以获得有关 conffile（默认配置）文件的更多信息（参见 参考资料, 第 15.1 节）。

2.2.5 Ubuntu 维护脚本

Ubuntu 维护脚本是一种可执行脚本，它在软件包安装之前或之後自动运行。它和一个名叫 control 的文件一起组成 Ubuntu 包文件的“管理”部分。

这些文件是：

preinst:: 在 Ubuntu 包文件解包之前，运行这个脚本。许多“preinst”脚本的任务是停止作用于待升级软件包的服务，直到软件包安装或升级完成。

postinst:: 该脚本的任务是完成 Debian 包文件解包文件的配置工作。通常，“postinst”脚本等待用户输入，或提醒用户，如果他接受当前默认值，要记得软件包安装完後返回重新配置。许多 “postinst”脚本负责执行有关命令为新安装或升级的软件重启服务。

prerm:: 该脚本负责停止与软件包关联的 daemon 服务。它在删除软件包关联文件之前执行。

postrm:: 该脚本负责修改软件包链接或文件关联，或删除由它创建的文件。(参阅虚拟软件包, 第 2.2.7 节)。

当前，所有的管理文件都存放在/var/lib/dpkg/info 目录。与 foo 软件包相关的文件，名字以“foo”打头，以 “preinst"、“postinst”等为扩展名。目录中的 foo.list 文件列出了 foo 软件包安装的所有文件。(注意这些文件的位置在由 dpkg 来确定，可能会因 Debina 版本而异)

2.2.6 软件包优先级

每个 Ubuntu 软件包均被发布者指点了一个**优先级**,作为软件包管理系统 的一个辅助参数,优先级的值有：

- **Required**（必须） 该级别软件包是保证系统正常运行必须的。

包含所有必要的系统修补工具。不要删除这些软件包，否则整个系统将受到损坏，甚至无法使用 dpkg 恢复。只安装 Required 级软件包的系统不可能满足所有的用途，但它可以启动起来，让系统管理员安装想要的软件。

- **Important**（重要） 在任何类 Unix 系统上均安装有该级别软件包。

系统若缺少这类软件，会运行困难或不好操作。该级别软件包并**不**包括 Emacs 或 X11 或 TeX 或其它大型应用程序，它们只是一些实现系统底层功能的程序。

- **Standard**（基本） 该级别软件包是任何 Linux 系统的标准件，它们组成一个小而精的字符模式的系统。

系统的默认安装就包括了它们。“Standard”级软件包不包括许多大型应用程序，但它包括 Emacs（它比其它应用程序更底层）和 Tex 及 LaTeX 的精巧版（不支持 X)。

- **Optional**（推荐） 该级别软件包包括那些你可能想安装的软件，即使对它们并不熟悉,但对它们没有特殊的要求。

它们包括 X11，TeX 完整发布版和许多应用程序。

- **Extra**（额外） 该级别软件包可能与其它高级别软件包冲突，仅当你知道其用途时才会使用它们，或者有运行它们有专门要求，这些都使它们不适合定为“Optional"级。

请注意软件包描述中“Priority: required"（优先级：必须)、“Section: base"（组件：基本)、“Essential: yes"（必要：是）的区别。“Section: base"（组件：基本）意味着在安装新系统时这个软件包要先于其它所有软件安装。大多数在“Section: base”中的软件包都被打上了“Priority: required"（优先级：必须）标签，或者至少是“Priority: important"（优先级：重要)；并且其中的很多也同时具有“Essential: yes"（必要：是）标签。“Essential: yes”意味着要用软件包管理系统的 dpkg 等程序删除它时，必须给出额外的强制选项才行。比如，libc6、mawk 和 makedev 软件包属于 “Priority: required" 和“Section: base"，但不是“Essential: yes"。

2.2.7 虚拟软件包

虚拟软件包是一个统称，它代指一组具有相近功能的软件包中的任何一个。例如，tin 和 trn 都是新闻组阅读软件，当系统中某个程序需要使用新闻阅读 器时，它们中的任何一个都可以满足要求。因此，这两个软件包一起提供了一个叫做 news-reader（新闻阅读器）的“虚拟软件包"。

类似的，许多 exim、exim4、sendmail、postfix 这样的软件包都提供邮件传输代理的功能。因此，它们在一起提供了一 个称为 mail-transport-agent（邮件传输代理）的虚拟包。安装了它们中的任何一个，都会满足其它倚赖于邮件传输代理功能的程序的需要。

Ubuntu 有个机制，如果系统中提供同种虚拟包的软件包安装了多个，系统管理员可以指定一个为首选软件。相关的命令是 update-alternatives，更详细的描述参阅 Alternative 命令，第 6.5.3 节。

2.2.8 软件包依赖关系

Ubuntu 软件包管理系统依赖声明，它描述了这一事实：一些软件包需要其它软件包被安装才能正常运行或运行得更好。

- 软件包 A **依赖**（depends）软件包 B：要运行 A 必须安装 B。在有些情况下，A 不仅依赖 B，还要求 B 的特定版本。版本依赖通常有最低版本限制，A 更依赖于 B 的最新版而非某个特定版本。
- 软件包 A **推荐**（recommends）软件包 B：软件包维护者认为所有用户都不会喜欢缺少某些功能的 A，而这些功能需要 B 来提供。
- 软件包 A **建议**（suggests）软件包 B：B 中某些文件与 A 的功能相关，并能够增强 A 的功能。这种关系通过声明软件包 B **增强 Enhances** 软件包 A 来表示。
- 软件包 A 与软件包 B **冲突**（conflicts）：如果系统中安装了 B 那么 A 无法运行。“Conflicts”常和“replaces”同时出现。
- 软件包 A **替换**（replaces）软件包 B：B 安装的文件被 A 中的文件移除和复盖了。
- 软件包 A **提供**（provides）软件包 B：A 中包含了 B 中的所有文件和功能。

上述术语使用方法的更详细的信息参阅 *Packaging Manual* 和 *the Policy Manual*。

注意，dselect 可以对 **recommends** 和 **suggests** 类软件包进行细操作，apt-get 只会简单的下载安装 **depends** 类软件包而不管 **recommends** 和 **suggests** 类软件包。这两个程序均正式使用 APT 作为其後台。

2.2.9 何为“Pre-depends”

dpkg 总是在配置一个有依赖关系的包之前，先对被依赖的包进行配置。 然而，dpkg 通常将归档文件随意解包，不顾依赖性。 （从归档文件中解包并提取文件，将他们放置到正确的位置。） 如果是 **Pre-Depends** 包，则在所依赖的其它包被解包和配置之前，Pre-Depends 包不会被解包。 使用这种依赖的目的是为了将依赖复杂性降至最底。

2.2.10 软件包状态

软件包有各种状态：“unknown”，“install”，“remove”，“purge”和“hold”。这些“希望”标记描述了用户打算如何操作这些软件包（既可以使用 dselect 的“Select”菜单，也可以直接调用 dpkg）。

它们的意思是：

- **unknown** - 用户并没描述他想对软件包进行什么操作。
- **install** - 用户希望对软件包进行安装或升级。
- **remove** - 用户希望删除软件包，但不想删除任何配置文件。
- **purge** - 用户希望完全删除软件包，包括配置文件。
- **hold** - 用户希望软件包保持现状，例如，用户希望保持当前的版本，当前的状态，

当前的一切。

2.2.11 阻止软件包升级

有两种方法阻止软件包升级，使用 dpkg，或者在 Woody 中使用 APT。

使用 dpkg，首先导出软件包选择列表：

```
dpkg --get-selections \* > selections.txt
```

接着编辑文件 selections.txt，修改想要恢复的软件所在的行，例如 libc6，则将：

```
libc6                           install
```

改为：

```
libc6                           hold
```

保存文件，将它装入 dpkg 数据库：

```
dpkg --set-selections < selections.txt
```

或者，如果你知道要恢复的软件包名称，执行：

```
echo libc6 hold | dpkg --set-selections
```

这个命令将在每个软件包安装过程中保持该软件包不变。

使用 dselect 也可以达到同样的效果。进入[S]elect 屏幕，找到想阻止其升级的软件包，按“=”键（或者“H”）。更改在你退出[S]elect 屏幕後立即生效。

Woody 中的 APT 系统有一个新机制来阻止软件包升级，在下载升级档进程中使用 Pin-Priority。参阅帮助页面 apt_preferences(5)，或阅读 http://www.debian.org/doc/manuals/apt-howto/或 apt-howto 软件包。

2.2.12 源码包

源码包发布在 source 目录中，既可以手工下载可以使用

```
apt-get source foo
```

获取它们（参阅 apt-get(8)帮助页面）。

2.2.13 编译源码包

对于 foo 软件包，从源码编译需要 foo_*.dsc、foo_*.tar.gz 和 foo_*.diff.gz 文件（注意，对于由 Debian 开发的软件包，没有.diff.gz 文件）。

当你得全了这些文件，如果你这安装了 dpkg-dev 软件包，运行：

```
$ dpkg-source -x foo_version-revision.dsc
```

它将软件包解压到一个名为 foo-version 的目录。

使用下列命令创建二进制软件包：

```
$ cd foo-version
$ su -c "apt-get update ; apt-get install fakeroot"
$ dpkg-buildpackage -rfakeroot -us -uc
```

然後，

```
# su -c "dpkg -i ../foo_version-revision_arch.deb"
```

安装新编译的软件包。参阅["../ch-package.zh-cn.html: 向系统引入软件包, 第 6.4.10 节]。

2.2.14 新建 Ubuntu 软件包

创建新软件包的详细信息，参阅 *New Maintainers' Guide*， 该文档在 maint-guide 包中，或浏览 http://www.debian.org/doc/manuals/maint-guide/。

2.3 Ubuntu 系统升级

Ubuntu 的设计目标之一就是提供一个平滑、安全和可靠的升级过程。软件包系统在升级过程中会将重要改变警告系统管理员，在某些情况下，会要系统管理员来做决定。

你也应该阅读发布手记（Release Notes），它存在于所有的 Ubuntu 光盘中，也可以通过互联网访问 http://www.Ubuntu.com/releases。

Ubuntu 软件包管理, 第 6 章提供了升级的实用指南，本节只提供一个大纲，由包工具开始 。

2.3.1 dpkg

它是操作软件包文件的主要工具；参阅 dpkg(8) 获得完整信息。

dpkg 由几个原始的辅助程序演化而来。

- dpkg-deb：操作.deb 文件。dpkg-deb(1)
- dpkg-ftp：一个旧的软件包获取命令。dpkg-ftp(1)
- dpkg-mountable：一个旧的软件包获取命令。dpkg-mountable(1)
- dpkg-split：将大软件包分割成小包。dpkg-split(1)

dpkg-ftp 和 dpkg-mountable 已被新的 APT 系统取代。

2.3.2 APT

APT（the Advanced Packaging Tool）是 Ubuntu 软件包管理系统的高级界面，由几个名字以“apt-”打头的程序组成。apt-get、apt-cache 和 apt-cdrom 是处理软件包的命令行工 具，它们也是其它用户前台程序的後端，如 dselect 和 aptitude。

更多信息，可安装 apt 软件包後阅读 apt-get(8)、apt-cache(8)、apt-cdrom(8)、apt.conf(5)、sources.list(5)、apt_preferences(5)(woody)以及 /usr/share/doc/apt/guide.html/index.html。

另一个资源是 APT HOWTO，如果安装了 apt-howto 包，可在/usr/share/doc/Debian/apt-howto/中找到它。

apt-get upgrade 和 apt-get dist-upgrade 只获取“Depends”类软件包，它忽略“Recommend”和“Suggests”类软件包，不想这样的话，可使用 dselect。

2.3.3 dselect

这个程序是 Ubuntu 软件包管理系统中菜单驱动的用户界面。特别适用于首次安装和大面积升级。参阅 dselect，第 6.2.4 节。

更多信息，可安装 install-doc 包後阅读/usr/share/doc/install-doc/dselect-beginner.en.html 或 dselect Documentation for Beginners。

2.3.4 不停机系统升级

Ubuntu 系统的内核（文件系统）支持替换使用中的文件。 当一个软件包升级时，如果由该软件包提供的服务在当前运行级下正在运行，则该服务将被重新启动。 Ubuntu 系统不要求用户在 single-user 模式下进行不停机升级。

2.3.5 下载和缓存.deb 文件

如果你手工下载包文件到硬盘(这完全没有必要，请阅读上面有关 dpkg-ftp 或 APT 的内容)，当你完成软件包安装工作後，可以从系统中删除.deb 文件。

如果是使用 APT，这些文件会缓存在/var/cache/apt/archives/目录中。你可以在完成安装後删除它们（apt-get clean）或者将它们拷贝到另一个机器的/var/cache/apt/archives/目录中以备以後的安装。

2.3.6 升级记录

dpkg 会对软件包的解包、配置、删除、完全删除进行记录，但不能（目前是这样）记录在包操作的过程中活跃终端的行为。

最简单的解决方法是在运行 dpkg、dselect、apt-get 等工具的会话中加入 script(1)程序。

2.4 Ubuntu 系统引导进程

2.4.1 init 程序

同所有的 Unix 一样， Ubuntu 启动要执行 init 程序。init 的配置文件（/etc/inittab)

中指定的第一个执行脚本应该是 /etc/init.d/rcS。

接下来将要发生要看是否安装了 sysv-rc 软件包或 file-rc 软件包。下面假设安装了 sysv-rc 软件包。（file-rc 含有它自己的 /etc/init.d/rcS 脚本，在 rc 目录里使用文件代替符号链接来控制哪个服务在哪个运行级别启动。）

sysv-rc 软件包里面的 /etc/init.d/rcS 运行 /etc/rcS.d/ 目录下的所有脚本来执行初始化，如： 检查并挂载文件系统、装载内核模块、启动网络服务、设定时钟等。 接着，为了兼容性考虑，它运行/etc/rc.boot/目录下的文件（除了那些文件名中包含“.”的文件），该目录中的脚本是保留给系统管理员使用，不 赞成使用该目录。 更多信息参见 系统初始化，第 9.1 节 和 Debian Policy Manual 中的 System run levels and init.d scripts

Ubuntu 没有使用 BSD 风格的 rc.local 文件。

2.4.2 运行级别

完成系统启动进程後，init 启动所有在默认运行级别配置为启动的服务。 默认运行级别由 /etc/inittab 中的 id 给出。 Ubuntu 使用 id=2。

Ubuntu 使用下列的运行级别：

- 1（单用户模式 single-user mode），
- 2 到 5（多用户模式 multi-user modes），
- 0（关闭系统），
- 6（重启系统）。

运行级 7、8 和 9 也能够使用，但是它们的 rc 目录在软件包安装的时候没有。

使用 telinit 命令来转换运行级别。

当进入一个运行级别时，所有在 /etc/rcrunlevel.d/ 目录下的脚本将被执行。 脚本名的第一个字母决定了该脚本的运行**方式**： 使用 K 开头的脚本，使用 stop 参数来运行。 使用 S 开头的脚本，使用 start 参数来运行。 这些脚本按照它们名字的字母顺序运行；因此，“stop”脚本比“start”脚本先运行。 在 K 或 S 之後的两个数字决定了脚本运行的先後次序，数字小的脚本先运行。

实际上，目录/etc/rcrunlevel.d/中的脚本都是指向/etc/init.d/的符号链接。这些脚本接受 “restart”和“force-reload”作为参数：“force-reload”的方式可以用来在系统启动後，重新启动服务或者强迫它们重新装 载它们的配置文件。

例如：

```
# /etc/init.d/exim4 reload
```

2.4.3 自定义运行级别

自定义运行级别是一个高级的系统管理任务。 下面的指示面向大部分服务。

在运行级 R 启动 service 服务，创建一个符号链接 /etc/rcR.d/Sxyservice 到目标文件 ../init.d/service。 xy 是序列号，是由软件包在安装的时候分配给服务的。

停止服务，重命名符号链接，将它的名字用 K 开头来代替 S， 它的序列号是 100 减 xy。 可以使用象 sysv-rc-conf 或 ksysv 这样的运行级别编辑器来方便的修改服务。

在一个特定的运行级别目录，可以将某个服务的 S 符号链接删除来代替重新命名它。 这种作法不停止该服务，但将把该服务留在一种 sysv-rc 初始化系统认为的“漂浮”状态： 当运行级别改变时，该服务即不会启动，也不会停止， 它将保留它原有的状态，不管它是在运行或者没有运行。 注意，处于这种漂浮状态的服务，如果它所属的软件包升级了， 这个服务将启动，不管它在升级前是否运行。 这是当前 Debian 系统一个已知的缺点。 注意：还需要在运行级 0 和 6 之间保留服务的 K 符号链接。如果删除了一个服务的所有符号链接，在升级该服务的软件包时， 所有的符号链接将恢复到它们的出厂默认模式。

不建议对 /etc/rcS.d/ 目录里的符号链接做任何改变。

2.5 多样性支持

Ubuntu 提供几种途径，在不破坏系统的前提下满足系统管理员各种要求。

- dpkg-divert，参阅 dpkg-divert 命令, 第 6.5.1 节。
- equivs，参阅 equivs 软件包, 第 6.5.2 节。
- update-alternative，参阅 Alternative 命令, 第 6.5.3 节。
- make-kpkg 可以支持多启动引导。参阅 make-kpkg(1)和 Ubuntu 标准方式, 第 7.1.1 节。

/usr/local/目录下的所有文件均属于系统管理员，Ubuntu 不会改动它们。/etc 下的大部分文件属于 conffiles，Ubuntu 不会在升级後复盖它们，除非系统管理员明确要求复盖。

2.6 国际化

Ubuntu 系统是国际化产品，不论是在控制台下还是在 X 下，它都提供多种语言的字符显示和输入支持。许多文档、帮助页面以及系统消息都被翻译成各种语言。在安装过程中，Ubuntu 提示用户选择安装语言（有时是当地语言变量）。

如果你安装的系统提供的语言特性支持不能满足你的需要，或者你要改变当前语言或安装别的键盘方案以适应你的语言，参阅 本地化（localization）, 第 9.7 节。

2.7 Ubuntu 和系统内核

参见 Ubuntu 下的 Linux 内核, 第 7 章。

2.7.1 编译非 Ubuntu 源码包内核代码

首先你必须了解 Ubuntu policy 有关头文件的规定。

Ubuntu C libraries 是依据**内核**头文件最新 **stable** 发布版创建的。

跟随内核源码发布的内核头文件位于/usr/include/linux/include/。

如果你编译某个程序所用的内核头文件比 libc6-dev 提供的头文件还新，在编译时你必须在命令行中加上- I/usr/src/linux/include/。这些情况是存在的，例如，对于 automounter daemon（amd）软件包而言，当新内核改变了对 NFS 的内部操作方式，amd 需要知道这些改变。这时就需要引用新的内核头文件。

2.7.2 自定义内核创建工具

对于希望（或必须）使用自定义内核的用户，推荐下载 kernel-package 软件包。该软件包包含了创建内核包的脚本。有了它，新建 Ubuntu 内核镜像包只需在内核源码目录的一级目录运行

```
# make-kpkg kernel_image
```

在内核源码所在目录的顶层，执行下述命令可获得有关帮助

```
# make-kpkg --help
```

或阅读 make-kpkg(8)帮助页面以及 Ubuntu 下的 Linux 内核, 第 7 章。

如果所需的 kernel-source-version(version 代表内核版本号)包不存在,用户就得从 Linux 文件站点分别下载最新的（或需要的）内核源码。

有关 kernel-package 包的用法详述见于/usr/doc/kernel-package/README。

2.7.3 多系统引导器

要使用多系统引导器如 grub 或 loadlin，请将编译好的 Linux 内核 bzimage 拷贝到相应的地方（例如/boot/grub 或 MS-DOS 分区）。

2.7.4 制做引导软盘

Ubuntu 的 mkrboot 软件包可以帮你制作一张自定义启动软盘，软件包中的 Shell 脚本按 syslinux 格式制作启动软盘。对于那些使用 MS-DOS 格式化的软盘，其主引导 扇区的记录将被修改为直接引导 Linux（或是其它在盘中 syslinux.cfg 文件里的操作系统）。该软件包中的其它脚本可制作急救盘甚至重建基本系 统。

2.7.5 模块加载规定

Ubuntu 的 modconf 软件包提供了一个 shell 脚本（/usr/sbin/modconf），它可以用来自定义内核模块配置。该脚本使用菜单界面，用户通过 它给出系统中可挂载设备驱动的有关细节，它再将这些细节信息生成/etc/modules.conf 文件（其中列出了别名 aliases 和其它参数，用于 连接各种模块），该配置文件用来加载/etc/modutils/目录下和/etc/modules（其中列出了需要在系统启动时加载的模块）目录的相关 模块。

新版的配置帮助文件 Configure.help 可为构造自定义内核提供帮助，同样，modconf 软件包中也有一系列帮助文件（位于/usr/share/modconf/目录下），告诉你如何对模块 设定合适的参数。参阅 模块化的 2.4 内核, 第 7.2 节中的例子。

2.7.6 卸载旧内核

uname -ra 可用来检查当前运行的内核版本，以确定是否与你打算卸载的内核版本相同。因此你可以使用如下命令删除不想要的内核镜像包：

```
# dpkg --purge --force-remove-essential linux-image-NNN
```

（当然，要将 NNN 替换成你的内核版本号和修订版号。）

第 3 章 - Ubuntu 系统安装提示

Ubuntu 的官方安装文档位于 http://doc.ubuntu.com/ubuntu/install/ ，里面包含了 amd64、hppa、i386、ia64、powerpc 和 sparc 的安装文档。中文翻译在 http://wiki.ubuntu.org.cn/ubuntu/install/

3.1 常规 Linux 安装提示

如果你在寻找精简的 Ubuntu 安装光盘，请别忘记检查 http://archive.ubuntu.com/ubuntu/dists/{发行版}/main/installer 架构}/current/images/netboot/。

使用最新的开发版的 Ubuntu 会增加遇到严重软件错误的风险。在机器上安装一个稳定版本的 Ubuntu 然后使用多重启动方案可有效控制风险。或者使用更安全的 chroot 技巧，参阅 chroot, 第 8.6.35 节。后者可实现同时在不同的控制台运行多种版本的 Ubuntu

3.1.1 硬件兼容性

Linux 兼容绝大多数 PC 硬件，而且几乎可以安装在任何系统架构上。对我而言它的安装过程和 Windwos 95/98/Me 一样简单。可兼容的硬件列表也在不断增加。

如果你有一部笔记本电脑，可以去 Linux on Laptops 查找对应品牌和型号的安装指导。

我推荐的台式机硬件通常“比较保守”：

- 用于工作目的时最好选 SCSI 而不是 IDE，IDE/ATAPI HD 适于个人使用。
- IDE/ATAPI CD-ROM（或 CD-RW）。
- 最好选 PCI 而不是 ISA，特别是网卡(NIC)。
- 选用便宜的 NIC。Tulip for PCI、NE2000 for ISA 都不错。
- 初学 Linux 安装时避免使用 PCMCIA （笔记本电脑）。
- 别使用 USB 键盘、鼠标.....除非你想证明自己的实力。

如果你的机器特别慢，最好把硬盘插到另一个速度快些的机器上安装。

3.1.2 确定 PC 硬件和芯片组

在安装过程中，可能会要你确定电脑的硬件或芯片组等。有时了解这些信息并非易事，这儿有个方法：

- 打开机箱亲自看看。
- 记下显卡芯片、网卡芯片、串口和 IDE 端口周围的芯片上印的数字。
- 记下印在 PCI 和 ISA 插卡背面的名称。

3.1.3 在 Ubuntu 下确定 PC 硬件

在 Linux 系统中使用下列命令，可获得机器当前使用的硬件及配置的相关信息。

```
$ pager /proc/pci
$ pager /proc/interrupts
$ pager /proc/ioports
```

```
$ pager /proc/bus/usb/devices
```

在安装过程中按 Alt-F2 进入控制台，可运行这些命令获得帮助。

在最初的安装结束之后，通过安装可选的软件包，比如 pciutils、usbutils 和 lshw，你就可以获得更多的系统信息。

```
$ lspci -v |pager
$ lsusb -v |pager
```

典型的中断用途

```
*   IRQ0: timer output (8254)
*   IRQ1: keyboard controller
*   IRQ2: cascade to IRQ8 - IRQ15 on PC-AT
*   IRQ3: secondary serial port (io-port=0x2F8) (/dev/ttyS1)
*   IRQ4: primary serial port (io-port=0x3F8) (/dev/ttyS0)
*   IRQ5: free [sound card (SB16: io-port=0x220, DMA-low=1, DMA-high=5)]
*   IRQ6: floppy disk controller (io-port=0x3F0) (/dev/fd0, /dev/fd1)
*   IRQ7: parport (io-port=0x378) (/dev/lp0)
*   IRQ8: rtc
*   IRQ9: software interrupt (int 0x0A), redirect to IRQ2
*   IRQ10: free [network interface card (NE2000: io-port=0x300)]
*   IRQ11: free [(SB16-SCSI: io-port=0x340, SB16-IDE: io-port=0x1E8,0x3EE)]
*   IRQ12: PS/2 Mouse
*   IRQ13: free (was 80287 math coprocessor)
*   IRQ14: primary IDE controller (/dev/hda, /dev/hdb)
*   IRQ15: secondary IDE controller (/dev/hdc, /dev/hdd)
```

对于旧的非 PnP ISA 卡，你可能需要在 BIOS 中设定 IRQ5，IRQ10 和 IRQ11 为非 PnP 设备。

对于 USB 设备，它们在 /proc/bus/usb/devices 中以 Cls=nn 的形式列出：

```
*   Cls=00 : Unused
*   Cls=01 : Audio (speaker etc.)
*   Cls=02 : Communication (MODEM, NIC, ...)
*   Cls=03 : HID (Human Interface Device: KB, mouse, joystick)
*   Cls=07 : Printer
*   Cls=08 : Mass storage (FDD, CD/DVD drive, HDD, Flash, ...)
*   Cls=09 : Hub (USB hub)
*   Cls=255 : Vendor specific
```

如果设备类别（device class）不是 255，则 Linux 可支持该设备。

3.1.4 在其它 OS 下确定 PC 硬件

还可从其它操作系统中获取硬件信息。

安装其它商业版 Linux，当前它们在硬件侦测方面做得比 Debian 好。

安装 Windows。用鼠标右键点“我的电脑”在菜单中选属性/设备管理，可获得硬件配置信息。记下所有的资源信息如 IRQ、I/O 端口地址和 DMA。有些旧的 ISA 卡可能要在 DOS 下配置。

3.1.5 Lilo 神话

"Lilo 受限于 1024 柱面。" 大错特错！

Ubuntu 之后使用的新版 lilo 支持 lba32。只要主板的 BIOS 版本支持 lba32，lilo 就可以突破 1024 柱面的限制。

如果你使用的是旧版 lilo.conf，请确定在你的 lilo.conf 文件开头的某处加了一行命令指示系统读取“lba32"。参阅 /usr/share/doc/lilo/Manual.txt.gz。

3.1.6 GRUB

GNU Hurd 项目组提供的新的系统引导工具 grub 可使用于 Debian Woody 系统。

<pre><nowiki>

要编辑 GRUB 菜单，可编辑 /boot/grub/menu.lst 文件。参阅 设置 GRUB 启动参数, 第 8.1.6 节获取有关启动参数设置信息，它和 lilo 的设置方法不太一样。

3.1.7 选择引导软盘

你可以使用 bootcd 包，来制作一个可以启动的软盘，当然我们更推荐你直接使用 LiveCD 来作为维护系统的工具。对于最新的 dapper 版本，你可以直接使用 Desktop CD 从光盘启动来维护，或者安装系统。

3.1.8 安装系统

这方面的官方文档在 http://doc.ubuntu.com/ubuntu/install/i386 ，中文翻译在：http://wiki.ubuntu.org.cn/ubuntu/install/i386

在安装 /dev/hda 的 mbr 时，我喜欢将 lilo 装在诸如 /dev/hda3 的位置，这样做可以减小引导信息被覆盖的风险。

下面是我在安装过程中所做的选择。

* MD5 passwords "yes"
* shadow passwords "yes"
* Install "advanced" (dselect **) and select
** 取消 emacs (如果它被选中了)、nvi、tex、telnet、talk(d);
** 选上 mc、vim，nano-tiny 和 elvis-tiny 任选一个。

参阅 dselect, 第 6.2.4 节。即使你是个 Emacs 迷也不要在安装阶段使用它，nano 可满足你的需要。也不要安装其它庞大的软件包如 Tex(Potato 曾在此阶段设置它为默认安装)。参阅 应急的编辑器, 第 11.2 节了解此时安装 nano-tiny 或 elvis-tiny 的原因。

* 在每个软件包安装对话框中，回答所有的配置提问均=“y”（替换当前值）
* exim：选第 2 项，因为我使用 ISP 的 SMTP 服务器发送邮件。

有关 dselect 的更多信息，参阅 dselect, 第 6.2.4 节。

3.1.9 网络所需的主机名和 IP 地址

LAN 配置的例子（C subnet: 192.168.1.0/24）：

<pre><nowiki>
Internet
|
+--- 外部 ISP 提供 POP 服务（使用 fetchmail 访问）
|
Access point ISP 提供 DHCP 服务和 SMTP 中继服务
| :
Cable modem   (Dialup)
| :
LAN 网关外部端口: eth0（IP 由 ISP 的 DHCP 提供）
使用老式笔记本电脑（IBM Thinkpad, 486 DX2 50 MHz, 20MB RAM）
运行 Linux 2.4 内核，提供 ext3 文件系统支持
运行“ipmasq”软件包（安装它的补丁、NAT 和 firewall）
运行“dhcp-client”软件包配置 eth0（覆盖 DNS 的设定）
运行“dhcp”软件包配置 eth1
运行“exim”作为 smarthost（mode 2）
运行“fetchmail”设一个较长的时间值（fallback）
运行“bind”作为高速域名服务器，在从 LAN 连入 Internet 时
作为认证域名服务器，在从 LAN 中连入网中某个域时
运行“ssh”使用端口 22 和 8080（从任何地点连接）
运行“squid”作为缓存服务器连接 Debian 包文档服务器（APT 需要连接它）
LAN 网关内部端口：eth1 （IP = 192.168.1.1，固定）
|
+--- LAN Switch (100 base T) ---+
| |
LAN 中一些固定 IP 客户端   LAN 中一些 DHCP 客户端
(IP == 192.168.1.2-127, 固定)   (IP == 192.168.1.128-200, 动态)

参阅 网络设置, 第 10 章 了解更多网络设置方面的信息。参阅 建立路由网关, 第 10.12 节 了解更多 LAN 网关服务器设置方面的信息。

3.1.10 用户帐号

为了让机器访问起来有一致的感觉，我的系统中开头几个帐号通常是不变的。

我首先创建的用户帐号名是“admin”（uid=100）。我通过该帐号转发所有的 root email。该帐号加入到 adm 用户组（参阅 “为什么 GNU su 命令不支持 wheel group”, 第 9.2.2 节），这个用户组为成员提供大量 root 特权的命令，通过 su 或 sudo 命令就能使用 PAM。详情参阅 添加一个用户, 第 4.1.3 节。

3.1.11 创建文件系统

3.1.11.1 硬盘分区

我更喜欢将不同的目录树分别装在不同的分区下，这样可以将系统崩溃造成的损失减到最小。例如：

/   == (/ + /boot + /bin + /sbin)
== 50MB+

```
/tmp       == 100MB+
/var       == 100MB+
/home      == 100MB+
/usr       == 700MB+ with X
/usr/local == 100MB
```

/usr 目录的大小很大程度上取决于 X window 应用程序和文档的数目。如果只运行控制台终端 /usr 可以小到 300MB，但如果你装了大量 Gnome 应用程序 2GB-3GB 也很正常。当 /usr/ 增长得太大，将 /usr/share/ 移到别的分区是最有效的解救方法。对新的 Linux2.4 内核包，/ 目录的大小可能要超过 200MB。

例如，当前我的 Internet 网关服务器的硬盘使用情况如下（由 df -h 命令输出）：

```
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda3             300M  106M  179M  38% /
/dev/hda7             100M   12M   82M  13% /home
/dev/hda8             596M   53M  513M  10% /var
/dev/hda6             100M  834k   94M   1% /var/lib/cvs
/dev/hda9             596M  222M  343M  40% /usr
/dev/hda10            596M  130M  436M  23% /var/cache/apt/archives
/dev/hda11            1.5G  204M  1.2G  14% /var/spool/squid
```

（有块大空间留给 /var/spool/squid 作为下载软件包时代理服务器的缓冲区。）

下面的 fdisk -l 输出提供了一个分区分配方案：

```
# fdisk -l /dev/hda # comment

/dev/hda1             1        41    309928+   6  FAT16 # DOS
/dev/hda2            42        84    325080   83  Linux # (not used)
/dev/hda3   *        85       126    317520   83  Linux # Main
/dev/hda4           127       629   3802680    5  Extended
/dev/hda5           127       143    128488+  82  Linux swap
/dev/hda6           144       157    105808+  83  Linux
/dev/hda7           158       171    105808+  83  Linux
/dev/hda8           172       253    619888+  83  Linux
/dev/hda9           254       335    619888+  83  Linux
/dev/hda10          336       417    619888+  83  Linux
/dev/hda11          418       629   1602688+  83  Linux
```

里面有一个未使用的分区。留作安装第二个 Linux 或为增长的目录树提供扩充余地。

3.1.11.2 挂载文件系统

下面的 /etc/fstab 文件可完成上述分区的挂载工作：

```
# /etc/fstab: 静态的文件系统信息。
#
# filesystem   mount point     type    options                dump pass
/dev/hda3       /               ext2    defaults,errors=remount-ro 0 1
/dev/hda5       none            swap    sw                         0 0
```

```
proc            /proc           proc    defaults                0 0
/dev/fd0        /floppy         auto    defaults,user,noauto    0 0
/dev/cdrom      /cdrom          iso9660 defaults,ro,user,noauto 0 0
#
# 各分区保持独立
/dev/hda7       /home           ext2    defaults                0 2
/dev/hda8       /var            ext2    defaults                0 2
/dev/hda6       /var/lib/cvs    ext2    defaults                0 2
# noatime 会提高读取文件时的访问速度
/dev/hda9       /usr            ext2    defaults,noatime        0 2
/dev/hda10      /var/cache/apt/archives ext2    defaults        0 2

# 为代理缓存设置的大分区
/dev/hda11      /var/spool/squid ext2   rw                      0 2

# 备份、可启动、DOS
/dev/hda1       /mnt/dos        vfat    rw,noauto               0 0
# 作为备份用的可启动的 Linux 系统（未设置）
/dev/hda2       /mnt/linux      ext2    rw,noauto               0 0
#
# nfs 挂载点
mickey:/        /mnt/mickey     nfs     ro,noauto,intr          0 0
goofy:/         /mnt/goofy      nfs     ro,noauto,intr          0 0
# minnie:/ /mnt/minnie smbfs ro,soft,intr,credentials={filename} 0 2
```

对于 NFS，我使用 noauto、intr 以及默认的 hard 项。如果有死连接，挂载进程可能会死掉，可以使用 Ctrl-C 恢复。

对于使用 Samba 连接的 Windows 机器（smbfs），rw,auto,soft,intr 是个好方案，参阅 Samba 设置, 第 3.5 节。

对于软驱，使用 noauto,rw,sync,user,exec 可以防止因退盘前未执行卸载命令而造成文件损坏，但它会降低写盘速度。

3.1.11.3 自动挂载

自动挂载的要点：

- 加载 vfat 模块，从而 /etc/auto.misc 可使用 -fstype=auto 参数：

```
# 在试图访问软驱之前
# modprobe vfat         ... 或者使这个设置自动完成，
# echo "vfat" >> /etc/modules        ... 重启系统。
```

- 设置 /etc/auto.misc 如下：

```
floppy -fstype=auto,sync,nodev,nosuid,gid=100,umask=000 :/dev/fd0       ... 此处 gid=100 表示 “users”。
```

- 在 /home/user 中创建链接文件 cdrom 和 floppy ，分别指向 /var/autofs/misc/cdrom 和 /var/autofs/misc/floppy。

- 将 user 加入“users”用户组。

3.1.11.4 挂载 NFS

外部 Linux NFS 服务器（goofy）处于防火墙（gateway）之后。在我的 LAN 内部，安全策略非常宽松，因为只有我自己一个用户。为了访问 NFS，NFS 服务器端要按如下方式加上 /etc/exports：

```
# /etc/exports: the access control list for filesystems which may be
#               exported to NFS clients.  See exports(5).      /
(rw,no_root_squash)
```

除安装和激活 NFS 服务器及客户机的软件包外，这一步对激活 NFS 服务器也是必要的。

为了简洁明了，我通常创建一个单独的 2GB 分区，用它进行实验或玩玩别的 Linux 系统。我有选择的共享了两个系统的 swap 和 /tmp 分区。多分区方案很难满足这种要求。如果只想装一个运行简单控制台模式的系统，分区留 500MB 就绰绰有余了。

3.1.12 DRAM 内存指导

下列是有关 DRAM 的简单指导。

4MB: Linux 内核运行的最低配置。
16MB: 控制台系统运行的最低配置。
32MB: 简单 X 系统运行的最低配置。
64MB: GNOME/KDE 系统运行最低配置。
128MB: 流畅运行 GNOME/KDE。
256MB: 如果你银子充裕干嘛不呢？DRAM 很便宜了。

使用启动选项 mem=4m（或 lilo append="mem=4m"）可以看看只用 4MB 内存时系统如何运行。如果机器的 BIOS 比较旧且内存数大于 64MB 启动时就得加 lilo 启动参数。

3.1.13 Swap 空间

我按下面的指导原则来划分 swap 空间：

- 每个 swap 分区均 <128MB（使用旧版 2.0 内核），<2GB（使用最新内核）
- 总容量 =（机器内存大小的 1 到 2 倍）或（128MB 到 2GB 之间）
- 将它们分散于不同的硬盘，在 /etc/fstab 中可使用 sw,pri=1 选项挂载它们。这样可确保内核以 RAID 方式使用 swap 分区，最大限度地发挥 swap 的性能。
- 如果可能，使用硬盘中间的扇区。

尽管你不一定真的需要，但为系统设置多一些 swap 空间（128MB）总要好点，至少运行有内存漏洞的坏程序，系统会先慢下来而不是马上死机。

3.2 Bash 设置

我按自己的喜好修改 shell 启动脚本：

```
/etc/bash.bashrc        换成你的喜好
/etc/profile            保持发布版设定 ( \w -> \W)
```

/etc/skel/.bashrc 换成你的喜好
/etc/skel/.profile 换成你的喜好
/etc/skel/.bash_profile 换成你的喜好
~/.bashrc 换成你的喜好，会改变所有用户的设置
~/.profile 换成你的喜好，会改变所有用户的设置
~/.bash_profile 换成你的喜好，会改变所有用户的设置

可以参考我提供的脚本样例。我喜欢系统看起来清清楚楚，所以将 umask 设为 002 或 022。PATH 依次由以下配置文件设定：

/etc/login.defs - 在 shell 之前设定 PATH
/etc/profile （会调用 /etc/bash.bashrc）
~/.bash_profile （会调用 ~/.bashrc）

3.3 鼠标设置

3.3.1 PS/2 鼠标

对于 ATX 主板上使用 PS/2 接口的鼠标，信号通路如下：

mouse -> /dev/psaux -> gpm -> /dev/gpmdata = /dev/mouse -> X

这里，创建指向 /dev/gpmdata 的符号链接 /dev/mouse，有助于简化配置操作。（例如，如果你决定不再使用 gpm 服务，只需在删掉 gpm 服务后，将符号链接 /dev/mouse 指向 /dev/psaux。）

信号通路使得对键盘和鼠标插拔操作，只要重启 gpm 就可以生效，不必重启 X!

处于 gpm 输出与 X 输入之间的信号通路协议有两种执行方式，“ms3”（使用 Microsoft 3-button serial mouse 协议）和“raw”（对于已连接的鼠标使用相同的协议），此处选择的协议将决定 X 配置中使用的协议。

目前 dapper 发行版并没有默认安装 gpm ，如果需要使用它，手工安装如下：

```
sudo apt-get install gpm
```

下面我将示范一下如何配置 Logitech 3-button（传统 Unix 风格的鼠标）PS/2 鼠标：

3.3.1.1 使用 ms3 协议方式

```
/etc/gpm.conf              | /etc/X11/xorg.conf
=========================+=====================================
device=/dev/psaux          | Section "InputDevice"
responsiveness=            |  Identifier "Configured Mouse"
repeat_type=ms3            |  Driver     "mouse"
type=autops2               |  Option     "CorePointer"
append=""                  |  Option     "Device"   "/dev/mouse"
sample_rate=               |  Option     "Protocol" "IntelliMouse"
                           | EndSection
```

如果使用这种方式，设置鼠标类型只需编辑 gpm.conf ，X 的设置将保持不变。参阅 我的样例脚本。

3.3.1.2 使用 raw 协议方式

```
/etc/gpm.conf            | /etc/X11/xorg.conf
=========================+=======================================
device=/dev/psaux        | Section "InputDevice"
responsiveness=          |  Identifier "Configured Mouse"
repeat_type=raw          |  Driver     "mouse"
type=autops2             |  Option     "CorePointer"
append=""                |  Option     "Device"   "/dev/mouse"
sample_rate=             |  Option     "Protocol" "MouseManPlusPS/2"
                         | EndSection
```

如果使用这种方式，设置鼠标类型可编辑 gpm.conf，同时也会改变 X 的设置。

3.3.1.3 如何设置不同的鼠标

gpm 的设备类型 autops2 可以自动检测出市面上大多数 PS/2 鼠标。不幸的是它也不是万能的，而且目前的发布版中并不包括它，这时可在 gpm.conf 中试试 ps2 或者 imps2。想看看 gpm 支持哪些类型的鼠标可输入：gpm -t help。参阅 gpm(8)。

如果使用的是 2 键 PS/2 鼠标，可选上 X 协议的 Emulate3Buttons 项。协议中 2 键鼠标和 3 键鼠标的区别在于每次按下中间键时，是自动检测还是自动模拟信号传给 gpm。

对于在 X 协议 使用 raw 协议方式，第 3.3.1.2 节 或不使用 gpm，可使用下面的设置：

- IntelliMouse: 串口鼠标（gpm 转换器使用“ms3”）
- PS/2: PS/2 鼠标（通常首先试试这项）
- IMPS/2: 任何 PS/2 鼠标（2 键、3 键或滚轴鼠标）
- MouseManPlusPS/2: Logitech PS/2 鼠标
- ...

更多信息可浏览 Mouse Support in XFree86。

典型的 Microsoft 滚轴鼠标，有报导说这样设置最好：

```
/etc/gpm.conf            | /etc/X11/XF86Config-4
=========================+=======================================
device=/dev/psaux        | Section "InputDevice"
responsiveness=          |  Identifier "Configured Mouse"
repeat_type=raw          |  Driver     "mouse"
type=autops2             |  Option     "CorePointer"
append=""                |  Option     "Device"   "/dev/mouse"
sample_rate=             |  Option     "Protocol" "IMPS/2"
                         |  Option     "Buttons" "5"
                         |  Option  "ZAxisMapping" "4 5"
                         | EndSection
```

对于某些最新的 Toshiba 超薄笔记本，在 System-V init 脚本中设置 gpm 先于 PCMCIA

激活可防止系统锁死。听起来古怪不过这是真的。

3.3.2 USB 鼠标

请确定你设置了所有必须的 kernel 选项，并在编译的时候编入内核或编成模块了：

- 在 “Input Core Support”选项下：
  - "Input core support" (CONFIG_INPUT, input.o),
  - "Mouse support" (CONFIG_INPUT_MOUSEDEV, mousedev.o),
- 在 “USB support”选项下：
  - "Support for USB" (CONFIG_USB, usbcore.o),
  - "Preliminary USB device filesystem" (CONFIG_USB_DEVICEFS),
  - "UHCI" or "OHCI" (CONFIG_USB_UHCI || CONFIG_USB_UHCI_ALT || CONFIG_USB_OHCI, usb-uhci.o || uhci.o || usb-ohci.o),
  - "USB Human Interface Device (full HID) support" (CONFIG_USB_HID, hid.o), and
  - "HID input layer support" (CONFIG_USB_HIDINPUT)

这儿，模块的名称为小写。

如果你没使用 devfs，则需按下面的方法创建一个设备节点 /dev/input/mice：

```
# cd /dev
# mkdir input
# mknod input/mice c 13 63
```

对于典型的滚轴 USB 鼠标，配置如下：

```
/etc/gpm.conf            | /etc/X11/XF86Config-4
=========================+======================================
device=/dev/input/mice   | Section "InputDevice"
responsiveness=          |  Identifier "Generic Mouse"
repeat_type=raw          |  Driver     "mouse"
type=autops2             |  Option     "SendCoreEvents" "true"
append=""                |  Option     "Device"   "/dev/input/mice"
sample_rate=             |  Option     "Protocol" "IMPS/2"
|  Option     "Buttons" "5"
|  Option  "ZAxisMapping" "4 5"
| EndSection
```

更多信息可浏览 the Linux USB Project。

3.3.3 触摸屏（Touchpad）

尽管触摸屏在笔记本电脑上默认是模拟 2 键 PS/2 鼠标的行为，仍可从 tpconfig 软件包获得对这类设备的完全操控。例如在 /etc/default/tpconfig 中设置 OPTIONS="--tapmode=0" 可屏蔽讨厌的“click by tap”行为。按如下方法设置 /etc/gpm.conf 可以控制台下使用触摸屏和 USB 外接鼠标。

```
device=/dev/psaux
responsiveness=
repeat_type=ms3
type=autops2
append="-M -m /dev/input/mice -t autops2"
sample_rate=
```

3.4 NFS 设置

安装 NFS 要设置/etc/exports。

```
# apt-get install nfs-kernel-server
# echo "/ *.domainname-for-lan-hosts(rw,no_root_squash,nohide)" \
>> /etc/exports
```

详情见我的样例脚本.

3.5 Samba 设置

参考资料：

- http://www.samba.org/
- samba-doc package

以“share”方式安装 Samba 比较容易，因为安装过程会创建 WfW-type 共享驱动器。但最好使用“user”模式来配置它。

可以用 debconf 或 vi 来设置 Samba：

```
# dpkg-reconfigure --priority=low samba
# vi /etc/samba/smb.conf
```

详情参阅我的脚本样例.

可通过 smbpasswd 向 smbpasswd 文件添加一个新用户：

```
# smbpasswd -a username
```

确保使用的加密密码有恰当的兼容性。

按下表中相应的值设定 os level（数字越大，服务器的优先级越高）：

```
0:      Samba with a loose attitude (will never become a master browser)
1:      WfW 3.1, Win95, Win98, Win/Me?
16:     Win NT WS 3.51
17:     Win NT WS 4.0
32:     Win NT SVR 3.51
33:     Win NT SVR 4.0
255:    Samba with mighty power
```

确认用户是共享目录所属组的成员，并且对目录赋予了执行权限。

3.6 打印机设置

传统打印方法是 lpr/lpd。新的 CUPS 系统（Common UNIX Pinting System）提供了另一方法 PDQ，详情参阅 Linux Printing HOWTO。

3.6.1 lpr/lpd

对于 lpr/lpd 的打印缓冲池（type spoolers)（ lpr、lprng 和 gnulpr)，如果它们连接的是 PostScript 或 text-only 打印机（最基本的打印机）可按如下方式设置 /etc/printcap：

```
lp|alias:\
:sd=/var/spool/lpd/lp:\
:mx#0:\
:sh:\
:lp=/dev/lp0:
```

上述各行的意思是：

- Head line: lp - spool 名称, alias = alias
- mx#0 - 不限最大文件尺寸
- sh - 不打印页眉
- lp=/dev/lp0 - 本地机印机，或 port@host 远程打印机

如果连接的是台 PostScript 打印机，上述设置就够用了。如果是 Windows 机器通过 Samba 打印，上述设置也适用于任何 Windows 支持的打印机(不支持双向通信)。你必须在 Windows 环境中对打印机做相应的配置。

如果你没有 PostScript 打印机，就得用 gs 安装过滤系统。有很多自动配置工具可用于配置 /etc/printcap，可选择下列任何一组：

- gnulpr、(lpr-ppd) 和 printtool— 我用这种
- lpr 和 apsfilter
- lpr 和 magicfilter
- lprng 和 lprngtool
- lprng 和 apsfilter
- lprng 和 magicfilter

想运行 GUI 配置工具如 printtool，需要 root 权限，参阅 X 下获取 root 权限, 第 9.4.12 节。printtool 可创建打印缓冲池，它使用 gs 模拟 PostScript 打印机，所以要使用 PostScript 打印驱动来访问它们。在 Windows 端，“Apple LaserWriter”是标准件。

3.6.2 CUPS™

使用 aptitude 安装 Common UNIX Printing System（或 CUPS™）和所有位于 "Tasks" -> "Servers" -> "Print Server". (Sarge) 下的软件包。 为了得到最佳的结果，你需要对 aptitude 进行如下的设置： "F10" -> "Options" -> "Dependency handling" -> "[X] Install Recommended packages automatically".

KDE 和 Gnome 桌面系统提供了简易的打印机设置。如果安装了 swat，你也可以用任何一种浏览器来设置。

```
$ mybrowser http://localhost:631
```

举个例子，将你的打印机联接到可访问打印机列表中：

- 在主页面上点击“Printers"，然后点“Add Printer"，
- 输入用户名和密码，进入“root"，
- 按提示添加打印机，
- 返回“Printers”页面，点“Configure Printer"，
- 设定打印纸尺寸、分辨率和其它参数。

更多信息可浏览 http://localhost:631/documentation.html 和 http://www.cups.org/cups-help.html。

对于 2.4 内核，参见 并行端口支持，第 7.2.6 节。

3.7 桌面 PC 的 CRON

Vixie cron 是计划任务默认安装的软件。 除非系统是 7×24 小时连续运行，该软件并不能够很好的工作。 对于桌面 PC， 需要在安装 cron 软件包的基础上再安装 anacron 来解决这个问题。 fcron 软件包可以作为 anacron 软件包的一个替代。

参见 日程安排（cron，at），第 8.6.27 节 来配置 CRON。

3.8 其它主机安装提示

3.8.1 初始化安装完成后再安装其他软件包

到现在，你已经拥有一个小巧但功能颇强的 Debian。接下来，可以安装那些较大的软件包了。

- 运行 tasksel。参阅 安装 tasks，第 6.2.2 节。

可按需选择：

  - End-user - X window system
  - Development - Cand C++
  - Development - Python
  - Development - Tcl/Tk
  - Miscellaneous — TeX/LaTeX environment
  - 对于其他软件包，我喜欢把 tasksel 作为安装指导来用，查看 <Task Info> 了解有关任务的安装列表 然后用 dselect 来选择安装。

- 运行 dselect。

在此，你最想做的事就是选择钟爱的编辑器和其它需要的程序。你可以同时安装 Emacs 的多个变种。参阅 dselect，第 6.2.4 节 和 流行的编辑器，第 11.1 节。 你也可以将某些默认的软件包替换成特定版本。

  - lynx-ssh (而不是 lynx)
  - ...

- ...

我通常编辑 /etc/inittab 来简化关机步骤。

```
...
# What to do when CTRL-ALT-DEL is pressed.
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -h now
...
```

3.8.2 模块

在初始安装期间可进行设备驱动模块配置。以后还可使用 modconf 进行配置，它是基于菜单界面的工具,可用来配置那些在初始安装时未曾配置的模块或完成新内核安装后的配置工作。

所有预加载(preloading)模块的名称要加入到 /etc/modules 列表。也可以使用 lsmod 和 depmod 进行手工控制。

对于 2.4 版内核别忘了在 /etc/modules 中加上几行来处理 IP 伪装（FTP 等）。参阅 模块化的 2.4 内核, 第 7.2 节，特别是 网络功能, 第 7.2.3 节。

3.8.3 CD-RW 基本步骤

对于在 2.4 版内核中使用 IDE 接口的 CD-RW，编辑下列文件：

```
/etc/lilo.conf  (添加 append="hdc=ide-scsi ignore=hdc",
执行 lilo 激活)
/dev/cdrom      (创建链接 # cd /dev; ln -sf scd0 cdrom)
/etc/modules    (加入 "ide-scsi"和"sg"。如果需要可再加上"sr"。)
```

详情参阅 刻录机, 第 9.3 节。

3.8.4 多内存和关机自动断电

编辑 /etc/lilo.conf，设置启动提示参数如下，实现识别大内存（适用于 2.2 版内核）和关机自动断电（适用于 APM）：

```
append="mem=128M apm=on apm=power-off noapic"
```

执行 lilo 完成上述设置。对称多处理器内核（SMP-kernel）需要 apm=power-off，而对我那堆糟糕的 SMP 硬件而言 noapic 也是需要的。系统启动时，在启动提示符后直接输入这些参数效果也一样。参阅 其它用于启动提示符的技巧, 第 8.1.5 节。

在 2.4 版内核中，如果 APM 是作为模块编译的，可在系统启动后运行 # insmod apm power_off=1 或设置 /etc/modules：

```
# echo "apm power_off=1" >>/etc/modules
```

还可以这样：编译新版内核时加入 ACPI 支持可达到同样的效果，而且这种方式更适合 SMP（只有较新的主板才支持 ACPI）。对于较新的主板 2.4 版内核可以直接检测到大内存。

```
CONFIG_PM=y
CONFIG_ACPI=y
...
CONFIG_ACPI_BUSMGR=m
CONFIG_ACPI_SYS=m
```

在 /etc/modules 中按如下顺序添加参数：

```
ospm_busmgr
ospm_system
```

或者重新编译内核，在配置时将上述内核选项均设为“y"。总之，有了 ACPI 支持就不再需要任何启动提示参数。

3.8.5 无法访问某些站点的怪问题

如果 内核开启 ECN,对于某些使用劣质路由器的站点,会出现无法访问的问题。新的 dapper 发行版默认关闭了 ECN，检查 ECN 状态可使用：

```
# cat /proc/sys/net/ipv4/tcp_ecn
... 或
# sysctl net.ipv4.tcp_ecn
```

将它关闭：

```
# echo "0" > /proc/sys/net/ipv4/tcp_ecn
... 或
# sysctl -w net.ipv4.tcp_ecn=0
```

每次启动时禁止 TCP ECN，可编辑 /etc/sysctl.conf，加上：

```
net.ipv4.tcp_ecn = 0
```

3.8.6 PPP 拔号设置

安装 pppconfig 软件包，设置 PPP 拨号访问。

```
# apt-get install pppconfig
# pppconfig
... 按提示配置 PPP 拨号
# adduser user_name dip
... 允许 user_name 进行 PPP 拨号访问
```

用户（user_name）进行 PPP 拨号访问：

```
$ pon ISP_name   # 开始 PPP 访问，接通你的 ISP
... 享受 Internet
$ poff ISP_name # 停止 PPP 访问，ISP_name 可选
```

详情参阅 设置 PPP 接口, 第 10.2.4 节。

ADSL 用户使用 pppoeconf，来配置 PPPOE 拨号访问。

```
# apt-get install pppoeconf
# pppoeconf
... 按提示配置 PPPoe 拨号
```

3.8.7 /etc/ 中的其它配置文件

Ubuntu 标准安装不包含 /etc/cron.deny 文件，想添加可拷贝 /etc/at.deny。

第 4 章 - Ubuntu 指南

这一节为真正的新手提供一个熟悉的 Ubuntu 世界，如果您已经使用了一段时间的类 unix 操作系统，那么您应该已经了解我在这里所说的一切。那么请用这个来做一个实战检验。

4.1 开始了

在您的电脑上安装完 Ubuntu 系统以后，您需要学习一点东西以便使用它，让我们来给您做一个快速的培训。

4.1.1 用超级用户登录到命令提示符

在重新启动系统的时候，您处在一个图形的登录界面，或者字符界面的登录界面，这个取决于您初始安装时所选择的发行版本，简单的说，如果您现在处在图形登录界面， 那么按下 Ctrl-Alt-F1 来获得字符界面。

假设您的主机名是 foo,那么登录提示符看起来是这样的：

```
foo login:
```

输入用户名也就是你安装时设置的用户名，我们假设用户名为 ubuntu，如果采用 oem 方式安装，则用户名为 oem ，然后按回车键，然后会提示输入密码，就是您在安装过程中所输入的密码。在 Ubuntu 系统中，按照 Unix 的习惯，密码是不可见的。然后系统就会输出欢迎 信息而且给出 $ 的命令提示符等待您的输入。

```
foo login: ubuntu
Password:
Linux backup 2.6.15-25-686 #1 SMP PREEMPT Wed Jun 14 11:34:19 UTC 2006 i686
GNU/Linux

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

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
You have new mail.
```

```
Last login: Tue Jul  4 13:24:05 2006 on tty3
ubuntu@foo:~$
```

现在输入 sudo -sH ，会再次输入密码，输入相同的登录密码后，回车，切换到 root 帐号：

```
ubuntu@foo:~$ sudo -sH
sudo: please use single character options
Password:
root@@foo:~#
```

您现在已经准备好通过root的命令提示符进行系统的管理。这个root帐户也被称为超级 用户或者特权用户。拥有这个帐户，您将可以做任何事情：

- 读，写和删除任何文件而不用去理会它们所要求的权限
- 设置文件的归属，以及系统里面任何文件的访问权限
- 登录到任何帐户而不需要他们的密码

通过和别人共享 root 用户的密码来共享 root 帐户是一个非常糟糕的想法。而使用程序比如 sudo(8)是共享管理权限的好方法。

请注意，优先使用非特权用户登录被认为是一个好的 Unix 习惯，哪怕是您要进行管理活动，在必要的时候您也可以使用 sudo, super,或者 su -c 来 获得有限的 root 权限。See 更安全地工作 - sudo, 第 9.2.4 节.

4.1.2 设置最小的新手环境

我认为学习一个电脑系统就像学习一门新的外语。尽管书本指南是有帮助的，但是您需要 练习使用帮助工具。在这个情况下，我认为安装一些额外的软件包是一个好主意，比如 mc, vim, lynx, doc-linux-text, and debian-policy.

```
# apt-get update
...
# apt-get install mc vim lynx doc-linux-text debian-policy
...
```

如果您已经安装了这些软件包，那么什么都不需要安装了。

4.1.3 添加一个用户

在安装过程中，您通常已经创建了一个非特权用户来接收发送给 root 帐户的 e-mail。 因此您也不希望用这个特殊的帐户来进行下面的训练，那么您需要创建另外一个帐户。

假设您希望新帐户的名字叫 penguin, 输入:

```
root@foo:root# adduser penguin
... 回答所有的问题
```

这样就可以了。 在更进一步之前，我们先来学习一点东西

4.1.4 在不同的控制台之间切换

在默认的 Ubuntu 系统中，有 6 个独立的伪终端，比如，您可以把 PC 的 VGA 字符终端当作 6 个可以选择的 VT-100 终端来使用。从一个切换终端到另外一个，你只需要同时按下左 Alt 键 和 F1 －:F6 键。任何一个伪终端都可以让不同的用户独立登录。多用户环境是 Unix 的一个很优秀的、使人迷恋的特性。

如果你偶然的在装有 X 装口系统的系统上按下了 Alt-F7，控制台就会切换到图形界面，然后 再按下 Ctrl-Alt-F1 可以重新回到字符界面。您可以尝试在在不同的控制台之间切换，然后 再换回到原来的那个，您慢慢就会习惯于这样做。

4.1.5 怎样关闭机器

就像任何其它现代的操作系统一样，任何文件操作都会在内存当中缓存数据，Ubuntu 操作系统也需要一个适当的过程，让电脑电源关闭之前保证这些文件的一致性，您可以在 root 命令提示符下使用下面的命令来关闭系统。

```
# shutdown -h now
```

上面是在正常的多用户模式下面的方法。如果您是处在单用户模式，您可以在 root 用户的命令提示符下使用下面的命令：

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

可选择的其它方式，比如您可以用 Ctrl-Alt-Delete 来关闭系统

默认系统会自动关闭电源，如果 APM 和 ACPI 选项在 BIOS 和 Linux 内核里面都配置有问题的话，等待系统在屏幕上输出"System halted"之后您就可以关闭电源了。可以看 多内存和关机自动断电, 第 3.8.4 节获得详细信息。

4.1.6 玩的时间

现在您已经准备好使用 Ubuntu 系统了，而不用担心有任何冒险，因为您使用的非特权用户 penguin。

让我们先登录到 penguin 用户。 如果您现在正在 root 用户的命令提示符下面，输入 exit 在 root 用户提示符下面关闭 root 的 shell 以后我们就返回了 ubuntu 用户提示符，再次输入 exit ，返回登录提示符。输入您刚创建的新 用户名 penguin 和对应的密码。 您将会获得如下的命令提示符

```
penguin@foo:penguin$
```

从现在开始，为了简单起见我们将使用简化的命令提示符，我们将使用：

- #： root 的命令提示符
- $： 非特权用户的命令提示符

我们将先用简单的方法 Midnight Commander (MC), 第 4.2 节来学习 Ubuntu ,稍后使用

较好的方法 类 Unix 工作环境, 第 4.3 节。

4.2 Midnight Commander (MC)

Midnight Commander (MC)是 Linux 控制台和其它终端环境中的 GNU“瑞士军刀”。这给予了新手一个菜单驱动控制台的体验，它比 Unix 标准命令容易学的多。

用命令来研究 Ubuntu 系统。这是最好的学习方法。请不使用键盘而只用鼠标和回车键来访问下面的地址：

- /etc 和它的字目录。
- /var/log 和它的字目录。
- /usr/share/doc 和它的字目录。
- /sbin and /bin

4.2.1 提高 MC

为了让 MC 在退出的时候改变工作目录，您需要修改 ~/.bashrc（或者/etc/bash.bashrc，查看.bashrc），使用-P 选项在它的 手册里面可以获取更详细的信息 mc(1).

4.2.2 启动 MC

```
$ mc
```

在 MC 中用户可轻而易举使用菜单完成所有的文件操作。可以按 F1 获得帮助，您可以只用鼠标和功能键来操作 MC。

4.2.3 MC 里的文件管理器

默认状态下，所有文件列于两个目录面板。一种实用的方式是将右边窗口设定为“information”，用来查看文件访问权限等信息。下面是一些基本的击键。如果 gpm daemon 在运行，你还可以使用鼠标。（在 MC 中进行剪切和粘贴操作时别忘了按下 shift 键。）

- F1: 帮助菜单
- F3: 内置文件阅读器
- F4: 内置编辑器
- F9: 激活折叠菜单
- F10: 退出 Midnight Commander
- Tab: 在两个窗口间移动
- Insert: 为多文件操作生成文件如拷贝
- Del: 删除文件（小心—请设置 MC 为安装删除模式）
- Cursor keys: 与各自的名字一致

4.2.4 MC 里的命令行技巧

- cd 命令可改变焦点屏幕中的目录位置。
- Control-Enter 或 Alt-Enter 可以将文件名拷贝到命令行。在编辑命令行时可与 cp 或 mv 命令结合使用。
- Alt-Tab 显示焦点文件所属文件或目录的文件名

- 可指定 MC 两个目录窗口的起始目录；例如 mc /etc /root。
- Esc + numberkey ===== Fn（例如 Esc + '1' ===== F1, etc.; Esc + '0' ==== F10）
- Esc- key ==== Alt-key (== Meta, M-); 例如 Esc + 'c'等价于 Alt-c

4.2.5 MC 里面的编辑器

内置编辑器的剪切-粘贴方式很有意思。按 F3 在起始处做标记，再次按 F3 在终止处做标记，这时中间的选中部分会高亮显示。然后你可以移动光标到某处 按下 F6，被选中部分就会移到该处。如果你按下的是 F5，选中部分就会拷贝到该处。F2 用来存盘，F10 退出，所有方向键的作为和它们的名字一样直观。

该编辑器可直接打开某个文件：

```
$ mc -e filename_to_edit
$ mcedit filename_to_edit
```

虽然它不是多窗口编辑器，但你可利用 Linux 多控制台的特性达到相同的效果。要在窗口间进行拷贝操作，可使用 Alt-Fn 切换虚拟控制台，然后使 用“File->Insert file”或“File->Copy to file”将一个文件中的内容拷贝到另一个文件。

可指定任何外部编辑器作为内置编辑器。

许多程序使用环境变量 EDITOR 或 VISUAL 来决定使用哪个编辑器。如果你用不惯 vim，可在 ~/.bashrc 中加上几行来指定新的 mcedit：

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

我强烈推荐将它们设定为 vim。在 Linux/Unix 世界里就该使用 vi(m)命令。

4.2.6 MC 里的阅读器

非常精巧的阅读器。它是在文档中搜索单词的利器。在/usr/share/doc 目录下我总是用它，面对大量的 Linux 资料用它浏览是最快的方法。阅读器可以直接找开文件：

```
$ mc -v filename_to_view
```

4.2.7 MC 的自动启动特性

在文件上按 Enter，会激活相关的程序操作该文件。这是 MC 的一个极方便的特点。

```
executable file:        执行命令
man, html file:    将文件内容传入阅读器程序
tar, gz, deb file: 象浏览子目录一样浏览它的内容
```

为了使这些阅读器和虚拟文件的属性能够被查看，不要将可阅读文件设成可执行文件。可在 MC 文件菜单中改变文件属性或使用 chmod 命令。

4.2.8 MC 里的 FTP 虚拟文件系统

MC 可通过 FTP 访问文件。按 F9 转到菜单栏，输入’p’激活 FTP 虚拟文件系统。按 username:passwd@hostname.domainname 格式输入 URL，远程文件目录就会以本地目录的方式显示出来。

在 URL 里试着用 http.us.debian.org/debian 来浏览 Debian 存档文件。看 Ubuntu 文件, 第 2.1 节 就知道这些是怎么被识别的了。

4.3 类 Unix 工作环境

尽管 MC 让您能够做绝大多数的事情，但是利用 shell 来学习命令行工具，和熟悉类 Unix 系统 的工作环境还是很重要的。

4.3.1 特殊的按键组合

在类 Unix 环境里，有一些按键有特殊的意思。

- Ctrl-U: 擦除一行光标前面的部分。
- Ctrl-H: 擦除光标前面的一个字符。
- Ctrl-D: 终止输入。(退出 shell，如果您正在使用 shell 的话)。
- Ctrl-C: 终止当前正在运行的程序。
- Ctrl-Z: 暂停程序。 (把它放到后台运行请看 command &, 第 4.3.10.1 节)
- Ctrl-S: 停止向屏幕输出。
- Ctrl-Q: 重新激活向屏幕输出。

默认的 shell，bash， 有历史编辑和 tab 补齐功能。

- up-arrow: 开始历史命令搜索。
- Ctrl-R: 开始增量历史命令搜索。
- TAB: 完整的把文件名输入到命令行。
- Ctrl-V TAB: 输入 TAB 而不是扩展命令行。

其他一些需要记住的按键组合。

- Ctrl-Alt-Del:挂起或者重新启动系统 初始化安装完成后再安装其他软件包, 第 3.8.1 节.
- Left-click-and-drag mouse: 选择并且拷贝到剪贴板。
- Click middle mouse button: 使用剪贴板的内容粘贴。
- Meta-key (Emacs terminology) 传统的是使用 Left-Alt-key. 但是有些系统使用 Windows-key 实现 Meta-key.

这里，为了在 Linux 字符界面下使用鼠标，您需要使用 gpm 服务。 查看 鼠标设置, 第 3.3 节。

4.3.2 基本 Unix 命令

让我们来学习基本的 Unix 命令。 使用非特权用来执行下面的所有的命令。 penguin：

- pwd
    - 显示当前工作路径。
- whoami
    - 显示当前用户名。
- file foo
    - 显示 foo 文件的属性。
- type -p commandname
    - 显示命令 commandname 所的的地方。
    - which commandname 也可以用来做这个 .
- type commandname
    - 显示命令 commandname 的信息。
- apropos key-word
    - 找到和 key-word 相关的命令。
    - man -k key-word 也可以做到
- whatis commandname
    - 显示该命令的一句话帮助。 commandname.
- man -a commandname
    - 显示命令的解释信息。 commandname. (Unix style)
- info commandname
    - 显示很长的命令解释 commandname. (GNU style)

ls

- 列出目录内容 (非以.开始的文件和目录)
- ls -a
    - 列出目录内容（所有的文件和目录）
- ls -A
    - 列出目录内容。(几乎所有的文件和目录，略去".." and ".")

- ls -la
    - 列出目录所有文件和目录的详细信息。查看 Ubuntu 中的文件系统概念, 第 4.5.2 节。
- ls -d *
    - 列出当前目录下的目录名称，而不是目录下的内容。
- lsof foo
    - 显示文件 foo 的打开状态。
- mkdir foo
    - 在当前目录下创建一个新的目录 foo。
- cd foo
    - 切换到当前目录下或者在变量 CDPATH 中列出来的目录 foo。在 builtins(7) 查看命令 cd。
- cd /
    - 切换到根目录。
- cd
    - 切换到用户主目录。
- cd /foo
    - 切换到绝对路径/foo 所指定的目录。
- cd ..
    - 切换到上一级目录。
- cd ~foo
    - 切换到用户主目录下的 foo 目录去。
- cd -
    - 切换到上一次所去的目录。
- </etc/motd pager
    - 使用默认的分页程序查看文件/etc/motd 的内容，参照 command < foo, 第 4.3.10.9 节。

- touch junkfile
    - 创建一个空文件 junkfile。
- cp foo bar
    - 拷贝一个已有的文件 foo 到新文件 bar。
- rm junkfile
    - 删除文件 junkfile
- mv foo bar
    - 把已有的文件 foo 重命名为 bar
- mv foo bar/baz
    - 把已有的文件 foo 移动到新位置并重命名为 bar/baz。目录 bar 必须存在。
- chmod 600 foo
    - 让已经存在的文件 foo 其他人不能读写。(所有人都 不能执行)。
- chmod 644 foo
    - 使文件 foo 其他的人可以读，但是不能写。(所有人 都不能执行)
- chmod 755 foo
    - 使文件 foo 其他的人能读不能写。(所有人都可以执行)
- top
    - 全屏显示进程信息。输入”q”退出。
- ps aux | pager
    - 用 BSD 风格输出所有正在运行的进程的信息。参照 command1 | command2, 第 4.3.10.2 节。
- ps -ef | pager
    - 用 system-V 风格来输出所有正在运行的进程的信息。
- ps aux | grep -e "[e]xim4*"
    - 显示 exim4 进程，或者运行 exim 的进程。输入 man grep 可以从 grep(1)的手册页学习正则表达式。
- ps axf | pager

- 用 ASCCI 艺术形式来显示运行所有进程信息。

- kill 1234
  - 杀死进程号为 1234 的进程。 查看 中止一个进程, 第 8.5.1 节。

- grep -e "pattern" *.html
  - 找到当前目录下面所有以.html 结尾的文件中含有"pattern"的行，并显示它们。

- gzip foo
  - 用 Lempel-Ziv(LZ77)压缩算法压缩 foo，生成 foo.gz。

- gunzip foo.gz
  - 将文件 foo.gz 解压缩生成 foo。

- bzip2 foo
  - 将文件 foo.bz2 解压缩生成 foo。

- tar -xvvf foo.tar
  - 从打包文件 foo.tar 解出文件来。

- tar -xvvzf foo.tar.gz
  - 从打包压缩的文件 foo.tar.gz 中解开文件。

- tar -xvvf --bzip2 foo.tar.bz2
  - 从文件 foo.tar.bz2 解压缩文件。

- tar -cvvf foo.tar bar/
  - 把目录 bar/的内容打包存放到 foo.tar 存档中。

- tar -cvvzf foo.tar.gz bar/
  - 把目录 bar/的内容打包并且压缩存放到 foo.tar.gz 存档中。

- tar -cvvf --bzip2 foo.tar.bz2 bar/
  - 把目录 bar/中的内容打包存放到 foo.tar.bz2 存档里面。

- zcat README.gz | pager
  - 实用默认的分页显示程序 pager 来显示压缩文件 README.gz 中的内容。

- zcat README.gz > foo
    - 使用文件 README.gz 解开后的内容创建一个文件 foo。
- zcat README.gz >> foo
    - 把文件 README.gz 解开后的内容追加到文件 foo 的后面(如果文件不存在的话，就会创建一个)。
- find . -name pattern
    - 用 shell 找到匹配 pattern 的文件名（慢一些)。
- locate -d . pattern
    - 用 shell 找到匹配 pattern 的文件名（使用已有的规则的数据库，快一些)。

请用上面的这些命令来查看您的系统的目录和其他的信息,以此来熟练一些操作。如果您 对上面的这些控制台命令有任何不明白的地方,请首先阅读帮助手册,比如下面的命令就是 好的开始：

```
$ man man
$ man bash
$ man ls
```

现在也是时候启动 vim 然后按下 F1 键了。您最少也应该阅读开始的 35 行。然后把光标移动到|tutor|，再按下 Ctrl-]就可以做在线测试了。查看 编辑器, 第 11 章可以学到更多关于编辑器的知识。

请注意许多来自于 GNU 和 BSD 的类 Unix 命令都会在您进行如下操作的时候（或者您没有给出任何参数）给出简单的帮助：

```
$ commandname --help
$ commandname -h
```

您也可以参照 Ubuntu 小技巧, 第 8 章里的例子来进行自我测试。

4.3.3 命令执行

现在您已经比较了解应该如何使用 Ubuntu 系统了。让我们来更深入的了解 Ubuntu 系统的命令执行结构。

4.3.4 简单的命令

一个简单的命令是如下面的序列

- 可变的声明 (可选)

- 命令的名字
- 参数 (可选)
- 重定向 (可选: >, >>, <, <<, 等等)
- 控制操作 (可选: &&, ||; <换新行>, ;, &, (,))

想了解更多复杂命令的解释和应用请查看 命令行处理, 第 13.2.6 节。

4.3.5 命令执行和环境变量

典型的使用 shell 来执行命令情况如下:

```
$ date
Sun Oct 26 08:17:20 CET 2003
$ LC_ALL=fr_FR date
dim oct 26 08:17:39 CET 2003
```

这里 date 程序是在前台执行的。环境变量 LC_ALL 是:

- 取消设置 (系统默认的和 C)作为第一个命令
- 设置为 fr_FR (French locale) 作为第二个命令。

绝大多数的命令并没有预先定义各种环境变量。像上面的例子选择如下方式:

```
$ LC_ALL=fr_FR
$ date
dim oct 26 08:17:39 CET 2003
```

正如您所看到的, 命令的输出和环境变量设置有关, 上面产生的是法语输出。如果您 想这个环境变量在子进程中也能够得到继承的话 (e.g.,执行 shell 脚本的时候), 您需要 使用下面的方式。

```
$ export LC_ALL
```

4.3.6 命令搜索路径

当您在 shell 提示符里面输入一个命令的时候, shell 就会在 PATH 环境变量所列出的目录里面去查找。PATH 环境变量的值也被叫做 shell 的查找目录。

在默认安装的 Debian 中, 用户的 PATH 环境变量里面也许没有包含/sbin/。因此如果您想运行一些命令比如/sbin/目录下的 ifconfig, 您就必须在 PATH 环境变量里面包含它。PATH 环境变量一般是在初始化文件~/.bash_profile 里面设置的, 参看 Bash 设置, 第 3.2 节。

4.3.7 命令行选项

一些命令带有参数, 参数部分以-或者--开始的被称做选项。可以用来控制命令的行为。

```
$ date
Mon Oct 27 23:02:09 CET 2003
```

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

这里命令行参数-R 改变了命令 date 的表现以让它输出符合 RFC-2822 的日期字符串。

4.3.8 Shell 通配符

通常您需要用命令来处理一组文件，但是您又不想输出所有的文件名。shell **通配符**使得这个成为可能。

- *
  - 这个匹配 0 个或者多个字符。
  - 它不会匹配以"."开始的文件名.
- ?
  - 这个仅匹配一个字符。
- [...]
  - 这个匹配[]里面的某个字符。
- [a-z]
  - 这个匹配字符 a 到 z 之间的某个字符。
- [^...]
  - 这个匹配任意不包含在[]里面的字符（不包含字符"^").

作为练习。请尝试着运行下面的命令并思考一下：

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

4.3.9 命令返回值

每个命令都返回一个值和它返回的状态。

- 返回 0 表示命令被正确执行
- 返回非 0 的值表示命令没有正确执行。

返回值可以在命令执行后马上用 shell 用变量$?来访问。

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

请注意，在 shell 中的逻辑上下文中使用这些返回值的时候 **success** 被看做逻辑值 **TRUE**。这个多少有一点不直观，因为 **success** 等价于值 **zero**。

参看 Shell 条件表达式, 第 13.2.5 节。

4.3.10 典型的命令序列

在我们阅读了这些惯用的 shell 命令以后，让我们试者记住它们。参看 Shell 参数, 第 13.2.3 节， Shell 重定向, 第 13.2.4 节， Shell 条件表达式, 第 13.2.5 节，和 命令行处理, 第 13.2.6 节。

4.3.10.1 command &

command 在子 shell 的 **background** 运行。后台任务让多成程序能够运行在一个 shell 里面。

管理这些后台任务的请求需要 shell 内建的： jobs，fg, bg,和 kill。请查看 bash(1)这一小节中的"SIGNALS", "JOB CONTROL","SHELL BUILTIN COMMANDS".的相关内容。

4.3.10.2 command1 | command2

command1 的标准输出被直接输入到 command2 的标准输入。 两个命令都可能**并行**地运行。这个被称作 **pipeline**。

4.3.10.3 command1 ; command2

command1command2 被有序的执行。

4.3.10.4 command1 && command2

command1 如果执行成功的话那么再执行 command2。只有当 command1 **并且** command2 都运行成功的话上面的命令序列才会成功返回。

4.3.10.5 command1 || command2

command1 被执行以后，如果不成功的话，command2 也会被执行。当 command1 **或者** command2 有一个执行成功的话，上面的序列就会返回真值。

4.3.10.6 command > foo

把 command 的标准输出重定向到文件 foo。（覆盖内容）

4.3.10.7 command >> foo

把 command 的标准输出重定向到文件 foo。（追加）

4.3.10.8 command > foo 2>&1

同时把 command 的标准输出和标准出错信息重定向到文件 foo。

4.3.10.9 command < foo

把 command 的标准输入重定向到一个文件 foo。

```
$ </etc/motd pager
... (the greetings)
$ pager </etc/motd
... (the greetings)
$ pager /etc/motd
... (the greetings)
$ cat /etc/motd | pager
... (the greetings)
```

尽管上面 4 个方法都显示同样的内容，但是最后一个多运行了 cat 命令。而且不必要的浪费了资源。

4.3.11 命令别名

您可以给一个命令序列起一个别名。比如：

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

现在， la 就成了 ls -la 命令的简写用来列出所有文件的详细信息。

您可以用 type 来显示命令的详细路径或者其他身份。比如：

```
$ 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
```

这里 ls 在最近被查找过了，而 file 没有，因此 ls 被"hashed"，i.e.,shell 有一个内部的记录可以用来快速的找到 ls 的地址。

4.4 类 Unix 文本处理

在类 Unix 的系统中,有几个文本处理工具经常用到。

- 非正则表达式的方法有：
    - head 显示文件的开始部分。
    - tail 显示文件的结尾部分。
    - sort 给文件中的每一行排序。
    - uniq 删除文件中重复的行。
    - tr 转换或者删除字符。
    - diff 把文件中的内容一行一行的比较。
- 基本的正则表达式 Basic regular expression (BRE) :
    - grep 按模式匹配文本。
    - ed 一个原始的行编辑器。
    - sed 一个流编辑器。
    - vi 一个屏幕编辑器。
    - emacs 一个屏幕编辑器。
- 扩展的正则表达式 Extended regular expression (ERE) is used:
    - egrep 按模式匹配文本。
    - awk 进行简单的文本处理。 查看 Awk, 第 13.3 节.
    - perl 做非常难以想像的文本处理。 查看 Perl, 第 13.4 节.

查看 正规表达式的置换, 第 8.6.13 节, 精巧的管道命令辅助脚本, 第 8.6.18 节, and 短小的 Perl 脚本, 第 8.6.20 节 可以找到一些脚本的例子。

4.4.1 正则表达式

正则表达式用在很多文本处理工具里面。它们和 shell 的通配符比较相似（查看 Shell 通配符, 第 4.3.8 节），但是它们更复杂也更强大。

正则表达式是由文本字符和**元字符**组成的,用来描述匹配模式。元字符是有特殊意义的字符 。它们有两种主要的形式 BRE 和 ERE，主要取决于 类 Unix 文本处理, 第 4.4 节里面是如何描述的。

在扩展的正则表达式（EREs）里面**元字符**包括"\ . [ ] ^ $ * + ? ( ) { } |"。正则表达式表示：

- c
    - 这个用来匹配非元字符 "c".
- \c
    - 这个用来匹配原本的字符"c".
- .

    - 这个用来匹配任意字符包括换行符。
- ^
    - 这个用来匹配字符串的开始。
- $
    - 这个用来匹配字符串的结尾。
- \<
    - 这个用来匹配一个单词的开始。
- \>
    - 这个用来匹配一个单词的结尾。
- [abc...]
    - 这个字符序列用来匹配 "abc..."中的任意字符。
- [^abc...]
    - 这个否定的字符序列匹配所有的字符除了 "abc...".
- r*
    - 这个匹配以"r" 开始的后面有 0 个或者多个字符的字符串。
- r+
    - 这个匹配以"r" 开始的后面有一个或者多个字符的表达式。
- r?
    - 以"r" 开始后面有 0 个或者 1 个其他的字符。
- r1|r2
    - 匹配"r1"或者"r2"。
- (r1|r2)
    - 匹配"r1"或者 "r2"并且把它当作一个**分类** 正则表达式。

在 BREs 里面**元字符**"+ ? ( ) { } |"不再具有它们特殊的含义，而是使用的有反斜杠的版本"\+ \? \(\) \{ \} \|"。因此分组(r1|r2)需要被表示成\(r1|r2\)。 因为 emacs，虽然基本上是 BRE，但是它把"+ ?"当作元字符。因此不需要特别表示它们。查看 替换表达式, 第 4.4.2 节来了解构造分组是如何使用的。

举个例子，grep 可以用正则表达式来搜索文本：

```
$ 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
```

4.4.2 替换表达式

在替换表达式里面，下面的字符有特殊的含义：

- &
    - 这个会替换正则表达式所匹配的部分。（在 emacs 里面用\&）
- \n
    - 这个会替换 n-th **括号**正则表达式所匹配的内容。

在 Perl 里面，$n 替换了\n，&也没有特殊的含义。

举个例子：

```
$ 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=&=
```

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文本处理工具里面是如何使用的。

这些正则表达式可以用来控制光标的运动和编辑器里面的文本替换。

请阅读所有相关手册来学习这些命令。

4.5 类 Unix 文件系统

在 GNU/Linux 和其他的类 Unix 操作系统里面**文件**都被放在**目录**。 所有的**文件**和**目录**都被排列在一棵很大的树里面， 即文件层次树，他的根是/。

这些文件和目录可以跨多个设备。mount(8)命令可以把一些设备挂载到文件系统树里面来。

反之 umount(8)可以把设备卸载。

4.5.1 Unix 文件基础

这里是一些最基础的：

- 文件名是区分大小写的，MYFILE 和文件 MyFile 是**不同**的文件。
- 根目录指的是/不要把这个”root”和 root 用户相混淆了。参看 用超级用户登录到命令提示符，第 4.1.1 节。
- 任何目录都有一个可以由任意字母或者符号组成的名字，但是/是**例外**。[31] 根目录是一个例外：它的名字是/（读做”slash″或者根目录）它不能被重命名。
- 任何一个目录都是有下面几种形式给出的，**完全限制的文件名**，**绝对文件名**，或者**路径**，给出所有需要经过的目录序列。这三种形式是等价的。所有的绝对文件名都以/目录开始，在目录和目录或者目录和文件之间用/隔开。最开始的/是一个目录的名字，但是后面的仅仅是文件名的分隔符。

上面的这些话看起来让人很费解。那么看看下面的例子吧：

/usr/share/keytables/us.map.gz

这就是一个完整限制的文件名；有些人把它叫做**路径**。然而人们经常 单独把 us.map.gz 作为文件名。

- 根目录有很多分支，比如/etc/和/usr/。这些子目录 又分出很多子目录来，比如/etc/init.d/和/usr/local/。 所有的加起来被称作**目录树**。

您可以想像，一个绝对文件名就是从树的根基(/)到一个分支的末端（文件）的路由。您也会听到别人把目录树叫做**家庭**树：这样子目录就有 **双亲**，路径就显示了所有文件完整的血缘关系。 除此之外，还有相对路径，它从其它的地方开始而不是根目录。您可能还记得 ../ 指的是上一级目录。

- 没有哪一个目录是和物理设备紧密关联的，比如您的磁盘。这个和DOS,CP/M,Windows系统不一样的，它们所有的路径都是以一个设备名开始的，比如 C:\，参看 Ubuntu 中的文件系统概念，第 4.5.2 节。

关于文件层次的详细信息以及最好的操作练习可以在这里找到 Filesystem Hierarchy Standard。作为一个初学者，您需要记住下面的事实：

- /
    - 简单的一个/表示根目录。
- /etc/
    - 这个是大多数系统配置文件存放的地方。
- /var/log/
    - 这个是系统日志存放的地方。

- /home/
    - 这个目录是存放所有非特权用户的主目录的。

4.5.2 Ubuntu 中的文件系统概念

按照 Unix 的传统， Ubuntu 为存放物理数据的磁盘或者其它存储设备，还有这些硬件设备之间的交互提供文件系统，比如控制台屏幕和远程串行终端就用联合的方式表示。

每个在 Ubuntu 系统上的文件，目录，命名管道，或者物理设备都有一个数据结构被称作 **inode**，它被用来描述设备用用的属性，比如设备所有者，所属于的组， 上次访问时间等等 。参看/usr/include/linux/fs.h来获得 struct inode 在 Debian GNU/Linux 系统中的详细信息。

这些表现出来的统一的物理入口是非常强大的，因为它们使得可以使用同样的命令和同样的操作来访问完全不同的设备。

您所有的文件都可以在一个次方上面，----或者您有 20 个磁盘，有些是在网络上其它的计算机上面， 在 GNU/Linux 系统中每个文件和目录都与其拥有者（主人）和拥有组相关联。所有的文件信息都保存在一个称为 **inode** 的数据结构中。

4.5.3 文件和目录的访问权限

文件和目录的访问权限对如下 3 类用户进行了分别定义：

- 文件**拥有者**(u)，
- 文件拥有者所在**用户组**中的其它成员(g)，和
- 所有**其它**用户(o)。

每个文件均拥有下列三种权限：

- **read** (r): 查看文件内容
- **write** (w): 修改文件
- **execute** (x): 如同命令一样执行文件

每个目录均拥有下列三种权限：

- **read** (r): 列出目录内容
- **write** (w): 在目录中增删文件
- **execute** (x): 访问目录中的文件

在此，对目录的 **execute** 权限，不仅意味着允许查看目录下文件的内容，还允许查看文件的其它信息如文件大小、修改时间。

ls 可用来显示目录和文件的这些信息。参阅 ls(1)。使用 ls 的-l 选项，就会按如下顺序显示下列信息：

- **文件类型**（第 1 个字符）
    - -: 普通文件
    - d: 目录

    - l: 符号链接
    - c: 字符型设备节点
    - b: 块设备节点
    - p: 命名管道
    - s: 套接字

- 文件访问**权限**（接下来的 9 个字符，每 3 个一组依次代表 user、group 和 other）。
- 文件的**硬链接数**
- 文件拥有 **user** 的用户名
- 文件所属 **group** 的用户组名
- 文件的字符数**大小** (bytes)
- 文件的**时间和日期** (mtime)
- 文件的**名称**

在 root 账号下可使用 chown 改变文件的拥有者。要改变文件的所属组，可以文件拥有者或 root 的身份运行 chgrp。要改变目录的访问权限，可以文件拥有者或 root 的身份运行 chmod。

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

细节请参见 chown(1)、 chgrp(1) 和 chmod(1)。

例如，可以 root 帐号下创建一个目录树，并使其拥有者为 foo，所属组为 bar：

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

下面是 3 个更特殊的权限：

- **set user ID** (s 或 S 代替 user's x),
- **set group ID** (s 或 S 代替 group's x),
- **sticky bit** (t 或 T 代替 other's x).

在此，如果隐藏在特殊权限后面的执行权限标位没有设置，则 ls -l 的输出中，这些标识位将使用大写字母。

为可执行文件设置 **set user ID** 位将允许用户以该文件拥有者的 ID 来执行该文件（例如以 **root** 身份）。同样，为可执行文件设置 **set group ID** 将允许用户以该文件所属组的 ID 来执行该文件（例如以 **root** 身份）。因为这些设置将引起安全风险，所以使用这些特性时要格外小心。

为目录设置 **set group ID**，则该目录会使用 BSD-like 文件创建方案，即目录中所有新创建的文件均属于该目录所属的 **group**。

为目录设置 **sticky bit** 可防止非文件拥有者移动目录中的文件。为确保全局可写目录如 /tmp 或组可写目录中的文件内容不被修改，不仅要关闭文件的**写**权限，还应设置目录的 **sticky bit**，否则，任何对该目录有写权限的用户均可以将该文件移动到别处，然后在原地

创建一个同名文件。

这儿有一些有关文件权限的有趣例子。

```
$ ls -l /etc/passwd /etc/shadow /dev/ppp /usr/sbin/pppd
crw-rw----    1 root     dip      108,    0 Jan 18 13:32 /dev/ppp
-rw-r--r--    1 root     root         1051 Jan 26 08:29 /etc/passwd
-rw-r-----    1 root     shadow        746 Jan 26 08:29 /etc/shadow
-rwsr-xr--    1 root     dip        234504 Nov 24 03:58 /usr/sbin/pppd
$ ls -ld /tmp /var/tmp /usr/local /var/mail /usr/src
drwxrwxrwt    4 root     root         4096 Feb  9 16:35 /tmp
drwxrwsr-x   10 root     staff        4096 Jan 18 13:31 /usr/local
drwxrwsr-x    3 root     src          4096 Jan 19 08:36 /usr/src
drwxrwsr-x    2 root     mail         4096 Feb  2 22:19 /var/mail
drwxrwxrwt    3 root     root         4096 Jan 25 02:48 /var/tmp
```

在 chmod(1) 命令里，有一种替代的数字模式来描述文件权限。这种数字模式使用 3 到 4 个八进制数字（以 8 为基）。 每个数字相应如下：

- 第一个可选数字： **set user ID** (=4)、 **set group ID** (=2) 和 **sticky bit** (=1) 之和
- 第二个数字： **user** 的 **read** (=4)、**write** (=2) 和 **execute** (=1) 权限之和
- 第三个数字： 同上，用于 **group**
- 第四个数字： 同上，用于 **other**

这听起来复杂，但实际上相当简单。 从 ls -l 命令的输出中，看第（2-10）列，把它们作为二进制（以 2 为基）文件权限（“-”表示“0”,“rwx” 表示“1"）的表示方式来读， 这种数字模式值将使你理解八进制（以 8 为基）的文件权限表示。 例如，尝试：

```
$ touch foo bar
$ chmod u=rw,go=r foo
$ chmod 644 bar
$ ls -l foo bar
-rw-r--r--    1 penguin  penguin  0 Nov  3 23:30  foo
-rw-r--r--    1 penguin  penguin  0 Nov  3 23:30  bar
```

默认的文件权限掩码使用 shell 内建命令 umask 设置。 参见 builtins(7)。

4.5.4 时间戳

GNU/Linux 的文件有 3 种类型的时间戳：

- **mtime**: 修改时间 (ls -l),
- **ctime**: 状态改变时间 (ls -lc)，以及
- **atime**: 最近访问时间 (ls -lu).

注意 **ctime** 并非文件创建时间。

- 覆盖一个文件会改变所有三类时间 **mtime**、**ctime** 和 **atime** 所有三类时间。
- 改变文件的访问权限或拥有者会改变文件的 **ctime** 和 **atime**。

- 读文件会改变文件的 **atime**。

注意，在 Debian 系统中，即便是简单的读文件通常会引起文件的写操作，从而更新 **inode** 上的 **atime** 信息。使用 noatime 选项来挂载文件系统，可使用系统忽略该操作，从而加速文件的访问和读取。参阅 mount(8)。

使用 touch(1) 命令来改变存在文件的时间戳。

4.5.5 链接

在 2 种方法将文件 foo 关联到不同的文件名 bar。

- **hardlink**（硬链接）相当于现存文件的另一个名字。(ln foo bar)，
- **symbolic link**（符号链接）,或者“symlink"，是通过名字指向另外一个文件的特殊文件。(ln -s foo bar)。

下面的例子显示了链接数的改变和使用 rm 命令时产生的微妙差异。

```
$ echo "Original Content" > foo
$ ls -l foo
-rw-r--r--    1 osamu    osamu           4 Feb  9 22:26 foo
$ ln foo bar     # 硬链接
$ ln -s foo baz  # 符号链接
$ ls -l foo bar baz
-rw-r--r--    2 osamu    osamu           4 Feb  9 22:26 bar
lrwxrwxrwx    1 osamu    osamu           3 Feb  9 22:28 baz -> foo
-rw-r--r--    2 osamu    osamu           4 Feb  9 22:26 foo
$ rm foo
$ echo "New Content" > foo
$ cat bar
Original Content
$ cat baz
New Content
```

上例中符号链接均拥有名义上的文件访问权限“rwxrwxrwx"，它们的有效访问权限均由它们所指向的文件来定义。

- 链接到它所属的目录,因此任何新目录的链接数都从 2 开始计算。.. 链接到父目录，因些目录链接数会随新的子目录数的增长而增长。

4.5.6 命名管道（FIFOs）

命名管道是一个行为象一个管道的文件。把某些东西放入命名管道文件，它从另外的一端出来。因此，它被称为 FIFO，或 First-In-First-Out：首先放入管道的东西将首先从另外一端出来。

如果写东西到一个命名管道，在写的东西在管道被读出之前，写的进程不会中止。如果从一个命名管道读，读的进程在中止之前，一直处于等待状 态，直到有东西可以读为止。命名管道的大小始终为零 --- 它不储存数据,象 shell | 一样,它仅仅连接两个进程。然而，这个管道有一个名字，两个进程没有必要在同一个命令行运行，或者由同一个用户运行。

做下面的操作来尝试：

```
$ cd; mkfifo mypipe
$ echo "hello" >mypipe & # 放入后台
[1] 5952
$ ls -l mypipe
prw-r--r--    1 penguin penguin  0 2003-11-06 23:18 mypipe
$ cat mypipe
hello
[1]+  Done                    echo hello >mypipe
$ ls mypipe
prw-r--r--    1 penguin penguin  0 2003-11-06 23:20 mypipe
$ rm mypipe
```

4.5.7 套接字

套接字类似于命名管道（FIFO），它允许进程交换信息。 对于套接字，那些进程不必要在同时运行，也没有必要是同一个祖先进程的子进程。 它是进程间通讯的端点。 信息交换可以通过网络发生在不同的主机之间。

4.5.8 设备文件

设备文件是系统上物理的或者虚拟的设备，比如说硬盘、显卡、显示器或者键盘。 一个虚拟设备的例子是控制台，由 /dev/console 表示。

有两种类型的设备：

- **字符设备**
    - 一次能够访问一个字符，那就是说，从设备读或者写的最小的数据单元是一个字符（byte）。
- **块设备**
    - 一定是访问一个叫块的大单元，它含有许多字符。 硬盘是一个块设备。

设备文件可以被读写，尽管设备文件包含二进制数据，而这些二进制数据对人类来说是费解的乱码。 向设备文件直接写数据，有时侯对解决硬件连接故障有用。 比如说，将一个文本文件导出到打印机设备 /dev/lp0， 或者发送调制解调器命令到一个适当的串口 /dev/ttyS0。 但是，除非是慎重的操作，它有可能造成一个大的破坏。 所以请小心。

4.5.8.1 /dev/null 等等

/dev/null 是一个特殊的设备文件，它忽略写给它的任何东西。 如果不需要某些东西，把它扔到 /dev/null。 它本质上是一个无底洞。 如果从 /dev/null 读东西，将会立即得到文件结束符（EOF）。

/dev/zero 是类似的，只是从它读的话，将会得到 \0 字符（不与数字零的 ASCII 码相同）。 参阅 空文件, 第 8.6.34 节。

4.5.8.2 设备号

按例子执行 ls，将显示设备号。

```
$ ls -l /dev/hda /dev/ttyS0 /dev/zero
brw-rw----    1 root     disk       3,   0 Mar 14  2002 /dev/hda
crw-rw----    1 root     dialout    4,  64 Nov 15 09:51 /dev/ttyS0
crw-rw-rw-    1 root     root       1,   5 Aug 31 03:03 /dev/zero
```

在这里：

- /dev/hda 主设备号为 3，次设备号为 0。 属于 disk 组的用户有读写访问权。
- /dev/ttyS0 主设备号为 4，次设备号为 64。 属于 dialout 组的用户有读写访问权。
- /dev/zero 主设备号为 1,次设备号为 5。 所有人都有读写访问权。

在老的系统中，安装过程使用 /sbin/MAKEDEV 命令来创建设备号。 参阅 MAKEDEV(8)。

在新的系统中， /dev 下的文件系统通过设备文件系统自动生成， 设备文件系统与 /proc 文件系统类似。

4.5.9 /proc 文件系统

/proc 文件系统是一个伪文件系统，它包含系统信息和正在运行的进程信息。

当注意到一个特殊的文件 /proc/kcore 时，人们经常恐慌，因为它通常很巨大。该文件（或多或少）是计算机内存的一个拷贝。 它被用来调试内核， 实际上它并不存在，所以不必担心它的大小。

参阅 通过 proc 文件系统调整内核, 第 7.3 节 和 proc(5)。

4.6 X 窗口系统

参阅 X, 第 9.4 节。

4.6.1 启动 X 窗口系统

X 窗口系统能够使用类似 xdm 的图形登录守护启动，或者在控制台下输入如下的命令启动:

```
$ exec startx
```

4.6.2 X 窗口系统下的菜单

X 环境能够与许多窗口管理器协作，各个窗口管理器的用户界面有很大不同。 请记住，右击根窗口将显示一个选择菜单。这个功能总是存在。

- 得到 shell 命令提示符，从菜单启动 Xterm ：
    - "XShells" --> "XTerm".

- 浏览有图形的网页，从菜单启动 Mozilla：
    - "Apps" --> "Net" --> "Mozilla Navigator".
- 浏览有图形的 PDF 文件，从菜单启动 Xpdf：
    - "Apps" --> "Viewers" --> "Xpdf".

如果没有发现菜单条目，请安装适当的软件包。参阅 Ubuntu 软件包管理基础, 第 6.2 节。

4.6.3 X 窗口系统键盘序列

当运行 X 窗口系统时，下面是一些需要记住的重要键盘序列。

- Ctrl-Alt-F1 到 F6: 切换到其它伪终端（从 X 系统、DOSEMU 等。）
- Alt-F7: 切换回 X 窗口
- Ctrl-Alt-minus: 改变 X 窗口的屏幕解析度（减号为数字键盘的键）
- Ctrl-Alt-plus: 在 X 窗口内以相反的方向改变屏幕解析度(加号为数字键盘的键)
- Ctrl-Alt-Backspace: 中止 X 服务器程序
- Alt-X, Alt-C, Alt-V: 通常在 Windows/Mac 下与 Ctrl- 键联合的粘贴、拷贝和剪切键，在 Netscape Composer 等程序中，使用 Alt- 键代替。

4.7 进一步学习

目前，推荐阅读来自 The Linux Documentation Project: Guides 的关键用户手册。

- "The Linux System Administrators' Guide",
    - 覆盖了系统运行、处理用户账号、备份和配置系统的各个方面的内容。
    - 软件包： sysadmin-guide
    - 文件： /usr/share/doc/sysadmin-guide/html/index.html
    - 网址： http://www.tldp.org/LDP/sag/index.html
- "The Linux Network Administrator's Guide, Second Edition",
    - 在 Linux 环境下网络管理的简单参考
    - 软件包： (not available)
    - 文件：(not applicable)
    - 网址：http://www.tldp.org/LDP/nag2/index.html
- "Linux: Rute User's Tutorial and Exposition"
    - 覆盖 GNU/Linux 系统管理的精装书（有在线版）
    - 作者：Paul Sheer
    - 出版：Prentice Hall
    - 软件包：rutebook (从 non-free)
    - 文件： /usr/share/doc/rutebook/

更多资源参阅 Ubuntu 技术支持, 第 15 章。

第 5 章 - 发行版升级到 Breezy、Dapper 或 Edgy

升级的官方发布通知位于 http://www.ubuntu.com/ubuntu/releases 和 http://www.ubuntu.com/news（不断更新中）。

将系统升级到 Breezy、Dapper 或 Edgy 需要几个步骤，而且必须按照下面的顺序：

- 升级到 Hoary （如果你的系统比 Hoary 要旧）
- 升级到 Breezy
- 升级到 Dapper
- 升级到 Edgy

Ubuntu 不支持省略中间发布的升级

5.1 升级到 Hoary

升级系统到 Hoary。

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

5.2 准备升级工作

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

生成一个干净的 stable 版存储列表：

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

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

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

调整 /etc/apt/sources.list 和 /etc/apt/preferences 的艺术请参阅 Ubuntu 软件包管理基础, 第 6.2 节 。

5.3 升级

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

软件包的实质性信息请参见 Ubuntu 软件包管理, 第 6 章，如果你遇到问题，请查看 APT 升级错误以及解决方法, 第 6.3.2 节。

5.3.1 使用 dselect

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

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

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

```
# dselect install
```

在安装过程中，必须回答一些有关软件包配置的问题，准备好你的笔记本花点时间处理它们。参阅 dselect, 第 6.2.4 节。

使用 dselect。**它能干得不赖 :)**

5.3.2 使用 apt-get

```
# apt-get update
# apt-get -t breezy upgrade
# apt-get -t breezy dist-upgrade
# apt-get -t dapper upgrade
# apt-get -t dapper dist-upgrade
# apt-get -t edgy upgrade
# apt-get -t edgy dist-upgrade
```

一旦你的系统到达 hoary ，使用 aptitude 代替 apt-get 是明智的。(aptitude 接受 apt-get 所接受的许多选项，包括上面列出的那些选项。)

升级到目前 dselect 的设置：

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

参阅 软件包依赖关系, 第 2.2.8 节。

第 6 章 - Ubuntu 软件包管理

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

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

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

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

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

在 chroot, 第 8.6.35 节 中描述的使用方法非常适合于需要同时确保系统的稳定性和使用最新软件的情况。

6.1 介绍

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

6.1.1 主要的包管理工具

dpkg - Debian 包安装工具
apt-get - APT 的命令行前端
aptitude - APT 的高级的字符和命令行前端
synaptic - 图形界面的 APT 前端
dselect - 使用菜单界面的包管理工具
tasksel - Task 安装工具

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

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

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

相反 aptitude 可以设置成安装所安装软件推荐的或者建议的软件包。

dselect 给使用者列出所安装软件推荐或建议的软件包，可以进行单独选择。 参阅 软件包依赖关系, 第 2.2.8 节.

6.1.2 方便的工具

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

6.2 Ubuntu 软件包管理基础

6.2.1 设置 APT

参考 准备升级工作, 第 5.2 节 来设置 sources.list。 请参考 Ubuntu 系统安装提示, 第 3 章, 发行版升级到 Breezy、Dapper 或 Edgy, 第 5 章, 和 应急的编辑器, 第 11.2 节.

6.2.2 安装 tasks

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

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

```
dselect update
```

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

6.2.3 aptitude

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

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

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

```
按键        动作
F10         菜单
?           按键命令帮助（完整的清单）
u           更新软件包信息
+           标记软件包为升级或者新安装
-           标记软件包为删除（保留配置文件）
_           标记软件包为完全删除（删除配置文件）
===           保持软件包的当前版本，阻止其被升级
U           标记所有可以升级的软件包为升级
```

g	下载和安装选择的软件包
q	退出当前屏幕，保存改变
x	退出当前屏幕，忽略改变
Enter	查看一个软件包的信息
C	查看一个软件包的更新日志
l	改变软件包树状显示限制
/	搜索第一个匹配的软件包
\	重复最后一次搜索

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

aptitude 的其他特点如下：

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

6.2.4 dselect

dselect 一直是主要的包维护工具。你可以考虑用 aptitude 代替。

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

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

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

使用 D 和 Q，你可以选择有冲突的选项。请小心使用这个命令。

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

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

6.2.5 使用 APT 来维护发行版本

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

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

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

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

关于限制特定软件在特定版本上，而其他软件随系统升级的设置在 examples subdirectory 找到，即 preferences.dapper 和 preferences.edgy。

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

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

Ubuntu 不支持将某个**软件包**降级到先前的发行版本。然而在新的软件包出问题时，你会发现你不得不安装旧的可用的软件包。你可以在本地的 /var/cache/apt/archives/ 或远端的 http://archive.ubuntu.com/ 中找到先前的版本。请参考 使用 dpkg 救助，第 6.3.3 节。

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

6.2.6 aptitude，apt-get 和 apt-cache 命令

还是以上面使用 testing 发行版的用户为例，可使用下列命令来管系统：

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

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

- apt-get dselect-upgrade

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

- aptitude install package/edgy

从 edgy 中安装 package，并由 dapper 版本提供安装依赖的包。

- aptitude install -t edgy package

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

- apt-cache policy foo bar ...

检查 foo bar ... 软件包的状态。

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

查看 foo bar ... 软件包的有关信息。

- aptitude install foo=2.2.4-1

安装 foo 软件包的特定版本 2.2.4-1。

- aptitude install foo bar-

安装 foo 软件包并删除 bar 软件包。

- aptitude remove bar

删除 bar 软件包，但保留其配置文件。

- aptitude purge bar

删除 bar 软件包及其所有配置文件。

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

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

使用 --no-act 可进行模拟升级，并不是进行真正的升级行为。

6.3 Ubuntu 生存命令

掌握了这些知识，你就能够享受无尽的“升级”了 :-)

6.3.1 检测程序错误寻求帮助

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

```
$ lynx https://launchpad.net/distros/ubuntu/+bugs/
$ lynx https://launchpad.net/distros/ubuntu/+bugs/package-name  # 如果你知道软件包的名字
$ lynx https://launchpad.net/distros/ubuntu/+bugs/bugnumber     # 如果你知道错误序号
```

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

如有疑问，可阅读帮助文件。设置 CDPATH 如下：

```
export CDPATH=.:/usr/local:/usr/share/doc
```
然后输入

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

更多技术支持资源列在 Debian 技术支持, 第 15 章。

6.3.2 APT 升级错误以及解决方法

从 edgy/dapper 进行升级时可能出现 升级, 第 5.3 节 中提到的软件包关联问题。多数情况下，是因为升级的软件包所需的新增的关联包没有安装。可使用如下方法解决：

```
# aptitude dist-upgrade
```

如果这招无效，可以重复下面的方法至到问题解决：

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

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

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

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

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

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

还可以用 equivs 包来解决此类问题。参阅 /usr/share/doc/equivs/README.Debian 和 equivs 软件包, 第 6.5.2 节。

6.3.3 使用 dpkg 救助

如果你在使用 APT 的时候遇到死胡同了，那么可以从 Ubuntu 的镜像站点下载软件包并使

用 dpkg 来安装。如果你不能访问网络，可以在 /var/cache/apt/archives/ 中找到被缓存的软件包。

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

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

6.3.4 恢复软件包选择状态的数据

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

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

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

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

6.3.5 /var 崩溃之后如何恢复系统

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

从相同或旧版本的最简 Ubuntu 系统中取得 /var 目录的内容框架，例如 var.tar.gz，然后它放入受损系统的 root 目录，接着

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

上述步骤可使系统恢复工作。使用 恢复软件包选择状态的数据, 第 6.3.4 节 中描述的技术加速软件包选择数据的恢复。([FIXME]：该过程需要更多的实践来检验)

6.3.6 为无法启动的系统安装软件包

使用 Ubuntu 急救软盘 /CD 或从多启动 Linux 系统其它分区启动。 参阅 启动系统, 第

8.1 节. 将无法启动的系统挂载到 /target 并使用 dpkg 的 chroot 安装模式。

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

接下来就可以着手配置并解决问题。

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

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

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

6.3.7 如果 dpkg 命令出错怎么办

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

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

对 i386，亦可用 http://packages.ubuntu.com/dpkg 作为 URL。

6.4 Ubuntu 必杀技

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

6.4.1 文件信息

在已安装的软件包中许找特定文件所属的软件包:

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

或者搜索 Ubuntu archive:

```
$ wget http://archive.ubuntu.com/ubuntu/dists/dapper/Contents-i386.gz
$ zgrep -e pattern Contents-i386.gz
```

或是用专门的软件包命令：

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

6.4.2 软件包信息

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

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

你也这可这样查看软件包信息（我用 mc 浏览）：

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

比较下面的文件可以确切了解最近的安装过程对系统造成了那些改变。

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

6.4.3 使用 APT 无人执守安装

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

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

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

安装完毕以后，可以用 重新配置已安装的软件包, 第 6.4.4 节 中的方法配置特定的软件

包。

6.4.4 重新配置已安装的软件包

使用下列方法重新配置已安装的软件包。

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

如果你想永久改变 debconf 对话框模式，可这么做。

某些程序用于生成特殊的配置脚本。

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

6.4.5 删除和清除软件包

删除软件包但保留其配置文件：

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

删除软件包并清除配置文件：

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

6.4.6 阻止旧软件包升级

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

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

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

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

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

该命令可以显示处于“阻止”状态的软件包：

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

6.4.7 breezy/dapper/edgy 混合系统

apt-show-versions 可以列出发行版中可用软件包的版本。

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

6.4.8 删除缓存包文件

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

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

6.4.9 记录/拷贝系统配置

对软件包选择情况进行本地备份：

```
$ dpkg --get-selections "*" >myselections   # 或使用 \*
```

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

你可将这个文件发送到另一台电脑并在那儿按文件中的选择进行软件包安装。

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

6.4.10 向 breezy 系统引入软件包

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

```
deb-src http://archive.ubuntu.com/ubuntu dapper \
main multiverse restricted universe
deb-src http://archive.ubuntu.com/ubuntu edgy \
main multiverse restricted universe
```

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

然后下载源码并在本地生成软件包：

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

$ dpkg-buildpackage -rfakeroot

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

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

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

现在，这些关联问题已被简化。例如，编译 pine 源码包：

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

6.4.11 本地软件包文件

为了创建与 APT 和 dselect 系统兼容的本地软件包文件，需要创建 Packages，包中文件

要放在特定的目录树中。

Ubuntu 官方包文件喜欢存放于本地 deb 仓库，下面就来创建仓库：

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

还有一种快速但是肮脏的方法来创建本地 deb 仓库：

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

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

6.4.12 转换或安装外来的二进制软件包

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

6.4.13 自动安装命令

auto-apt 是一种请求式软件包安装工具。

```
$ sudo auto-apt update
... 升级数据库
$ auto-apt -x -y run
```

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

6.4.14 校验已安装的软件包

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

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

per Joerg Wendland joergland@debian.org (untested).

6.4.15 优化 sources.list

简而言之，我尝试过用各种优化方法来创建 sources.list，但任何一种方法对我这个住在美国的人来说都没有明显的改善。最后我还是用 apt-setup 手工选择近一点的站点。

apt-spy 会根据站点回应时间和带宽自动创建 sources.list。netselect-apt 会创建一个更完整的 sources.list 文件，但它使用更落后的方法来选择镜像站点（比较 ping 时间）。

```
# aptitude install apt-spy
# cd /etc/apt ; mv sources.list sources.list.org
# apt-spy -d dapper -l sources.apt
```

6.5 其他 Ubuntu 的特性

6.5.1 dpkg-divert 命令

使用文件**转移**(diversions)的方法可以强令 dpkg 将文件安装到 **转移** 目录而非默认目录。对于某个引起冲突的文件，可以在 Ubuntu 软件包脚本中使用 **Diversions** 将它安装到别的目录。系统管理员还可以使用 diversion 来重载软件包配置文件，或者用来保留某些旧配置文件（这些文件没有在 **conffiles** 中登记）当安装新版软件时这些文件会被覆盖。（参阅保存本地配置, 第 2.2.4 节）。

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

记住，不到万不得已不要使用 dpkg-divert。

6.5.2 equivs 软件包

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

equivs。

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

6.5.3 Alternative 命令

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

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

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

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

想设置你喜爱的 X window 环境，执行 update-alternatives 来指定 /usr/bin/x-session-manager 和 /usr/bin/x-window-manager。详情参阅 自定义 X 会话, 第 9.4.5.1 节。

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

6.5.4 运行级别 Runlevel

安装好之后，大部分 Ubuntu 软件包的服务被设定为在 runlevel 2 到 5 时运行。所以，在没有定制过的 Ubuntu 系统中，runleve 2、3、4、5、6 是没有区别的 Ubuntu 保留这些给本地管理员使用。 自定义运行级别, 第 2.4.3 节 说明如何定制 runlevels。 这样的 runlevels 系统和其他流行的 GNU/Linux 发行版本完全不同。你可能要做的改变之一就是取消 runlevel 2 上的 xdm 和 gdm，使得在完成启动之后 X 显示管理去不会自动启动；然后你可以通过切换到 runlevel 3 来启动 X 显示管理器。

参阅 运行级别, 第 2.4.2 节 来获得更多关于 runlevels 的信息。

6.5.5 停止 daemon 服务

Ubuntu 发行版非常注重系统安全，并期望系统管理员能担此重任。它将系统的易用性放在

了第二位，许多 daemon 服务都定位在最高安全级别，因而，默认安装状态下系统只启动最少的（甚至没有）可用的服务。

如果拿不定把握（有关 Exim、DHCP...），可执行 ps aux 或检查 /etc/init.d/* 和 /etc/inetd.conf 下的内容,还可以使用 用 PAM 来控制登录, 第 9.2.1 节 中提到的方法检查 /etc/hosts.deny。pidof 命令也很有用（参阅 pidof(8)）

在 Ubuntu 系统中，默认状态下 X11 不允许 TCP/IP（远程）连接。参阅 在 TCP/IP 中使用 X, 第 9.4.6 节，但是使用 SSH 进行 X 传送是允许的，参阅 联接远程的 X 服务器 - ssh, 第 9.4.8 节。

第 7 章 - Ubuntu 下的 Linux 内核

Ubuntu 使用自己的方法来编译内核及相关模块。参阅 Ubuntu 和系统内核, 第 2.7 节。

7.1 内核编译

Ubuntu Edgy 发行版中的 gcc、binutils 和 modutils 可用来编译最新的 Linux 内核。这方面的官方信息，参阅 /usr/share/doc/kernel-package/README.gz，特别是文件的后半部分。

内核编译是个很困难的议题，由于目标在不断的变化，即使是最受人尊敬的开发者也会有不同的见解。

对于单机内核编译，initrd 不是必须的。我用它是希望我新编译的内核与相应的内核镜像一模一样。如果使用 initrd，请先阅读一下 mkinitrd(8) 和 mkinitrd.conf(5)。

7.1.1 Ubuntu 标准方式

关心一下有关 kernel-package、gcc、binutils 和 modutils 的错误报告。在需要时使用较新的版本。

在 Ubuntu 系统中用源码编译自定义内核要特别小心。用 make-kpkg 的 --append_to_version 选项来创建多重内核镜像比较安全。

```
# apt-get install debhelper modutils libncurses5-dev
# apt-get install linux-source-2.6.15   # 使用最新版本
# apt-get install fakeroot
# vi /etc/linux-pkg.conf                # 输入我的名字和 email
$ cd /usr/src                           # 创建目录
$ tar --bzip2 -xvf linux-source-2.6.15.tar.bz2
$ cd linux-source-2.6.15                # 如果这是你的内核源码
$ cp /boot/config-2.4.18-386 .config    # 将当前配置设定为默认配置
$ make menuconfig                       # 按自己的喜好来定制
$ make-kpkg clean                       # 必须执行这步(per: man make-kpkg)
$ fakeroot make-kpkg --append_to_version -486 --initrd \
--revision=rev.01 kernel_image \
modules_image # modules_image 可以是 pcmcia-cs* 等。
```

```
$ cd ..
# dpkg -i linux-image*.deb pcmcia-cs*.deb # 安装
```

make-kpkg kernel_image 实际上执行了 make oldconfig 和 make dep。如果没使用 initrd 就不要使用 --initrd 选项。

如果想加载 pcmcia-cs 模块或内核 pcmcia 源码中没有的模块，应该在 make menuconfig 后选“General setup —>”进入“PCMCIA/CardBus support —>”,配置“< >PCMCIA/CardBus support”选项（例如，取消复选项）。

对于 SMP 机器，参照 kernel-pkg.conf(5) 的说明设置 CONCURRENCY_LEVEL。

7.1.2 经典方式

从下列地址获得干净的源代码：

- Linux: http://www.kernel.org/
- pcmcia-cs: http://pcmcia-cs.sourceforge.net/

或使用 Ubuntu 所附的等价的源代码：

```
# cd /usr/src
# tar xfvz linux-whatever.tar.gz
# rm -rf linux
# ln -s linux-whatever linux
# tar xfvz pcmcia-cs-whatever.tar.gz
# ln -s pcmcia-cs-whatever pcmcia
# cd linux
# make menuconfig
... 配置内核选项 ...
# make dep
# make bzImage
... 编辑 lilo/grub ...
... 移动 /usr/src/linux/arch/i386/boot/bzImage 到 boot ...
... /sbin/lilo or whatever you do for grub
# make modules; make modules_install
# cd ../pcmcia
# make config
# make all
# make install
... 添加需要的模块名称到 /etc/modules
# shutdown -r now
... 启动到新内核 ...
```

7.1.3 内核头文件

绝大多数“普通”程序不需要内核头文件，事实上如果直接引用它们会出错。这些程序应该引用那些**编译 glibc** 所用的头文件，它们位于 Ubuntu 系统的 /usr/include/linux 和 /usr/include/asm 目录下。

故不要在 /usr/src/linux 目录中创建指向 /usr/include/linux 和 /usr/include/asm 的链接，一些过时的文档曾建议创建它们。

如果某些内核类应用程序**需要**特定的内核头文件，可修改 Makefile(s)，使其包含指向“特定内核头文件目录/include/linux”和“特定内核头文件目录/include/asm”的路径。

7.2 模块化的 2.4 内核

kernel-image-2.4.NN 提供了新版的 Debian 2.4 内核，该版内核模块化程度极高。你必须激活相关的模块才能获得想要的内核功能。

尽管在接下来的部分中提供了许多通过配置 /etc/modules 来解决问题的范例。但据说，在 /etc/modutils/ 中用一个文件来提供所有的设备别名，就可解决这类有关模块问题，当前的内核有足够多的别名供你使用。某些模块也可以被硬件探测程序自动激活，例如 discover。参阅 X 服务器的硬件侦测, 第 9.4.2 节。

参阅 模块加载规定, 第 2.7.5 节和 Linux 内核源码目录中的 Documentation/*.txt 获取详细信息。

7.2.1 PCMCIA

要使一些老的 PCMCIA 卡能正常工作，你需在 /etc/modules 中包含下列内容：

```
# ISA PnP driver
isa-pnp
# New Low level PCMCIA driver
# yenta_socket # 我的机器上似乎不需要
```

剩下的工作就由 PCMCIA 脚本（来自 pcmcia-cs 软件包）、depmod 和 kmod 负责了。我需要 isa-pnp 因为我的笔记本电脑使用的是旧 ISA-PCMCIA。较新的笔记本电脑使用 CardBus/PCMCIA，不再需要它。

参阅 Linux PCMCIA HOWTO 和 网络设置和 PCMCIA, 第 10.8.5 节。

7.2.2 SCSI

[没有测试过] 想要 SCSI 工作，请在 /etc/modules 中包含如下内容：

```
# SCSI core
scsi_mod
# SCSI generic driver
sg
# SCSI disk
sd_mod
# All other needed HW modules
...
```

可用 depmod 来操作上述某些模块。

7.2.3 网络功能

/etc/modules 中需要包含如下内容以扩充网络功能：

```
# net/ipv-4
ip_gre
ipip

# net/ipv-4/netfilter
# iptable (in order)
ip_tables
ip_conntrack
ip_conntrack_ftp
iptable_nat
iptable_filter
iptable_mangle
#
ip_nat_ftp
ip_queue
#
ipt_LOG
ipt_MARK
ipt_MASQUERADE
ipt_MIRROR
ipt_REDIRECT
ipt_REJECT
ipt_TCPMSS
ipt_TOS
ipt_limit
ipt_mac
ipt_mark
ipt_multiport
ipt_owner
ipt_state
ipt_tcpmss
ipt_tos
ipt_unclean
#
#ipchains
#ipfwadm
```

上述内容并没有进行优化。可用 depmod 来操作上述某些模块。

7.2.4 EXT3 文件系统（> 2.4.17）

对预编译内核镜像包（> 2.4.17）执行下述操作可激活 EXT3 日志文件系统。

```
# cd /etc; mv fstab fstab.old
# sed 's/ext2/ext3,ext2/g' <fstab.old >fstab
# vi /etc/fstab
```

```
... 将 root 文件系统类型设置成“auto”而非“ext3,ext2”
# cd /etc/mkinitrd
# echo jbd >>modules
# echo ext3 >>modules
# echo ext2 >>modules
# cd /
# apt-get update; apt-get install kernel-image-2.4.17-686-smp
... 安装最新内核并配置 boot（lilo 从这儿运行）
# tune2fs -j -i 0 /dev/hda1
# tune2fs -j -i 0 /dev/hda2
... 将所有 EXT2 FS 转化成 EXT3
# shutdown -r now
```

现在就可使用 EXT3 日志文件系统了。 在 fstab 的“type”中使用 ex3、ext2 的是为了保险起见，如果内核不支持非 root 分区采用 EXT3 还可退回到 EXT2。

如果你已安装了 2.4 版内核并且不想再次重装,执行上述步骤中 apt-get 命令之前的步骤就行了。接着：

```
# mkinitrd -o /boot/initrd.img-2.4.17-686-smp /lib/modules/2.4.17-686-smp
# lilo
# tune2fs -j -i 0 /dev/hda1
# tune2fs -j -i 0 /dev/hda2
... 将所有 EXT2 FS 转化成 EXT3
# shutdown -r now
```

现在 EXT3 日志文件系统已生效。

如果没有设置 /etc/mkinitrd/modules 就 mkinitrd 运行，最好在系统启动时加载一些模块：

```
... 当 initrd 提示获取 shell 时（5 秒钟），输入 RETURN
# insmod jbd
# insmod ext3 # modprobe ext3 会负责一切
# insmod ext2
# ^D
... 继续启动
```

激活 EXT3 功能会造成某些系统发生严重内核死锁的情况，不过我没遇到过这种问题（我的内核是 2.4.17）。

7.2.5 2.4 版内核对 Realtek RTL-8139 的支持

不知何故，RTL-8139 支持模块已不再叫 rtl8139，现在它叫 8139too。从 2.2 版内核升级到 2.4 版时，请记得在 /etc/modules 中做相应修改。

7.2.6 并行端口支持

对于 kernel-image-2.4.*，并行端口支持已被模块化，要激活可执行：

```
# modprobe lp
# echo lp >> /etc/modules
```

参阅 Linux 内核源码目录中的 Documentation/parport.txt。

7.3 通过 proc 文件系统调整内核

Linux 内核行为可以在运行状态下通过 proc 文件系统进行调节。

有关在 /proc 文件系统下修改内核参数的基础知识，可参阅 Linux 源码包中的 Documentation/sysctl/* 文件。

调整内核参数的例子，可参考 /etc/init.d/networking 和无法访问某些站点的怪问题，第 3.8.5 节。

参阅 sysctl.conf(5) 了解如何使用 /proc 文件系统和脚本 /etc/init.d/procps.sh 来设置内核开机时的配置。此脚本由 /etc/rcS.d/S30procps.sh 执行。

7.3.1 打开了太多文件

Linux 内核有时会报告“Too many open files"，起因是 file-max 默认值（8096）太小。要解决这个问题，可以 root 身份执行下列命令：(或将它们加入/etc/rcS.d/*下的 init 脚本。)

```
# echo "65536"  > /proc/sys/fs/file-max  # 适用于 2.2 和 2.4 版内核
# echo "131072" > /proc/sys/fs/inode-max # 仅适用于 2.2 版内核
```

或将下列内容放入 /etc/sysctl.conf，做永久性的更改：

```
file-max=65536   # 适用于 2.2 和 2.4 版内核
inode-max=131072 # 仅适用于 2.2 版内核
```

7.3.2 磁盘缓存清除时间（Disk flush intervals）

可通过 proc 文件系统来修改磁盘缓存清除时间。下面的操作将默认的 5 秒时间间隔缩短到 1 秒。

```
# echo "40 0 0 0 100 30000 60 0 0"  > /proc/sys/vm/bdflush
```

这可能对文件 I/O 性能产生一点儿负面影响。但它能保证文件内容是最近 1 秒的，比默认的 5 秒更短。对日志文件系统来说更是如此。

7.3.3 迟缓的小内存旧机器

对某些小内存的旧机器来说，在 proc 文件系统中打开内存的 over-commit 功能会很有效

果：

```
# echo 1 > /proc/sys/vm/overcommit_memory
```

7.4 2.6 版内核和 udev

udev 是取代 /dev/ 的动态设置的系统。我们可以选择很短的设备名字。而 2.4 版内核使用的 devfs 已经被淘汰。

安装 Ubuntu 新版的 linux-image-2.6.NN 和 udev 就能启用这个功能。

第 8 章 - Ubuntu 小技巧

8.1 启动系统

关于系统启动提示的详细信息请参见 LDP BootPrompt-HOWTO。

8.1.1 “我忘记了 root 密码！”(一)

只要能访问控制台键盘，不需要 root 密码也可以启动系统并以 root 帐号登录。（这里假设没有 BIOS 密码或 lilo 之类的启动引导器密码用于控制系统启动。）

下面是一个不需要额外的启动盘或对 BIOS 启动设置进行修改的过程。这里的“Linux”是代表在 Ubuntu 默认安装系统中启动 Linux 内核的标签。

在 lilo 的启动屏幕中，当 boot: 一出现时（在某些系统中，您必须按 shift 键以阻止自动启动；如果 lilo 使用 framebuffer，您需要按 TAB 键才能看到自己输入的选项），就输入：

```
boot: Linux init=/bin/sh
```

这会让系统启动内核并运行 /bin/sh 而非其标准的 init。现在你已获得 root 权限和一个 root shell。由于当前 / 是以只读方式挂载，而其它的硬盘分区均未挂载，故你必须完成下列步骤才能获得一个有适当功能的系统。

```
init-2.03# mount -n -o remount,rw /
init-2.03# mount -avt nonfs,noproc,nosmbfs
init-2.03# cd /etc
init-2.03# vi passwd
init-2.03# vi shadow
```

（如果 /etc/passwd 文件中所有用户的第二个域的数据都为“x”，就表明系统使用了影子（shadow）密码，必须编辑 /etc/shadow。）要删除 root 密码，请编辑密码文件中第二个数据域，将它设置为空白。这样重启系统不用密码就能登录到 root。当系统启动进入 runlevel 1 时，Ubuntu 需要密码。

在 /bin 下装一个小编辑器是个好习惯，因为有时 /usr 是无法访问的（参阅 应急的编辑器, 第 11.2 节）。

另外可以安装 sash 软件包，当系统无法启动时，还可执行：

```
boot: Linux init=/bin/sash
```

当 /bin/sh 不可用时，sash 可作为 sh 的交互式替代品，它是静态链接，内建了许多标准工具（在系统提示符下输入“help”可获得参考列表）。

8.1.2“我忘记了 root 密码！”(二)

从急救盘启动系统。假设 /dev/hda3 是原始 root 分区，可用下面的方法编辑密码文件，与上述方法一样容易。

```
# mkdir fixit
# mount /dev/hda3 fixit
# cd fixit/etc
# vi shadow
# vi passwd
```

与上面的方法相比，该方法的好处在于不需要知道 lilo 密码（如果有的话）。但如果系统没有预先设置为从软盘或 CD 启动，就需要访问 BIOS 的权限。

8.1.3 无法启动系统

没在安装过程中制作启动盘？没关系。如果 lilo 损坏了，从 Debian 安装套件中拿出启动盘，用它来启动系统。假设你的 root 分区在 /dev/hda12，你想进入 runlevel 3，在启动提示符后输入：

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

接下来，系统使用软盘上的内核启动，你可登录到一个几乎拥有全部功能的系统了。（可能有少量特性或模块不可用。）

如果系统崩溃，亦可参阅 为无法启动的系统安装软件包, 第 6.3.6 节。

如果想做张自定义启动盘，参阅急救盘中的 readme.txt 文档。

8.1.4“我不想直接启动到 X！”

玩 edgy/dapper 很有趣，但在启动进程中执行不稳定的 xdm、gdm、kdm 和 wdm 会让你焦头烂额。

首先，在启动提示符后输入如下指令获得 root shell：

```
boot: Linux vga=normal s
```

这里的 Linux 是你要启动的内核的标记，“vga=normal”告诉 lilo 在普通 VGA 屏幕下运行，“s"（或“S"）是传给 init 的参数，告诉它进入单用户模式。在提示符后输入 root 密码。

有多种方法禁用 X 启动 daemons：

- 运行 update-rc.d -f ?dm remove ; update-rc.d ?dm stop 99 1 2 3 4 5 6 .
- 在所有 /etc/init.d/?dm 文件的最前面加上“exit 0"。
- 把所有的 /etc/rc2.d/S99?dm 文件改名为 /etc/rc2.d/K99?dm。
- 删除所有的 /etc/rc2.d/S99?dm 文件。
- 运行 :>/etc/X11/default-display-manager

其中，rc2.d 中的数字必须与 /etc/inittab 中指定的 runlevel 一致。而 ?dm 的意思是你要将同一个命令运行多次，每次将其替换成 xdm、gdm、kdm 和 wdm 中的一个。

在 Ubuntu 下只有第一种方法是“唯一正确的方法"。最后一种方法比较简单但只适用于 Ubuntu，而且还需要使用 dpkg-reconfigure 重新设置一次。其它方法都是通用的中 daemons 的方法。

你仍可在任何控制台 shell 中用 startx 命令启动 X。

8.1.5 其它用于启动提示符的技巧

使用 lilo 启动提示符，可指定系统启动到特定的 runlevel 和配置。详情参阅 BootPrompt-HOWTO (LDP)。

如果希望系统启动到 runlevel 4，可以 lilo 启动提示符后输入：

```
boot: Linux 4
```

如果希望系统启动到正常功能的单用户模式，而且你知道 root 密码，可在 lilo 启动提示符后输入下列任一参数。

```
boot: Linux S
boot: Linux 1
boot: Linux -s
```

如果希望系统以少于实际内存数的内存启动（也就是说机器有 64MB 内存，只分配 48MB 给系统使用），在 lilo 启动提示符后输入：

```
boot: Linux mem=48M
```

注意，不要指定大于实际内存数的内存，否则内核会崩溃。如果你有多于 64MB 的内存，如 128MB，应在系统启动时执行 mem=128M 或在/etc/lilo.conf 中添加类似的命令行，否则旧内核或使用旧 BIOS 的主板将无法使用大于 64MB 的内存。

8.1.6 设置 GRUB 启动参数

GRUB 是 Hurd 项目开发的新型启动管理器，比 Lilo 更灵活，不过启动参数也与之稍有不同。

```
grub> find /vmlinuz
grub> root (hd0,0)
grub> kernel /vmlinuz root=/dev/hda1
grub> initrd /initrd
grub> boot
```

请注意 Hurd 中的设备名：

HURD/GRUB	Linux	MSDOS/Windows
(fd0)	/dev/fd0	A:
(hd0,0)	/dev/hda1	C: (usually)
(hd0,3)	/dev/hda4	F: (usually)
(hd1,3)	/dev/hdb4	?

详情参阅/usr/share/doc/grub/README.Debian 和/usr/share/doc/grub-doc/html/。

8.2 活动记录

8.2.1 记录 shell 活动

比起普通的个人电脑环境，Unix 环境的系统管理包含了更多细致的任务。必须掌握所有基本的配置方法以便进行系统故障恢复。基于 X11 的 GUI 配置工具看上去又好又方便，但不适用于紧急状况。

记录 shell 活动是个好习惯，特别是 root 用户。

Emacs：使用 M-x shell 在缓冲区中开始记录，使用 C-x C-w 将缓冲区中的记录写入文件。

Shell：使用 screen 命令和 用 screen 来定制控制台，第 8.6.28 节 中描述的“^A H”；或者使用 script 命令。

```
$ script
Script started, file is typescript
... do whatever ...
Ctrl-D
$ col -bx <typescript >savefile
$ vi savefile
```

还可使用下面的方法：

```
$ bash -i 2>&1 | tee typescript
```

8.2.2 记录 X 活动

如果需要 X 应用程序的活动记录图，包括 xterm 屏显，可使用 gimp（GUI）。它可以对每个

窗口或整个屏幕进行拍照。还可以使用 xwd（xbase-clients)、import（imagemagick） 和 scrot（scrot)。

8.3 拷贝及创建子目录

这些拷贝和归档命令提供系统和数据备份的基本功能。在 the example scripts 中提供了一个名为 backup 的简单备份脚本例子。

8.3.1 拷贝整个子目录的基本命令

如果想重新整理文件组织结构，可使用下面的方法移动文件及文件链接：

标准方法：
```
# cp -a /source/directory /dest/directory # 要求 GNU cp
# (cd /source/directory && tar cf - . ) | \
(cd /dest/directory && tar xvfp - )
```
如果包含硬链接，则需要更严谨的方法：
```
# cd /path/to/old/directory
# find . -depth -print0 | afio -p -xv -0a /mount/point/of/new/directory
```
如果是远程操作：
```
# (cd /source/directory && tar cf - . ) | \
ssh user@host.dom (cd /dest/directory && tar xvfp - )
```
如果没有链接文件：
```
# scp -pr user1@host1.dom:/source/directory \
user2@host2.dom:/dest/directory
```

其中，scp <==> rcp，ssh <==> rsh。

下面的有关拷贝整个子目录的信息由 Manoj Srivastava srivasta@debian.org 发表于 debian-user@lists.debian.org。

8.3.2 cp

传统上，cp 并不能真正完成这个任务，因为它既没对符号链接进行区别对待，又不能保存硬链接。另一件需要注意的事就是稀疏文件（有洞的文件)。

GNU cp 克服了这缺陷，然而对于非 GNU 系统，cp 仍存在问题。而且使用 cp 无法生成小巧轻便的文档包。

```
% cp -a . newdir
```

8.3.3 tar

Tar 克服了 cp 在处理符号链接时出现的问题，然而，cpio 可以处理特殊文件，传统的 tar 却不行。

对于某个有多重硬链接的文件，tar 的处理方法是只将其中一个链接拷贝到磁带上，所以日后你只能找回拷贝中所保留那个的链接所指的文件；cpio 会为每个链接做一个拷贝，日后你可以找回任意一个链接所指的文件。

8.3.4 pax

全新的，符合 POSIX（IEEE Std 1003.2-1992, pages 380－388 (section 4.48) and pages 936－940 (section E.4.48)）标准的，众望所归的，轻便的文档包交互工具。pax 可以读、写以及列出文档包的成员，并能拷贝文件目录层次。pax 的操作独立于特定的文 档包格式，支持各种各样不同的文档包格式。

pax 工具刚刚成形，还很新。

```
# apt-get install pax
$ pax -rw -p e . newdir
or
$ find . -depth  | pax -rw -p e  newdir
```

8.3.5 cpio

cpio 从 cpio 或 tar 文档包提取/放入文件。该文档包可以是硬盘上的另一个文件，也可以是磁带或管道。

```
$ find . -depth -print0 | cpio --null --sparse -pvd new-dir
```

8.3.6 afio

afio 更善于处理 cpio 格式的文档包。通常它比 cpio 要快，且提供了更多磁带选项，并且能更友好的处理有讹误的输入数据。它支持交互式处理多卷文档包。用 afio 制作压缩文档包比压缩 tar 或 cpio 文档包 更安全。在备份处理脚本中 afio 是更佳的“文档处理引擎”。

```
$ find . -depth -print0 | afio -px -0a new-dir
```

对所有的磁带备份我都使用 afio。

8.4 差异备份与数据同步

要进行差异备份和数据同步可使用下列几种方法：

- rcs：备份并进行历史记录，只支持文本。
- rdiff-backup：备份并进行历史记录。支持链接。
- pdumpfs：对文件系统进行备份和历史记录。支持链接。
- rsync：单路同步。
- unison：双路同步。
- cvs：多路同步服务器备份并进行历史记录，只支持文本，技术成熟。参阅并行版本系统 (CVS)，第 12.1 节。
- arch：多路同步服务器备份并进行历史记录，但包括“处于工作中的目录”。
- subversion：多路同步服务器备份并进行历史记录，专用于 Apache。

有关将这些方法与文档包操作结合应用的讨论参阅拷贝及创建子目录，第 8.3 节，有关自动进行备份的讨论参阅日程安排（cron，at），第 8.6.27 节。

我只讲解三个较容易使用的工具。

8.4.1 使用 rdiff 进行差异备份

rdiff-backup 提供了简单好用的方法对任何文件包括链接进行历史差异备份。例如要对~/目录下的所有文件备份到/mnt/backup：

```
$ rdiff-backup --include ~/tmp/keep --exclude ~/tmp  ~/ /mnt/backup
```

从该文档包中取出 3 天前的旧数据恢复到~/old 目录：

```
$ rdiff-backup -r 3D /mnt/backup ~/old
```

参阅 rdiff-backup(1)。

8.4.2 使用 pdumpfs 进行每日备份

pdumpfs 是一种简单的每日备份系统，与 Plan9 的 dumpfs 一样，它每天都保存系统快照。任何时候都可以用它来恢复到某天的系统状态。请使用 pdumpfs 和 cron 来备份你的 home 目录。

在目标目录中，pdumpfs 以 YYYY/MM/DD 的方式来组织系统快照。 当 pdumpfs 第一次运行时，它将所有源文件拷贝到快照目录。从每二次运行起，pdumpfs 仅拷贝更新的或新建的文件，对于没有改变的文件用硬链接方式指向前一天的系 统快照，以此来节省硬盘空间。

```
$ pdumpfs src-dir dest-dir [dest-basename]
```

参阅 pdumpfs(8)。

8.4.3 使用 RCS 进行定期差异备份

Changetrack 会定期对 RCS 文档包中基于文本格式的配置文件的变化进行记录。参阅 changetrack(1)。

```
# apt-get install changetrack
# vi changetrack.conf
```

8.5 系统冻结恢复

8.5.1 中止一个进程

运行 top 看看什么进程的活动有异常。按“P”以 CPU 使用率排序，“M”以内存使用率排序，“k”可以中止一个进程。还有一种方法，使用 BSD 风格的 ps aux | less 或 System V 风格的 ps -efH | less。System V 风格的排列会显示父进程 ID （PPID），这对中止出错的（死掉的）子进程十分有用。

知道了进程的 ID，就可使用 kill 中止（或发信号给）某个进程，killall 的作用正如其名

一样。经常使用的信号有：

1: HUP，重启 daemon
15: TERM，普通中止
9: KILL，强令中止

8.5.2 Alt-SysRq

内核编译选项“Magic SysRq key”提供系统强心针。在 i386 机器上按下 ALT-SysRq 组合键后，试试按下列各键 r 0 k e i s u b，奇迹产生了：

Un'r'aw 让键盘从 X 崩溃中重生。将控制台 loglevel 改为'0'以减少错误信息。sa'k'(system attention key) 中止当前虚拟控制台的所有进程。t'e'rminate 中止当前终端除 init 外的所有进程。k'i'll 中止除 init 外的所有进程。

'S'ync，'u'mount 和 re'b'oot 帮你逃离真正的险境。

本文写作之时，Ubuntu 默认安装的内核并未将该选项编译进去，需要重新编译内核激活该功能。详情参阅 /usr/share/doc/kernel-doc-version/Documentation/sysrq.txt.gz 或 /usr/src/kernel-version/Documentation/sysrq.txt.gz。

8.6 记住这些可爱的小命令

8.6.1 Pager

less 就是默认的 pager(文件内容浏览器)。按“h”可获得帮助。它比 more 更有用。在 shell 启动脚本中运行 eval $(lesspipe)或 eval $(lessfile)可以让 less 活力四射。详情参阅 /usr/share/doc/lessf/LESSOPEN。使用-R 选项可输出生癖字符 and enables ANSI color escape sequences.参阅 less(1)。

对于某些编码系统（EUC）w3m 可能是更好的选择。

8.6.2 释放内存

free 和 top 能让你了解内存资源的许多有用信息。别担心“Mem:”行中“used”的大小，看看它下面的数字（本例的数字是 38792)。

```
$ free -k # 用 256MB 内存的机器
total       used       free     shared    buffers cached
Mem:        257136     230456      26680      45736     116136 75528
-/+ buffers/cache:      38792     218344
Swap:       264996          0     264996
```

物理内存的准确大小可通过 grep '^Memory' /var/log/dmesg 得到，本例将显示“Memory: 256984k/262144k available (1652k kernel code, 412k reserved, 2944k data, 152k init)”。

```
Total         ==== 262144k ==== 256M (1k=1024, 1M=1024k)
Free to dmesg ==== 256984k ==== Total - kernel - reserved - data - init
```

Free to shell ==== 257136k ==== Total - kernel - reserved - data

约有 5MB 内存系统不能使用，因为内核需要它。

8.6.3 设定时间（BIOS）

```
# date MMDDhhmmCCYY
# hwclock --utc --systohc
# hwclock --show
```

设定系统时间和硬件时间为 MM/DD hh:mm, CCYY。显示时间为本地时间而硬件时间使用 UTC。

如果硬件（BIOS）时间设置为 GMT，在文件 /etc/default/rcS 中改变设置 UTC=yes。

8.6.4 设定时间（NTP）

参考：Managing Accurate Date and Time HOWTO。

8.6.4.1 拥有永久 Internet 连接的系统设置时间

设置系统时钟通过远程服务器自动对时：

```
# ntpdate server
```

如果你的系统拥有永久的 Internet 连接，应该将该命令加入/etc/cron.daily/。

8.6.4.2 偶尔进行 Internet 连接的系统设置时间

使用 chrony 软件包。

8.6.5 如何禁用屏幕保护程序

禁用屏幕保护程序,使用下面的命令。

对于 Linux 控制台：

```
# setterm -powersave off
```

启动 kon2 （kanji）控制台可执行：

```
# kon -SaveTime 0
```

运行 X 可执行：

```
# xset s off
或
# xset -dpms
```

或
```
# xscreensaver-command -prefs
```

控制其它的控制台特征，请参阅相关 man 页面。 改变和显示终端行设置，请参阅 stty(1)。

8.6.6 搜索系统管理数据库

Glibc 提供了 getent(1)搜索管理数据库的各类项目。例如 passwd、group、hosts、services、protocols、networks。

```
getent database [key ...]
```

8.6.7 禁用声音（响铃）

最直接的方法是拔掉 PC 喇叭。;-) 对于 Bash shell 可执行：

```
echo "set bell-style none">> ~/.inputrc
```

8.6.8 控制台上的错误信息

不想看屏幕显示的错误信息，首选的方法是检查/etc/init.d/klogd，在该脚本中设置 KLOGD="-c 3"然后运行/etc/init.d/klogd restart。另一种方法是执行 dmesg -n3。

这儿是各种错误级别的含义：

- 0: KERN_EMERG, 系统不可用
- 1: KERN_ALERT, 必须立即执行
- 2: KERN_CRIT, 紧急状态
- 3: KERN_ERR, 错误状态
- 4: KERN_WARNING, 警告状态
- 5: KERN_NOTICE, 正常状态且十分重要
- 6: KERN_INFO, 报告
- 7: KERN_DEBUG, debug-level 信息

如果你很厌恶详细而无用的错误信息，可以试试这个小补丁 shutup-abit-bp6（位于样例脚本子目录）。

另一个该看看的地方是/etc/syslog.conf;，检查一下是否有信息记录被发送到了控制台设备。

8.6.9 正确设置控制台类型

在类 Unix 系统中，访问控制台屏幕通常要调用库例程，这就为用户提供了一种独立于终端的方式来优化字符的屏幕更新过程。参阅 ncurses(3X)和 terminfo(5)。

在 Ubuntu 系统中，有大量预定义项目：

```
$ toe | less                    # 所有项目
```

```
$ toe /etc/terminfo/ | less    # 用户可再配置的项目
```

你的选择可导出到环境变量 TERM。

当登录到远程 Debian 系统时，如果 xterm 的 terminfo 项目在非 Debian 的 xterm 中失效，请将终端类型改为支持较少特性的版本如“xterm-r6”。 参阅 /usr/share/doc/libncurses5/FAQ。“dumb”是 terminfo 的最小公分母。

8.6.10 恢复控制台的健壮性

如果执行 cat some-binary-file 后屏幕一片混乱（命令的返回值与你的输入大相径庭）：

```
$ reset
```

8.6.11 将 DOS 下的文本文件转换为 Unix 类型

将 DOS 文本文件（行尾=M^J）转换成 Unix 文本文件（行尾=J）。

```
# apt-get install sysutils
$ dos2unix dosfile
```

8.6.12 使用 recode 转化文本文件

下面将在 DOS、 Mac 和 Unix 的文本文件中转换行结尾格式：

```
$ recode /cl../cr <dos.txt >mac.txt
$ recode /cr.. <mac.txt >unix.txt
$ recode ../cl <unix.txt >dos.txt
```

自由的 recode 在各种各样的字符集和界面中转换：

```
$ recode charset1/surface1..charset2/surface2 \
<input.txt >output.txt
```

使用的通用字符集设置是（参阅 Locales 简介, 第 9.7.3 节） ：

- us — ASCII (7 bits)
- l1 — ISO Latin-1 (ISO-8859-1, Western Europe, 8 bits)
- EUCJP — EUC-JP for Japanese (Unix)
- SJIS — Shift-JIS for Japanese (Microsoft)
- ISO2022JP — Mail encoding for Japanese (7 bits)
- u2 — UCS-2 (Universal Character Set, 2 bytes)
- u8 — UTF-8 (Universal Transformation Format, 8 bits)

使用的通用界面 [38] ：

- /cr — Carriage return as end of line (Mac text)
- /cl — Carriage return line feed as end of line (DOS text)
- / — Line feed as end of line (Unix text)

- /d1 — Human readable bytewise decimal dump
- /x1 — Human readable bytewise hexidecimal dump
- /64 — Base64 编码文本
- /QP — Quoted-Printable 编码文本

更多信息，请参阅 info recode 中的相关描述。

有更多的专业转换工具：

- 字符集转换：
    - iconv — 本地编码转换
    - konwert — 特殊编码转换
- 二进制文件转换：
    - uuencode and uudecode — 用于 Unix
    - mimencode — 用于邮件

8.6.13 正规表达式的置换

将所有文件 FILES ...中的所有 FROM_REGEX 字段替换成 TO_TEXT 字段。

```
$ perl -i -p -e 's/FROM_REGEX/TO_TEXT/g;' FILES ...
```

-i 表示“就地编辑”，-p 表示“在 FILES...各文件中循环”。如果置换很复杂，应使用参数 -i.bak 而非-i，这有助于出错恢复；它会将每个原始文件保存为以.bak 为后缀的备份文件。

8.6.14 使用脚本来编辑文件

下面的脚本将删除 5 - 10 行以及 16 - 20 行。

```
#!/bin/bash
ed $1 <<EOF
16,20d
5,10d
w
q
EOF
```

在此，ed 命令与 vi 命令模式下的是一样的，从外部编辑文件的方式使它更适于脚本化。

8.6.15 提取源文件修改部分合并到更新包

下面的操作将根据文件位置，提取源文件的修改部分并创建统一的 diff 文件 file.patch0 或 file.patch1：

```
$ diff -u file.old file.new1 > file.patch0
$ diff -u old/file new1/file > file.patch1
```

diff 文件（也称补丁文件）通常用于发送程序更新。收到的补丁文件可使用下面的方法更新另一个文件：

```
$ patch -p0 file < file.patch0
$ patch -p1 file < file.patch1
```

如果有 3 个版本的源代码，使用 diff3 来合并效率更高：

```
$ diff3 -m file.mine file.old file.yours > file
```

8.6.16 分割大文件

```
$ split -b 650m file    # 将大文件分块成多个 650MB 的小文件
$ cat x* >largefile     # 将所有小文件合并成一个大文件
```

8.6.17 从文本格式的表格中抽取数据

假设有一个文本文件名为 DPL，其中存放着所有前 Debian 项目领导人的名字和他们的上台日期，表格格式是以空格做为分隔的。

```
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
```

Awk 经常用于从这类文件中提取数据。

```
$ 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
```

象 Bash 这种 Shell 也能够用来分析这种文件：

```
$ while read first last month year; do
echo $month
```

```
done <DPL
... 跟第一个 Awk 例子有相同的输出
```
在这里，read 内建命令使用字符 $IFS （internal field separators 内部列分隔符）来将行分开为单词。

如果你改变 IFS 为 ":"，你能够用 shell 漂亮的分析 /etc/passwd：

```
$ oldIFS="$IFS"   # 保存旧值
$ IFS=":"
$ while read user password uid gid rest_of_line; do
if [ "$user" === "osamu" ]; then
echo "$user's ID is $uid"
fi
done < /etc/passwd
osamu's ID is 1001
$ IFS="$oldIFS"   # 恢复旧值
```

（如果使用 Awk 作同样的事情，使用 FS=":" 来设置列分隔符。）

shell 也使用 IFS 来分开参数扩展、命令替换和算术扩展的结果集。 但在被单引号或双引号引用的单词内,不会发生这种情况。默认的 IFS 值是：<space>、<tab> 和 <newline> 。

请小心使用这个 shell IFS 技巧。 当 shell 解释部分脚本作为它的**输入**时，奇怪的事情将会发生。

```
$ IFS=":,"                        # 使用 ":" 和 "," 作为 IFS
$ echo IFS=$IFS,   IFS="$IFS"     # echo 是 Bash 内建的
IFS==  , IFS=:,
$ date -R                         # 只是一个命令输出
Sat, 23 Aug 2003 08:30:15 +0200
$ echo $(date -R)                 # 子 shell --> 输入到主 shell
Sat  23 Aug 2003 08 30 36 +0200
$ unset IFS                       # 重设 IFS 为默认的
$ echo $(date -R)
Sat, 23 Aug 2003 08:30:50 +0200
```

8.6.18 精巧的管道命令辅助脚本

下列脚本做为管道的一部分十分有用。

```
find /usr | egrep -v "/usr/var|/usr/tmp|/usr/local"
# 查找 /usr 下的所有文件，排除某些文件
xargs -n 1 command   # 将所有项作为标准输入来执行命令
xargs -n 1 echo |    # 将空格隔离的项分开为行
xargs echo       |   # 合并所有的行到一行里面
grep -e pattern|     # 提取含有 pattern 的行
cut -d: -f3 -|
# 提取用 : 分开的第 3 列（比如说 passwd 文件）
awk '{ print $3 }' | # 提取用空格分开的第 3 列
```

```
awk -F'\t' '{ print $3 }' |
# 提取用 tab 分开的第 3 列
col -bx |            # 删除退格键，扩展 tab 为空格
expand -|            # 扩展 tab
sort -u|             # 排序并删除重复行

tr '\n' ' '|         # 将多行连接为一行
tr '\r' ''|          # 删除 CR
tr 'A-Z' 'a-z'|      # 转化大写字母为小写
sed 's/^/# /'|       # 将每行变为注释
sed 's/\.ext//g'|    # 删除 .ext
sed  -n -e 2p|       # 显示第 2 行
head -n 2 -|         # 显示头两行
tail -n 2 -|         # 显示最后两行
```

8.6.19 循环每一个文件的脚本片段

下面的方法将循环处理匹配 *.ext 的每一个文件，确保适当处理文件名包含空格的奇怪文件，并且执行相同的操作。

- Shell 循环 （在 PS2=" " 时使用多行格式的例子。 如果在一行内输入这些命令，需要在每一个换行处加一个分号。）：

```
for x in *.ext; do
if test -f "$x"; then
command "$x"
fi
done
```

- find 和 xargs 结合：

```
find . -type f -maxdepth 1 -name '*.ext' -print0 | \
xargs -0 -n 1 command
```

- 加 -exec 选项的 find 和一个命令结合：

```
find . -type f -maxdepth 1 -name '*.ext' \
-exec command '{}' \;
```

- 加 -exec 选项的 find 和一个短的 shell 脚本结合：

```
find . -type f -maxdepth 1 -name '*.ext' \
-exec sh -c "command '{}' && echo 'successful'" \;
```

8.6.20 短小的 Perl 脚本

虽然任何 Awk 脚本均可使用 a2p(1) 自动重写成 Perl， 但最好用手工的方法将一行 Awk

脚本转化为一行 perl 脚本。例如：

```
awk '($2=="1957") { print $3 }' |
```

与下列任意一行相当：

```
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"' |
```

其实上面各行中所有 perl 参数中的空格均可去掉，这得益于 Perl 的数字字符串自动转换功能。

```
perl -lane 'print$F[2]if$F[1]eq+1957' |
```

有关命令行参数的信息可参阅 perlrun(1)，在 http://perlgolf.sourceforge.net 有更多令人着魔的 Perl 脚本，你会感兴趣的。

8.6.21 从网页上获取文本或邮件列表文档

下面的操作将网页转化为文本文件。从网上拷贝配置文件时十分有用。

```
$ lynx -dump http://www.remote-site.com/help-info.html >textfile
```

links 和 w3m 也可以这么用，只是生成的文本样式可能略有不同。

如果是邮件列表文档，可使用 munpack 从文本获得 mime 内容。

8.6.22 打印网页

下面的操作将网页内容打印成 PostScript 文件或发送到打印机。

```
$ apt-get install html2ps
$ html2ps URL | lpr
```

参阅 lpr/lpd, 第 3.6.1 节。还可使用 a2ps 和 mpage 软件包生成 PostScript 文件。

8.6.23 打印帮助页面

下面的操作将帮助页面打印成 PostScript 文件或发送到打印机。

```
$ man -Tps some-manpage | lpr
$ man -Tps some-manpage | mpage -2 | lpr
```

8.6.24 合并两个 PostScript 或 PDF 文件

可以将两个 PostScript 文件或 PDF 文件合并。

```
$ 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
```

8.6.25 命令耗时

显示某进程的耗时

```
# time some-command >/dev/null
real    0m0.035s        # 挂钟时间（过去的真实时间）
user    0m0.000s        # 用户模式时间
sys     0m0.020s        # 核心模式时间
```

8.6.26 nice 命令

使用 nice（来自 GNU shellutils 软件包）可设置命令启动时的 nice 值。renice（bsdutils）和 top 可以重设进程的 nice 值。nice 值为 19 代表最慢的（优先级最低的）进程；负值就“not-nice"，如-20 代表非常快的（优先级高的）进 程。只有超级用户可以设定负 nice 值。

```
# nice  -19 top                                          # very nice
# nice --20 cdrecord -v -eject speed=2 dev=0,0 disk.img # very fast
```

有时极端的 nice 值对系统弊大于利，所以使用该命令要小心。

8.6.27 日程安排（cron，at）

在 Ubuntu 下使用 cron 和 at 进行任务日程安排。参阅 at(1)、crontab(5)、crontab(8)。

执行命令 crontab -e 创建或编辑 crontab 文件，为规律事务（按周期循环的事务）安排日程。下面的一个 crontab 文件样例：

```
# 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
```

执行 at 命令为偶然任务（只执行一次的任务）安排日程：

```
$ echo 'command -args'| at 3:40 monday
```

8.6.28 用 screen 来定制控制台

screen 程序允许在单一的物理终端或终端模拟窗口运行多个伪终端，每个伪终端都拥有自己的交互式 shell。即便可以使用 Linux 伪终端或多个 xterm 窗口，研究一下如何设置 screen 丰富的特性仍很有益，这些特性包括：

- 回溯历史显示，
- 拷贝和粘贴，
- 输出到日志，
- 图形入口，
- 将终端与整个 screen 会话**分离**，稍后再连接。

8.6.28.1 远程访问方案

如果你经常从远程终端登录到 Linux 机器或使用 VT100 终端程序，screen 的 **detach**（分离）特性将简化你的生活。

- 通过拨号连接登录，运行了一个非常复杂的 screen 会谈，打开了好几个窗口，有编辑器和其它一些程序。
- 突然你需要离开终端一下，但你并不想挂断连接中止工作。
- 输入 A d**离开**会话，然后登出系统。（或者更简单些，输入A DD 离开会话并自动登出系统）
- 当你回来时，需要再次登录，可输入命令 screen -r，screen 会如魔法般地重新连接上所有打开的窗口。

8.6.28.2 典型的 screen 命令

一但打开了 screen 程序，除了命令按键（默认为 A）所有的键盘输入都被送到当前窗口，所有的 screen 命令均按特定方式输入：A 加一个单键命令[加一些参数]的。常用的命令有：

```
^A ?      显示帮助屏幕（显示命令集）
^A c      创建并切换到新窗口
^A n      跳到下一个窗口
^A p      跳到上一个窗口
^A 0      跳到 0 号窗口
^A w      显示窗口列表
^A a      将 Ctrl-A 做为键盘输入发送到当前窗口
^A h      对当前窗口做硬拷贝写入到文件
^A H      开始/中止将当前窗口事件记录到文件
<sup>A </sup>X     锁定终端（密码保护）
^A d      从终端分离屏幕会话
^A DD     分离屏幕会话并退出登录
```

以上只是 screen 命令的一个很小的子集。只要是你认为 screen 能干的事，没准它真就可以! 详情参阅 screen(1)。

8.6.28.3 screen 会话中的退格键和 Ctrl-H

在运行 screen 时，如果发现退格键和/或 Ctrl-H 无法正常工作，可编辑/etc/screenrc，找到这行：

```
bindkey -k kb stuff "\177"
```

将这注释掉（例如在句首添加“#”）。

8.6.28.4 X 下与 screen 等价的程序

找找 xmove。参阅 xmove(1)。

8.6.29 网络测试基础

安装 netkit-ping、traceroute、dnsutils、ipchains（适用于 2.2 版内核）、iptables（适用于 2.4 版内核）和 net-tools 软件包，然后执行：

```
$ ping yahoo.com            # 检查 Internet 连接
$ traceroute yahoo.com      # 跟踪 IP 数据包
$ ifconfig                  # 检查主机设置
$ route -n                  # 检查路由设置
$ dig [@dns-server.com] host.dom [{a|mx|any}] |less
# 检查 dns-server.com 的 host.dom DNS 记录
# 查找{a|mx|any}记录
$ ipchains  -L -n |less     # 检查包过滤(2.2 kernel)
$ iptables -L -n |less      # 检查包过滤(2.4 kernel)
$ netstat -a                # 查找系统上所有打开的端口
$ netstat -l --inet         # 查找系统监听的端口
$ netstat -ln --tcp         # 查找系统监听的 TCP 端口（端口数字）
```

8.6.30 从本地缓冲池中清空（flush）邮件

从本地缓冲池中清空邮件：

```
# exim -q    # 清空待读邮件
# exim -qf   # 清空所有邮件
# exim -qff  # 清空冻结邮件
```

-qff 选项用在/etc/ppp/ip-up.d/exim 脚本中效果更好。 在 Sarge 中，使用 exim4 代替 exim。

8.6.31 删除本地缓冲池中的冻结邮件

删除本地缓冲池中的冻结邮件并返回出错信息：

```
# exim -Mg `mailq | grep frozen | awk '{ print $3 }'`
```

在 Sarge 中，使用 exim4 代替 exim。

8.6.32 再分发 mbox 中的信件

如果 home 目录没有空间继续处理邮件，procmail 将失败， 就需要对磁盘空间进行扩容，扩容完成后需要手工分发/var/mail/username 目录中的邮件到 home 目录中的分类邮箱，执行：

```
# /etc/init.d/exim stop
# formail -s procmail </var/mail/username
# /etc/init.d/exim start
```

在 Sarge 中，使用 exim4 代替 exim。

8.6.33 清空文件内容

要清空某些文件如日志文件的内容，千万不要使用 rm 删除文件然后再创建一个新的空文件，因为在两次操作的间隔，系统可能需要访问该文件。下面是清空文件内容的安全方法：

```
$ :>file-to-be-cleared
```

8.6.34 空文件

下面的命令可以创建空文件：

```
$ dd if=/dev/zero    of=filename bs=1k count=5 # 5KB 零内容
$ dd if=/dev/urandom of=filename bs=1M count=7 # 7MB 随机内容
$ touch filename #  创建一个 0 B 大小的文件（如果文件存在，更新该文件的修改时间）
```

例如，最实用的用法是从 Debian 启动软盘的 shell 中执行下列命令将硬盘/dev/hda 的内容完全清空。

```
# dd if=/dev/urandom of=/dev/hda ; dd if=/dev/zero of=/dev/hda
```

8.6.35 chroot

chroot 程序，chroot(8)，不需要重启系统，就可以在单独的系统上同时运行多个不同的 GNU/Linux 环境。

还可以在较快主机的 chroot 下运行某些需耗大量系统资源内存的程序如 apt-get 或 dselect，并将较慢子机的硬盘通过 NFS 方式挂载到主机，开放读/写权限，在主机上以 chroot 方式操作子机。

8.6.35.1 用 chroot 来运行不同版本的 Ubuntu

在老版本中使用 debootstrap 命令很容易构造 chroot Debian 体系。对于 Hoary 的后续发行版，用 cdebootstrap 命令加上适当的选项来代替 debootstrap。例如，在一台拥有快速 Internet 连接的机器的 /sid-root 下创建一个 dapper chroot：

```
main # cd / ; mkdir /dapper-root
main # debootstrap dapper /dapper-root http://archive.ubuntu.com/ubuntu/
... 看它下载整个系统
main # echo "proc-dapper /dapper-root/proc proc none 0 0" >> /etc/fstab
main # mount proc-dapper /dapper-root/proc -t proc
main # cp /etc/hosts /dapper-root/etc/hosts
main # chroot /dapper-root /bin/bash
chroot # cd /dev; /sbin/MAKEDEV generic ; cd -
chroot # apt-setup # 创建 /etc/apt/sources.list 文件
chroot # vi /etc/apt/sources.list # 将源指向 dapper
chroot # dselect  # 可以使用 aptitude，安装 mc 和 vim :-)
```

现在你就拥有了一个全功能 Ubuntu 子系统，可以尽情享受而不必担心主 Ubuntu 受到不利影响。

该 debootstrap 应用技巧还可以实现在没有 Ubuntu 安装盘的情况下，从另一个 GNU/Linux 发行版下安装 Ubuntu。参阅 http://www.debian.org/releases/stable/i386/apcs04.html.en。

8.6.35.2 设置 chroot 登录

输入 chroot /sid-root /bin/bash 非常简单，但这将保留当前的所有环境变量，你可能并不希望这样并且有时还会出问题。更好的方法是，在别的虚拟终端上执行另一个登录进程，登录到 chroot 目录。

在默认的 Ubuntu 系统中，从 tty1 到 tty6 运行 Linux 控制台，tty7 运行 X Window 系统，在本例中，我们将 tty8 设置成 chroot 控制台。按照 用 chroot 来运行不同版本的 Ubuntu, 第 8.6.35.1 节中的描述创建好 chroot 系统后，就可以在主系统的 root shell 中输入：

```
main # echo "8:23:respawn:/usr/sbin/chroot /sid-root "\
"/sbin/getty 38400 tty8"  >> /etc/inittab
main # init q    # 重启 init
```

8.6.35.3 配置 chroot 下的 X

想在 chroot 下安全地运行最新版的 X 和 GNOME 吗？完全可以！下面的例子将实现在虚拟终端 vt9 下运行 GDM。

首先，按照 用 chroot 来运行不同版本的 Ubuntu, 第 8.6.35.1 节中描述的方法安装好 chroot 系统，从主系统的 root 下拷贝关键配置文件到 chroot 系统。

```
main # cp /etc/X11/xorg.conf /dapper-root/etc/X11/xorg.conf
main # chroot /dapper-root # or use chroot console
chroot # cd /dev; /sbin/MAKEDEV generic ; cd -
chroot # apt-get install gdm gnome x-window-system
chroot # vi /etc/gdm/gdm.conf # do s/vt7/vt9/ in [servers] section
chroot # /etc/init.d/gdm start
```

在此，编辑/etc/gdm/gdm.conf，使其在 vt7 到 vt9 上创建虚拟终端。

现在可以很容易地能过切换 Linux 虚拟终端来实现在主系统的 X 环境和 chroot 系统的 X 环境之间转换，例如使用 Ctrl-Alt-F7 和 Ctrl-Alt-F9。酷吧！

[FIXME] 在 chroot 系统下 gdm 的 init 脚本中添加一条注释和一条链接。

8.6.35.4 使用 chroot 来运行其它发行版

很容易创建一个包含其它发行版的 chroot 环境。使用其它发行版的安装程序将它们安装到单独的硬盘分区中。例如 root 分区位于/dev/hda9：

```
main # cd / ; mkdir /other-dist
main # mount -t ext3 /dev/hda9 /other-dist
main # chroot /other-dist /bin/bash
```

接下来按照 用 chroot 来运行不同版本的 Ubuntu, 第 8.6.35.1 节、 设置 chroot 登录, 第 8.6.35.2 节和 配置 chroot 下的 X, 第 8.6.35.3 节 处理。

8.6.35.5 使用 chroot 来编译软件包

这儿有一个很特殊的 chroot 软件包 pbuilder，它构造一个 chroot 系统并在其中编译软件包。该体系可用于检查软件包编译时关联关系是否正确，并确保编译生成的软件包中没有不必要的或错误的关联关系。

8.6.36 怎样检查硬链接

检查两个文件是否是指向同一个文件的两个硬链接：

```
$ ls -li file1 file2
```

8.6.37 mount 硬盘上的镜像文件

如果 file.img 文件是硬盘内容的镜像文件，而且原始硬盘的配置参数为 xxxx === (bytes/sector) * (sectors/cylinder)，那么，下面的命令将其挂载到/mnt：

```
# mount -o loop,offset=xxxx file.img /mnt
```

注意绝大部分的硬盘都是 512 bytes/sector。

8.6.38 Samba

获取 Windows 文件的基本方法：

```
# mount -t smbfs -o username=myname,uid=my_uid,gid=my_gid \
//server/share /mnt/smb  # 挂载 Windows 的文件到 Linux
# smbmount //server/share /mnt/smb \
-o "username=myname,uid=my_uid,gid=my_gid"
# smbclient -L 192.168.1.2 # 列出某个计算机的共享目录
```

可从 Linux 检查 Samba 网上邻居：

```
# smbclient -N -L ip_address_of_your_PC | less
# nmblookup -T "*"
```

8.6.39 外来文件系统的操作工具

Linux 内核支持多种外来文件系统，想访问它们只需将其挂载到合适的文件系统下就行了。对某些文件系统，还提供专门工具不需要挂载，只依靠用户空间的程序，不需要内核提供文件系统支持，就能完成访问。

- mtools: for MSDOS filesystem (MS-DOS, Windows)
- cpmtools: for CP/M filesystem
- hfsutils: for HFS filesystem (native Macintosh)
- hfsplus: for HFS+ filesystem (modern Macintosh)

对于创建和检查 MS-DOS FAT 文件系统 dosfstools 非常有用。

8.7 需要注意的典型错误

这里有一些危险行为的例子。如果你是使用特权账号 root 的话，负面影响将会更大。

8.7.1 rm -rf .*

在象"rm -rf .*" 的命令行参数中使用通配符文件名，有可能造成危险的结果， 因为 ".*" 扩展为 "." 和 "..". 比较幸运的是，在 Debian 发行版中，当前版本的 "rm" 命令会检查文件名参数的健全性，会拒绝删除 "." 和 "..". 但这种检查并不一定在其它地方存在。尝试下面的操作来参看通配符文件名是怎样工作的。

- "echo .": 列出当前目录自身
- "echo *": 列出当前目录下所有不以点开头的文件和目录
- "echo .[^.]*": 列出当前目录下所有以点开头的文件和目录
- "echo .*": 列出父目录下的所有东西和父目录自身

8.7.2 rm /etc/passwd

由于你的过失，丢失象 /etc/passwd 这样的重要文件，是一件非常痛苦的事。Ubuntu 系统周期性的将他们备份到 /var/backups/。 当你恢复这些文件的时候，你需要设置适当的权限。

```
# cp /var/backups/passwd /etc/passwd
# chmod 644 /etc/passwd
```

参阅 恢复软件包选择状态的数据，第 6.3.4 节。

第 9 章 - Ubuntu 系统微调

本章讲述了基本的基于命令行界面的系统配置方法。在学习本章前，你需要先阅读 Ubuntu

系统安装提示, 第 3 章.

如是你很关心安全方面的信息,你应该阅读 Securing Debian Manual,它在 harden-doc 软件包中。

9.1 系统初始化

Ubuntu 使用 System V 的 init 脚本系统。参阅 init 程序, 第 2.4.1 节的介绍。

9.1.1 自定义 init 脚本

最简单的控制 init 脚本的方法是改变 /etc/default 目录下,与 init 脚本同名的文件里的环境变量设置。 例如,/etc/default/hotplug 可以控制 /etc/init.d/hotplug 的行为。文件 /etc/default/rcS 可以用来定制 motd、sulogin 等为默认启动。

如果你不能通过设置这些变量来得到你所需要的行为,你可以去修改 init 脚本:它们都是配置文件。

9.1.2 自定义系统日志

可通过 /etc/syslog.conf 配置系统日志记录方式。如果想为日志文件上色可安装 colorize 软件包。参阅 syslogd(8) 和 syslog.conf(5)。

9.1.3 硬件存取优化

有一部分硬件优化的配置工作 Ubuntu 留给了系统管理员。

- hdparm
  - 硬盘存取优化。效果显著。
  - 危险。务必先阅读 hdparm(8)。
  - hdparm -tT /dev/hda 测试硬盘存取速度。
  - hdparm -q -c3 -d1 -u1 -m16 /dev/hda 加速新型 IDE 系统。(有一定风险。)
- setcd
  - 压缩磁盘存取优化。
  - setcd -x 2 减速至 2x speed。
  - 参阅 setcd(1)。
- setserial
  - 串行接口管理工具集。
- scsitools
  - SCSI 硬件管理工具集。
- memtest86

    - 内存硬件管理工具集。

- hwtools
    - 低级硬件管理工具集。
        - § irqtune：修改设备的 IRQ 优先级，使那些需要高优先级和快速服务的硬件（例如，串行接口、调制解调器）获得它所要的资源。对串口/调制解调器加速后获得原来 3 倍的吞吐量。
        - § scanport：扫描 I/O 空间的 0x100 至 0x3ff 地址段，查找已安装的 ISA 设备。
        - § inb：一个小巧的黑客工具，用来阅读 I/O 端口信息并将其值转换成十六进制和二进制。

- schedutils
    - Linux 日程安排工具包。
    - 包括 taskset、irqset、lsrt 和 rt。
    - 再加上 nice 和 renice（不包括在工具包内），就可对进程的日程安排进行全面的管理。

使用 noatime 选项挂载文件系统可有效提高文件的读取速度。参阅 fstab(5) 和 mount(8)。

通过 proc 文件系统，Linux 内核可直接调节某些硬件参数。参阅 通过 proc 文件系统调整内核, 第 7.3 节。

Ubuntu 中有许多专门的硬件配置工具包。其中有不少是针对笔记本电脑的。这儿有一些有趣的软件包：

- tpconfig - 一个配置触摸屏设备的程序
- apmd - 高级电源管理（APM）工具
- acpi - 显示 ACPI 设备信息
- acpid - ACPI 使用工具
- lphdisk - 识别 Phoenix NoteBIOS 下的隐藏分区。
- sleepd - 笔记本电脑处于非工作状态时进入休眠
- noflushd - 让空闲硬盘进入减速状态
- big-cursor - X 下的巨型鼠标指针
- acme - 激活笔记本电脑上的“多媒体按钮”
- tpctl - IBM ThinkPad 硬件配置工具
- mwavem - Mwave/ACP modem 支持
- toshset - 访问大部分 Toshiba 笔记本电脑的硬件接口
- toshutils - Toshiba 笔记本电脑工具集
- sjog - 激活 Sony Vaio 笔记本电脑上“Jog Dial”功能的程序
- spicctrl - Sony Vaio 控制器程序可增亮 LCD 背光

ACPI 是一种比 APM 更新的电源管理系统。

某些软件包需要专门的内核模块。它们已经包含在许多最新的内核源码中。如果遇此问题，则需要手动打上最新的内核补丁。

9.2 访问限制（Restricting access）

9.2.1 用 PAM 来控制登录

PAM(Pluggable Authentication Modules 可嵌入认证模块)允许你控制用户是如何登录的。

```
/etc/pam.d/*            # PAM 管理文件
/etc/pam.d/login        # PAM 登录管理文件
/etc/security/*         # PAM 模块参数
/etc/securetty          # 管理通过控制台进行的 root 登录(login)
/etc/login.defs         # 管理登录行为(login)
```

如果想在控制台终端不用密码直接登录系统，可按下面的方法修改 /etc/pam.d/login 文件的内容，风险自负。

```
#auth       required    pam_unix.so nullok
auth       required    pam_permit.so
```

该方法亦可用于 xdm、gdm，实现无密码 X 控制台。

相反，如果你希望强化密码政策，可安装 cracklib2 并按下面的方法修改 /etc/pam.d/passwd：

```
password required       pam_cracklib.so retry=3 minlen=6 difok=3
```

使用一次性登录密码激活帐户也很有用。要实现该功能，在 passwd 命令后加上 -e 参数，参阅 passwd(1)。

要设置系统最大进程数，可在 Bash shell 中设定 ulimit -u 1000 或设置 PAM 的 /etc/security/limits.conf 文件。其它参数如 core 等的设置方法与之类似。PATH 的初始值可在 /etc/login.defs 中先于 shell 启动脚本设置。

PAM 的文档位于 libpam-doc 软件包内。其中 *The Linux-PAM System Administrator's Guide* 一文涵盖了 PAM 配置、可用模块等内容，文档中还包括了 *The Linux-PAM Application Developers' Guide* 和 *The Linux-PAM Module Writers' Guide*。

9.2.2 “为什么 GNU su 命令不支持 wheel group”

这是 Richard M. Stallman 的一句名言，位于旧版 info su 页面末尾。别担心：在 Ubuntu 中，当前版本的 su 使用 PAM，因此你可以用 /etc/pam.d/su 下的 pam_wheel.so 来限制任何用户组使用 su 的能力。下面的操作将在 Ubuntu 系统中赋予 adm 用户等同于 BSD wheel 用户组的权限，而且该组成员不需要密码就能使用 su 命令。

```
# anti-RMS configuration in /etc/pam.d/su
auth       required    pam_wheel.so group=adm

# Wheel members to be able to su without a password
auth       sufficient pam_wheel.so trust group=adm
```

9.2.3 各标准用户组的目的

一些有趣的用户组：

- 如果 pam_wheel.so 不带任何 group== 参数，root group 就是 su 默认的 wheel group。
- adm group 可以阅读日志文件。
- cdrom group 可在本地赋予一组用户访问 CD-ROM 驱动器的权限。
- floppy group 可在本地赋予一组用户访问软盘驱动器的权限。
- audio group 可在本地赋予一组用户访问声音设备的权限。
- src group 拥有源代码以及 /usr/src 目录下的文件。它可以在本地赋予某个用户管理系统源代码的权限。
- 对于管理桌面或低级别的系统管理员，可设置他们为 staff 成员，该类成员可以在 /usr/local 下工作并且可以在 /home 下创建目录。

完整列表参阅 Securing Debian Manual 的“FAQ”章节，亦见于 Woody 中的 harden-doc 软件包。新的 base-passwd (>3.4.6)软件包亦包含了权威列表：/usr/share/doc/base-passwd/users-and-groups.html。

9.2.4 更安全地工作 - sudo

使用 sudo 最主要的目的是保护自己少做蠢事，我认为使用系统时使用 sudo 比总是使用 root 帐号更好。

Ubuntu 系统默认使用 sudo，来工作，你建立的第一个帐户将自动加入到 admin 组， admin 组的成员拥有 sudo 的权限。

安装 sudo 然后编辑 sudoers 中有关选项激活它。还可在 /usr/share/doc/sudo/OPTIONS 中查看 sudo 的用户组特性。

样例中的配置，设定“staff”用户组成员可通过 sudo 执行任何 root 权限的命令而“src”用户组成员只可执行规定的一部分 root 权限的命令。

使用 sudo 的好处在于只需一个普通用户密码登录，并且所有的活动都受到监控。用它为低级别的系统管理员赋权是个好主意。例如：

```
$ sudo chown -R myself:mygrp .
```

当然，如果你知道 root 密码（绝大部分在家安装系统的用户都会知道），就可以在普通用户下执行任何 root 命令：

```
$ su -c "shutdown -h now"
Password:
```

（我想我该严格限制 admin 帐号的 sudo 特权，但对于家中的服务器，就不用考虑那么多了。）

想了解其它允许普通用户执行 root 权限命令的程序，可以看看 super 软件包。

9.2.5 服务的访问限制

对于 Internet *超级服务器*，inetd 会在系统启动时通过 /etc/rc2.d/S20inetd（for RUNLEVEL=2）加载，S20inetd 是一个指向 /etc/init.d/inetd 的符号链接。本质上，inetd 允许一个运行中的守护进程(daemon)调用其它多个守护进程，以减轻系统的负载。

当某个服务请求到达，系统会查询 /etc/protocols 和 /etc/services 中的数据库，确定该请求所指定的相关协议和服务，接着 inetd 会在 /etc/inetd.conf 数据库中查找普通 Internet 服务或 /etc/rpc.conf 中查找基于 Sun-RPC 的服务。

为了系统安全，请在 /etc/inetd.conf 中关闭所有不用的服务。涉及到 NFS 和其它基于 RPC 的程序时需要激活 Sun-RPC 服务。

有时，inetd 并不直接打开请求的服务，而是在 /etc/inetd.conf 中将该服务名作为的参数，打开 tcpd TCP/IP 守护进程包装程序。这时，tcpd 首先登记请求并使用 /etc/hosts.deny 和 /etc/hosts.allow 进行附加的检查，然后再运行相应的服务程序。

如果新版的 Ubuntu 系统进行远程访问时出现问题，可以在 /etc/hosts.deny 中注释掉“ALL: PARANOID”，如果有该行的话。

更多信息参阅 inetd(8)、inetd.conf(5)、protocols(5)、services(5)、tcpd(8)、hosts_access(5) 和 hosts_options(5)。

有关 Sun-RPC 的更多信息参阅 rpcinfo(8)、portmap(8) 和 /usr/share/doc/portmap/portmapper.txt.gz。

9.2.6 集中式验证 - LDAP

使用轻形目录访问控制协议（LDAP） 参阅：

- OpenLDAP
- OpenLDAP 管理员指南在软将包 openldap-guide 中
- LDP: LDAP Linux HOWTO
- LDP: LDAP Implementation HOWTO
- OpenLDAP, extensive use reports
- Open LDAP with Courier IMAP and Postfix

9.3 刻录机

ATAPI/IDE 接口的刻录机是时下非常流行的配件，它是极好的系统备份工具，特别是对于那些单个文件容量一般 < 640 MB 的家庭用户。更多权威的信息，请参阅 LDP CD-Writing-HOWTO。

9.3.1 概述

首先需要说明的是，在向刻录机发送数据过程中，任何数据中断都会对光盘造成无法挽回的损坏。所以应选购缓冲区尽可能大的刻录机。如果资金充裕，就别 再考虑 ATAPI/IDE 型的，买台 SCSI 型的没错。如果可以连接 IDE 接口，就使用 PCI 总线（例如，在主板上）而别用 ISA 总线（SB16 声卡使用的就是它）。

当刻录机连接到 IDE，驱动它的通常是 IDE-SCSI 驱动而非旧式的 IDE CD 驱动，所以，需要激活 SCSI 通用驱动。有两种方法激活它，假设系统使用的是较新版本的内核（如 2001 年三月的版本）。

对于 Linux 2.6 的内核，你应该使用 IDE 驱动并直接使用 /dev/hdx 这些设备名称来访问。这种方式你可以使用 DMA。

9.3.2 方法一：modules + lilo

如果使用的是 Ubuntu 原装内核，将下面的内容添加到 /etc/lilo.conf，如果有多个选项，列出时要将它们用空格分隔开：

```
append="hdx=ide-scsi ignore=hdx"
```

在此，刻录机使用 ide-scsi 驱动访问，hdx 就代表它，其中 x 代表下列任何一种设备：

```
hda          接第一个 IDE 接口作主盘
hdb          接第一个 IDE 接口作从盘
hdc          接第二个 IDE 接口作主盘
hdd          接第二个 IDE 接口作从盘
hde ... hdh  接扩展 IDE 接口或 ATA66/100 IDE 接口
```

完成上述配置工作后以 root 身份运行下列命令激活设备

```
# lilo
# shutdown -h now
```

9.3.3 方法二：重编译内核

Ubuntu 用 make-kpkg 创建新内核，使用 make-kpkg 时加上新的 --append_to_version 参数可创建多重内核镜像。参阅 Ubuntu 下的 Linux 内核, 第 7 章。

make menuconfig 后执行下列步骤：

- bzImage
- 包含 IDE CD driver（不是必须的，但这样更简单）
- 将 ide-scsi 和 sg 编译进内核，或编译成模块

9.3.4 配置步骤

下列步骤可让系统在启动时激活内核对刻录机的支持：

```
# echo ide-scsi >>/etc/modules
# echo sg      >>/etc/modules
# cd /dev; ln -sf scd0 cdrom
```

手工激活可以这样做：

```
# modprobe ide-scsi
# modprobe sg
```

重启以后，用下列方法检查安装情况：

```
$ dmesg|less
# apt-get install cdrecord
```

```
# cdrecord -scanbus
```

[Per Warren Dodge]如果机器上同时有 CD-ROM 和 CD-R/RW,这时 ide-scsi 和 ide-cd 可能会产生冲突，请试试在 /etc/modutils/aliases 中加上下面的内容，然后运行 update-modules 并重启系统。

```
pre-install      ide-scsi      modprobe ide-cd
```

上述指令指示系统在加载 ide-scsi 前先加载 IDE 驱动。IDE 驱动 ide-cd 接管所有 ATAPI CD-ROM——对指明**忽略的**设备除外。剩下的设备才由 ide-scsi 来管理。

9.3.5 光盘镜像文件（可引导光盘）

将 target-directory/ 下的文件制作成光盘镜像文件 cd-image.raw（可引导系统、Joliet TRANS.TBL-enabled 格式的光盘；如果不需要引导系统功能，可去掉 -b 和 -c 选项），在第一个软驱中插入启动软盘然后执行：

```
# dd if=/dev/fd0 target-directory/boot.img
# mkisofs -r -V volume_id -b boot.img -c bootcatalog -J -T \
-o cd-image.raw target_directory/
```

一个有趣的黑客尝试是制作一盘 DOS 引导光盘。如果上述的 boot.img 文件中包含了通用 DOS 引导软盘镜像，光盘就可以象插在软驱（A:）中的 DOS 软盘一样引导 DOS 系统。如果再加上 freeDOS 就更有趣。

想检查该光盘镜像文件，可以在回送设备（loop device）上加载它。

```
# mount -t iso9660 -o ro,loop cd-image.raw /cdrom
# cd /cdrom
# mc
# umount /cdrom
```

9.3.6 刻录光盘（R，R/W）：

首先进行设备测试（假设是双倍数刻录）

```
# nice --10 cdrecord -dummy speed=2 dev=0,0 disk.img
```

如果测试通过，执行下面的命令刻录 CD-R

```
# nice --10 cdrecord -v -eject speed=2 dev=0,0 disk.img
```

或执行下面命令刻录 CD-RW

```
# nice --10 cdrecord -v -eject blank=fast speed=2 dev=0,0 disk.img
```

某些型号的 CD-RW 刻录机用下面的命令更好

```
# nice --10 cdrecord -v blank=all speed=2 dev=0,0 disk.img
```

接下来执行

```
# nice --10 cdrecord -v -eject speed=2 dev=0,0 disk.img
```

分两步做是必要，这可以防止在刻录时遇到数据空白产生 SCSI 超时错误。nice 参数可时也要做一些调整。

9.3.7 制作光盘镜像文件

某些 CD-R 和商业光盘在数据末尾追加了空白扇区（junk sectors），使用 dd 无法拷贝这些光盘（Windows98 CD 就是其中之一）。cdrecord 软件包中有一个 readcd 命令，用它可以将任何光盘内容拷贝成镜像文件。对于数据盘，先挂载，运行 df 查看它的实际大小，再将显示的数字（in blocks, === 1024 bytes）除以 2 得到实际光盘扇区数（2048 bytes），带参数运行 readcd 用该硬盘镜像文件烧制 CD-R/RW。

```
# readcd dev=target,lun,scsibusno # select function 11
```

其中，大部分情况下命令行中三个参数都为 0。有时 readcd 给出的扇区数会偏多！此时使用前面用挂载镜像的方法得出的大小值来对上述参数赋值效果更好。

应该提醒的是，如果你对 CD-ROM 使用 dd 的话，会有不少问题。第一次执行 dd 时可能会产生错误信息并丢失光盘镜像末端的一些数据。再次执行 dd 时，如果没有指定镜像的大小的话，在一些系统上会产生一个过大光盘镜像，其末端都是垃圾。只有第二次运行 dd 时，使用正确的镜像大小并在看到错误信息之后不弹出光盘，才能避免这些问题。例如，假设用 df 得到镜像的大小为 46301184 blocks，则执行两次下面的命令可以得到正确的镜像（这是我的经验）：

```
# dd if=/dev/cdrom of=cd.img bs=2048 count=$((46301184/2))
```

9.3.8 Ubuntu 安装盘镜像

有关 Ubuntu CDs 的最新信息，请浏览 Ubuntu CD site。

如果有较快的 Internet 连接，可考虑用下面的引导方法从网络安装系统：

- 一些软盘镜像。
- 一个迷你型可引导光镜像。

如果没有较快的 Internet 连接，可考虑从光盘分发免费获取安装光盘。

请不要浪费带宽来下载标准光盘镜像（即使是使用新的 jigdo 方式），除非你是光盘镜像测试员。

有一个很有名的光盘镜像 KNOPPIX - Live Linux Filesystem On CD。该光盘可以启动一个全功能的 Debian 系统而且不需要在硬盘上安装。

9.3.9 将系统备份到 CD-R

想要将重要的配置文件和数据备份到 CD-R，可使用 backup 中的“backup”脚本。亦可参阅 差异备份与数据同步，第 8.4 节。

9.3.10 将音乐 CD 刻录到 CD-R

我没测试过：

```
# apt-get install cdrecord cdparanoia
# cdparanoia -s -B
# cdrecord dev=0,0,0 speed=2 -v -dao -eject defpregap=1 -audio *.wav
```

或

```
# apt-get install cdrdao #disk at once
# cdrdao read-cd --device /dev/cdrom --paranoia-mode 3 my_cd # read cd
# cdrdao write --device /dev/cdrom --speed 8 my_cd    # write a new CD
```

cdrdao 与拷贝不同（如没有数据间隙，等...）。

9.3.11 刻录 DVD-R、DVD-RW 和 DVD+RW

刻录 DVD 光盘有两种方式：

- 使用 growisofs 并搭配 mkisofs。
- 按照 /usr/share/doc/cdrecord/README.DVD.Debian 的说明重新编译 cdrecord，并创建加入了 dvd 支持的本地软件包。

9.4 X

X 窗口系统由 XFree86 提供。Ubuntu 的最早系统中 X 服务器有两个主要版本：XFree86 Version 3.3 (XF3)和 XFree86 Version 4.x series (XF4)，它们都是基于 X.Org 制定的 X11R6 标准。现在全部使用 xorg 。

想了解 X 的基础知识，可参阅 X(7)，LDP XWindow-User-HOWTO 和 Remote X Apps mini-HOWTO。 对 Debian 用户专门的指南，可阅读 xfree86-common 软件包中提供的 /usr/share/doc/xfree86-common/FAQ.gz，其中 Branden Robinson 有一些关于 key binding 的有趣且权威的讨论。

X 服务器，第 9.4.3 节:: 该程序存在于那些需要在用户显示器（CRT，LCD）上显示 X 窗口和桌面并接收键盘和鼠标输入的本地主机上。

X 客户端，第 9.4.4 节:: 该程序存在于那些需要运行与 X 兼容的应用程序的（本地或远程）主机上。

这正好将常规的“服务器”和“客户机”关系倒转过来。

有几种途径让“X server”（显示端）接收远程“X client”（应用端）的连接请求：

- xhost 方式

    - 主机列表机制（很不安全）。
    - 协议不加密（易受到网络监听攻击）
    - 尽量不要使用该方式。
    - 参阅 联接远程的 X 服务器 - xhost, 第 9.4.7 节 和 xhost(1x)。

- **xauth** 方式

    - MIT magic cookie 机制（不安全但比 xhost 强点）。
    - 协议不加密（易受到网络监听攻击）
    - 仅用于本地连接，它所需的 CPU 消耗比 ssh -X 低。
    - 参阅 X 下获取 root 权限, 第 9.4.12 节 和 xauth(1x)。

- xdm, wdm, gdm, kdm, ... 方式

    - MIT magic cookie 机制（和 xauth 一样不安全）
    - 参阅 xdm(1x) 和 Xsecurity(7) 获得更多有关 X 显示访问控制的基础知识
    - 参阅 wdm(1x)、gdm(8) 和 kdm.options(5) 获得更多信息，当然先得装上它们。
    - 参阅 自定义运行级别, 第 2.4.3 节了解如何在不删除 xdm 包的情况下禁用它，使系统启动到控制台。

- **ssh -X** 方式

    - 基于安全 shell 的端口发送机制（**安全**）。
    - 加密协议（在本地使用很耗系统资源）。
    - 使用它进行远程连接。
    - 参阅 联接远程的 X 服务器 - ssh, 第 9.4.8 节。

除了 ssh，所有的远程连接方式，都需要 X 服务器开启 TCP/IP 连接。参阅 在 TCP/IP 中使用 X, 第 9.4.6 节。

9.4.1 X 软件包

提供了下列几个（meta）软件包来简化 X 的安装。

x-window-system-core:: 该综合包提供一些基本组件,用于在单一工作站上运行 X Window 系统，其中包括 X 函数库、一个 X 服务器（xserver-xfree86)、一套字体、一组基本的 X 客户端及工具。

x-window-system:: 该综合包提供 XFree86 项目开发的所有 X Window 系统的组件，以及一套经久不衰的辅助程序。(注意，它包含了 x-window-system-core、twm 和 xdm 等组件,故安装了它就不用再安装 x-window-system-core 了。)

xserver-common-v3:: XFree86 3.x X 服务器（X3）相关的程序和工具。

xserver-*:: XF3 服务器软件包的补充包，包含了对那些新的 XF4 服务器（xserver-xfree86）不支持的硬件的支持。如 XF4 不支持某些老式的 ATI mach64 卡，某些视频卡在 Woody 版的 XF4 中无法工作等等。(要获得可用软件包，可执行 apt-cache search xserver-|less。所有这些 XF3 服务器均是基于 xserver-common-v3 的。)

大多数情况下，应该安装 x-window-system（如果要通过控制台登录，需禁用 xdm，具体方法参阅 “我不想直接启动到 X!”，第 8.1.4 节。）

9.4.2 X 服务器的硬件侦测

在安装 X 系统之前安装下列软件包，就能在 X 配置阶段实现硬件侦测：

- discover - 硬件识别系统。
- mdetect - 鼠标自动侦测工具。
- read-edid - VESA PnP 监视器硬件信息收集工具。

9.4.3 X 服务器

有关 X 服务器的信息，参阅 xorg(1x)。

从本地控制台调用 X 服务器：

```
$ startx -- :<display> vtXX
e.g.:
$ startx -- :1 vt8 -bpp 16
... start on vt8 connected to localhost:1 with 16 bpp mode
```

--后面的参数用于设置 X 服务器。

注意，在使用 ~/.xserverrc 脚本定制 X 服务器启动进程时，请确保 exec 调用的是真正的 X 服务器。如果没这么做会导致 X 服务器启动缓慢及退出。例如：

```
#!/bin/sh
exec /usr/bin/X11/X -dpi 100 -nolisten tcp
```

9.4.3.1 配置 X 服务器（版本 4）

（重新）配置 XF4 服务器，

```
# dpkg-reconfigure --priority=low xserver-xorg
```

该命令会生成 /etc/X11/xorg.conf 文件并调用 dexconf 脚本来配置 X。

9.4.3.2 配置 X 服务器（版本 3）

（重新）配置 XF3 服务器。例如，针对 ATI mach64，

```
# dpkg-reconfigure --priority=low xserver-common-v3
# dpkg-reconfigure --priority=low xserver-mach64
```

该命令会生成 /etc/X11/XF86Config 文件并调用 xf86config-v3 脚本来配置 X。

9.4.3.3 手工配置 X 服务器

往文件 /etc/X11/xorg.conf 中添加用户自定义内容时,**不要在配置文件的定义段落中进行编辑**:

```
### BEGIN DEBCONF SECTION
[snip]
### END DEBCONF SECTION
```

正确是做法是将用户定义内容加在定义段落之前。例如，要添加自定义视频卡，可在文件*开头*添加类似下面的内容：

```
Section "Device"
Identifier        "Custom Device"
Driver            "ati"
Option            "NoAccel"
EndSection

Section "Screen"
Identifier  "Custom Screen"
Device      "Custom Device"
Monitor     "Generic Monitor"
DefaultDepth 24
Subsection "Display"
Depth       8
Modes       "1280x960" "1152x864" "1024x768" "800x600" "640x480"
EndSubsection
Subsection "Display"
Depth       16
Modes       "1280x960" "1152x864" "1024x768" "800x600" "640x480"
EndSubsection
Subsection "Display"
Depth       24
Modes       "1280x960" "1152x864" "1024x768" "800x600" "640x480"
EndSubsection
EndSection

Section "ServerLayout"
Identifier        "Custom"
Screen            "Custom Screen"
InputDevice       "Generic Keyboard" "CoreKeyboard"
InputDevice       "Configured Mouse" "CorePointer"
EndSection
```

对于 Sarge（撰写本文时是 testing)，如果你希望在升级的时候保留用户的自定义的 /etc/X11/xorg.conf 设置，请用 root 运行下列命令：

```
# cp /etc/X11/xorg.conf /etc/X11/xorg.conf.custom
# md5sum /etc/X11/xorg.conf > /var/lib/xfree86/xorg.conf.md5sum
```

```
# dpkg-reconfigure xserver-xorg
```

如果想美化字体，请按照 X 下的 TrueType 字体，第 9.4.13 节中的说明来修改 /etc/X11/xorg。

请同时检查 X 设置中的其他部分。不良的显示器设置甚至比难看的字体更让人头痛，所以请确保你设置的刷新率是你显示器能处理的最高刷新率（85 Hz 很好，75 Hz 还可以，60 Hz 就很糟糕了）。

9.4.4 X 客户端

绝大多数 X 客户端程序都可以用类似下面的命令启动：

```
client $ xterm -geometry 80x24+30+200 -fn 6x10 -display hostname:0 &
```

命令行中各参数的含义如下：

- -geometry WIDTHxHEIGHT+XOFF+YOFF：窗口初始尺寸和位置。
- -fn FONTNAME：显示文本的字体。FONTNAME 的赋值有如下几个：
    - a14: 普通字体
    - a24: 大号字体
    - ... （使用 xlsfont 检查可用字体。）
- -display displayname：X 服务器名称。 displayname 的赋值有如下几个：
    - hostname:D.S 表示在名为 hostname 的主机的显示器 D 上显示的屏幕 S; 工作于该显示器的 X 服务器监听 TCP 端口 6000+D。
    - host/unix:D.S 表示在 host 主机的显示器 D 上显示的屏幕 S；工作于该显示器的 X 服务器监听 UNIX domain socket /tmp/.X11-unix/XD（故只能从主机访问它）。
    - :D.S 等价于 host/unix:D.S，其中 host 代表本地主机名。

默认的 X 客户端程序（应用端）的 displayname 可通过 DISPLAY 环境变量来设置。例如: 在运行某 X 客户端程序之前，执行下列命令之一就可以完成设置工作：

```
$ export DISPLAY=:0
# 默认情况下，本地机器使用第一个 X 屏幕
$ export DISPLAY=hostname.fulldomain.name:0.2
$ export DISPLAY=localhost:0
```

程序启动方式可以在~/.xinitrc 中进行自定义。例如：

```
xrdb -load $HOME/.Xresources
xsetroot -solid gray &
xclock -g 50x50-0+0 -bw 0 &
xload -g 50x50-50+0 -bw 0 &
xterm -g 80x24+0+0 &
xterm -g 80x24+0-0 &
```

twm

正如 自定义 X 会话, 第 9.4.5.1 节中所描述的，当使用 startx 启动 X 时，该脚本将重载 Xsession 所做的所有常规操作，通常用 ~/.xsession 来代替，而该方法仅作为最后的手段使用。参阅 xsetroot(1x)、xset(1x) 和 X 资源, 第 9.4.10 节。

9.4.5 X 会话

X 会话（X 服务器 + X 客户端）可使用下列方法启动：

- startx：xinit 的脚本化命令（wrapper script command），负责从 Linux 字符型控制台启动 X 服务器和客户端。如果 ~/.xinitrc 文件不存在，/etc/X11/xinit/xinitrc 会调用并执行 /etc/X11/Xsession。
- xdm、gdm、kdm 或 wdm：X 显示管理器守护进程，负责启动 X 服务器和客户端，并管理来自 GUI 屏幕的登录行为。直接执行 /etc/X11/Xsession。

想使用控制台参阅 “我不想直接启动到 X!”, 第 8.1.4 节。

9.4.5.1 自定义 X 会话

默认的启动脚本 /etc/X11/Xsession 是 /etc/X11/Xsession.d/50xfree86-common_determine-startup 和 /etc/X11/Xsession.d/99xfree86-common_start 的高效的结合体。

/etc/X11/Xsession 的执行会受 /etc/X11/Xsession.options 的影响，从本质上讲，它使用 exec 命令执行系统中按下面的次序排序，排在第一位的程序:

- ~/.xsession or ~/.Xsession，如果它被定义。
- /usr/bin/x-session-manager，如果它被定义。
- /usr/bin/x-window-manager，如果它被定义。
- /usr/bin/x-terminal-emulator，如果它被定义。

Ubuntu 选择系统 Ubuntu alternative system ）对这些命令的确切定义进行了描述，参阅 Alternative 命令, 第 6.5.3 节。例如：

```
# update-alternatives --config x-session-manager
... 或
# update-alternatives --config x-window-manager
```

如果想定义某 X 窗口管理器为默认窗口管理器，同时保留已安装的 GNOME 和 KDE 会话管理器，按下面的方法编辑 /etc/X11/Xsession.options 来禁用 X 会话管理器：

```
# /etc/X11/Xsession.options
#
# configuration options for /etc/X11/Xsession
# See Xsession.options(5) for an explanation of the available options.
# Default enabled
allow-failsafe
allow-user-resources
```

```
allow-user-xsession
use-ssh-agent
# Default disabled (enable them by uncommenting)
do-not-use-x-session-manager
#do-not-use-x-window-manager
```

如果不想按上述方法修改系统，由于 gnome-session 和 kdebase 软件包包含了那些 X 会话管理器。所以删除它们，X 窗口管理器就成了默认窗口管理器了。(废话，还更好的主意吗？)

对于那些 /etc/X11/Xsession.options 中仅包含一行 allow-user-xsession 的系统,任何定义了 ~/.xsession 或 ~/.Xsession 的用户,均可以自定义 /etc/X11/Xsession 的行为。

~/.xsession 文件中排在最后的命令,其格式应该为 exec some-window/session-manager,用来启动你喜欢的 X 窗口/会话管理器。

/usr/share/doc/xfree86-common/examples/xsession.gz 给出了一个不错的 ~/.xsession 脚本样例。

我使用它来为每个用户设置窗口管理器、屏幕访问和语言支持。参阅 针对用户启动 X 会话, 第 9.4.5.2 节、 X 下获取 root 权限, 第 9.4.12 节、 多语言的 X 窗口系统范例, 第 9.7.9 节。

如果你希望一些 X 客户端程序能自动启动,参阅 X 客户端, 第 9.4.4 节 的范例并将其写在 ~/.xsession 而不是 ~/.xinitrc。

用户自己添加的 X 资源保存在 ~/.Xresources，而系统级的 X 资源保存于 /etc/X11/Xresources/*。参阅 xrdb(1x)。

用户可以在 ~/.xmodmaprc 中自定义键盘布局和鼠标按键布局，参阅 xmodmap(1x)。

9.4.5.2 针对用户启动 X 会话

遵循 自定义 X 会话, 第 9.4.5.1 节中描述的原则，要激活用户特定的 X 会话/窗口管理器，需要安装相应的软件包并在 ~/.xsession 文件末尾添加如下内容（我爱用 blackbox/fluxbox，它简单快捷。)：

- 默认 X 会话管理器
    - 参阅 Alternative 命令, 第 6.5.3 节.
    - exec /usr/bin/x-session-manager
- 默认 X 窗口管理器
    - 参阅 Alternative 命令, 第 6.5.3 节.
    - exec /usr/bin/x-window-manager
- GNOME 会话管理器(loaded)
    - 需安装软件包：gnome-session

    - exec /usr/bin/gnome-session

- KDE 会话管理器(loaded)

    - 需安装软件包：kdebase (or kdebase3 for KDE3)
    - exec /usr/bin/kde2

- Blackbox 窗口管理器(lightweight, slick).

    - 需安装软件包：blackbox
    - exec /usr/bin/blackbox

- Fluxbox 窗口管理器(lightweight, new blackbox).

    - 需安装软件包：fluxbox
    - exec /usr/bin/fluxbox

- Xfce 窗口管理器(Mac OS-X, SUN CDE like).

    - 需安装软件包：xfce
    - exec /usr/bin/xfwm

- IceWM 窗口管理器(lightweight, GNOME alternative)

    - 需安装软件包：icewm
    - exec /usr/bin/X11/icewm

- FVWM2 虚拟窗口管理器(lightweight, Win95 like)

    - 需安装软件包：fvwm
    - exec /usr/bin/fvwm2

- Windowmaker 窗口管理器(somewhat NexT like)

    - 需安装软件包：wmaker
    - exec /usr/bin/wmaker

- Enlightenment 窗口管理器(loaded).

    - 需安装软件包：enlightenment
    - exec /usr/bin/enlightenment

参阅 Window Managers for X.

9.4.5.3 配置 KDE/GNOME

要配置完整的 KDE 或 GNOME 环境，下列的综合包很有用：

- KDE：安装 kde 软件包
- GNOME：安装 gnome 软件包

请使用能操作 Recommends 类软件包的安装工具安装这些软件包，如 dselect 和 aptitude，比起 apt-get 它们能提供更丰富的软件供你选择。

如果想从控制台登录，必须禁用 X 显示管理器，例如 kdm、gdm 和 wdm 这会牵扯到一些关联问题，有关信息参阅 “我不想直接启动到 X!”，第 8.1.4 节。

如果想将系统的默认环境由 KDE 换成 GNOME，请用 Alternative 命令，第 6.5.3 节中所述的方法配置 x-session-manager。

9.4.6 在 TCP/IP 中使用 X

由于不加密的远程 TCP/IP 套接字连接易受到窃听攻击，新版的 Debian 安装 X 时默认是禁用 TCP/IP 套接字口的。建议使用 ssh 进行远程 X 连接(参阅 联接远程的 X 服务器 - ssh，第 9.4.8 节)。

通常不推荐使用本节所述的方法，除非系统处于防火墙之后且所处网络中全是绝对可信任的用户。使用下面的命令检查当前 X 服务器的 TCP/IP 套接字口的设置：

```
# find /etc/X11 -type f -print0 | xargs -0 grep nolisten
/etc/X11/xinit/xserverrc:exec /usr/bin/X11/X -dpi 100 -nolisten tcp
```

删除 -nolisten 就可以恢复 X 服务器对 TCP/IP 的监听。

9.4.7 联接远程的 X 服务器 - xhost

xhost 允许通过主机名访问。该方式极不安全。下面的方法将关闭主机验证功能，只要 TCP/IP 套接字连接功能是打开的（参阅 在 TCP/IP 中使用 X，第 9.4.6 节）本机就会接收来自任何地方的连接请求。

```
$ xhost +
```

要重新打开主机验证功能可执行：

```
$ xhost -
```

xhost 无法区分远程主机上不同的用户，而且远程连接的主机名（实际上是地址）也可以是伪造的。

如果处于一个不可信的网络环境（例如通过 PPP 拨号连接到 Internet），即使在网络中成为主机受到一定标准的限制，也应尽量避免使用该连接方式。参阅 xhost(1x)。

9.4.8 联接远程的 X 服务器 - ssh

使用 ssh 可以在本地主机和远程应用服务器之间建立一个安全的连接通道。

- 如果不想每次执行相同的命令行选项，可在远程主机的 /etc/ssh/sshd_config 文件中，打开 X11Forwarding 和 AllowTcpForwarding 选项。
- 启动本地主机的 X 服务器。
- 在本地主机上开一个 xterm 进程。
- 运行 ssh 建立与远程站点的连接。

```
localname @ localhost $ ssh -q -X -l loginname remotehost.domain
Password:
.....
```

- 在远程站点上运行 X 应用程序命令。

```
loginname @ remotehost $ gimp &
```

该连接方式使得远程 X 客户机上的屏幕输出，看上去就好象是通过本地 UNIX 域套接字的方式连接到服务器的客户机输出。

9.4.9 X 终端模拟器 - xterm

学习 xterm 可以去 http://dickey.his.com/xterm/xterm.faq.html。

9.4.10 X 资源

许多老式的 X 程序，如 xterm，使用 X 资源数据库配置它们的外观。~/.Xresources 文件用于保存用户资源定义。登录后该文件自动合并到默认的 X 资源中。系统范围的缺省配置存储在 /etc/X11/Xresources/* 中，应用程序缺省的配置存储在 /etc/X11/app-defaults/*。使用这些设置作为学习的起点。

这儿是一些有用的设置，可加到~/.Xresources 文件中：

```
! Set the font to a more readable 9x15
XTerm*font: 9x15

! Display a scrollbar
XTerm*scrollBar: true

! Set the size of the buffer to 1000 lines
XTerm*saveLines: 1000

! Large kterm screen
KTerm*VT100*fontList: -*-fixed-medium-r-normal--24-*,\
-*-gothic-medium-r-normal--24-*,\
-*-mincho-medium-r-normal--24-*
```

要使上述设置立即生效，可用下面的命令将它们合并到数据库：

```
xrdb -merge ~/.Xresources
```

参阅 xrdb(1x)。

9.4.11 X 中键盘和指针按钮的映射

xmodmap 程序用来编辑和显示键盘修订表和按键映射表的，客户端程序用它们来把按键代码事件(event keycodes)转换成 X 中的 keysyms。

```
$ xmodmap -pm
... 显示当前按键修订表
$ xmodmap -pk | pager
... 显示当前按键映射表
$ xmodmap -e "pointer === 3 2 1" # 设置为惯用左手鼠标
$ xmodmap ~/.xmodmaprc # 用 ~/.xmodmaprc 中的描述设置键盘
```

通常从用户的会话中启动脚本，从 ~/.xsession 中执行。

要获得按键代码(keycode)，请在 X 中运行 xev 并按键。要想获得 keysym 的含义，请从宏定义文件 /usr/include/X11/keysymdef.h 中查找。该文件中所有的 #define 声明都用 XK_ 命名，伪装成 keysym 的名字。

参阅 xmodmap(1x)。

9.4.12 X 下获取 root 权限

如果运行 GUI 程序时需要 root 权限，请用下面的步骤在用户的 X 服务器上显示程序输出。**千万不要直接使用 root 帐号启动 X 服务器**以避免承担不必要的安全风险。

以普通用户身份启动 X 服务器，开一个 xterm 控制台窗口，执行：

```
$ XAUTHORITY=$HOME/.Xauthority
$ export XAUTHORITY
$ su root
Password:*****
# printtool &
```

非 root 用户以 su 方式运用该技巧时，要确保该非 root 用户所在用户组对 ~/.Xauthority 文件有读权限。

想要系统自动执行该命令序列，请在用户帐号下创建 ~/.xsession 文件，编辑文件如下：

```
# This makes X work when I su to the root account.
if [ -z "$XAUTHORITY" ]; then
XAUTHORITY=$HOME/.Xauthority
export XAUTHORITY
fi
unset XSTARTUP
# If a particular window/session manager is desired, uncomment following
# and edit it to fit your needs.
#XSTARTUP=/usr/bin/blackbox
# This starts x-window/session-manager program
if [ -z "$XSTARTUP" ]; then
if [ -x /usr/bin/x-session-manager ]; then
XSTARTUP=x-session-manager
elif [ -x /usr/bin/x-window-manager ]; then
XSTARTUP=x-window-manager
elif [ -x /usr/bin/x-terminal-emulator ]; then
XSTARTUP=x-terminal-emulator
```

```
fi
fi
# execute auto selected X window/session manager
exec $XSTARTUP
```

接着在用户的 xterm 窗口中运行 su（不是 su -）。现在从该 xterm 启动的 GUI 程序就可以在该用户的 X window 环境中显示以 root 权限运行的程序输出。只要执行了默认的 /etc/X11/Xsession，就可以使用该方法。如果用户使用 ~/.xinitrc 或 ~/.xsession 来配置自定义环境，需要将上面提到的环境变量 XAUTHORITY 加到这些脚本中去。

还有一种方法，sudo 可用于自动执行上面的命令序列：

```
$ sudo xterm
... 或
$ sudo -H -s
```

这时 /root/.bashrc 中应包含：

```
if [ $SUDO_USER ]; then
sudo -H -u $SUDO_USER xauth extract - $DISPLAY | xauth merge -
fi
```

即使对那些 home 目录位于 NFS 上的用户，它也能正常工作。因为 root 不用读 .Xauthority 文件。

还有一些用于该目的的专用软件包：kdesu、gksu、gksudo、gnome-sudo 和 xsu。其它方法也可以达到同样的目的：如在 /root/.Xauthority 和相应用户文件之间创建一个符号链接；使用 sux 脚本；或对 root 初始化脚本执行“xauth merge ~USER_RUNNING_X/.Xauthority"。

更多方法参阅 debian-devel mailing list。

9.4.13 X 下的 TrueType 字体

xorg 中标准的 xfs 能完美地驱动 TrueType 字体，如果你使用的是 XFree86-3，就得安装第三方字体服务器如 xfs-xtt。

不论什么应用程序，如果要使用 TrueType 字体，就要与 libXft 或 libfreetype 建立链接（如果你使用的是已编译好的 .deb 包，就不用在这方面操心了）。

首先进行字体支持的基础设置：

- 安装软件包 x-ttcidfont-conf 和 defoma。它们能自动生成文件 fonts.scale 和 fonts.dir。

```
# apt-get install x-ttcidfont-conf
```

- 编辑 /etc/X11/xorg.conf 的 Section "Files" 部分，如下显示：

```
Section "Files"
FontPath   "/var/lib/defoma/x-ttcidfont-conf.d/dirs/TrueType"
FontPath   "/usr/share/fonts/truetype"
FontPath   "/usr/lib/X11/fonts/CID"
FontPath   "/usr/lib/X11/fonts/Speedo"
FontPath   "/usr/lib/X11/fonts/misc"
FontPath   "/usr/lib/X11/fonts/cyrillic"
FontPath   "/usr/lib/X11/fonts/100dpi:unscaled"
FontPath   "/usr/lib/X11/fonts/75dpi:unscaled"
FontPath   "/usr/lib/X11/fonts/Type1"
EndSection
```

第一行使 xorg 使用任何你从 Ubuntu 软将包安装的 TrueType 字体。因为 xorg 渲染 Type1 字体的效果很差,Type1 字体的目录就放在后面了。点阵字体的小技巧 :unscaled 对于新的 XF4 来说是不需要的，我把它包含在内是为了确保万无一失。 为了保留手工修改的 /etc/X11/xorg.conf 文件，请按照 手工配置 X 服务器, 第 9.4.3.3 节的说明操作。

然后安装 DFSG 字体软件包：

- 西方 TrueType 字体：
  - ttf-bitstream-vera：一套由 Bitstream, Inc 创造的高质量的 TrueType 字体。 [40]
  - ttf-freefont：一套高质量的 TrueType 字体，包含了 UCS 字符集。
  - ttf-thryomanes：一套包括了 Latin、Greek、Cyrillic 和 IPA 的Unicode TrueType 字体。
- 亚洲字体：
  - tfm-arphic-bsmi00lp: Chinese Arphic "AR PL Mingti2L Big5" TrueType font TeX font metric data
  - tfm-arphic-bkai00mp: Chinese Arphic "AR PL KaitiM Big5" TrueType font TeX font metric data
  - tfm-arphic-gbsn00lp: Chinese Arphic "AR PL SungtiL GB" TrueType font TeX font metric data
  - tfm-arphic-gkai00mp: Chinese Arphic "AR PL KaitiM GB" TrueType font TeX font metric data
  - ttf-baekmuk: Korean Baekmuk series TrueType fonts
  - hbf-jfs56: Chinese Jianti Fangsong 56x56 bitmap font (GB2312) for CJK
  - hbf-cns40-b5: Chinese Fanti Song 40x40 bitmap font (Big5) for CJK
  - hbf-kanji48: Japanese Kanji 48x48 bitmap font (JIS X-0208) for CJK

由于供**自由**使用的字体有时很有限，Debian 用户也可以安装或共享某些商业 TrueType 字体。为了简化安装这类字体的工序，于是产生了一些方便的软件包：

- ttf-commercial

- msttcorefonts (>1.1.0) [41]

请慎重选择 TT 字体，以免自由系统受到不自由字体的污染。

所有这些 Debian 字体软件包不用设置就能工作，并且对于使用“core”字体系统的 X 程序来说，它都是可用的。包括 Xterm、Emacs 和其他一些非 KDE 和 gnome 的程序。

现在运行 xfontsel，在 fndry 菜单中选中任何一个 TrueType 字体，你可以在 “fmly" 菜单中看到很多项目。

对于 KDE2.2 和 GNOME1.4（搭配 libgdkxft0 使 GTK 1.2 能对字体进行反锯齿的渲染），同样的你需要配置 Xft1。Xft1 非常的过时了，基本上只有 GNOME1.4 和 KDE2.2 在使用。编辑 /etc/X11/XftConfig 文件，在其他“dir”之前加入下面这一行东西。

```
dir "/var/lib/defoma/x-ttcidfont-conf.d/dirs/TrueType"
```
[42]

对于 GNOME2 和 KDE3（Sarge 之后的版本），你需要设置 fontconfig，Xft2 用它来查找字体。 你不必为此再安装其他额外的软件包，因为所有用到 fontconfig 的软件包都会依赖与它的。

首先，查看 /etc/fonts/fonts.conf。里面应该有如下的一行内容。如果没有，就打开 /etc/fonts/local.conf，在 <fontconfig> 行后面添加这些内容。

```
<dir>/var/lib/defoma/x-ttcidfont-conf.d/dirs/TrueType</dir>
```

Fontconfig 应该能直接获得字体信息,“fc-list"能列出你的新字体。另外一个 fontconfig 的特色是，你能把字体放在 ~/.fonts/ 中，而所有字体可设置的程序都能立即访问它们。

如果你在 X 中手动安装了新的一个 TrueTyep 字体，而没有使用 Ubuntu 的软件包，运行

```
# xset fp rehash
```

让 xorg 重新检查目录下面的内容并找到新的字体。

9.4.14 X 中的网页浏览器

Dapper 发行版中包含了下面这些拥有图形处理能力的网页浏览器：

- mozilla Mozilla 浏览器
- mozilla-firefox Mozilla 浏览器变体（独立的）
- epiphany-browser Mozilla 浏览器变体（Gnome）
- galeon 基于 Mozilla 的使用 Gnome UI 的浏览器（新增）
- konqueror KDE 浏览器
- amaya W3C 参考浏览器
- ...

在 Dapper 或 Edgy 中，将会遇到 mozilla 变体浏览器的版本匹配问题，因为这些变体浏览器需要匹配共享库的版本。

安装诸如 mozilla 浏览器的插件，可手工将“*.so”装到 plug-in 目录下，然后重启浏览器。

Plug-in 资源：

- Java plug-in: 从 http://java.sun.com 安装二进制的“J2SE”。
- Flash plug-in: 从 http://www.macromedia.com/software/flashplayer/ 安装二进制的“Macromedia Flash Player 5”
- freewrl: VRML 浏览器和 Netscape plug-in
- ...

9.4.15 X 图形界面下的邮件客户端（MUAs）

在 Sarge 发行版下，有几个客户端软件包有图形显示界面：

- mozilla-thunderbird 独立的邮件客户端
- kmail KDE 邮件客户端
- evolution Novell 里面的 groupware 套件
- ...

9.5 SSH

SSH(Secure SHell)是在 Internet 中建立连接的安全途径。OpenSSH 是一个自由的 SSH 实现软件，它包含 在 Debian 的 ssh 软件包中。

9.5.1 SSH 基础

首次安装 OpenSSH 服务器和客户机。

```
# apt-get update && apt-get install ssh
```

要运行 OpenSSH 服务器，还得屏蔽掉 /etc/ssh/sshd_not_to_be_run。

SSH 有两个验证协议：

- SSH 协议 第 1 版：
    - Potato 发布版仅支持该版协议
    - 可用的验证方法：
        - RSA 验证：基于 RSA 密钥的用户验证
        - Rhosts 验证：基于 .rhosts 的主机验证（不安全，有缺陷）
        - RhostsRSA 验证：.rhosts 验证与 RSA 主机密钥相结合（有缺陷）
        - ChallengeResponse 验证：RSA Challenge-response 验证
        - Password 验证：基于 password 的验证
- SSH 协议 第 2 版
    - Woody 后继版本将以该版协议为主
    - 可用的验证方式：

§ Pubkey 验证：基于公共密钥的用户验证
§ Hostbase 验证：.rhosts 或 /etc/hosts.equiv 验证与公共密钥客户端主机验证相结合（有缺陷）
§ ChallengeResponse 验证：challenge-response 验证
§ Password 验证：基于 password 的验证

如果系统正迁移到 Woody 或使用非 Debian 系统，请注意版本差异。

更多信息请参阅 /usr/share/doc/ssh/README.Debian.gz、ssh(1)、sshd(8)、ssh-agent(1) 和 ssh-keygen(1)。

下面是一些关键的配置文件：

- /etc/ssh/ssh_config：默认的 SSH 客户机。参阅 ssh(1)。其中重要的项目有：
    - Host：作用于所有与该关键字后所列出的主机相匹配的主机，它们须遵守下面（处于本 host 关键字之后下一个 host 关键字之前的内容）所列的各项条款。
    - Protocol：规定所使用的 SSH 协议的版本。默认为“2,1”。
    - PreferredAuthentications：规定 SSH2 客户端验证方式。默认为“hostbased,publickey,keyboard-interactive,password”。
    - PasswordAuthentication：如果想使用密码登录，须确认该选项没有设置成 no。
    - ForwardX11：默认为关闭状态。可使用命令行选项“-X”重载它。
- /etc/ssh/sshd_config：默认的 SSH 服务器。参阅 sshd(8)。其中重要的项目有：
    - ListenAddress：规定 sshd 监听的本地地址。允许多重指定。
    - AllowTcpForwarding：默认为关闭状态。
    - X11Forwarding：默认为关闭状态。
- $HOME/.ssh/authorized_keys：默认公共密钥列表，客户机可使用这些密钥连接本主机的该用户帐号。参阅 ssh-keygen(1)。
- $HOME/.ssh/identity：参阅 ssh-add(1) 和 ssh-agent(1)。

下面的操作将从客户机建立一个 ssh 连接。

```
$ ssh username@hostname.domain.ext
$ ssh -1 username@hostname.domain.ext # Force SSH version 1
$ ssh -1 -o RSAAuthentication=no -l username foo.host
# force password on SSH1
$ ssh -o PreferredAuthentications=password -l username foo.host
# force password on SSH2
```

在用户眼里，ssh 的功能相当于一个更灵巧更安全的 telnet（will not bomb with ^]）。

9.5.2 发送端口 SMTP/POP3 微调

在本地机器上执行下面的命令，可以建立一个连接本地主机 4025 端口和远程服务器 25 端口的管道，以及一个连接本地主机 4110 端口和远程服务器 110 端口的 ssh 连接。

```
# ssh -q -L 4025:remote-server:25 4110:remote-server:110 \
username@remote-server
```

在 Internet 上可使用该方法建立与 SMTP/POP3 服务器的安全连接。记得在远程主机的 /etc/ssh/sshd_config 中设置 AllowTcpForwarding 值为 yes。

9.5.3 用更少的密码建立连接 - RSA

使用 RSAAuthentication（SSH1 协议）或 PubkeyAuthentication（SSH2 协议）可不必记住每个远程系统的连接密码。

在远程系统上，在 /etc/ssh/sshd_config 中分别设置“RSAAuthentication yes”或“PubkeyAuthentication yes"。

然后在本地生成验证密匙，在远程系统上安装公共密钥：

```
$ ssh-keygen          # RSAAuthentication: RSA1 key for SSH1
$ cat .ssh/identity.pub | ssh user1@remote \
"cat - >>.ssh/authorized_keys"
...
$ ssh-keygen -t rsa   # PubkeyAuthentication: RSA key for SSH2
$ cat .ssh/id_rsa.pub | ssh user1@remote \
"cat - >>.ssh/authorized_keys"
...
$ ssh-keygen -t dsa   # PubkeyAuthentication: DSA key for SSH2
$ cat .ssh/id_dsa.pub | ssh user1@remote \
"cat - >>.ssh/authorized_keys"
```

今后可用“ssh-keygen -p”来改密码。最后记得检查一下设置，可做个连接测试，如遇问题，执行“ssh -v"。

你可以通过在 authorized_keys 里添加选项来限制主机及运行指定的命令。详情参阅 sshd(8)。

注意 SSH2 有 HostbasedAuthentication，要使它工作，必须同时在服务器端的 /etc/ssh/sshd_config 文件中和客户机端的 /etc/ssh/ssh_config 或 $HOME/.ssh/config 文件中设置 HostbasedAuthentication 为 yes。

9.5.4 处理外来的 SSH 客户端

下面是其它一些非类 Unix 平台的免费 SSH 客户端。

Windows:: puTTY (GPL)

Windows (cygwin):: SSH in cygwin (GPL)

Macintosh Classic:: macSSH (GPL) [注意 Mac OS X 包含 OpenSSH;在终端应用程序中使用 ssh]

参阅 SourceForge.net 的站点文档，“6. CVS Instructions”。

9.5.5 设置 ssh-agent

使用 passphrase 来保护 SSH 认证密匙会更安全，如果还没有设置，可使用 ssh-keygen -p 来设置。

用更少的密码建立连接 - RSA，第 9.5.3 节中描述了如何使用一个基于密码的远程主机连接，将公共密钥（例如~/.ssh/id_rsa.pub）放入远程主机的 ~/.ssh/authorized_keys。

```
$ ssh-agent bash # 或者用 zsh/tcsh/pdksh 这些程序代替。
$ ssh-add ~/.ssh/id_rsa
Enter passphrase for /home/osamu/.ssh/id_rsa:
Identity added: /home/osamu/.ssh/id_rsa (/home/osamu/.ssh/id_rsa)
$ scp foo user@remote.host:foo
... no passphrase needed from here on :-)
$^D
... terminating ssh-agent session
```

对于 X 服务器，普通 Ubuntu 启动脚本会将 ssh-agent 作为一个父进程执行。所以只需执行一次 ssh-add 即可。

详情参阅 ssh-agent(1) 和 ssh-add(1)。

9.5.6 SSH 问题处理

如果遇到问题，检查一下配置文件的访问权限，并使用“-v”选项运行 ssh。

如果是 root 身份，遇到连接防火墙出错的情况，可使用“-P”选项；它规定 ssh 使用服务器的 1 - 1023 以外的端口。

如果与远程站点的 ssh 连接突然停止工作，很可能是因为系统管理员修补系统造成的，host_key 在系统维护过程中被更改。在查明了事情真象并确定并不是有人试图冒充远程主机非法入侵之后，从本地机器的 $HOME/.ssh/known_hosts 中删除 host_key 项目就可以恢复连接了。

9.6 邮件

邮件系统配置分为三类：

- 邮件传输代理（MTA）：exim4, exim, postfix, sendmail, qmail, ssmtp, nullmailer, ...
- 邮件工具：procmail, fetchmail, mailx, crm114, ...
- 邮件用户代理（MUA）：mutt, emacs+gnus,

9.6.1 邮件传输代理（MTAs）

对于全功能的 MTA 来说，请使用 exim4。 参考资料：

- 针对软件包 exim4 的 exim4-doc 和 exim4-doc-html。
- http://www.exim.org/

如果你对安全性有高要求的话，唯一一个可替代的 MTA 是 postfix。Ubuntu 软件包还提供了 sendmail 和 qmail，但并不推荐你使用它们。

有时并不需要 MTA 的所有功能，如在一个卫星系统中的一台笔记本电脑，可以考虑下列几种轻量级软件包：

- ssmtp：需要 SMTP 连接并支持别名功能，或者
- nullmailer：可以存信但不支持别名。

此刻，我发现 exim 对于我作为个人工作站的笔记本电脑再合适不过了。

如果要安装这些软件包必须先删除 exim，它们有冲突：

```
# dpkg -P --force-depends exim
# apt-get install nullmailer         # or ssmtp
```

9.6.1.1 Smarthost

如果你在一台通过用户级别的服务连接的机器上运行 exim4 或 exim，请确保在发信的时候通过 ISP 或者其他人提供的 smarthost。 [45] 这样做有许多好处：

- ISP 的 smarthost 通常有更可靠的连接，可以确保 SMTP 重发。
- 避免使用**动态 IP** 发送信件，这类信件会被 dial-up spam 列表过滤的。
- 节省寄出多个收信人的邮件的带宽。

唯一可能的坏处是：

- 你 ISP 的 SMTP 出问题时的紧急处理。
- 用于学习目的的实验。
- 你的主机是专业的主机服务器。

9.6.1.2 Exim 的基础设置

为了将 exim 作为 MTA，可按下列步骤进行配置：

```
/etc/exim/exim.conf      使用“eximconfig”创建及编辑（exim）
/etc/exim4/*         使用“dpkg-reconfigure exim4-config”创建及编辑（exim4）
/etc/inetd.conf          注释掉 smtp，将 exim 作为 daemon 运行
/etc/email-addresses     添加伪来源地址列表
```

检查邮件过滤器可使用 exim4 搭配 -brw, -bf, -bF, -bV, ... 等等

9.6.1.3 在 Exim 下设置一个收集不存在的邮件地址的容器

在 /etc/exim/exim.conf 文件（Woody 或后继版本）的 DIRECTORS 部分的末尾（localuser:director 之后）添加一个 catch-all director，将所有前面的 director 无

法解析的地址收集到一起(per Miquel van Smoorenburg)：

```
catchall:
driver === smartuser
new_address === webmaster@mydomain.com
```

如果要为每个虚拟域指定更精细的处理方法或其它什么的，可在/etc/exim/exim.conf 末尾添加（我没仔细测试过）：

```
*****@yourdomain.com ${lookup{$1}lsearch*{/etc/email-addresses} \
{$value}fail} T
```

接着在/etc/email-addresses 中加上一行“*"。

9.6.1.4 在 Exim 下设置寄出邮件的地址重写

可用 exim 为发出的邮件指定特定的“From:”信头，在 /etc/exim/exim.conf 文件的末尾编辑：

```
*****@host1.something.dyndns.org \
"${if eq {${lookup{$1}lsearch{/etc/passwd}{1}{0

{1} \

{$0}{$1@something.dyndns.org}}" frFs </nowiki></pre>
```

该语句将作用于所有符合*@host1.something.dyndns.org 的邮件。

- 在/etc/password 中搜索，以确定 local part（$1）是否为本地用户。
- 如果是本地用户，它将用第一个域（$0）中的内容重写地址
- 如果不是本地用户，重写域部分。

9.6.1.5 在 Exim 中设置 SMTP 认证

某些 SMTP 服务如 yahoo.com 需要 SMTP 认证。可用下面的方法配置 /etc/exim/exim.conf：

```
remote_smtp:
driver === smtp
authenticate_hosts === smtp.mail.yahoo.com
...

smarthost:
driver === domainlist
transport === remote_smtp
route_list === "* smtp.mail.yahoo.com bydns_a"
...

plain:
driver === plaintext
```

```
public_name === PLAIN
client_send === "<sup>cmatheson3</sup>this_is_my_password"
```

别忘了最后一行的双引号。

9.6.2 收取邮件 - Fetchmail

fetchmail 以 daemon 方式运行，用 ISP 提供的 POP3 帐号将邮件收到本地邮件系统。配置：

```
/etc/init.d/fetchmail
/etc/rc?.d/???fetchmail run update-rc.d fetchmail default priority 30
/etc/fetchmailrc        configuration file (chown 600, owned by fetchmail)
```

在 Potato 中，有关如何在 init.d 脚本中配置 fetchmail 以 daemon 方式运行的信息，十分混乱，Woody 解决了这个问题。参阅 example scripts 中 /etc/init.d/fetchmail 和 /etc/fetchmailrc 样例文件。

如果你的邮件信头被 ISP 的邮件工具以^M 污染，可在 $HOME/.fetchmailrc 中添加“stripcr”选项：

```
options fetchall no keep stripcr
```

9.6.3 处理邮件 - Procmail

procmail 是一个本地邮件分发过滤程序。使用时，需要为每个使用它的用户创建 $HOME/.procmailrc，样例：_procmailrc。

9.6.4 用 crm114 处理垃圾邮件

crm114 软件包提供了 /usr/share/crm114/mailfilter.crm 脚本。该脚本是用 CRM114 撰写的，提供了非常有效的垃圾信件过滤器，并可以通过喂食垃圾邮件来调节它的处理能力。

CRM114 是一种专门为写过滤器设计的小语言，可以把它当做拥有超能力的 grep 版本。参阅 crm(1)。

9.6.5 阅读邮件 - Mutt

用 mutt 做用户邮件代理（MUA）与 vim 结合使用。使用 ~/.muttrc 进行自定义；例如：

```
# use visual mode and "gq" to reformat quotes
set editor="vim -c 'set tw=72 et ft=mail'"
#
# header weeding taken from the manual (Sven's Draconian header weeding)
#
ignore *
unignore from: date subject to cc
unignore user-agent x-mailer
```

```
hdr_order from subject to cc date user-agent x-mailer
auto_view application/msword
....
```

在 /etc/mailcap 或 $HOME/.mailcap 中添加下列内容，就能显示 HTML 邮件和内嵌的 MS Word 附件：

```
text/html; lynx -force_html %s; needsterminal;
application/msword; /usr/bin/antiword '%s'; copiousoutput;
description="Microsoft Word Text"; nametemplate=%s.doc
```

9.7 本地化（localization）

Ubuntu 是国际化的操作系统，它所支持的语言和地区惯例的数目正在不断增加。接下来的部分列出了当前 Ubuntu 对各种差异形式的支持，接着再讨论**本地化**，该过程负责定制你的工作环境，根据你所选的语言确定当前系统的输入输出方式，并按照你所在地区的惯例转化日期、数字、货币格式以及系统中其它相关方面。

9.7.1 本地化基础

定制系统的本地化和国家语言支持包括以下几个方面。

9.7.1.1 键盘本地化

Ubuntu 发布版中包含了二十多种键盘布局方案。重新配置键盘可使用：

- dpkg-reconfigure --priority=low console-data # console
- dpkg-reconfigure --priority=low xserver-xfree86 # XF4
- dpkg-reconfigure --priority=low xserver-common-v3 # XF3

9.7.1.2 资料文件本地化

绝大多数 Ubuntu 软件包都能使用 non-US-ASCII 字符，它们通过 glibc 中的 **locale** 技术，用 LC_CTYPE 环境变量来操作这些字符。

- 纯 8-bit 字符：应用于所有程序中
- 其它拉丁字符集（例如：ISO-8859-1 或 ISO-8859-2）：应用于绝大多数程序中
- 多字节语言如中文、日文或韩文：应用于较新的应用程序中

9.7.1.3 显示本地化

X 可以显示包括 UTF-8 在内的许多编码并支持所有的字体。列表中包含了所有的 8-bit 字体和 16-bit 字体诸如中文、日文或韩文。 其他 X 输入法, 第 9.7.10 节 机制支持多字节输入法。参阅 多语言的 X 窗口系统范例, 第 9.7.9 节和 支持 UTF-8 的 X 终端机, 第 9.7.12 节。

kon2 软件包可实现在 (S)VGA 图形化控制台中显示日文 EUC 编码。另一个替代品是 jfbterm，它也使用 FB 控制台。在控制台环境里，必须由应用程序来提供对日文输入的支持。所以要为 Emacs 加装 egg 软件包，可使用日文化的 jvim 软件包作为 Vim 环境。

安装非 Unicode 字体到 X 就能在 X 下显示任何编码的文件。所以不用太担心字体的编码问题。

9.7.1.4 信息和文档的本地化

许多在 Ubuntu 系统中显示的文本信息和文档被翻译成了各种译本，如出错信息、标准程序输出、菜单以及帮助页面。当前 Ubuntu 支持德语、西班牙语、芬兰语、法语、匈牙 利语、意大利语、日语、韩语、波兰语、葡萄牙语、汉语以及俄语帮助页面，可通过安装 manpages-LANG 软件包实现这些支持（此处 LANG 代表双位的 ISO 国家代码。使用 apt-cachesearch manpages-|less 获得可用的 Unix 帮助页面列表。）

要访问 NLS 帮助页面，用户必须将环境变量 LC_MESSAGES 设置成相应的字串。例如，要访问意大利语的帮助页面，需要将 LC_MESSAGES 设置成 it，这时 man 程序会在 /usr/share/man/it/ 目录下搜索帮助页面。

9.7.2 Locales

Ubuntu 支持 **locale** 技术。locale 机制允许程序按照该地区惯例来提供输出和其它特殊功能如字符集、日期时间显示格式，货币符号等等。该机制使用环境变量来确定其相关的行为。例如，假设你同时在系统上安装了美式英语和法语 locales，许多程序的出错信息都以双语显示：

```
$ LANG="en_US" cat foo
cat: foo: No such file or directory
$ LANG="de_DE" cat foo
cat: foo: Datei oder Verzeichnis nicht gefunden
```

Glibc 以函数库的形式向程序提供该功能的支持。参阅 locale(7)。

9.7.3 Locales 简介

完整的 locale 描述包括三个部分：xx_YY.ZZZZ。

- **xx**: ISO 639 个语言代码（小写）
- **YY**: ISO 3166 个国家代码（大写）
- **ZZZZ**: 编码集，例如，字符集或编码标志。

关于语言代码和国家代码，请参阅 info gettext 中的相关描述。

请注意，为了完成跨平台的兼容性，这个编码集部分可能被“内部标准化”。移除了所有 -，把所有字符都转化成小写的。典型的编码集：

- **UTF-8**: 适合所有区域的 Unicode 码，通常是 1-3 个八进制数（新的事实上的标准）
- **ISO-8859-1**: western Europe (de facto old standard)
- **ISO-8859-2**: eastern Europe (Bosnian, Croatian, Czech, Hungarian, Polish, Romanian, Serbian, Slovak, Slovenian)
- **ISO-8859-3**: Maltese
- **ISO-8859-5**: Macedonian, Serbian
- **ISO-8859-6**: Arabic
- **ISO-8859-7**: Greek
- **ISO-8859-8**: Hebrew
- **ISO-8859-9**: Turkish

- **ISO-8859-11**: Thai (=TIS-620)
- **ISO-8859-13**: Latvian, Lithuanian, Maori
- **ISO-8859-14**: Welsh
- **ISO-8859-15**: western Europe with euro
- **KOI8-R**: Russian
- **KOI8-U**: Ukrainian
- **CP1250**: Czech, Hungarian, Polish (MS Windows origin)
- **CP1251**: Bulgarian, Byelorussian (MS Windows origin)
- **eucJP**: Unix style Japanese (=ujis)
- **eucKR**: Unix style Korean
- **GB2312**: Unix style Simplified Chinese (=GB, ==eucCN) for zh_CN
- **Big5**: Traditional Chinese for zh_TW
- **sjis**: Microsoft style Japanese (Shift-JIS)

询问基本编码系统术语的意思：

- **ASCII**: 7 bits (0-0x7f)
- **ISO-8859-?**: 8 bits (0-0xff)
- **ISO-10646-1**: Universal Character Set (UCS) (31 bits, 0-0x7fffffff)
- **UCS-2**: First 16 bit of UCS as straight 2 Octets (Unicode: 0-0xffff)
- **UCS-4**: UCS as straight 4 Octets (UCS: 0-0x7fffffff)
- **UTF-8**: UCS encoded in 1-6 Octets (mostly in 3 Octets)
- **ISO-2022**: 7 bits (0-0xff) with the escape sequence. ISO-2022-JP is the most popular encoding for the Japanese e-mail.
- **EUC**: 8 bits + 16 bits combination (0-0xff), Unix style
- **Shift-JIS**: 8 bits + 16 bits combination (0-0xff), Microsoft style.

ISO-8859-?、EUC、ISO-10646-1、UCS-2、UCS-4 和 UTF-8 对于 7 位的字符使用和 ASCII 相同的编码。 EUC or Shift-JIS uses high-bit characters (0x80-0xff) to indicate that part of encoding is 16 bit. UTF-8 also uses high-bit characters (0x80-0xff) to indicate non 7 bit character sequence bytes and this is the most sane encoding system to handle non-ASCII characters.

Please note the byte order difference of Unicode implementation:

- **Standard UCS-2, UCS-4**: big endian
- **Microsoft UCS-2, UCS-4**: little endian for ix86 (machine-dependent)

See 使用 recode 转化文本文件, 第 8.6.12 节 for conversion between various character sets. For more see Introduction to i18n.

9.7.4 激活 locale 支持

Ubuntu 并**不**在系统中编译所有可用的 locales,检查 /usr/lib/locale 确定哪个 locales（除了默认的“C”）已在系统上编译安装。如果所需的 locale 并不在其中，有两个解决办法：

- 编辑 /etc/locale.gen 添加需要的 locale，然后以 root 身份运行 locale-gen 编译它。参阅 locale-gen(8)以及该帮助页面中“SEE ALSO”一节所列的命令。
- 运行 dpkg-reconfigure locales 可以重新配置 locales 软件包。如果还没有安装 locales,locales 安装程序会调出 debconf 界面让你选择所需的 locales 并编译

相关数据库。

9.7.5 激活特定 locale

相关的环境变量按如下次序将特定 locale 值赋给程序：

- LANGUAGE：该环境变量由一个用冒号分隔、以优先级排序的地区名称列表组成。仅当 POSIX 地区值与“C”地区值相异时才使用到它[在 Woody 中；在 Potato 版本中通常其优先级高于 POSIX locale]。(GNU 扩展名)
- LC_ALL：如果为非空值，其值将作用于所有 locale 项目。(POSIX.1) 通常为“ ”（空值）。
- LC_*：如果为非空值，其值将作用于相应的 locale 项目。(POSIX.1) 通常为“C"。

LC_*变量有：

  - LC_CTYPE：字符分类和环境转换
  - LC_COLLATE：校正命令
  - LC_TIME：时间显示格式
  - LC_NUMERIC：非货币型数字格式
  - LC_MONETARY：货币符号
  - LC_MESSAGES：常规信息、诊断消息和交互响应信息的格式
  - LC_PAPER：纸张尺寸
  - LC_NAME：姓名格式
  - LC_ADDRESS：地址格式和地区信息
  - LC_TELEPHONE：电话号码格式
  - LC_MEASUREMENT：度量单位（公制或其它）
  - LC_IDENTIFICATION：有关地区信息的元数据

- LANG：如果为非空值且 LC_ALL 也没有定义，则该值作用于所有没有定义的 LC_* 地区项目。(POSIX.1) 通常为“C"。

注意，有些应用程序（例如 Netscape 4）忽略 LC_*设置。

locale 程序可显示当前激活的地区设置和可用的 locale；参阅 locale(1)。(注意：locale -a 将列出系统已知的所有的 locales；这并不代表它们都已在系统中编译了！参阅 激活 locale 支持, 第 9.7.4 节。)

9.7.6 ISO 8601 日期格式

名为 en_DK “丹麦英语的” locale 提供了对国际标准日期格式 yyyy-mm-dd（ISO 8601 日期格式）的支持（听起来有点搞笑:-))。它仅工作于 ls 的控制台屏幕。

9.7.7 US（ISO-8859-1）例子

将下列语句添加到 ~/.bash_profile：

```
LC_CTYPE=en_US.ISO-8859-1
export LC_CTYPE
```

9.7.8 带 Euro 符号的 France（ISO-8859-15）的例子

将下列语句添加到 ~/.bash_profile：

```
LANG=fr_FR@euro
export LANG
LC_CTYPE=fr_FR@euro
export LC_CTYPE
```

按 键盘本地化，第 9.7.1.1 节中描述的方法，将键盘设置成 French "AZERTY"。安装 manpages-fr 包添加法语帮助页面。US 中的 Right-Alt 键在 Europe 中称为 Alt-Gr，它与其它键组成的组合键可用于输出大量特殊字符，例如 Alt-Gr+E 可以输出欧元符号。

可使用类似的方法配置绝大多数西欧语言环境。

参阅 Debian Euro HOWTO 了解有关对新欧洲货币方面的支持，有关对法语环境的支持请参阅 Utiliser et configurer Debian pour le français。

9.7.9 多语言的 X 窗口系统范例

让我们来创建一个多语言 X 窗口系统，能在不同的控制台中搭配 EUC、UTF-8 和 ISO-8859-1 来同时支持日语、英语、德语和法语。

我会向你展示用 Ubuntu menu 系统进行的自定义设置 Ubuntu menu 系统的详细信息参阅 /usr/share/doc/menu/html/index.html。在例子中，我也创建了 mozilla 网络浏览器的快捷键。 [46]

- 使用 本地化（localization），第 9.7 节中描述的方法添加对日文 ja_JP.eucJP locale 和其他需要的 locale 的支持。
- 安装 Kana-to-Kanji 转换系统和字典（针对日语）：
    - canna - 本地服务器 (free bear license)，或
    - freewnn-jserver - 加入网络功能的服务器 (Public Domain)
- 安装日文输入系统（针对日语）：
    - kinput2-canna - X 下或
    - kinput2-canna-wnn - X 下和
    - egg - 在控制台下能搭配 Emacsen 直接输入日文（额外）
- 安装日文兼容终端机：
    - xterm - X（支持 ISO-8859-1 和 UTF-8），
    - kterm — X （支持日文 EUC），和
    - mlterm — X （支持多语言）
- 添加全部日文字库包。（针对所有语言）
- 创建~/.xsession，用户可通过配置它来指定 X 环境，如 自定义 X 会话，第 9.4.5.1 节中描述的（针对所有语言）：

```
#!/bin/sh
# This makes X work when I su to root.
```

```
if [ -z "$XAUTHORITY" ]; then
XAUTHORITY=$HOME/.Xauthority
export XAUTHORITY
fi

# Set specific environment through debian menu system.
# Reset locale
unset LC_CTYPE LC_NUMERIC LC_TIME LC_COLLATE LC_MONETARY LC_MESSAGES
unset LC_PAPER LC_NAME LC_ADDRESS LC_TELEPHONE LC_MEASUREMENT
unset LC_IDENTIFICATION LC_ALL LANG LANGUAGE PAGER
# set locale default in X
LANG=C
# export locale
export LC_CTYPE LC_NUMERIC LC_TIME LC_COLLATE LC_MONETARY LC_MESSAGES
export LC_PAPER LC_NAME LC_ADDRESS LC_TELEPHONE LC_MEASUREMENT
export LC_IDENTIFICATION LC_ALL LANG LANGUAGE PAGER
###
# activate input method for Japanese with kinput2
kinput2 &
XMODIFIERS="@im=kinput2"
export XMODIFIERS
# How about blackbox window manager (lightweight)
exec blackbox
#exec xfwm
#exec wmaker
```

- 在 ~/.bash_profile 中为 Linux 控制台设置 locale（针对所有语言）。
- 删除 ~/.bashrc 中的 locale 设置，如果其存在的话（针对所有语言）。
- 在 /etc/menu/ 中建立新文件（针对所有语言）。
    - /etc/menu/xterm-local:（在菜单中加入新项目）

```
?package(xterm):\
needs=x11\
section=XShells\
longtitle="XTerm: terminal emulator (en_US.ISO-8859-1)"\
title="XTerm (en_US.ISO-8859-1)"\
command="sh -c 'LC_ALL=en_US.ISO-8859-1 xterm'"
?package(xterm):\
needs=x11\
section=XShells\
longtitle="XTerm: terminal emulator (de_DE.ISO-8859-1)"\
title="XTerm (de_DE.ISO-8859-1)"\
command="sh -c 'LC_ALL=de_DE.ISO-8859-1 xterm -T xterm-de'"
?package(xterm):\
needs=x11\
section=XShells\
longtitle="XTerm: terminal emulator for X with Unicode support (Japanese)"\
title="UXTerm (ja_JP.UTF-8)"\
command="sh -c 'LC_ALL=ja_JP.UTF-8 uxterm'"
```

    - /etc/menu/kterm:（覆盖系统默认的）

```
?package(kterm):\
needs="x11"\
section="XShells"\
command="sh -c 'LC_ALL=ja_JP.eucJP PAGER=w3m /usr/X11R6/bin/kterm -xim'" \
title="Kanji Terminal"
?package(kterm):\
needs="x11"\
section="XShells"\
command="sh -c 'LANG=ja_JP.eucJP \
LC_MESSAGES=en_US.ISO-8859-1 PAGER=w3m /usr/X11R6/bin/kterm -xim'" \
title="Kanji Terminal (bilingal)"
```

    - /etc/menu/mozilla-local:（添加新的快捷键）

```
?package(mozilla-browser):needs="x11" section="/" \
title=" Mozilla Navigator" command="mozilla-1.5" hints="Web browsers" \
icon=/usr/share/pixmaps/mozilla.xpm
```

    - 以 root 帐号运行 update-menus。

- 在 ~/.muttrc 中加入下列内容（针对日语）：

```
# UTF-8 support is not popular in popular Japanese EMACS environment
# 7 bit encoding of iso-2022-jp is easier for everyone
# default encoding order === us-ascii --> iso-8859-1 --> iso-2022-jp
set send_charset="us-ascii:iso-8859-1:iso-2022-jp"
set allow_8bit=no
```

- 为 X 应用程序激活 XIM kinput2（针对日语）：
    - 在 X 资源文件 ~/.Xresources（不知何故，看上去 Debian 好象在自动配置它）添加 *inputMethod: kinput2 和 KTerm*VT100*OpenIm:true。
    - 某些应用程序（如 mlterm）也允许设置 *inputMethod: 和其它运行时的动态信息（在 mlterm 中按下 **Ctrl-MouseButton-3**）。
- 用 startx 或任何显示管理器（xdm、gdm、kdm、wdm...）启动 X（针对所有语言）
- 打开日文兼容应用程序：VIM6、(x)emacs21、mc-4.5、mutt-1.4...（针对日语）。（通常 Emacs 是最流行的平台，尽管我不怎么用它。）
- 按“**Shift+Space**”可切换日文输入法。（针对日语）。
- 在本地化了的控制台中阅读本地化手册（针对所有语言）。

关于其他 CJK 语言的支持，参阅下面的章节和 SuSE pages for CJK。

9.7.10 其他 X 输入法

还有许多其他可用的 X 输入法支援软件包：

Language	LC_CTYPE	XIM server	XMODIFIERS	Start key
Japanese	ja_JP*	kinput2	"@im=kinput2"	Shift-Space
Korean	ko_KR*	ami	"@im=Ami"	Shift-Space
Chinese(T)	zh_TW.Big5	xcin	"@im=xcin-zh_TW.big5"	Ctrl-Space
Chinese(S)	zh_CN.GB2312	xcin	"@im=xcin-zh_CN.GB2312"	Ctrl-Space

日语输入法 kinput2 由软件包 kinput2-canna-wnn、kinput2-canna 和 kinput2-wnn 提供。输入法还需要例如 canna 和 freewnn-jserver 这样的字典服务器才能使用。

9.7.11 X 终端模拟机

还有许多 X 终端机支持 8-bit 编码，安装相应的字体就能正常显示：

- xterm - X 终端机
- gnome-terminal - Gnome 的 xterm
- konsole - KDE 的 xterm
- rxvt - VT102 终端 (lighter)
- aterm - Afterstep WM 的 VT102 终端
- eterm - Enlightment WM 的 VT102 终端
- wterm - WindowMaker WM 的 VT102 终端

在 UTF-8 编码的情况下，xterm 提供多编码的 X 终端的支持（ 支持 UTF-8 的 X 终端机，第 9.7.12 节）。其他传统编码的支持还在开发中（2003 年时）。下面的软件包都提供了传统编码的支持：

- aterm-ml - Multi-lingual
- kterm - Multi-lingual (Japanese, ...)
- rxvt-ml - Multi-lingual
- wterm-ml - Multi-lingual
- cxterm-big5 - Chinese (Trad., Big5)
- cxterm-gb - Chinese (Simp., GB)
- cxterm-ks - Chinese (KS)
- cxterm-jis - Japanese
- hanterm-classic - Korean (Hangul)
- hanterm-xf - Korean (Hangul)
- hztty - Chinese (GB, Big5, zW/HZ)

对于 kterm（和其他终端），你可以通过在菜单上按下 Ctrl+鼠标中键来激活 XIM。

9.7.12 支持 UTF-8 的 X 终端机

xorg 的软件包 xterm 中的 X 终端机 uxterm 提供了对 UTF-8 的支持。它支持所有语言。它是 xterm(1) 程序的一个 wrapper，使得启动的 xterm 使用了“UXTerm”的 X 资源集。

例如，为了提供英文、俄文、日文、中文和韩文大字体的完美显示，需要在安装相应的字体后在 ~/.Xresources 中添加下列内容：

```
! set large font
```

```
UXTerm*font: -misc-fixed-medium-r-normal-*-18-120-100-100-c-90-iso10646-1
! Use XIM for Japanese
*****inputMethod: kinput2
```

然后按照 X 资源，第 9.4.10 节 中描述的，运行 xrdb -merge ~/.Xresources 来更新 X 资源。

虽然大部分的终端程序，例如 vim、mutt 和 emacs 都已经能兼容 UTF-8 了（Woody-Sarge）。但是还有不少 mc 这样的程序不兼容 UTF-8，只支持 8-bit 编码。如果你编辑未知或者混合编码文件的 7-bit ASCII 部分，使用 locale 不敏感的支持 8-bit 的编辑器比较安全。

参阅 The Unicode HOWTO。

9.7.13 FB 控制台下 UTF-8 的例子

在 debian-installer 中 bterm 提供了在 FB 控制台对 UTF-8 的支持。

9.7.14 超越 locales

当你第一次在系统上安装“**国家语言环境**”（national language environment）时，请注意使用 tasksel 或 aptitude 查看一下，在选择相关语言环境任务项时都有哪些软件包被选上了，这些选择信息非常有用特别是进行多语言设置时。如果遇到某些关联包与运行良好的系统上的某 些软件发生冲突，就不要安装那些引起冲突的包。由于新装的软件比原来的软件具有更高的优先级，所以必须用 update-alternative 结合有关命令让系统恢复到原来状态。

大部分较新的使用 glibc2.2 的程序都已支持国际化了。所以不必再为诸如基于 VIM 的 jvim 等程序指定 loclae，因为 X 下的 vim 6.0 版已提供了该功能。事实上，比起另一个版本它显得有点粗糙，jvim 有个版本直接将日文输入法(canna)支持编译进去了，而且还集成了大量成熟的日文特性，很值得你期待:-)

有时为了获得更好的工作环境，仅通过 locale 来配置程序是不够的。language-env 软件包和 set-language-env 命令可以大大简化你的工作。

亦可参阅有关国际化的文档 Introduction to i18n，虽然其目标读者为开发人员但对系统管理员也十分有用。

9.8 多语言化（Multilingualization, m17n）

由 language-env 等软件包所实现的 本地化（localization），第 9.7 节是以单一语言的本地化为目标的。这些软件包使用传统的文本编码（encoding）方式。你在这样的环境下不能在文本中混合使用法语和日语，因为它们分 别使用 ISO-8859-1 和 EUC-JP 的编码方式，而这两种编码是不兼容的。

如果你选择在一个可用的 UTF-8 区域设置（locale）下启动 GNOME 或 KDE，你就可以得到一个多语言化的 UTF-8 桌面环境。在这样的环境下，只要你用的软件是符合 UTF-8 标准的，你就可以自由地混合使用英语，汉语，俄语，日语等各种文字。

m17n-env 是一个用来帮助建立多语言化环境的程序。你可以在 root 或普通用户下运行 set-m17n-env 命令来设置多语言化的 UTF-8 环境。 [50]

在这样的环境下，最好使用基于 scim 的多语言化输入法。scim 所提供的各种输入法可以一起用 Ctrl-Space 开启或关闭。具体的输入法引擎可以通过点击 SCIM 面板来选择和切换。

你仍然可以通过 m17n-env 创建的自定制区域设置终端而方便地使用传统的编码环境。这在你需要编辑旧的 EUC-JP 或 ISO-8859-1 编码文件时十分有用。

第 10 章 - 网络设置

这一章重点在于 Ubuntu 的网络管理。请阅读 Net-HOWTO 来了解一般 GNU/Linux 的网络设置。

为了让 Ubuntu 主机能够访问 Internet，它的网络接口需要被正确的设置。

首先要确认内核支持这个设备，例如以太网卡、无线网卡(Wi-Fi)和调制解调器。为了获得这些支持，你可能需要重新编译内核或者给内核增加模块，如 Ubuntu 下的 Linux 内核，第 7 章中描述的。

下面说明如何设置网络设备。

10.1 IP 网络设置基础

一个 Ubuntu 主机可能有很多有不同 Internet 协议(IP)地址的网络接口。接口可能有很多种，如：

- Loopback：lo
- Ethernet：eth0、eth1 ……
- Wi-Fi：wlan0、wlan1、wifi0 ……
- Token Ring：tr0、tr1 ……
- PPP：ppp0、ppp1 ……

其他可用的网络设备还有很多，包括 SLIP、PLIP（串行和并行 IP)、控制某种网络接口流量的“shaper”设备、帧中继、AX.25、X.25、ARCnet 和 LocalTalk。

每个直接连接到 Internet(或任何基于 IP 的网络)的网络接口都用唯一的 32 位的 IP 地址来识别。 IP 地址可分为网络地址和主机地址两个部分。如果你拿到一个 IP 地址，把网络地址部分全部设为 1，而主机地址部分全部设为 0，则你将得到这个网络的子网掩码。

传统意义上，IP 网络按照网络地址的长度分为 8、16、24 位三个组别。这个系统缺乏灵活性，浪费了很多 IP 地址，所以现在的 IPv4 网络是由可变长度的网络号来分配的。

```
IP addresses                    net mask      length
Class A   1.0.0.0     - 126.255.255.255  255.0.0.0     ===  /8
Class B   128.0.0.0   - 191.255.255.255  255.255.0.0   === /16
Class C   192.0.0.0   - 223.255.255.255  255.255.255.0 === /24
```

IP 地址不在这个范围内的被用作特殊目的。

每一个组别中都有保留给本地网络(LANs)使用的地址范围。这些地址不会和 Internet 上的发生冲突。(同理，如果主机被分配到这类地址的话，这些主机就不能直接访问 Internet，需要通过一个作为代理的网关或网络地址转换服务(NAT)才能访问 Internet。) 这些地址范围在下表中列出，包含每个组别中这些地址范围的数目。

network	addresses		length	how many
Class A	10.x.x.x		/8	1
Class B	172.16.x.x -	172.31.x.x	/16	16
Class C	192.168.0.x -	192.168.255.x	/24	256

IP 网络中 IP 地址的第一个值就是网络本身，最后一个值是该网络的广播地址。 其余所有的 IP 地址都可以分配给网络中的主机。通常 IP 地址的第一个和最后一个都会留给该网络的 Internet 网关。

路由表包含了关于内核如何把 IP 包发送到它们目的地的信息。这儿有一个位于本地网络(LAN)，IP 地址为 192.168.50.x/24 的 Debian 主机的路由表。另一台主机 192.168.50.1(也在 LAN 中)是公司网络 172.20.x.x/16 的路由器，主机 192.168.50.254(也在 LAN 中)是负责访问 Internet 的路由器。

```
# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
127.0.0.0       *               255.0.0.0       U     0      0        2 lo
192.168.50.0    *               255.255.255.0   U     0      0      137 eth0
172.20.0.0      192.168.50.1    255.255.0.0     UG    1      0        7 eth0
default         192.168.50.254  0.0.0.0         UG    1      0       36 eth0
```

- 第一行说明传送目的地是 127.x.x.x 的话，则会通过 lo 回环网络接口来路由。
- 第二行说明传送目的地是 LAN 的话，则会通过 eth0 来路由。
- 第三行说明传送目的地是公司网络的话，则会通过 eth0 来路由，最后发送到网关 192.168.50.1。
- 第四行说明传送目的地是 Internet 的话，则会通过 etho 来路由，最后发送到网关 192.168.50.254。

路由表中的 IP 地址也可以用名称表示，这些名称从 /etc/networks 或通过 resolver (C Library) 来获得。

除了路由之外，内核能实现网络地址转换(NAT)、流量控制和包过滤。

参阅 Net-HOWTO 和 other networking HOWTOs 来了解背后运行的原理。

10.2 底层网络设置

GNU/Linux 上传统的底层网络设置工具是 ifconfig 和 route，它们在 net-tools 这个软件包中。目前这些工具被软将包 iproute 中的 ip 代替。ip 可以在 Linux 2.2 或更新的内核上运行，有着比老的工具更好的兼容性。然而，这些传统的设置工具还是能用的而且大家也更加熟悉。

10.2.1 底层网络设置 - ifconfig 和 route

下面演示如何把网络接口 eth0 的 IP 地址从 192.168.0.3 改为 192.168.0.111；设置 eth0 的路由，通过 192.168.0.1 访问 10.0.0.0 这个网络。 执行 ifconfi 和 route 时不带网络接口参数，则显示所有网络接口和路由的现状。

```
# ifconfig
eth0 Link encap:Ethernet  HWaddr 08:00:46:7A:02:B0
inet addr:192.168.0.3  Bcast:192.168.255.255  Mask:255.255.0.0
UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
RX packets:23363 errors:0 dropped:0 overruns:0 frame:0
TX packets:21798 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:100
RX bytes:13479541 (12.8 MiB)  TX bytes:20262643 (19.3 MiB)
Interrupt:9

lo   Link encap:Local Loopback
inet addr:127.0.0.1  Mask:255.0.0.0
UP LOOPBACK RUNNING  MTU:16436  Metric:1
RX packets:230172 errors:0 dropped:0 overruns:0 frame:0
TX packets:230172 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:22685256 (21.6 MiB)  TX bytes:22685256 (21.6 MiB)
# route
Kernel IP routing table
Destination  Gateway      Genmask          Flags Metric Ref Use Iface
192.168.0.0  *            255.255.0.0      U     0      0     0 eth0
default      192.168.0.1  255.255.255.255  UG    0      0     0 eth0
```

首先我们关闭网络接口。

```
# ifconfig eth0 inet down
# ifconfig
lo   Link encap:Local Loopback
... （没有 eth0 这个条目了）
# route
... （没有路由表了）
```

接下来我们启动 eth0 并给予其新的 IP 地址和路由。

```
# ifconfig eth0 inet up 192.168.0.111 \
netmask 255.255.255.0 broadcast 192.168.0.255
# route add -net 10.0.0.0 netmask 255.0.0.0 gw 192.168.0.1 dev eth0
```

结果是：

```
# ifconfig
eth0 Link encap:Ethernet  HWaddr 08:00:46:7A:02:B0
inet addr:192.168.0.111  Bcast:192.168.0.255  Mask:255.255.255.0
UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

```
...

lo   Link encap:Local Loopback
inet addr:127.0.0.1  Mask:255.0.0.0
...
# route
Kernel IP routing table
Destination  Gateway      Genmask        Flags Metric Ref Use Iface
192.168.0.0  *            255.255.255.0  U     0      0     0 eth0
*****0.0.0     192.168.0.1  255.0.0.0      UG    0      0     0 eth0
```

更多信息请参阅 ifconfig(8) 和 route(8).

10.2.2 底层网络设置 - ip

ip 和先前的 ifconfig 和 route 有相同功能的命令如下：

- ip link show
- ip route list
- ip link set eth0 down
- ip addr del dev eth0 local 192.168.0.3
- ip addr add dev eth0 local 192.168.0.111/24 broadcast 192.168.0.255
- ip link set eth0 up
- ip route add dev eth0 to 10.0.0.0/8 src 192.168.0.111 via 192.168.0.1

运行的时候使用 help 参数，能让 ip 打印出命令的语法。 例如，ip link help 打印出：

```
Usage: ip link set DEVICE { up | down | arp { on | off } |
dynamic { on | off } |
multicast { on | off } | txqueuelen PACKETS |
name NEWNAME |
address LLADDR | broadcast LLADDR |
mtu MTU }
ip link show [ DEVICE ]
```

参阅 ip(8).

10.2.3 设置无线网卡(Wi-Fi)接口

对于无线网卡(Wi-Fi)接口，除了 ifconfig 或 ip 之外，你还需要 iwconfig 这个程序。此程序在 wireless-tools 中。

参阅 iwconfig(8).

10.2.4 设置 PPP 接口

如果你是通过连接在拨号电话机上的调制解调器，并使用 Point-to-Point 协议 (PPP) 来上网的。那么这样的网络连接就是通过 ppp0、ppp1 等网络接口来实现的。

PPP 接口是由 pppd 这个 PPP 服务来管理的。你可以在 ppp 中找到该程序。 所以，对于用户来说，设置 PPP 接口也就是对 pppd 进行设置。

10.2.4.1 手动设置 pppd

为了建立网络连接，我们需要打开一个通讯端口（通常是串口），需要把命令传输到通讯设备上（通常是调制解调器），需要拨某个电话号码，对于外部的 PPP 服务还需要进行身份验证，需要内核创建 PPP 接口，最后需要修改路由表。只有这样信息才能在这条连接上传递。pppd 能完成上述所有动作，因而会有一堆的设置参数。相关参数参见 pppd(8)。

在 Ubuntu 系统上，全局的设置在 /etc/ppp/options 里面。用户的特定设置放在 ~/.ppprc。依赖于通讯端口的设置全部在 /etc/ppp/options.partname。例如，假设你有两个调制解调器 —通过 /dev/LT-modem 来访问的内置 Lucent LT 调制解调器和通过 /dev/ttyS0 来访问的外置调制解调器。 建立以下这两个文件。

```
# cat > /etc/ppp/options.LT-modem <<EOF
115200
init "/usr/sbin/chat -f /etc/chatscripts/setup-LT-modem"
EOF
# cat > /etc/ppp/options.ttyS0 <<EOF
115200
init "/usr/sbin/chat -f /etc/chatscripts/setup-ttyS0"
EOF
```

这些涉及到下面的 chat scripts。 首先，/etc/chatscripts/setup-LT-modem。

```
ABORT ERROR
'' ATZ
OK 'ATW2X2 S7=70 S11=55'
OK AT
```

其次，/etc/chatscripts/setup-ttyS0。

```
ABORT ERROR
'' ATZ
OK 'ATL1M1Q0V1W2X4&C1&D2 S6=4 S7=70 S11=55 S95=63 S109=1 +FCLASS=0'
OK AT
```

显然，这些文件的内容依赖于你的硬件。

选项也可以被当为参数传递给 pppd。

在 Ubuntu 中通常用 pon 来启动 pppd。pon 使用的时候，它的第一个参数表示位于 /etc/ppp/pears/ 里面的配置文件的名称，这个文件同样也被 pppd 读取。[54] 这儿就是你为特定连接设置特殊选项的地方 — 例如，一个特别的网络服务提供商(ISP)。

假设你往来于 Amsterdm 和 Den Haag 这两座城市。在每个城市，你要求能访问两个 ISP 服务— Planet 和 KPN。首先为每个 ISP 创建基本的配置文件。

```
# cat > /etc/ppp/peers/KPN <<EOF
remotename KPN
noauth
user kpn
noipdefault
ipparam KPN
EOF
# cat > /etc/ppp/peers/Planet <<EOF
remotename Planet
auth
user user3579@planet.nl
noipdefault
mru 1000
mtu 1000
ipparam Planet
EOF
```

这些文件设置了两个 ISPs 中不同的部分。相同的部分可以放在 /etc/ppp/options 或于接口有关的某个设置文件中。

现在为每个城市里的每个 ISP 创建配置文件。在我们的例子中，从一个城市连接某个 ISP 和从另外一个城市连接这个 ISP 唯一的差别就是所需要的 chatscript。(chatscript 不同是因为当地访问的电话号码不同。)

```
# cat > /etc/ppp/peers/KPN-Amsterdam <<EOF
connect "/usr/sbin/chat -v -f /etc/chatscripts/KPN-Amsterdam"
file /etc/ppp/peers/KPN
EOF
# cat > /etc/ppp/peers/KPN-DenHaag <<EOF
connect "/usr/sbin/chat -v -f /etc/chatscripts/KPN-DenHaag"
file /etc/ppp/peers/KPN
EOF
# cat > /etc/ppp/peers/Planet-Amsterdam <<EOF
connect "/usr/sbin/chat -v -f /etc/chatscripts/Planet-Amsterdam"
file /etc/ppp/peers/Planet
EOF
# cat > /etc/ppp/peers/Planet-DenHaag <<EOF
connect "/usr/sbin/chat -v -f /etc/chatscripts/Planet-DenHaag"
file /etc/ppp/peers/Planet
EOF
```

file 命令显示了各个配置，包括先前列出过的配置。connetc 命令详细说明了 pppd 用来建立连接需要的特殊参数。我们通常使用 chat 这个程序来做这些事情，使 chatscript 适合这个 ISP。这里是给 Den Haag 的 chatscripts；给 Amsterdam 用的 chatscripts 也类似，除了电话号码不一样。不过当这个 ISP 通过本地的其他公司来提供服务的话，也许 chatscripts 就有所区别了。

```
# cat > /etc/chatscripts/KPN-DenHaag <<EOF
ABORT BUSY
```

```
ABORT 'NO CARRIER'
ABORT VOICE
ABORT 'NO DIALTONE'
ABORT 'NO DIAL TONE'
ABORT 'NO ANSWER'
ABORT ERROR
OK-AT-OK ATDT 0676012321
CONNECT \d\c
EOF
# cat > /etc/chatscripts/Planet-DenHaag <<EOF
ABORT BUSY
ABORT 'NO CARRIER'
ABORT VOICE
ABORT 'NO DIALTONE'
ABORT 'NO DIAL TONE'
ABORT 'NO ANSWER'
ABORT ERROR
OK-AT-OK ATDT 0676002505
CONNECT \d\c
EOF
```

为了能连接上这些 ISP，你需要用户名和密码以便让 pppd 在需要的时候能提供这些资料。这些信息不是被存储在 /etc/ppp/pap-secrets（如果使用 PAP 协议）就是在 /etc/ppp/chap-secrets（如果使用 CHAP 协议）。虽然 CHAP 更加的安全，但是 PAP 仍然是使用最为广泛的。因为这些文件包含有“秘密"，所以群组和其他用户应该不被允许读写这些文件。这些文件的格式在 pppd(8) 中有解释。“秘密”（第三格）是通过用户名（第一格）和/或服务器名称（第二格）来查找的。当连接到一个 ISP 时通常是不知道这个服务器的名字的，所以我们用用户名代替；上面 peers/KPN 和 peers/Planet 中的 user 那一行就是完成这个动作的。

```
# client name       server name  secret
kpn                 *            kpn
user3579@planet.nl  *            myfavoritepet
```

详细信息，参阅 /usr/share/doc/ppp/README.Debian.gz。

10.2.4.2 使用 pppconfig 设置 pppd

一个快速设置 pppd 方法就是使用 pppconfig 程序，该程序来自同名的软件包。pppconfig 先使用菜单界面来询问使用者一些问题，然后设置上面提到过的这些文件。

10.2.4.3 使用 wvdial 设置 PPP 接口

另一种使用 pppd 的处理方法是从 wvdial 来运行 pppd，wvdial 在软件包 wvdial 中。不同于 pppd 使用 chat 来拨号和协商连接，wvdial 在完成拨号和初始化协商之后才运行 pppd 去完成剩余的工作。只要给出电话号码、用户名和密码，大多数情况下，wvdial 都能成功建立连接。

10.3 命名主机

10.3.1 主机名

主机名是由内核维护的。初始化脚本 /etc/init.d/hostname.sh 在系统启动的时候根据 /etc/hostname 中存储的名称设置主机名（使用 hostname 这个命令）。这个文件应该**只**包含系统的主机名，而不是完整的域名。

运行 hostname（不带任何参数）可以打印出当前的主机名。

10.3.2 邮件名

主机的**邮件名**是于邮件相关的程序用来确认主机的。/etc/mailname 包含了该名称并以新空行结尾。邮件名通常是主机的完整的域名之一。参阅 mailname(5)。

电子邮件接受者看到的你这台 Debian 主机发送的邮件信头 From: 的内容，取决于你机器上邮件用户代理(MUA)和邮件传输代理(MTA)的设置。假设本地用户 foo 从邮件名为 myhost.dom 的主机上发送了一封邮件。送出去的电子邮件的信头 From: 会是：

- “From: foo@myhost.dom” 如果 MUA 没有 From: 的设置；
- “From: bar@myhost.dom” 如果 MUA 有“From: bar”的设置；
- “From: bar@bogus.dom” 如果 MUA 有“From: bar@bogus.dom”的设置。

就算 MUA 中设置了 From:，MTA 还是会加入"Sender:foo@herman.dom" 来表示真正的来源。

当然，任何复杂的 MTA 在执行地址重写的时候，如同在 Exim 下设置一个收集不存在的邮件地址的容器，第 9.6.1.3 节和在 Exim 下设置寄出邮件的地址重写，第 9.6.1.4 节中讨论的，收件者看到的邮件地址是可以任意改变的。

10.4 域名服务(DNS)

主机由域名和 IP 地址来查询。DNS 是一套客户端-服务器系统，在这套系统中域名解释器访问域名服务器从而把域名和 IP 地址或是其他合适的主机联系在一起。 GNU C Library resolver(3) 也能够在文件中或通过网络信息服务(NIS)来查找 IP 地址。

某些程序（如，GNOME）就希望主机名能被解析为一个 IP 地址并且拥有一个合法的域名。这样真的是非常不合适的，因为主机名和域名是两个完全不同的的东西。为了支持这些软件，我们需要确保系统主机名能够被解析。通常的做法就是在 /etc/hosts 中加入一行带有 IP 地址和系统主机名的内容。如果你的系统有一个永久的 IP 地址，那就用这个地址，否则使用 127.0.0.1 这个地址。

```
127.0.0.1 localhost
127.0.1.1 uranus
```

使用 hostname --fqdn 来查看你的系统的主机名能否被解析为一个 IP 地址并拥有一个有效的域名。

10.4.1 域名解析器

域名解析器的工作是查找某个域名所对应的 IP 地址。大部分常用的域名解析器是 GNU C Library 中的 resovler 提供的功能(resolver(3))。另一个是由 libfiredns 软件包提供

的 FireDNS resolver。还有其他的。

GNU LIBC 的域名解析器对域名的解析是由 /etc/nsswitch.conf 中的 hosts 这一行配置决定的。该行列出了解析域名用的服务：例如 dns、files、nis、nisplus。参阅 nsswitch.conf(5)。 即使在使用 files 的情况下，域名解析器的行为也是由 /etc/hosts 这个配置文件控制的。参阅 hosts(5)。

上述文件都是静态的，你可以用你喜欢的编辑器修改。

在使用 dns 服务的情况下，域名解释器的行为也是由 /etc/resolv.conf 这个配置文件控制的。参阅 resolv.conf(5)。resolv.conf 的一个重要功能就是提供一个域名服务器的 IP 地址列表，通过查询这些服务器来获得域名解析。这一列表常常依赖于网络环境，而且在你机器运行的时候，网络环境时常发生变化。pppd 和 dhclient 这类程序都能添加或删除 resolv.conf 中的信息。但是这些功能不是每次都能正常工作而且两者还会有冲突。软件包 resolvconf 采用了更好的方法解决了这个问题，并提供了一个标准的框架来更新 resolv.conf。参阅管理域名服务器信息 - resolvconf, 第 10.4.2 节。

10.4.2 管理域名服务器信息 - resolvconf

软件包 resolvconf 提供了一个框架，能动态的管理关于可用域名服务器的信息。它解决了长久以来如何维护一个给域名解析器和 DNS 缓存使用的动态的域名服务器列表的问题。Resolvconf 把它自己设为控制网络接口和提供域名服务信息的程序与需要域名服务信息的应用程序的中间媒介。

resolvconf 被设计成不需要任何手动设置就能工作。但是，这个软件包还是很新的，可能需要一些手工的干预才能正常的工作。如果你曾经定制过软件包，而且它们更新了 /etc/resolv.conf 的话：你就需要去掉这部分定制。更多信息参阅 /usr/share/doc/resolvconf/README.gz。

10.4.3 缓存查询过的域名 - nscd、dnsmasq、pdnsd、bind9

如果你的域名服务器响应速度非常慢，你可能需要使用 nscd 来缓存域名解析器 libc6 查询到的结果。

如果你希望为你本地网络中的其他主机缓存结果的话，你可能要去运行一个缓存转发域名服务器(caching forwarding nameserver)。就像 dnsmasq 或 pdnsd。

如果你愿意，你也可以用软件包 bind9 中的 named 来做缓存转发域名服务器(caching forwarding nameserver)。但是这是一个很庞大的程序，除非你需要它高级功能，否则还是使用上面提到的那些程序比较好。

所有这些软件包都能和 resolvconf 一起工作。

10.4.4 提供域名解析服务 - bind

如果你希望给一个域提供一个权威的域名服务的话，你就需要一个完善的域名服务器，例如软件包 bind9 中的 named。

如果你安装了 bind9，你也应该安装 dnsutils。 你可能还需要安装这样一些工具软件包：bind9-host； dns-browse； dnscvsutil； nslint。 你可能还需要安装说明文档： bind9-doc。 你可能还需要安装开发文档：libbind-dev； libnet-dns-perl。 如果你是使用 DHCP，

下面这个软件包会对你有所帮助： dhcp-dns。

安装 bind9 或者用 dpkg-reconfigure bind9 来进行基本的设置。设置包括编辑文件 name。在 Debian 中，你可以在 /etc/bind/ 找到这个文件，它主要是用来设置基本的 DNS 域的；它包含了其他两个文件： named.conf.local，用来定义本地区域，和 named.conf.options，用来设置选项的。(后者的执行需要 resolvconf 来产生 /var/run/bind/named.options 文件，除了 forwarders 的说明是一个当前可用的非本地域名服务器列表之外，其余都和原先的一样。要利用这个，可以修改 named.conf 中的 include 这一样，使其包含 /var/run/bind/named.options。参阅管理域名服务器信息 - resolvconf，第 10.4.2 节。)

在 named.conf* 文件中用到的数据库文件，如果没有指定完整的路径，则该数据库文件会被存储在 /var/cache/bind/。这是一个正确的存储 named 产生的文件的地方。例如：某个域的从服务器使用的数据库文件。/etc/bind/ 下面的那些静态的数据库文件，需要在 named.conf 中有完整的路径才能被找到。详情参阅 /usr/share/doc/bind9/README.Debian.gz。

10.5 使用 DHCP 来配置网络接口

底层的网络接口设置可以用 Dynamic Host Configuration Protocol (DHCP) 来自动设置。你的防火墙或路由器或宽带 ISP 可能用这个方法来配置 IP 地址和其他参数。

要做这个工作你必须安装下列软件包的其中一个：

- dhcp3-client (version 3, Internet Software Consortium)
- dhcpcd (Yoichi Hariguchi and Sergei Viznyuk)
- pump (Red Hat)

pump 简易且被广泛应用。 dhcp3-client 复杂，但是可配置程度更高。 [55]

10.6 Debian 的高级网络设置

10.6.1 使用 ifupdown 进行高级网络设置

为了让网络设置更加简单，Debian 提供了一个标准的高级网络设置工具，包含 ifup 和 ifdown 程序和 /etc/network/interfaces 文件。 如果你选择用 ifupdown 来配置你的网络，那么就**不要**同时使用底层工具去配置。这也意味着你不应该用其他高级配置工具，如 whereami、divine、intuitively 等。他们调用的也是底层配置工具。ifupdown 程序在设计的时候，是假设仅有这样一个程序会被用来设置网络接口的。

更新接口设置是执行：

```
# ifdown eth0
# editor /etc/network/interfaces  # 做你需要的调整
# ifup eth0
```

更多信息参阅 interfaces(5)、 /usr/share/doc/ifupdown/examples/network-interfaces.gz 和 ifup(8)。

10.6.1.1 用固定 IP 地址为接口进行设置

假设你要配置一个以太网接口，使其拥有一个固定的 IP 地址 192.168.0.111。这个 IP 地址以 192.168.0 为开头，所以它肯定在一个 LAN 内。进一步假设 192.168.0.1 是 LAN 上面 Internet 网关的地址。编辑 /etc/network/interfaces，使其包含类似下面这段的内容：

```
iface eth0 inet static
address 192.168.0.111
netmask 255.255.255.0
gateway 192.168.0.1
```

在接口被激活或是在激活之前，你都可以配置接口的其他部分或者进行其他操作。只要你在"up"和"down"那几行中设置合适的命令。

```
iface eth0 inet static
address 192.168.0.111
netmask 255.255.255.0
gateway 192.168.0.1
up route add -net 10.0.0.0 netmask 255.0.0.0 gw 192.168.0.2 dev $IFACE
down route del -net 10.0.0.0 netmask 255.0.0.0 gw 192.168.0.2 dev $IFACE
up echo Interface $IFACE going up | /usr/bin/logger -t ifup
down echo Interface $IFACE Going down | /usr/bin/logger -t ifdown
```

你也可以选择把命令插入到 /etc/network/if-up.d 和 /etc/network/if-down.d 目录下的脚本中。这些脚本也能执行扩展的选项。详情参阅 interfaces(5)。例如，软件包 resolvconf 包含的脚本允许你在接口被激活的同时，往 /etc/resolv.conf 添加指定的 DNS 信息：

```
iface eth0 inet static
address 192.168.0.111
netmask 255.255.255.0
gateway 192.168.0.1
dns-search somedomain.org
dns-nameservers 195.238.2.21 195.238.2.22
```

dns-search 选项的参数 somedomain.org 符合 resolv.conf(5) 中所说的 search 选项的参数。 dns-nameservers 选项的参数 195.238.2.21 和 195.238.2.22 符合选项 nameserver 的参数。其他可以识别的选项是 dns-domain 和 dns-sortlist。参阅 管理域名服务器信息 - resolvconf, 第 10.4.2 节。

10.6.1.2 用 DHCP 配置接口

为了使用 DHCP 配置接口，请编辑 /etc/network/interfaces，使其包含一下这段内容：

```
iface eth0 inet dhcp
```

为了让 DHCP 能工作，你需要安装一个 使用 DHCP 来配置网络接口, 第 10.5 节中提及的 DHCP 客户端程序。

10.6.1.3 配置无线网卡(Wi-Fi)接口

软件包 wireless-tools 包含了一个钩子脚本 /etc/network/if-pre-up.d/wireless-tools，使得在接口被激活之前，对无线网卡(802.11a/b/g)进行设 置变为可能。使用 iwconfig 程序来完成设置，参阅 iwconfig(8)。任何一个 iwconfig 的有效参数，你都可以把它包含在 /etc/network/interfaces 中，并在原有的参数名字前加上“wireless-”这个前缀。例如，要设置 eth0， 使得 eth0 在被 DHCP 激活之前， ESSID 设定为 myessid, encryption key 设定为 123456789e，请编辑 /etc/network/interfaces，加入一下这段内容：

```
iface eth0 inet dhcp
wireless-essid myessid
wireless-key 123456789e
```

注意!如果你使用 waproamd 来设置这个接口的话,你不应该使用这个方法来设置 ESSID 和 key。在 ifup 执行时，waproamd 就已经设置好了 ESSID 和 key。参阅 使用 waproamd 启动网络设置, 第 10.8.4 节。

10.6.1.4 设置 PPP 接口

ifup 和 ifdown 程序使用 pon 和 poff 来添加和删除 PPP 接口，所以先阅读 设置 PPP 接口, 第 10.2.4 节。

假设你已经设定了 PPP 和 myisp 一起工作。请编辑 /etc/network/interfaces，使其包含如下这段内容：

```
iface ppp0 inet ppp
provider myisp
```

这样设置好后，ifup ppp0 会完成

```
pon myisp
```

遗憾的是,目前无法在 /etc/network/interfaces 中的 ppp 段落里面提供额外的 pppd 选项。

目前无法使用 ifupdown 来为 PPP 接口提供辅助的设置。因为在 pppd 完成连接之前 pon 就已经存在了，ifup 执行激活脚本之后 PPP 接口才可用。到这个 bug 被修正之前，还是需要在 /etc/ppp/ip-up 或 /etc/ppp/ip-up.d/ 中进行额外的设置。

10.6.1.5 设置 PPPoE 接口

许多宽带因特网服务提供商(ISP)使用 PPP 协议来连接，即使用户的机器通过以太网和/或 ATM 网络连接他们。 这是通过 PPPoE 的技术来完成的，即把 PPP 祯封装在以太网卡(Ethernet)的祯里面。 假设你的 ISP 被称为 myisp。首先为 myisp 设置 PPP 和 PPPoE。最简单的方法就是安装 pppoeconf，然后从终端中运行 pppoeconf。之后编辑 /etc/network/interfaces 使其包含如下这段内容：

```
iface eth0 inet ppp
provider myisp
```

有时候最大传输单位 Maximum Transmit Unit (MTU) 和 PPPoE over Digital Subscriber Line (DSL) 有关。详情参阅 DSL-HOWTO。

注意！如果你的宽带调制解调器包含路由功能。那么当调制解调器/路由器自己处理 PPPoE 连接时，在 LAN 中它就表现的和简单的连接 Internet 的以太网网关一样。

10.6.1.6 为网关配置多个以太网接口

假设 eth0 已经用 DHCP-configured IP 地址连接到 Internet，并且 eth1 使用一个固定 IP 地址 192.168.1.1 连接到 LAN。编辑 /etc/network/interfaces 使其包含如下内容：

```
iface eth0 inet dhcp

iface eth1 inet static
address 192.168.1.1
netmask 255.255.255.0
```

如果按照 建立路由网关, 第 10.12 节中描述的去激活主机上的 NAT，那么你就能和 LAN 中的其他主机一起享用互联网连接了。

10.6.1.7 设置虚拟接口

使用虚拟接口，你可以设置一个以太网卡使其成为拥有很多 IP 子网的接口。例如，假设你的主机在 LAN 网络上(192.168.0.x/24)。你想要让主机连接到互联网，并用已经存在的以太网卡通过 DHCP 来获得公网 IP 地址。编辑 /etc/network/interfaces 使其包含如下一段内容：

```
iface eth0 inet static
address 192.168.0.1
netmask 255.255.255.0
network 192.168.0.0
broadcast 192.168.0.255

iface eth0:0 inet dhcp
```

eth0:0 接口是一个虚拟的接口。当它被激活的时候，它的真实硬件 eth0 也会被激活。

10.6.2 使用 ifupdown 的逻辑接口定义进行高级网络设置

下列内容中，对于读者而言了解**物理接口(physical interface)**和**逻辑接口(logical interface)**之间的不同是重要的。 **物理(physical)**接口就是我们所说的“接口”，是由内核命名为 eth0、eth1、ppp0 或其他。**逻辑(logical)**接口是一套可以用来对物理接口的可变参数进行设置的值的集合。如果你觉得还不清楚，那么在阅读的时候就用“用 X 配置文件来设置接口”去代替“设置逻辑接口 X”。

在 /etc/network/interfaces 中 iface 的定义实际上是逻辑接口的定义，而不是物理接口

的。如果你从来不去重新配置你的接口，那么你就可以忽略这个细节。因为物理接口 foo 缺省会被设置成逻辑接口 foo。

假设你的电脑是台笔记本，你需要在家里和工作的地方之间穿梭。那么当你的电脑连接到公司的网络或家里的网络时，你都要相应地对 eth0 进行设置。

首先定义两个逻辑接口 home 和 work（取代 eth0，就像我们先前做的），它们分别描述了在家中的和公司的网络中如何设置接口。

```
iface home inet static
address 192.168.0.123
netmask 255.255.255.0
gateway 192.168.0.1

iface work inet static
address 81.201.3.123
netmask 255.255.0.0
gateway 81.201.1.1
```

然后通过适当的设置，并在命令行中指定这些设置。物理接口 eth0 就能在家庭网络中被激活了：

```
# ifup eth0=home
```

针对公司网络重新设置 eth0 只要运行这些命令：

```
# ifdown eth0
# ifup eth0=work
```

注意！如果 interfaces 中的内容如上述所写的，那么我们就不能单独执行 ifup eth0 来激活 eth0。理由是 ifup 使用物理接口名作为缺省的逻辑接口名，但是现在在我们的例子中，没有关于逻辑接口 eth0 的定义。

10.6.3 使用 ifupdown 进行自动的网络设置

在 ifup 运行的时候，接口的名称可以被“映射(mapped)”为别的名称。至于映射成什么名称，这个视情况决定。因此 ifup 能够被设置为用预设的逻辑接口集合中的一个合适的逻辑接口来激活物理接口。

逻辑接口名称映射产生的情况如下：

- 如果执行 ifup 的时候没有给定逻辑接口名称，那么物理接口名称就会被用作初始的逻辑接口名称。
- 如果逻辑接口名称符合 mapping 描述的 glob-pattern，那么就会映射到新生成的逻辑接口名称中去。对于每段映射都是这样按顺序进行的。
- 如果最终的逻辑接口名称是 /etc/network/interfaces 中定义的一个逻辑接口的标签，那么物理接口就被当作这个逻辑接口来激活。否则 ifup 会打印"Ignoring unknown interface"随后退出。

mapping 的语法：

```
mapping glob-pattern
script script-name
[map script input]
```

mapping 段落中的 script，总是把**物理**接口的名称作为它的参数。其他“map”行中的内容（不包含“map”本身）都会作为它的标准输入。该 script 在退出之前会把映射的结果作为标准输出打印出来。

例如，下面这段 mapping 会让 ifup 用逻辑接口 home 来激活接口 eth0。

```
mapping eth0
script /usr/local/sbin/echo-home
```

/usr/local/sbin/echo-home 的内容为：

```
#!/bin/sh
echo home
```

因为映射是由脚本来完成的，所以自动选择逻辑接口是可能的 — 基于一些选择测试。参阅使用 guessnet 来选择逻辑接口, 第 10.6.3.1 节中的范例。

10.6.3.1 使用 guessnet 来选择逻辑接口

安装软件包 guessnet。然后在 /etc/network/interfaces 中加入如下一段内容：

```
mapping eth0
script guessnet-ifupdown
map home
map work
```

现在，当你 ifup eth0 的时候，guessnet 会检测 eth0 是否能用 home 或 work 来激活。它用存储在逻辑接口定义中的信息来完成这项工作。

10.6.4 使用 laptop-net 进行自动的网络设置

软件包 laptop-net 采用不同的方法处理自动的网络设置。Laptop-net 不用 ifupdown 的逻辑接口，取而代之的是它自己的一套配置"方案"和“配置文件”系统。不过，Laptop-net 还是会使用 ifup 和 ifdown 来设置物理接口。更多详细文档请安装 laptop-net-doc。

10.6.5 使用 network-manager 进行自动的网络设置

network-manager 这个软件现在是由 Fedora 的开发者们开发的，Ubuntu 已经对其进行了打包。有朝一日它也会出现在 debian 中，到时候我们应该放弃 ifupdown 和其他过时的朋友们了。

10.7 处理内核对接口命名的不一致性

eth0、eth1 这类设备的名称是由内核指定的，内核是按照创建这些接口的顺序来命名的。在开机的时候，被检测到的适配器通常都是按照一样的顺序被检测到的，所以每次都被 指定为同一个名称。但是，对于热拔插的适配器情况就不是这样了。在不同情况下，它们可能以任意的顺序被检测到，于是内核就给它们指定不同的名称。

因为这个关系，在一个网卡适配器是热拔插设备的系统中，使用 /etc/network/interfaces 给 eth0、eth1 这类接口定义逻辑接口和依靠缺省的映射关系都是不可能完全正常工作的。要取代这个做法,你必须给逻辑接口设置一个唯一的名称,并使用下列两个方法中的一个 来限制哪些逻辑接口会被指定给哪些是适配器。

一个方法是使用 nameif（在 net-tools 软件包中）工具或另外一个更灵活的 ifrename（在 ifrename 软件包中）工具，使内核按照适配器的属性来指定接口名称。使用这个命名方案的话，物理接口的名称可以被用来推测出接口下面的适配器的名称。

另外一种方法是使用 ifup 映射机制。这种情况下就会根据将要被激活的物理接口所在的适配器的某些属性来选择逻辑接口。

假设，你有两个网络适配器，分别在网络 net1 和 net2 中使用。
/usr/share/doc/ifupdown/examples/ 目录下面包含了一个映射脚本,能够根据适配器的媒体访问控制地址（MAC 地址）来选择逻辑接口。首先安装脚本到适合的目录。

```
# install -m770 /usr/share/doc/ifupdown/examples/get-mac-address.sh \
/usr/local/sbin/
```

然后在 /etc/network/interfaces 中加入如下一段内容：

```
mapping eth0
script /usr/local/sbin/get-mac-address.sh
map 02:23:45:3C:45:3C net1
map 00:A3:03:63:26:93 net2
```

更多，更复杂的例子参阅 多阶段(Multi-stage)映射, 第 10.9 节。

不管采用那种方法，通常都是用 MAC 地址来识别适配器的。

10.8 启动(triggering)网络设置

我们已经知道了接口是如何设置和重新设置的。这些动作需要在适当的时候完成。

传统上，网络是在开机的时候由 /etc/rcS.d/S40networking 这个脚本设置的，而且极少重新设置。其他需要网络的服务随后启动。在关机或者重启的时候，initscripts 按照相反的循序执行。

然而现在，GNU 和 Linux 正朝着支持动态硬件更换和突发事件的方向发展。首先是为可替换的 PCMCIA 卡提供支持。目前在添加 hotplug 机制后，很多外设都能在电脑运行的时候进行替换。这也包括了网络硬件。注意！当你拔插可热拔插设备的时候，涉及到此硬件的服务需要在插入之后启动或删除之 前关闭。这就意味着这类服务需要从 System V init 系统

中删除，并时期处于 ifupdown 的控制之下。

例如，假设受 initscript /etc/init.d/foo 控制的服务 foo 依赖于动态设置的网络接口 eth0。

- 首先，从 init 系统中删除 foo。如果你是使用 sysv-rc init 系统的话，那么请完成下面几个事项。

```
# rm /etc/rc[2345].d/S??foo
```

- 然后通过在/etc/network/interfaces 中 eth0 段落里加入 up 和 down 的选项，将 foo 置于 ifupdown 的控制之下。此文件还能呼叫 foo 的初始化脚本 (initscript)：

```
iface eth0 inet dhcp
up /etc/init.d/foo start
down /etc/init.d/foo stop
```

10.8.1 在开机的时候启动(triggering)网络设置

在启动的时候 init 脚本 /etc/rcS.d/S40networking 运行了 ifup -a 命令。这个命令激活了所有在 /etc/network/interfaces 中 auto 段落里罗列了的物理接口。

现在使用动态的方式来处理网络设置是一种更好方法。一旦这个支持硬件动态更换的机制在适当的位置上，处理静态的硬件就变得非常的简单，就和 处理动态的一样。启动过程也可以被当作另外一个热拔插事件。(参阅 使用 hotplug 启动(triggering)网络设置, 第 10.8.2 节。)

尽管如此，大多数情况下，至少 lo loopback 接口需要在开机时被激活。所以，需要确定 /etc/network/interfaces 中包含如下一段内容：

```
auto lo

iface lo inet loopback
```

如果你想让其他物理接口也在开机的时候被激活，请把它们加入到 auto 的段落中去。**绝对不要**把 PCMCIA 接口放在 auto 段落中。在开机启动顺序中，PCMCIA 的 cardmgr 晚于 /etc/rcS.d/S40networking 启动。

10.8.2 使用 hotplug 启动(triggering)网络设置

安装 hotplug 软件包来获得热拔插(hot-plug)支持。

你可以在开机的时候或是把一张卡（例如，PCMCIA 卡）插入机器之后或者在 discover 这类的工具启动并加载了必要的驱动模块之后，热拔插你的网络硬件。

当内核检测到新的硬件，它会初始化这个硬件并执行 hotplug 程序去配置这个硬件。当硬件被删除的时候,内核以不同的环境变量设置再次执行 hotplug。在 Debian 中,hotplug 被

呼叫时，它会执行 /etc/hotplug/ 和 /etc/hotplug.d/ 中的脚本。细节参阅 hotplug(8)。

新安装的网络硬件是由脚本 /etc/hotplug/net.agent 设置的。 假设你的 PCMCIA 网卡已经插入，并生成了可用的接口 eth0。则 /etc/hotplug/net.agent 做了下面的事情。 ：

```
ifup eth0=hotplug
```

除非你在 /etc/network/interfaces 中加入了名称为 hotplug 的逻辑接口定义或映射，否则这个命令不会做任何事。为了让这个命令能用来设置 eth0，在 /etc/network/interfaces 中加入如下一段内容：

```
mapping hotplug
script echo
```

如 使用 ifupdown 的逻辑接口定义进行高级网络设置, 第 10.6.2 节中解释的，这样会映射上述的命令，使其等同于如下的命令：

```
ifup eth0=eth0
```

（如果你使用 hotplug 启动 ifplugd 或 waproamd 来控制这个接口，请**不要**包含这样的映射内容。）

如果你希望在热拔插的时候仅仅激活 eth0，而不是其他接口，那么请用 grep 取代 echo，做法如下：

```
mapping hotplug
script grep
map eth0
```

更多技巧，参阅 使用 ifupdown 进行自动的网络设置, 第 10.6.3 节和 /usr/share/doc/hotplug/README.Debian。

10.8.3 使用 ifplugd 启动(triggering)网络设置

ifplugd 守护进程根据相关的硬件有没有接入网络来激活或关闭接口。这个程序能够检测网线是否已经插入或无线网卡(Wi-Fi)是否能访问 AP（虽然 waproamd 比较适合后一种情况）。当 ifplugd 发现连接状态改变时，它能运行一个代理脚本，缺省会呼叫 ifup 或 ifdown。

10.8.4 使用 waproamd 启动网络设置

waproamd 守护进程和 ifplugd 类似，只是它是针对无线网卡(Wi-Fi)设计的。它主动的扫描无线网卡能访问到的 AP。当访问完成之后，waproamd 执行 ifup。

如果你正在使用 waproamd，那么通常来说你是通过 waproamd 而不是 /etc/network/interfaces 里面的 wireless-* 选项来设置的。

10.8.5 网络设置和 PCMCIA

有许多可用的方法来处理 PCMCIA 网络接口的设置（对于 2.4 和 2.6 的内核）。

- 对于 32 位 PCI (CardBus) PCMCIA 网卡：
  - ifupdown 由 hotplug 控制
    - 在 Woody 和 Sarge 中，你需要在本地启用 hotplug 对 ifupdown 的控制，通过在文件 /etc/network/interfaces 里加入 使用 hotplug 启动(triggering)网络设置, 第 10.8.2 节中描述的映射段落即可。
- 对于 16 位 ISA PCMCIA 网卡：
  - ifupdown 由 hotplug 控制，并通过 pcmcia-cs 的限制来加载模块。
    - **推荐的**
    - 在 Woody 和 Sarge 中，你必须通过在 /etc/pcmcia/network 文件开始处，加入一行 exit 0 来取消缺省状态下 pcmcia-cs 对 ifupdown 的控制。同时，你必须在本地启用 hotplug 对 ifupdown 的控制，通过在文件 /etc/network/interfaces 里加入 使用 hotplug 启动(triggering)网络设置, 第 10.8.2 节中描述的映射段落即可。
  - 缺省状态下 pcmcia-cs 通过配置文件 /etc/pcmcia/network 来控制 ifupdown
    - **不赞成**使用，但 Woody 和 Sarge 缺省使用这个方式。
  - pcmcia-cs 通过 /etc/pcmcia/network 中的特殊代码来控制底层工具。
    - **不赞成**使用
    - 在 Woody 和 Sarge 中，编辑 /etc/pcmcia/network.opts 就能启用这些特殊代码。

对于 16 位的卡,推荐的处理方法充分利用了 2.4 内核的热拔插子系统对 PCMCIA 的支持。

PCMCIA 网卡是可以热拔插的。因此，任何需要通过 PCMICA 卡来获得网络的服务的服务，应该设置为要在卡插入之后启动并在卡移除的时候停止。通常通过安排服务在 ifup 时启动和 ifdown 时停止来完成这件事情。然而,有些人说服他们自己使用冷拔插(code plugging)他们的 PCMCIA 网卡：他们在系统启动之前插入网卡，在开机过程中陆续启动需要通过这卡来获得网络的那些服务。如果你是这类人，为了确保在启动这些服务器前网卡已经设置好 了，你需要做下列工作：

- 在 /etc/default/pcmcia 中设置 CARDMGR_OPTS="-f",强制 cardmgr 在前台运行。
- 修改 /etc/rc?.d/S20pcmcia 的名称，改成类似 /etc/rc?.d/S12pcmcia 的样子。

这些只适合于 16 位的 PCMCIA 卡。

注意！如果你使用 16 位的 PCMICA 卡，软件包 pcmcia-cs 还是需要的。该软件包包括了 cardmgr 守护进程，用来管理 socket 和加载驱动模块。我们只是不希望它通过 /etc/pcmcia/network 来呼叫网络设置程序。

为了让 cardmgr 能正常工作，你可能需要编辑 /etc/pcmcia/config.opts 来设置 16 位 PCMCIA 卡的资源。更多信息参阅 PCMCIA, 第 7.2.1 节和 Linux PCMCIA HOWTO。

10.9 多阶段(Multi-stage)映射

首先，假设你的网络适配器可以热拔插，你启用了 使用 hotplug 启动(triggering)网络设置, 第 10.8.2 节中描述的自动设置。其次，进一步假设你需要依照物理接口下面的适配器（如同 处理内核对接口命名的不一致性, 第 10.7 节中描述的）和接口上连接的网络（例如，使用 guessnet 来选择逻辑接口, 第 10.6.3.1 节中描述的）来把逻辑接口映射到“物理”接口。你就可以用多阶段映射来完成。

如果接口是可以热拔插的，映射的第一个阶段是利用 hotplug 的组名称并输出内核指定的接口名称。映射的第二阶段是利用内核指定的接口名称并输出适配器的名称。第三个阶段就是依照网络环境，把适配器名称映射到逻辑接口名称上去。

```
# 允许 hotplug 激活接口
mapping hotplug
        script echo

# 确定那个接口是有线的那个是无线的
mapping eth?
script /usr/local/sbin/get-mac-address.sh
map 02:23:45:3C:45:3C wired
map 00:A3:03:63:26:93 wifi

# 检测有线网络是否可用
mapping wired
script guessnet-ifupdown
map work-wired
map home

# 检测哪个无线网络可用
mapping wifi
script ifscout
map starbucks
map work-wireless

iface work-wired inet static
        ...
```

10.10 网络服务设置

桌面和家用服务器典型的网络服务设置包括：

- Internet *super-server*和 TCP/IP daemon wrapper，参阅 服务的访问限制, 第 9.2.5 节。
    - /etc/inetd.conf
- ssh: OpenSSH secure shell，参阅 SSH, 第 9.5 节。

    - /etc/ssh/ssh_config
    - /etc/ssh/sshd_config

- exim：邮件传输代理(MTA)，参阅 邮件名，第 10.3.2 节和 邮件传输代理（MTAs），第 9.6.1 节。

    - /etc/exim/exim.conf
    - /etc/mailname
    - /etc/aliases
    - /etc/email-addresses

- fetchmail：从 POP3 帐户中收取邮件的守护进程，参阅 收取邮件 - Fetchmail，第 9.6.2 节。

    - /etc/fetchmailrc

- procmail：本地邮件传递和过滤程序，参阅 处理邮件 - Procmail，第 9.6.3 节。

    - ~/.procmailrc

- 主机名和 DNS (proxy, cache, ...)，参阅 主机名，第 10.3.1 节和 域名服务(DNS)，第 10.4 节。

    - /etc/host.conf
    - /etc/hostname
    - /etc/hosts
    - /etc/hosts.allow
    - /etc/hosts.deny
    - /etc/resolv.conf
    - /etc/bind/named.conf（编辑）
    - /etc/bind/db.lan (add for LAN hosts)
    - /etc/bind/db.192.168.0 (add for LAN reverse)

- DHCP，参阅 使用 DHCP 来配置网络接口，第 10.5 节。

    - /etc/dhcp3/dhclient.conf（DHCP 客户端）
    - /etc/default/dhcp3-server（DHCP 服务器端）
    - /etc/dhcp3/dhcpd.conf（DHCP 服务器端）

- cvs：当前版本控制系统，参阅 并行版本系统 (CVS)，第 12.1 节.

    - /etc/cvs-cron.conf
    - /etc/cvs-pserver.conf

- nfs-kernel-server：网络文件系统，参阅 NFS 设置，第 3.4 节。(针对类 Unix 系统)

    - /etc/exports

- samba：Windows 网络文件和打印机共享，参阅 Samba 设置，第 3.5 节和 Samba，第 8.6.38 节。

    - /etc/samba/smb.conf

- 打印机守护进程系统，参阅 打印机设置, 第 3.6 节。
    - /etc/printcap (for lpr)

- apache 和 apache2：Web 服务器。
    - /etc/apache/*
    - /etc/apache2/*

- squid： Web 代理和缓存服务器。
    - /etc/squid/*

10.11 网络故障排除

如果你遇到了问题，首先执行下列命令来检查输出的结果：

```
# ifconfig
# cat /proc/pci
# cat /proc/interrupts
# dmesg | more
```

同时参阅 网络测试基础, 第 8.6.29 节下面的章节。

如果你无法浏览特定的站点，参阅 无法访问某些站点的怪问题, 第 3.8.5 节。

10.12 建立路由网关

一个 Debian 主机可以作为一个全能的网关，它可以承担网络地址转换(NAT，通常也称为 IP 伪装)、邮件传输、DHCP、DNS 缓存、HTTP 代理缓存、CVS 服务、NFS 服务和 Samba 服务。参阅 网络所需的主机名和 IP 地址, 第 3.1.9 节中的例子来完成上述设置。

10.12.1 Netfilter 设置

在 Linux 2.4 及其后继版本中加入了 netfilter/iptables 项目，作为一个防火墙子系统。参阅 Netfilter，那儿有许多有关其配置的讨论和解释。

10.12.1.1 netfilter 基础

Netfilter 内建了 5 条链路来处理数据包，它们分别是：PREROUTING、INPUT、FORWARD、OUTPUT 和 POSTROUTING：

```
routing
decision
IN ------> PRE ---> ------> FORWARD -----> ----> POST -----> OUT
interface  ROUTING  \       filter       /       ROUTING     interface
DNAT       |        tracking      ^       SNAT
REDIRECT   |                      |       MASQUERADE
```

```
v                     |
INPUT                 OUTPUT
| filter              ^ filter,DNAT
v                     |
\--> Local Process --/
user-space programs
```

10.12.1.2 过滤表(Netfilter table)

数据包在每条内建的链路中传输时按如下过滤表中的规则进行处理。

- filter（数据包过滤器，链路中默认的过滤器）
    - INPUT（作用于进入本机的数据包）
    - FORWARD（作用于路由到本机的数据包）
    - OUTPUT（作用于本地产生的数据包）
- nat（网络地址转换）
    - PREROUTING（作用于刚进入的待转换数据包）
    - OUTPUT（作用于在路由之前待转换的本地产生的数据包）
    - POSTROUTING（作用于待发出的已转换的数据包）
- mangle （network address mangling，适合于 2.4.18 之后的内核）
    - 适用于所有 5 条链路。

10.12.1.3 过滤目标(Netfilter target)

防火墙规则有许多目标：

- 4 个基本目标：
    - ACCEPT 允许数据包通过。
    - DROP 阻拦数据包。
    - QUEUE 允许数据包进入用户空间（userspace）（如果内核支持的话）。
    - RETURN 表示停止读取该链路并回到前一个（呼叫的）链路的下一条规则。
- 扩展目标：
    - LOG 打开内核日志。
    - REJECT 回送错误数据包并阻拦该数据包。
    - SNAT 修改数据包源地址，仅作用于 POSTROUTING 链路。（仅适用于 nat 过滤表）

```
--to-source ipaddr[-ipaddr][[UbuntuHelp:port-port]]
```

    - MASQUERADE 作用和 SNAT 一样，但面向使用动态 IP 请求建立的连接（拔号连接）。（仅适用于 nat 过滤表）

--to-ports port[-port]

- DNAT 修改数据包目的地址，仅作用于 PREROUTING、OUTPUT 链路以及由它们调用的用户自定义链路。（仅适用于 nat 过滤表）

--to-destination ipaddr[-ipaddr][[UbuntuHelp:port-port]]

- REDIRECT 修改数据包目标地址使其发送给本机。

--to-ports port[-port]

10.12.1.4 Netfilter 命令

iptables 的基本命令有：

```
iptables -N chain                    # 创建一个链路

iptables -A chain \                  # 添加链路的规则
-t table \                  # 使用过滤表(filter, nat, mangle)
-p protocol \               # tcp、udp、icmp 或所有，
-s source-address[/mask] \
--sport port[[UbuntuHelp:port]] \      # 如果 -p 是 tcp 或 udp，指定源的端口
-d destination-address[/mask] \
--dport port[[UbuntuHelp:port]] \      # 如果 -p 是 tcp 或 udp，指定目的地端口
-j target \                 # 如果匹配作何处理
-i in-interface-name \      # 针对 INPUT、FORWARD、PREROUTING
-o out-interface-name       # 针对 FORWARD、OUTPUT、POSTROUTING
```

10.12.1.5 网络地址转换

LAN 中的机器可以通过能把 LAN 地址转换为可用的 Internet 上的 IP 地址的网关来访问 Internet 的资源。

```
# apt-get install ipmasq
```

执行样例规则来加强 ipmasq 的保护机制。 参阅 /usr/share/doc/ipmasq/examples/stronger/README。 对于在 Woody 中使用 Debian 的 kernel-image-2.4，请确认加载了相应的模块。 Sarge 里面的 ipmasq 已经修正了这个问题。相关配置指导请参阅 网络功能, 第 7.2.3 节。

对于使用 2.2 版内核镜像的 Debian，可按下面的方法编辑 /etc/masq/rules 中的 Z92timeouts.rul 文件，以保证可长时间连接远程站点（如发送大容量的 email，等）：

```
# tcp, tcp-fin, udp
# 2hr, 10 sec, 160 sec - default
# 1 day, 10 min, 10 min - longer example
```

```
$IPCHAINS -M -S 86400 600 600
```

同样，如果是通过 PCMCIA NIC 访问网络，ipmasq 需要从 /etc/pcmcia/network.opts（阅读：/usr/share/doc/ipmasq/ipmasq.txt.gz）或 /etc/network/interfaces（阅读： 网络设置和 PCMCIA, 第 10.8.5 节和 启动(triggering)网络设置, 第 10.8 节） 启动。

10.12.1.6 重定向 SMTP 连接（2.4 版内核）

假设你将一台笔记本电脑重新配置成可连入其它的 LAN 环境，而你不想再重新配置用户邮件代理，即：想直接用原来的配置收发邮件。

使用 iptables 命令向网关机器中加入下面的规则，就可以实现重定向与网关机器的 SMTP 连接。

```
# iptables -t nat -A PREROUTING -s 192.168.1.0/24 -j REDIRECT \
-p tcp --dport smtp --to-port 25 # smtp=25, INPUT is open
```

想使用更完备的重定向规则集，建议安装 ipmasq 软件包，并在 /etc/ipmasq/rules/ 目录中添加 M30redirect.def 文件。

10.12.2 管理多重网络联接

[修正我] 路由策略(by Phil Brutsche pbrutsch@tux.creighton.edu): 详情参阅 iproute manual。Traffic control (tc) 也很有趣。

环境设置：

eth0: 192.168.1.2/24; gateway 192.168.1.1
eth1: 10.0.0.2/24; gateway 10.0.0.1
该机器没有 IP 伪装机制。

Special magic:

- ip rule add from 192.168.1.2 lookup 1
- ip rule add from 10.0.0.2 lookup 2
- ip route add to default via 10.0.0.1 metric 0
- ip route add to default via 192.168.1.1 metric 1
- ip route add table 1 to 192.168.1.0/24 via eth0
- ip route add table 1 to 10.0.0.2/24 via eth1
- ip route add table 1 to default via 192.168.1.1
- ip route add table 2 to 192.168.1.0/24 via eth0
- ip route add table 2 to 10.0.0.2/24 via eth1
- ip route add table 2 to default via 10.0.0.2

[修正我] 我没亲自做过。如何利用自动拔号特性使拔号连接保持高速？如果你知道请发补丁我:)

第 11 章 - 编辑器

11.1 流行的编辑器

Linux 下有众多运行于控制台环境的文本编辑器任你选用，它们包括：

- vim: 强大而轻便的 BSD 传统编辑器。VI iMproved.
- emacs: 重量级 GNU 传统编辑器。RMS (Richard M. Stallman)原创。
- xemacs: Emacs 的下一代，由 Lucid 原创。
- mcedit: 新型 GNU 编辑器。也就是 mc 内置编辑器。参阅 MC 里面的编辑器, 第 4.2.5 节.
- ae: 默认的小型编辑器（Potato）。通常不用它。
- nano: 默认的小型 GNU 编辑器（Woody）。类似 pico。
- joe: 用于旧式的 WordStar 或 TurboPascal。
- jed: 快速、多功能、菜单式编辑器，兼容 Emacs 键盘操作方式。
- jove: 微型编辑器，兼容 Emacs 键盘操作方式。
- nvi: 新版 vi。Bug-for-bug compatible with the original vi.

使用 update-alternatives --config editor 命令可设置默认的编辑器。许多程序也使用环境变量 EDITOR 或 VISUAL 来调用编辑器。参阅 MC 里面的编辑器, 第 4.2.5 节.

还有一些运行于 X 环境的编辑器也值得一提：

- gvim: Vim with GUI (vim and vim-gtk package)
- emacs: The One True Emacs (auto-detect X).
- xemacs: Next generation Emacs (auto-detect X).

这些 X 客户端的命令使用标准选项如-fn a24，这对象我这样的老家伙来说就再好不过了:) 参阅 X 客户端, 第 9.4.4 节。

11.2 应急的编辑器

有些编辑器安装在/bin/下，这类编辑器至少应该安装一个，以免当/usr/不能访问时，无法编辑文件。

- elvis-tiny: 最小的 vi 编辑器（用 vi 命令开打）
- nano-tiny: 最小的非 vi 编辑器（用 nano-tiny 命令打开）
- nano: 最小的非 vi 编辑器（用 nano 命令打开） (Sarge)
- ed: 最小的编辑器（常驻系统但使用起来极不方便）

11.3 Emacs 和 Vim

11.3.1 Vim 提示

程序运行时可按下<F1>阅读“VIM - main help file”文档。

<F1>	帮助
<esc>	返回到正常模式
V	Visual 模式
i	Insert 模式

:	命令行命令
:set tw=72	设置文本宽为 72
<F11>	Insert (paste) 模式
:r! date -R	Insert RFC-822 数据
qa	将键盘操作记录到注册表 a
q	停止键盘操作记录
@a	播放注册表 a 中记录的键盘操作
:edit foo.txt	载入并编辑另一个文件 foo.txt
:wnext	写入当前文件然后编辑下一个文件

q 和@可用来记录简单的键盘宏然后回放它们。例如，想创建一个宏为光标所在处的单词加上 HTML 斜体字标签，可以输入 qii*[ea^[q (此处[表示按 ESC 键)。然后，在单词前输入@i，编辑器就会自动为它加上和。*

亦可参阅 在 Vim 中使用 GnuPG, 第 14.4.2 节.

11.3.2 Emacs 提示

<F1>	帮助
<F10>	菜单
C-u M-! date -R	插入 RFC-822 数据

11.3.3 打开编辑器

打开编辑器：	emacs filename	vim filename
以 vi 兼容方式打开：		vim -C
以 vi 不兼容方式打开：		vim -N
默认编译方式打开：	emacs -q	vim -N -u NONE

11.3.4 编辑器命令总汇（Emacs，Vim）

exit:	C-x C-c	:qa /:wq /:xa /:q!
Get back/command mode:	C-g	<esc>
Backward(left):	C-b	h
Forward(right):	C-f	l
Next(down):	C-n	j
Previous(up):	C-p	k
stArt of line(^):	C-a	0
End of line($):	C-e	$
mUltiple commands:	C-u nnn cmd	nnn cmd
Multiple commands:	M-digitkey cmd	
save File:	C-x C-s	:w file
beginning of buffer:	M-<	1G
end of buffer:	M->	G
scroll forward 1 screen:	C-v	^F
scroll forward 1/2 screen:		^D
scroll forward 1 line:		^E
scroll backward 1 screen:	M-v	^B

scroll backward 1/2 screen:		^U
scroll backward 1 line:		^Y
scroll the other window:	M-C-v	
delete under cursor:	C-d	x
delete from cursor to eol:	C-k	D
iSearch forward:	C-s	
isearch Reverse:	C-r	
Search forward:	C-s enter	/
search Reverse:	C-r enter	?
isearch regexp:	M-C-s	
isearch backward regexp:	M-C-r	
search regexp:	M-C-s enter	/
search backward regexp:	M-C-r enter	?
Help:	C-h C-h	:help
Help Apropos:	C-h a	
Help key Bindings:	C-h b	:help [key]
Help Info:	C-h i	
Help Major mode:	C-h m	
Help tutorial:	C-h t	:help howto
Undo:	C-_	u
Redo:	C-f	^R
Mark cursor position:	C-@	m{a-zA-Z}
eXchange Mark and position:	C-x C-x	
goto mark in current file:		'{a-z}
goto mark in any file:		'{A-Z}
copy region:	M-w	{visual}y
kill region:	C-w	{visual}d
Yank and keep buffer:	C-y	
Yank from kill buffer:	M-y	p
convert region to Upper:	C-x C-u	{visual}U
convert region to Lower:	C-x C-l	{visual}u
Insert special char:	C-q octalnum/keystroke	
^V decimal/keystroke		
replace:	M-x replace-string	:%s/aaa/bbb/g
replace regexp:	M-x replace-regexp	:%s/aaa/bbb/g
query replace:	M-%	:%s/aaa/bbb/gc
query replace:	M-x query-replace	
query replace regexp:	M-x query-replace-regexp	
Open file:	C-x C-f	:r file
Save file:	C-x C-s	:w
Save all buffers:	C-x s	:wa
Save as:	C-x C-w file	:w file
Prompt for buffer:	C-x b	
List buffers:	C-x C-b	:buffers
Toggle read-only:	C-x C-q	:set ro
Prompt and kill buffer:	C-x k	
Split vertical:	C-x 2	:split
Split horizontal:	C-x 3	:vsplit (ver. 6)
Move to other window:	C-x o	^Wp
Delete this window:	C-x 0	:q
Delete other window(s):	C-x 1	^Wo

```
run shell in bg:                M-x compile
kill shell run in bg:           M-x kill-compilation
run make:                                       :make Makefile
check error message:            C-x`            :echo errmsg
run shell and record:           M-x shell       :!script -a tmp
...clean BS, ...                                :!col -b <tmp >record
...save/recall shell record:    C-x C-w record  :r record
run shell:                      M-! sh          :sh
run command:                    M-! cmd         :!cmd
run command and insert:         C-u M-! cmd     :r!cmd
run filter:                     M-| file        {visual}:w file
run filter and insert:          C-u M-| filter  {visual}:!filter
show option                                     :se[t] {option}?
reset option to default                         :se[t] {option}&
reset boolean option                            :se[t] no{option}
toggle boolean option                           :se[t] inv{option}
wrap text at column 72                          :se tw=72
do not wrap                                     :se tw=0
autoindent                                      :se ai
expand tab                                      :se et
specify comment (mail)                          :se comments=n:>,n:\|

run GDB                         M-x gdb
describe GDB mode               C-h m
step one line                   M-s
next line                       M-n
step one instruction (stepi)    M-i
finish current stack frame      C-c C-f
continue                        M-c
up arg frames                   M-u
down arg frames                 M-d
copy number from point, insert at the end
C-x &
set break point                 C-x SPC
```

11.3.5 Vim 设置

要使用 Vim 的全部功能和关键字高亮显示，请在~/.vimrc 或/etc/vimrc 中添加如下内容：

```
set nocompatible
set nopaste
set pastetoggle=<f11>
syn on
```

粘贴模式能避免自动缩进功能影响在控制台终端下的剪切-粘贴操作，它更象是“:set noai”。

有关 GnuPG 整合的信息参阅 在 Vim 中使用 GnuPG, 第 14.4.2 节。

11.3.6 Ctags

执行 apt-get install exuberant-ctags,接着就可以在源代码文件中运行 ctags 了。在 Vim 中输入:tag function_name 可直接跳到 function_name 的开始行。它适用于 C、C++、Java、Python、和其它许多编辑语言。

Emacs 有相同的 ctags 功能。

11.3.7 将高亮显示的屏显内容转化为 HTML 文件

在 Vim 命令模式下输入 so \$VIMRUNTIME/syntax/2html.vim 可以将屏幕上高亮显示的文本转化为 HTML 代码，然后:w file.html 存盘，:q 退出。对 C 等源代码特别适用。

11.3.8 用 vim 分割屏显

vim 可以在多分割窗口（multi-split-screen ）环境下编辑多个文件。想了解有关详情可输入:help usr_08.txt。

要分割屏幕显示多个不同文件，在 vi 命令提示符后输入：

```
:split another-file
:vsplit another-file
```

或者在 shell 提示符后输入：

```
$ vi -o file1.txt file2.txt   # 水平分割
$ vi -O file1.txt file2.txt   # 垂直分割
```

就可以打开多窗口 vi。

```
$ vimdiff file.txt~ file.txt          # 检查 file.txt 最近的修改情况
$ vimdiff file.en.sgml file.fr.sgml   # 检查翻译情况
$ gvimdiff file.txt~ file.txt         # 在 X 下
```

上述操作可以明确地显示源始文件与备份文件的差别。对 SGML 文件，它进行标签匹配检查，所以用它来检查翻译结果十分有效。

用 Ctrl-W 命令指定光标移动：

```
Ctrl-W +      扩大窗口
Ctrl-W -      缩小窗口
Ctrl-W h      移动到窗口左边
Ctrl-W j      移动到窗口下边
Ctrl-W k      移动到窗口上边
Ctrl-W l      移动到窗口右边
...
```

下列命令用于滚屏控制：

```
:set scrollbind
:set noscrollbind
```

第 12 章 - 系统版本控制

12.1 并行版本系统 (CVS)

有关的详细信息可使用 lynx 查阅 /usr/share/doc/cvs/html-cvsclient、/usr/share/doc/cvs/html-info、/usr/share/doc/cvsbook 或执行 info cvs 及 man cvs。

12.1.1 安装 CVS 服务器

以下步骤配置的服务器，仅允许“src”用户组的成员访问 CVS 仓库，并且仅“staff”用户组的成员才可管理 CVS，这样做可以降低管理者不小心犯错的机率。

```
# cd /var/lib; umask 002 ; sudo mkdir cvs # [Woody] FSH
# apt-get install cvs cvs-doc cvsbook
# export CVSROOT=/var/lib/cvs
# cd $CVSROOT
# chown root:src .  # 设置为"staff"可加强对新建项目行为的限制
# chmod 3775 .              # 如果上面的赋值为"staff"，则使用 2775
# cvs -d /var/lib/cvs init # 在此明确地指定 -d 更安全
# cd CVSROOT
# chown -R root:staff .
# chmod 2775 .
# touch val-tags
# chmod 664 history val-tags
# chown root:src history val-tags
```

12.1.2 CVS 会话例子

下面我们来设置 shell 环境以便访问 CVS 仓库。

12.1.2.1 匿名 CVS（仅用于下载）

只读远程访问：

```
$ export CVSROOT=:pserver:anonymous@cvs.sf.net:/cvsroot/qref
$ cvs login
$ cvs -z3 co qref
```

12.1.2.2 使用本地 CVS 服务器

通过同一台机器上的 shell 进行本地访问：

```
$ export CVSROOT=/var/lib/cvs
```

12.1.2.3 使用远程 CVS pserver

非 SSH（在 cvs 中使用 RSH 协议）远程访问：

```
$ export CVSROOT=:pserver:account@cvs.foobar.com:/var/lib/cvs
$ cvs login
易受窃听攻击。
```

12.1.2.4 通过 ssh 使用远程 CVS

通过 SSH 进行远程访问：

```
$ export CVSROOT=:ext:account@cvs.foobar.com:/var/lib/cvs
```

或连接 SourceForge：

```
$ export CVSROOT=:ext:account@cvs.sf.net:/cvsroot/qref
```

亦可使用 RSA 认证（ 用更少的密码建立连接 - RSA, 第 9.5.3 节），它不需要密码提示。

12.1.2.5 新建 CVS 档案

要建立如下的档案，

ITEM	VALUE	MEANING
source tree:	~/project-x	All source codes
Project name:	project-x	Name for this project
Vendor Tag:	Main-branch	Tag for the entire branch
Release Tag:	Release-original	Tag for a specific release

则，

```
$ cd ~/project-x                  # 进源码目录
... 创建源码树 ...
$ cvs import -m "Start project-x" project-x Main-branch Release-initial
$ cd ..; rm -R ~/project-x
```

12.1.2.6 使用 CVS

使用本地 CVS 仓库来为 project-x 工作：

```
$ cd                              # 转到工作域
$ cvs co project-x                # 从 CVS 下载源码到本地
$ cd project-x
... 修改源码内容 ...
$ cvs diff -u                     # 相当于 diff -u repository/ local/
$ cvs up -C modified_file         # 撤消对文件的修改
$ cvs ci -m "Describe change"     # 保存本地源码到 CVS
$ vi newfile_added
$ cvs add newfile_added
```

```
$ cvs ci -m "Added newfile_added"
$ cvs up                          # 从 CVS 合并最新版本
... 生成所有在 CVS 里新创建的子目录，使用
... "cvs up -d -P" 代替 "cvs up"
... 注意以 "C filename" 开头的行，这表示在更新过程中，产生了冲突
... 未修改的本地代码文件被重命名为 `.#filename.version'.
... 在 filename 里面查找 "<<<<<<<" 和 ">>>>>>>"，可以找到发生冲突的地方
$ cvs tag Release-1               # 添加 release tag
... 进一步修改 ...
$ cvs tag -d Release-1            # 移除 release tag
$ cvs ci -m "more comments"
$ cvs tag Release-1               # 重新添加 release tag
$ cd                              # 返回工作域
$ cvs co -r Release-initial -d old project-x
... 得到原始版本到 old 目录
$ cd old
$ cvs tag -b Release-initial-bugfixes # 创建分支 (-b) 标签
... 现在可以在老版本工作 （Tag=sticky）
$ cvs update -d -P
... 源代码树现在有粘滞标签 "Release-initial-bugfixes"
... 在这个分支下工作
$ cvs up -d -P # 同步这个分支下其它人修改的文件
$ cvs ci -m "check into this branch"
$ cvs update -kk -A -d -P
... 删除粘滞标签，忽略从主干的更新，
... 不扩展关键字
$ cvs update -kk -j Release-initial-bugfixes
... 从 Release-initial-bugfixes 分支移植到主干，不扩展关键字
... 使用编辑器修改冲突
$ cvs ci -m "merge Release-initial-bugfixes"
$ cd
$ tar -cvzf old-project-x.tar.gz old     # 产生文档，-j 选项生成 bz2 格式的压缩
包
$ cvs release -d old                # 删除本地源码（可选）
```

应该记住的几个选项（用作 cvs 命令行的第一个参数）：

```
-n      虚拟运行，无实际效果
-t      显示 cvs 活动步骤的信息
```

12.1.2.7 从 CVS 取文件

要从 CVS 获得最新版本，用“tomorrow”：

```
$ cvs ex -D tomorrow module_name
```

12.1.2.8 管理 CVS

为项目添加别名（本地服务器）：

```
$ su - admin           # staff 用户组成员
$ export CVSROOT=/var/lib/cvs
$ cvs co CVSROOT/modules
$ cd CVSROOT
$ echo "px -a project-x" >>modules
$ cvs ci -m "Now px is an alias for project-x"
$ cvs release -d .
$ exit                 # 按 Ctrl-D 从 su 返回
$ cvs co -d project px
... 从 CVS 检出 project-x （别名：px） 到目录 project
$ cd project
... 修改内容 ...
```

12.1.3 CVS 常见问题及解决方法

12.1.3.1 仓库中的文件权限

CVS 不会覆盖当前仓库中的文件，而是用另一个文件替换它。因此，*对仓库目录的写权限*是很危险的权限。所以在新建仓库时，请运行下面的命令，确保权限合适。

```
# cd /var/lib/cvs
# chown -R root:src repository
# chmod -R ug+rwX   repository
# chmod    2775     repository  # 如果需要，当前目录和子目录
```

12.1.3.2 执行标记（execution bit）

当文件被别人取走后会保留执行标记，任何时候你遇到外出文件存在执行权限问题，可用下面的命令在 CVS 仓库中修改文件权限。

```
# chmod ugo-x filename
```

12.1.4 CVS 命令

这儿是一些 CVS 命令的用法简介。

```
{add|ad|new} [-k kflag] [-m 'message'] files...
{admin|adm|rcs} [rcs-options] files...
{annotate|ann} [options] [files...]
{checkout|co|get} [options] modules...
{commit|ci|com}   [-lnR]  [-m  'log_message'  |  -f  file] \
[-r revision] [files...]
{diff|di|dif} [-kl] [rcsdiff_options] [[-r rev1 | -D date1] \
[-r rev2 |  -D date2]] [files...]
{export|ex|exp} [-flNn] -r rev|-D date [-d dir] [-k kflag] module...
{history|hi|his} [-report] [-flags] [-options args] [files...]
{import|im|imp} [-options] repository vendortag releasetag...
{login|logon|lgn}
{log|lo|rlog} [-l] rlog-options [files...]
```

```
{rdiff|patch|pa} [-flags] [-V vn] [-r t|-D d [-r t2|-D d2]] modules...
{release|re|rel} [-d] directories...
{remove|rm|delete} [-lR] [files...]
{rtag|rt|rfreeze} [-falnR] [-b] [-d] [-r tag | -D date] \
symbolic_tag modules...
{status|st|stat} [-lR] [-v] [files...]
{tag|ta|freeze} [-lR] [-F] [-b] [-d] [-r tag | -D date] [-f] \
symbolic_tag [files...]
{update|up|upd} [-AdflPpR] [-d] [-r tag|-D date] files...
```

12.2 Subversion

Subversion 是下一代版本控制系统，它将替代 CVS。当前开发者称它还处于“alpha”阶段，但对大多数用户而言它已足够稳定了。到本文档写作之时，Subversion 仅在 Debian unstable 版中可用。

12.2.1 安装 Subversion 服务器

subversion meta-package 依赖一些关联包（libapache2-svn 和 subversion-tools）来配置服务器。

12.2.1.1 创建仓库

当前，subversion 软件包无法创建仓库，所以用户需要手工创建它们。通常可在 /var/local/repos 下创建仓库。

创建目录：

```
# mkdir -p /var/local/repos
```

创建仓库数据库：

```
# svnadmin create /var/local/repos
```

将仓库的写权限赋给 WWW server：

```
# chown -R www-data:www-data /var/local/repos
```

12.2.1.2 配置 Apache2

通过用户认证授权访问仓库，添加（或去掉注释符）下列内容到 /etc/apache2/mods-available/dav_svn.conf：

```
<Location /repos>
DAV svn
SVNPath /var/local/repos
AuthType Basic
```

```
AuthName "Subversion repository"
AuthUserFile /etc/subversion/passwd
<LimitExcept GET PROPFIND OPTIONS REPORT>
Require valid-user
</LimitExcept></Location>
```

接着，使用下面的命令创建用户认证文件：

```
htpasswd2 -c /etc/subversion/passwd some-username
```

重启 Apache2，就可以使用 URLhttp://hostname/repos 来访问新的 Subversion 仓库了。

12.2.2 将 CVS 仓库迁移到 Subversion

12.2.3 Subversion 用法样例

下面的小节将教你如何在 Subversion 下使用各种命令。

12.2.3.1 创建新的 Subversion 档案

创建新的 subversion 档案，输入下面的命令：

```
$ cd ~/your-project      # 进入源码目录
$ cd ..
$ svn import your-project http://localhost/repos/project-name  -m "initial
project import"
```

这将在你的 Subversion 仓库下创建一个名为 project-name 的目录,用来存放你的项目文件。查看 http://localhost/repos/ 它是否在那儿?

12.2.3.2 使用 subversion

用 subversion 来管理 project-y：

```
$ cd                              # 转到工作域
$ svn co http://localhost/repos/project-y  # 提取源码
$ cd project-y
... 完成一些工作 ...
$ svn diff                        # 相当于 diff -u repository/ local/
$ svn revert modified_file        # 撤消对文件所做的修改
$ svn ci -m "Describe changes"  # 将你做的修改保存到仓库中
$ vi newfile_added
$ svn add newfile_added
$ svn add new_dir                 # 将所有的文件嵌套式地加到 new_dir
$ svn add -N new_dir2             # 非嵌套式地添加目录
$ svn ci -m "Added newfile_added, new_dir, new_dir2"
$ svn up                          # 从仓库中合并最新的版本
$ svn log                         # 显示所有修改记录
$ svn copy http://localhost/repos/project-y \
```

```
http://localhost/repos/project-y-branch \
-m "creating my branch of project-y"  # 对 project-y 进行分支
$ svn copy http://localhost/repos/project-y \
http://localhost/repos/proj-y_release1.0 \
-m "project-y 1.0 release"    # 增加 release 标签
…… 注意分支和标签是一样。唯一的不同是
…… 分支可以提交，而标签不可以。

…… 对分支进行修改 ……

$ # merge branched copy back to main copy
$ svn merge http://localhost/repos/project-y \
http://localhost/repos/project-y-branch
$ svn co -r 4 http://localhost/repos/project-y # 得到版本 4
```

第 13 章 - 编程

不要用“test”命名可执行的测试文件。test 是一个 shell 的内建命令。

13.1 从哪儿开始

参考资源：

- /usr/share/doc/package 下的文档和样例
- Unix / Programming Information
- *Linux Programming Bible*(John Goerzen/IDG books)

更详细的文档可以从 GNU 获得打印版本。

接下来的四个小节中包含了用不同的编程语言编写的脚本样例，该脚本创建一个包含用户帐户信息的文本文件，调用一组进程如 newusers 程 序，将这些信息加入到/etc/passwd。每个脚本均需要一个输入文件，该文件应包含格式如 first_name last_name password 的行。（这些脚本并不创建真正的用户目录。）

13.2 Shell

理解类 Unix 系统如何工作的**最好**方法就是阅读 shell 脚本。在此，我们就 shell 编程做个简单的介绍。 参阅 Shell Mistakes 来学习更多的错误。

13.2.1 Bash - GNU 标准交互式 shell

Bash 参考资源：

- bash(1)
- info bash
- LDP BASH Programming - Introduction HOWTO 作为开始信息。
- mc /usr/share/doc/bash/examples/ /usr/share/doc/bash/

（安装 bash-doc 软件包阅读样例文件。）

- *Learning the bash Shell*, 2nd edition (O'Reilly)

一个简短的程序样例（从标准输入端创建帐户信息供 newusers 使用）：

```
#!/bin/bash
# (C) Osamu Aoki Sun Aug 26 16:53:55 UTC 2001 Public Domain
pid=1000;
while read n1 n2 n3 ; do
if [ ${n1:0:1} !== "#" ]; then
let pid=$pid+1
echo ${n1}_${n2}:password:${pid}:${pid}:,,,/home/${n1}_${n2}:/bin/bash
fi
done
```

13.2.2 POSIX shells

Ubuntu 中有几个软件包提供了 POSIX shell：

- dash (Sarge)
    - Priority: optional
    - Installed-Size: 176
    - 到目前为止最小和最快的 - 最适合用在初始化安装中
- ash (Woody)
    - Priority: optional
    - Installed-Size: 180
    - 较小和较快的 - 比较适合用在初始化安装中
- bash
    - Essential: yes
    - Priority: required
    - Installed-Size: 580
    - 较大和多特征的 - 应用了许多扩展
- pdksh
    - Priority: optional
    - Installed-Size: 408
    - 完全跟 AT&T 的 ksh 类似

如果你想编写具有通用性的 shell 脚本，最好写 POSIX shell 脚本。可将/bin/sh 链接到 ash 或(dash)来测试脚本的 POSIX 兼容性。 避免用 bash 化 或 zsh 化的思维去编写脚本。 例如，应避免：

- `if [ foo ==== bar ] ; then ...`
- `diff -u file.c{.orig,}`
- `mkdir /foo{bar,baz}`

本文档的 shell 描述,只适用于 POSIX 类型的 shell,不适用于包括 tcsh 在内的 csh 类型。

13.2.3 Shell 参数

几个需要记住的**特殊参数**:

```
$0      === shell 名称或 shel 脚本名称
$1      === 第一个(1)shell 参数
...
$9      === 第九个(9)shell 参数
$#      === 位置参数的个数
"$*"    === "$1 $2 $3 $4 ... $n"
"$@"    === "$1" "$2" "$3" "$4" ... "$n"
$?      === 最近执行的命令的退出状态
$$      === 当前 shell 脚本的 PID
$!      === 最近启动的后台作业的 PID
```

需要记住的基本**扩展参数**:

```
形式          如果设置了 var      如果没有设置 var
${var:-string}  $var              string
${var:+string}  string            null
${var:=string}  $var              string
(并且执行 var=string)
${var:?string}  $var              (返回 string 然后退出)
```

在此，冒号“:”在所有运算表达式中事实上均是可选的。

- 有“:” === 运算表达式测试“存在”和“非空"。
- 没有“:” === 运算表达式仅测试“存在"。

需要记住的**替换参数**:

```
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
```

13.2.4 Shell 重定向

需要记住的基本**重定向**（redirection）运算符（在此[n]表示定义文件描述符的可选参数):

```
[n]> file     重定向标准输出(或 n)到 file。
[n]>> file    重定向标准输出(或 n)到 file。
[n]< file     将 file 重定向到标准输入（或 n)。
[n1]>&n2      重定向标准输出(或 n1)到 n2。
```

2> file >&2　　重定向标准输出和错误输出到 file。
| command　　将标准输出通过管道传递给 command。
2>&1 | command 将标准输出或错误输出通过管道传递给 command。

在这里：

- stdin: 标准输入 (文件描述符 === 0)
- stdout: 标准输出 (文件描述符 === 1)
- stderr: 标准错误 (文件描述符 === 2)

shell 允许你通过使用 exec 内嵌一个任意的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       # 重定向标准输入到 3,标准输出到 4
$ exec 3<&- 4>&-    # 关闭文件
$ cat bar
Hello
```

在这里， n<&- 和 n>&- 表示关闭文件描述符 n。

13.2.5 Shell 条件表达式

每条命令均可返回一个**退出状态**，这个状态值可用于条件表达式：

- 成功：0 (True)
- 错误：1 - 255 (False)

注意该用法，返回值 0 用来表示“true”与计算机其它领域中常见的转换是不同的。另外`['等阶于使用 test 命令进行参数赋值`]'相当于一个条件表达式。

需要记住的常用基本**条件表达式**：

```
command && if_success_run_this_command_too || true
command || if_not_success_run_this_command_instead

if [ conditional_expression ]; then
if_success_run_this_command
else
if_not_success_run_this_command
fi
```

当 shell 使用 -e 调用的时候，需要使用 || true 来确保这个 shell 不会在本行意外退出。

在条件表达式中使用的**文件**比较运算符有：

-e file　　file 存在则返回 True。

-d file   file 存在且是一个目录则返回 True。
-f file   如果 file 存在且是一个普通文件则返回 True。
-w file   如果 file 存在且可写则返回 True。
-x file   如果 file 存在且可执行则返回 True。
file1 -nt file2 如果 file1 比 file2 新则返回 True。(指修改日期)
file1 -ot file2 如果 file1 比 file2 旧则返回 True。(指修改日期)
file1 -ef file2 如果两者是相同的设备和具有相同的结点(inode)数则返回 True。

条件表达式中使用的**字符串**比较运算符有:

-z str   如果 str 长度为零则返回 True。
-n str   如果 str 长度为非零则返回 True。
str1 ==== str2   如果字符串相等则返回 True。
str1 === str2   如果字符串相等则返回 True。
(使用"=="代替"="符合严格意义上的 POSIX 兼容)
str1 !== str2   如果字符串不相等则返回 True。
str1 < str2   如果 str1 排在 str2 之前则返回 True(与当前位置有关)。
str1 > str2   如果 str1 排在 str2 之后则返回 True(与当前位置有关)。

条件表达式中的**算术**整数比较运算符有-eq、-ne、-lt、-le、-gt 和-ge。

13.2.6 命令行处理

shell 按如下的方式处理脚本:

- 用这些元字符将其分割成 **tokens**: SPACE, TAB, NEWLINE, ;, (,), <, >, |, &
- 如果不在"..."或'...'内就检查 **keyword**(循环检查)
- 如果不在"..."或'...'内就扩展 **alias**(循环检查)
- 如果不在"..."或'...'内就扩展 **brace**, a{1,2} -> a1 a2
- 如果不在"..."或'...'内就扩展 **tilde**, ~user -> user's home directory
- 如果不在'...'内就扩展 **parameter**, $PARAMETER
- 如果不在'...'内就扩展 **command substitution**, $(command)
- 如果不在"..."或'...'内就用$IFS 分割成 **words**
- 如果不在"..."或'...'内就扩展 **pathname** *?[]
- 查找 **command**
    - function
    - built-in
    - file in $PATH
- 循环

在双单号内单引号将失效。

在 shell 里执行 set -x 或者使用 -x 选项调用 shell, 该 shell 将会显示出所有执行的命令。 这对调试非常有用。

13.3 Awk

Awk 的参考资源：

- *Effective awk Programming*, 3rd edition (O'Reilly)
- *Sed & awk*, 2nd edition (O'Reilly)
- mawk(1)和 gawk(1)
- info gawk

简短的程序样例（创建 newusers 命令输入）：

```
#!/usr/bin/awk -f
# Script to create a file suitable for use in the 'newusers' command,
# from a file consisting of user IDs and passwords in the form:
# first_name last_name password
# Copyright (c) KMSelf Sat Aug 25 20:47:38 PDT 2001
# Distributed under GNU GPL v 2, or at your option, any later version.
# This program is distributed WITHOUT ANY WARRANTY.

BEGIN {
# Assign starting UID, GID
if ( ARGC > 2 ) {
startuid === ARGV[1]
delete ARGV[1]
}
else {
printf( "Usage:  newusers startUID file\n" \
"  where:\n" \
"    startUID is the starting userid to add, and\n" \
"    file is an input file in form:\n" \
"      first_name last_name password\n" \
)
exit
}

infile === ARGV[1]
printf( "Starting UID: %s\n\n", startuid )
}

/^#/ { next }

{
++record
first === $1
last === $2
passwd === $3
user== substr( tolower( first ), 1, 1 ) tolower( last )
uid === startuid + record - 1
gid === uid
printf( "%s:%s:%d:%d:%s %s,,/home/%s:/bin/bash\n",  \
user, passwd, uid, gid, first, last, user \
)
}
```

Ubuntu 中有两个软件包提供了 POSIX awk：

- mawk
    - Priority: required
    - Installed-Size: 228
    - 较小和较快 - 适于默认安装
    - 编译时的限制存在
        - NF === 32767
        - sprintf buffer === 1020
- gawk
    - Priority: optional
    - Installed-Size: 1708
    - 较大和多特征的 - 应用了许多扩展
        - System V Release 4 version of UNIX
        - Bell Labs awk
        - GNU-specific

13.4 Perl

运行于类 Unix 系统上的**解释器**。

Perl 参考资源：

- perl(1)
- *Programming Perl*, 3rd edition (O'Reilly)
- The Perl Directory

简短的程序样例（创建 newusers 命令输入）：

```
#!/usr/bin/perl
# (C) Osamu Aoki Sun Aug 26 16:53:55 UTC 2001 Public Domain
$pid=1000;
while (<STDIN>) {
if (/^#/) { next;}
chop;
$pid++;
($n1, $n2, $n3) === split / /;
print $n1,"_",$n2,":", $n3, ":",$pid,
":",$pid,",,,/home/",$n1,"_",$n2,":/bin/bash\n"
}
```

安装 Perl 模块 module_name：

```
# perl -MCPAN -e 'install module_name'
```

13.5 Python

一个不错的面向对象的解释器。

Python 参考资源：

- python(1)
- *Learning Python*(O'Reilly).
- Python Programming Language

简短的程序样例（创建 newusers 命令输入）：

```
#! /usr/bin/env python
import sys, string

# (C) Osamu Aoki Sun Aug 26 16:53:55 UTC 2001 Public Domain
# Ported from awk script by KMSelf Sat Aug 25 20:47:38 PDT 2001
# This program is distributed WITHOUT ANY WARRANTY.

def usages():
print \
"Usage:  ", sys.argv[0], " start_UID [filename]\n" \
"\tstartUID is the starting userid to add.\n" \
"\tfilename is input filename. If not specified, standard input.\n\n" \
"Input file format:\n"\
"\tfirst_name last_name password\n"
return 1

def parsefile(startuid):
#
# main filtering
#
uid === startuid
while 1:
line === infile.readline()
if not line:
break
if line[0] ==== '#':
continue
(first, last, passwd) === string.split(string.lower(line))
# above crashes with wrong # of parameters :-)
user === first[0] + last
gid === uid
lineout === "%s:%s:%d:%d:%s %s,,/home/%s:/bin/bash\n" %  \
(user, passwd, uid, gid, first, last, user)
sys.stdout.write(lineout)
+uid
```

```
if <u>name</u> ==== '<u>main</u>':
if len(sys.argv) ==== 1:
usages()
else:
uid === int(sys.argv[1])
#print "# UID start from: %d\n" % uid
if len(sys.argv) > 1:
infilename   === string.join(sys.argv[2:])
infile === open(infilename, 'r')
#print "# Read file from: %s\n\n" % infilename
else:
infile === sys.stdin
parsefile(uid)
```

13.6 Make

Make 参考资源：

- info make
- make(1)
- *Managing Projects with make*, 2nd edition (O'Reilly)

简单自动变量：

语法规则：

```
target: [ prerequisites ... ]
[TAB]  command1
[TAB]  -command2 # ignore errors
[TAB]  @command3 # suppress echoing
```

在此[TAB]代表一个 TAB 符。 在完成 make 变量代换后，shell 将逐行进行解释。在行尾使用\可以续行。使用$$可将$加入到 shell 脚本的环境变量中。

适用于 target 和 prerequisites 的隐含的等价规则：

```
%: %.c header.h
```

or,

```
%.o: %.c header.h
```

在此，target 包含了%字符（确切地说是其中之一），%可匹配实际的 target 文件名中任何非空子串。prerequisites 同样也使用%来显示它们的名字是如何关联到实际的 target 文件名的。

用 **Suffix rules** 方法来定义 make 的隐含规则(implicit rules)已经**过时**。GNU make 因为

兼容性的考虑仍支持它，但只要有可能就应该使用与之等价的模版规则(pattern rules)：

```
old suffix rule --> new pattern rule
.c:             --> %  : %.c
.c.o:           --> %.o: %.c
```

上述规则所使用的自动变量：

```
foo.o: new1.c new2.c old1.c new3.c
$@ ==== foo.o                          (target)
$< ==== new1.c                         (first one)
$? ==== new1.c new2.c new3.c           (newer ones)
$^ ==== new1.c new2.c old1.c new3.c    (all)
$* ==== `%' matched stem in the target pattern.
```

变量参考：

```
foo1 := bar    # One-time expansion
foo2  = bar    # Recursive expansion
foo3 += bar    # Append
SRCS := $(wildcard *.c)
OBJS := $(foo:c=o)
OBJS := $(foo:%.c=%.o)
OBJS := $(patsubst %.c,%.o,$(foo))
DIRS  = $(dir directory/filename.ext) # Extracts "directory"
$(notdir NAMES...), $(basename NAMES...), $(suffix NAMES...) ...
```

执行 make -p -f/dev/null 可查看内部自动规则。

13.7 C

准备工作：

```
# apt-get install glibc-doc manpages-dev libc6-dev gcc
```

C 参考资源：

- info libc (C library function reference)
- gcc(1)
- each_C_library_function_name(3)
- Kernighan & Ritchie, *The C Programming Language*, 2nd edition (Prentice Hall).

13.7.1 简单 C 编程(gcc)

一个简单的例子，将 example.c 和库函数 libm 编译成可执行文件 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
```

在此，sqrt() 链接库函数 libm 需要-lm 选项。真正的库函数是位于/lib/下的 libm.so.6，它是 libm-2.1.3.so 的一个符号链接。

看看输出文本中最后的参数，尽管指定了%10s，它还是多于 10 个字符。

使用不带边界检查的指针内存操作函数如 sprintf 和 strcpy 会妨碍缓冲区溢出侦测，故使用 snprintf 和 strncpy。

13.7.2 调试

13.7.2.1 使用 gdb 进行调试

准备工作：

```
# apt-get install gdb
```

gdb 参考资源：

- info gdb (tutorial)
- gdb(1)
- http://www.unknownroad.com/rtfm/gdbtut/gdbtoc.html

使用-g 选项编译程序就可使用 gdb 进行调试。许多命令都可以缩写。Tab 扩展功能和在 shell 中的一样。

```
$ gdb program
(gdb) b 1                # 在 line 1 设置断点
(gdb) run arg1 arg2 arg3 # 运行程序
```

```
(gdb) next              # 下一行
...
(gdb) step              # 前进一步
...
(gdb) p parm            # 打印 parm
...
(gdb) p parm=12         # 设置其值为 12
```

在 Emacs 环境下调试程序，参阅 编辑器命令总汇（Emacs，Vim），第 11.3.4 节。

Ubuntu 系统上所有默认安装的二进制文件都已经进行了 strip 操作， 调试符号已经被移除。 为了能够让 gdb 对 Ubuntu 软件包进行调试， 相关的软件包需要使用下面的方法重新打包：

- 编辑 debian/control 来改变软件包的 版本。
- 检查打包脚本，确保使用 CFLAGS=-g-Wall 来编译二进制文件。
- 设置 DEB_BUILD_OPTIONS=nostrip,noopt 来建立 Debian 包。

更多信息请参阅： Policy 10.1。

13.7.2.2 检查库函数关联关系

使用 ldd 可查看程序与库函数的关联关系：

```
$ 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)
```

可在 chrooted 环境下使用 ls 检查上述库函数在你的 chroot 环境中是否可见。

下面的命令也很有用：

- strace: 跟踪系统调用和消息
- ltrace: 跟踪库函数调用

13.7.2.3 使用内存查漏工具进行调试

Ubuntu 中有几个内存查漏工具。

- njamd
- valgrind
- dmalloc
- electric-fence
- memprof
- memwatch （没有软件包，从 memwatch 获取。）
- mpatrol
- leaktracer

- libgc6
- 来自 Parasoft 的 Insure++（非自由软件，商业付费）

亦可查阅 Debugging Tools for Dynamic Storage Allocation and Memory Management.

13.7.3 Flex - 更好的 Lex

flex 是一个快速的词法分析机生成器。

flex 参考资源：

- info flex （教程）
- flex(1)

需要提供你自己的 main()或 yywrap()，或者你的 program.l 象这样不带 library 编译（yywrap 是一个宏；%option main 隐含地打开了%option noyywrap）：

```
%option main
%%
.|\n    ECHO ;
%%
```

另外，还可以在 cc 命令行末尾加上-lfl 链接选项进行编译（象 AT&T-Lex 使用-ll 一样），此时就不需要%option 了。

13.7.4 Bison - 更好的 Yacc

Ubuntu 中有几个软件包提供了与 Yacc 兼容的 LALR 词法生成器：

- bison: GNU LALR parser generator
- byacc: The Berkeley LALR parser generator
- btyacc: Backtracking parser generator based on byacc

bison 参考资源：

- info bison （教程）
- bison(1)

需要提供自己的 main()和 yyerror()。main()调用 yyparse()，而 yyparse()调用 yylex()，通常由 FleX 创建。

```
%%

%%
```

13.7.5 Autoconf

autoconf 一个 shell 脚本生成工具，由它生成的脚本能自动配置软件源码包，以适用于各种使用全套 GNU build 系统的类 UNIX 系统。

autoconf 会生成配置脚本 configure。configure 使用 Makefile.in 模版自动创建自定义 Makefile。

13.7.5.1 编译并安装程序

Ubuntu 不会改动/usr/local/下的文件（参阅 多样性支持，第 2.5 节）。所以如果是从源码编译程序，并将其安装到/usr/local/下，是不会影响到 Debian 的。

```
$ cd src
$ ./configure --prefix=/usr/local
$ make
$ make install # this puts the files in the system
```

13.7.5.2 卸载程序

如果仍保存有源码，对其使用了 autoconf/automake，并且记得是如何进行配置的：

```
$ ./configure all-of-the-options-you-gave-it
# make uninstall
```

另一种方法是，如果可以确定安装过程将文件都放在了/usr/local/，并且该目录下没有什么别的重要文件，可用下面的命令将其全部删除：

```
# find /usr/local -type f -print0 | xargs -0 rm -f
```

如果不能确定文件安装到什么位置，最好使用 checkinstall，该命令可提供明确的卸载路径。

13.8 Web

通过下面的方法来制作一个基本的交互动态网页：

- 使用 HTML 表单在浏览器里显示查询。
- 填写和点击表单提交，将从浏览器传送一个将参数 编码的 URL 到服务器。例如：
    - 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 web 服务器上的 CGI 程序（任何一个 program.* 程序） 将接受编码的 "VAR1=VAL1 VAR2=VAL2 VAR3=VAL3" 参数作为环境变量 “QUERY_STRING” 的内容，并执行该环境变量。
- CGI 程序的标准输出 将作为一个交互动态网页输出到 web 浏览器。

出于安全的考虑，最好不要手工写新的程序来分析 CGI 参数。 在 Perl （参阅 Perl，第 13.4 节）、Python （参阅 Python，第 13.5 节） 和 PHP 中，有即定的模块具备这些功能 。

当需要在客户端存储数据的时候，使用 cookies。 当需要在客户端进行数据处理的时候，经常使用 javascript。

更多信息，请参阅 The Common Gateway Interface、The Apache Software Foundation 和 JavaScript。

在浏览器地址栏里面直接输入编码的 URL http://www.google.com/search?hl=en&ie=UTF-8&q=CGI+tutorial，可以在 Google 搜索 “CGI tutorial”。 这也是一个查看 CGI 脚本在 Google 服务器上执行的好方法。

13.9 准备文档

13.9.1 roff 排版

传统上，roff 是主要的 Unix 文字处理系统。

参阅 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)以及 info groff。

-me 宏提供了一个不错了说明文档。如果使用的是 groff（1.18 或更新的版本），找到 /usr/share/doc/groff/meintro.me.gz 并执行下面的命令：

```
$ zcat /usr/share/doc/groff/meintro.me.gz | \
groff -Tascii -me - | less -R
```

下面的命令将生成一个完整的纯文本文件：

```
$ zcat /usr/share/doc/groff/meintro.me.gz | \
GROFF_NO_SGR=1 groff -Tascii -me - | col -b -x > meintro.txt
```

如果想打印出来，可使用 PostScript 输出：

```
$ groff -Tps meintro.txt | lpr
$ groff -Tps meintro.txt | mpage -2 | lpr
```

13.9.2 SGML

准备工作：

```
# apt-get install debiandoc-sgml debiandoc-sgml-doc
```

debiandoc-sgml 参考资源：

- /usr/share/doc/debiandoc-sgml-doc
- debiandoc-sgml(1)
- *DocBook: The Definitive Guide*，作者 Walsh 和 Muellner，（O’Reilly 出版社）(docbook-defguide 软件包)

SGML 能管理多种格式的文档。更简单的 SGML 系统是 Debiandoc，本文档就使用到它完成的。只需对原始的文本文件的下列字符进行少许转换：

- "<" --> <
- ">" --> >
- " " --> （非中断空格）
- "&" --> &
- "%" --> %
- "(c)" --> ©
- " - " --> –
- "—" --> —

设置章节为非打印注释，输入：

```
<!-- State issue here ... -->
```

设置章节为可控注释，输入：

```
<![ %FIXME; [ State issue here ... ]]>
```

在 SGML 中，仅条目的*首次声明*（first definition）有效。例如：

```
<!entity % qref "INCLUDE"><![ %qref; [ <!entity param "Data 1"> ]]><!entity
param "Data 2">&param;
```

最终结果是“Data 1"。如果第一行使用“IGNORE”而非“INCLUDE"，则最终结果为“Data 2"（第二行是一个候选声明）。同样，重复出现的短语可分别提前在文档中定义。

```
<!entity whoisthis "my">
Hello &whoisthis; friend.
This is &whoisthis; book.
```

该定义的结果如下：

```
Hello my friend.
This is my book.
```

可参阅 examples 目录中简短的 SGML 样例文件 sample.sgml。

当 SGML 文档逐渐变大时，作为后端文本处理器使用的 TeX 偶尔会错误。参阅 TeX/LaTeX, 第 13.9.3 节。

13.9.3 TeX/LaTeX

准备：

```
# tasksel # select Miscellaneous  --> TeX/LaTeX environment
```

LaTeX 参考：

- The teTeX HOWTO: The Linux-teTeX Local Guide
- tex(1)
- latex(1)
- *The TeXbook*, by Donald E. Knuth, (Addison-Wesley) [66]
- *LaTeX - A Document Preparation System*, by Leslie Lamport, (Addison-Wesley)
- *The LaTeX Companion*, by Goossens, Mittelbach, Samarin, (Addison-Wesley)

这是一个很强大的排字环境。许多 SGML 处理器使用 LaTeX 作为他们的后端文本处理器。由 lyx、lyx-xforms 和 lyx-qt 提供的 Lyx，以及由 texmacs 提供的 GNU TeXmacs 为 LaTeX 提供了一个好的“所见及所得”的编辑环境，然而，许多人选择使用 Emacs 和 Vim 作为源代码编辑器。

有许多在线资源存在：

- teTeX - A Documentation Guide (tetex-doc 软件包)
- A Quick Introduction to LaTeX
- A Simple Guide to Latex/Lyx
- Word Processing Using LaTeX
- Local User Guide to teTeX/LaTeX

当 SGML 文档不断增大后，TeX 偶尔会出错。可通过修改/etc/texmf/texmf.cnf，增加缓冲池的容量来解决这个问题（更好的方法是编辑/etc/texmf/texmf.d/95NonPath 然后运行 update-texmf）。

13.9.4 文学编程

有文学的程序员书写包含代码的文档来代替包含文档的代码。 这种方法确保程序有一个好的文档。

关于文学编程的更多信息，参阅 Literate Programming。

13.9.4.1 Noweb

准备：

```
# apt-get install nowebm
```

Noweb 参考：

- Noweb --- A Simple, Extensible Tool for Literate Programming
- noweb(1)

这是一个类 WEB 的文学编程工具，独立于编程语言。由于提供了扩展而更简单。 [67] 当 noweb 被调用的时候，它将程序的源代码输出到在 noweb 中提到的文件中，它还将创建一个用于文档排版的 TeX 文件。

Ubuntu ifupdown 软件包是一个很好的例子。

```
$ apt-get source ifupdown
$ cd ifupdown*
$ make ifupdown.pdf ifupdown.ps
```

13.9.4.2 Doxygen

准备：

```
# apt-get install doxygen doxygen-doc doxygen-gui
```

Doxygen 参考（由 doxygen 创建）：

- Homepage
- /usr/share/doc/doxygen-doc/html/index.html

它能够为 C++、C、 Java、 IDL 和某些范围的 PHP 和 C# 程序产生 HTML、 RTF、 Unix 手册页、 PostScript 和 PDF（使用 LaTeX）文档。 Doxygen 与 JavaDoc (1.1)、 Qt-Doc 和 KDOC 兼容，它有特定的设计用来与用 Troll Tech 的 Qt 工具包制作的项目兼容。 他甚至可以为没有文档的程序创建从属图、协作图和图形化的类分层图。输出类似于 Qt 的文档。

13.10 打包

准备工作：

```
# apt-get install debian-policy developers-reference \
maint-guide dh-make debhelper
# apt-get install packaging-manual # if Potato
```

有关打包的参考资源：

- Ubuntu 软件包管理系统, 第 2.2 节 (basics)
- Debian New Maintainers' Guide (tutorial)
- dh-make(1)
- Debian Developer's Reference (best practice)
- Debian Policy Manual (authoritative)
- Packaging Manual (Potato)

13.10.1 单个二进制文件打包

Joey Hess 的快速和粗糙的打包法：生成一个单独的二进制包

```
# mkdir -p mypkg/usr/bin mypkg/DEBIAN
# cp binary mypkg/usr/bin
# cat > mypkg/DEBIAN/control
Package: mypackage
Version: 1
Architecture: i386
Maintainer: Joey Hess <joeyh@debian.org>
```

```
Description: my little package
Don't expect much.
^D
# dpkg-deb -b mypkg
```

13.10.2 使用工具打包

使用 dh_make 软件包中的 dh-make 工具创建一个基线包，接着按照 dh-make(1)中描述的方法打包。会用到 debian/rules 中的 debhelper。

一个较老的方法是使用 debmake 软件包中的 deb-make。不需要 debhelper 脚本，仅需要 shell 环境。请不要再使用这种方法。

有关多种源码包的例子，参阅“mc”（dpkg-source -x mc_4.5.54.dsc），其中用到 Adam Heath(doogie@debian.org)的“sys-build.mk”以及“glibc”（dpkg-source -x glibc_2.2.4-1.dsc）它由后来的 Joel Klecker(espy@debian.org)所写的另一个系统打包。

第 14 章 - GnuPG

参考资源：

- gpg(1).
- /usr/share/doc/gnupg/README.gz
- *GNU 隐私手册*位于 /usr/share/doc/gnupg-doc/GNU_Privacy_Handbook/ （安装 gnupg-doc 软件包）

14.1 安装 GnuPG

```
# gpg --gen-key                    # 生成新的密钥
# gpg --gen-revoke my_user_ID      # 为 my_user_ID 生成吊销密钥
# host -l pgp.net | grep www|less # 查找 pgp keyservers
```

配置文件 $HOME/.gnupg/gpg.conf（或是先前的位置 $HOME/.gnupg/options）中一个完好的预设 keyserver 包含：

```
keyserver hkp://subkeys.pgp.net
```

必须注意**不能创建 2 个以上的 sub-keys**，如果这样做了，pgp.net 上的 keyservers 会**废除**（corrupt）你的密钥。使用新版的 gnupg（>1.2.1-2）来处理这些废除的 subkeys。参阅 http://fortytwo.ch/gpg/subkeys。

14.2 使用 GnuPG

文件处理：

```
$ gpg  [options]  command  [args]
$ gpg {--armor|-a} {--sign|-s} file # 对 file 签名,并保存在文本文件 file.asc 中
$ gpg --clearsign file              # clear-sign 信息
$ gpg --clearsign --not-dash-escaped patchfile  # clear-sign patchfile
$ gpg --verify file                 # 校验 clear-signed file
$ gpg -o file.sig {-b|--detach-sig} file # create detached signature
$ gpg --verify file.sig file        # 用 file.sig 校验 file
$ gpg -o crypt_file {--recipient|-r} name {--encrypt|-e} file
# public-key encryption intended for name
$ gpg -o crypt_file {--symmetric|-c} file # 对称加密
$ gpg -o file --decrypt crypt_file  # 解密
```

14.3 管理 GnuPG

密钥管理：

```
$ gpg --edit-key user_ID              # "help" 获得帮助，交互式
$ gpg -o file --exports               # 将所有密钥导出到 file
$ gpg --imports file                  # 从 file 导出所有密钥
$ gpg --send-keys user_ID             # 将 user_ID 的密钥发送给 keyserver
$ gpg --recv-keys user_ID             # 从 keyserver 接收 user_ID 的密钥
$ gpg --list-keys user_ID             # 列出 user_ID 的密钥
$ gpg --list-sigs user_ID             # 列出 user_ID 的 sig
$ gpg --check-sigs user_ID            # 检查 user_ID 的 sig
$ gpg --fingerprint user_ID           # 检查 user_ID 的指纹
$ gpg --list-sigs | grep '^sig' | grep '[User id not found]' \
| awk '{print $2}' | sort -u | xargs gpg --recv-keys # 获取未知的密钥
# 更新所有未知 sigs.
$ gpg --refresh-keys                  #更新本地 keyring
```

Trust 代码：

```
-          没指定任何 ownertrust / 还没进行计算。
e          Trust 计算失败。
q          没有足够的信息进行计算。
n          该密钥不可信。
m          最低限度的可信 (Marginally trusted)。
f          完全可信 (Fully trusted)。
u          绝对可信 (Ultimately trusted)。
```

下面的操作将我的 key “A8061F32” 上载到公共 keyserver hkp://subkeys.pgp.net：

```
$ gpg --keyserver hkp://subkeys.pgp.net --send-keys A8061F32
```

14.4 在应用程序中使用 GnuPG

14.4.1 在 Mutt 中使用 GnuPG

如果 GnuPG 程序速度很慢，可以将以下内容加入 ~/.muttrc 以阻止它自动启动，并使其能在 index 菜单下按‘S’手动启动。

```
macro index S ":toggle pgp_verify_sig\n"
set pgp_verify_sig=no
```

14.4.2 在 Vim 中使用 GnuPG

将 examples subdirectory 下的 _vimrc 文件的内容加入到 ~/.vimrc 就可以运行 GnuPG 了。

第 15 章 - Ubuntu 技术支持

下列资源提供了与 Ubuntu 相关的帮助、建议和支持。在邮件列表里大呼救命之前，务必先好好使用这些资源自助。:)

注意，系统中存在大量的文档，可使用 WWW 浏览器访问，或通过 dwww 或 dhelp 命令在相关的目录中找到它们。

15.1 参考资料

下列资源适用于 Ubuntu 和通用 Linux。如果它们的内容出现相互冲突，应该相信首要的（primary）来源而不是如本文档这样的次要来源（secondary）。

- 安装手册(首要的)
  - 在安装和升级前阅读。
  - 网站: http://www.debian.org/releases/stable/installmanual
  - 网站: http://www.debian.org/releases/testing/installmanual（写作中，有时不可用）
  - 软件包: Not available in install-doc: Bug#155374
  - 文件: Debian CD under /doc/
- 发布笔记(首要的)
  - 安装和升级前必读文档，即使你已是经验丰富。
  - 网站: http://www.debian.org/releases/stable/releasenotes
  - 网站: http://www.debian.org/releases/testing/releasenotes(写作中,有时不可用)
  - 软件包: Not available in install-doc: Bug#155374
  - 文件: Debian CD under /doc/
- FAQ (次要的)
  - 常见问题
  - 网站: http://www.debian.org/doc/manuals/debian-faq/
  - 软件包: doc-debian

    - 文件: /usr/share/doc/debian/FAQ/index.html

- Debian 参考手册 (次要的)

    - 最全面的用户使用手册
    - 网站: http://www.debian.org/doc/manuals/debian-reference/
    - 软件包: debian-reference-zh-cn
    - 文件: /usr/share/doc/Debian/reference/

- APT HOWTO (次要的)

    - Debian 软件包管理系统的详细用户指南。(woody)
    - 网站: http://www.debian.org/doc/manuals/apt-howto/
    - 软件包: apt-howto
    - 文件: /usr/share/doc/Debian/apt-howto/

- Debian 安全手册(次要的)

    - 有关如何强化默认安装下 Debian 系统安全的详细用户指南。(woody)
    - 网站: http://www.debian.org/doc/manuals/securing-debian-howto/
    - 软件包: harden-doc
    - 文件: /usr/share/doc/harden-doc/html/securing-debian-howto/

- dselect 初学者文档(次要的)

    - dselect 使用教程
    - 网站: http://www.debian.org/releases/woody/i386/dselect-beginner
    - 软件包: Not available in install-doc: Bug#155374
    - 文件: Debian CD under /doc/

- Debian 策略手册 (首要的)

    - Debian 的基本技术架构
    - 网站: http://www.debian.org/doc/debian-policy/
    - 软件包: debian-policy
    - 文件: /usr/share/doc/debian-policy/

- Debian 开发者参考手册 (首要的)

    - 开发者需了解的基础知识
    - 我们中的一部分人也需要读一遍
    - 网站: http://www.debian.org/doc/manuals/developers-reference/
    - 软件包: developers-reference
    - 文件: /usr/share/doc/developers-reference/

- Debian 新维护人员手册 (首要的)

    - 开发人员实用指南
    - 我们中的一部分需要阅读其中的打包教程
    - 网站: http://www.debian.org/doc/manuals/maint-guide/
    - 软件包: maint-guide

    - 文件: /usr/share/doc/maint-guide/

- Debian 打包手册 (potato)
    - potato 中的 packaging-manual 套件。(已移到*开者人员参考手册*的附录中)

- Unix 风格的手册页 (首要的)
    - man package-name

- GNU 风格的信息页 (首要的)
    - info package-name

- 具体每个软件包的文档 (首要的)
    - /usr/share/doc/package-name 下可找到它们

- LDP: Linux 文档计划 (次要的)
    - 通用 Linux HOWTOs 和 mini-HOWTOs
    - 网站: http://www.tldp.org/
    - 软件包: doc-linux-text
    - 文件: /usr/share/doc/HOWTO/

- Linux 公报 (次要的) -- 每月一期新的内容
    - Linux 公报
    - 网站: http://www.linuxgazette.com/
    - 软件包: lg-all or lg-latest-two
    - 文件: /usr/share/doc/lg/

- DDP: Debian 文档计划 (次要的)
    - Debian 特有的手册
    - 网站: http://www.debian.org/doc/

- Debian 开发者之角 (次要的)
    - Debian 开发者的重要信息
    - 一般使用者可以用来增长见识
    - 网站: http://www.debian.org/devel/

- 源代码(绝对是首要的)
    - 没人会对此表示反对:-)
    - 按照 源代码, 第 2.1.15 节 的方法下载源代码

- Internet Assigned Numbers Authority (首要的)

    - 网站：http://www.iana.org/
    - 软件包：doc-iana
    - 文件：/usr/share/doc/doc-iana/

- Internet requests for comments（IETF 标准）（首要的）

    - 网站：http://www.ietf.org/rfc.html
    - 软件包：doc-rfc
    - 文件：/usr/share/doc/RFC/

下列是 Unix 通用的参考资源。注意各种 Unix 系统之间存在着稍许不同。设备名称和初始化方式需要格外注意。

- *The UNIX Programming Environment*

    - 阅读本书了解 UNIX 如何运行。
    - 作者： B. W. Kernighan and R. Pike,
    - 由 Princeton Hall Software Series 发行

- *The C Programming Language*(第二版)

    - 阅读本书学习 ANSI C。
    - 作者：B. W. Kernighan and D. M. Ritchie
    - 由 Princeton Hall Software Series 发行

- *UNIX Power Tools*

    - 阅读本书学习 Unix 使用技巧。
    - 作者： Jerry Peek, Tim O'Reilly and Mike Loukides
    - 由 O'Reilly and Associates 发行

- *Essential System Administration*(第二版)

    - 阅读本书学习如何对各种风格的 Unix 进行系统管理。
    - 作者： Aeleen Frisch
    - 由 O'Reilly and Associates 发行

- *Linux: Rute User's Tutorial and Exposition*

    - 一本很好的管理 GNU/Linux 系统的精装版在线书籍
    - 作者：Paul Sheer
    - 由 Prentice Hall 出版
    - 软件包：rutebook (from non-free)
    - 文件：/usr/share/doc/rutebook/

- Bell Labs: Computing Sciences Research

    - 有关 Unix 历史的翔实文献
    - 主要文献：http://cm.bell-labs.com/cm/cs/
    - 技术报告精选：http://cm.bell-labs.com/cm/cs/cstr.html
    - 一些论文：http://cm.bell-labs.com/cm/cs/papers.html

- Linux 在线通用技术支持资料
    - Debian Planet
    - debianHELP
    - Linux.com
    - The Linux Home Page at Linux Online
    - Red Hat (commercial Linux vender) (RPM, Sys-V init)
    - SuSE, Inc. (commercial Linux vender) (RPM, Sys-V init)
    - Slackware (TGZ, BSD-style init)
- Unix 在线通用向导及资料
    - The UNIX System by The Open Group
    - A UNIX Introductory Course from Ohio State University
    - UNIXhelp from The University of Edinburgh
    - Unix / Programming Information
    - comp.unix.questions FAQ
    - comp.unix.user-friendly FAQ
    - FreeBSD Documentation
    - The FreeBSD Handbook
    - UNIX GUIDE
    - The Unix Heritage Society
- 自由软件项目主页
    - GNU 计划
    - Linux 文档计划
    - Linux 内核归档
    - XFree86 项目，公司
    - GNOME
    - K 桌面环境
    - Red Hat 公司的 GNU 软件
    - Mozilla
    - FreeBSD
    - OpenBSD
    - NetBSD

15.2 查词意

Debian 中使用了大量术语和缩写，下面的命令将会解决大部分问题（要求安装 dict 软件包和其它相关软件包）

```
$ dict put-a-weird-word-here
```

15.3 查找流行的 Debian 软件包

Debian 中有许多软件包，有时很难决定该装哪一个。参阅 Debian 软件包流行度调查结果可了解别人都在用什么软件。亦可安装 popularity-contest 软件包参加投票。

15.4 Debian bug 跟踪系统

Debian 发行版有一个 bug 跟踪系统 (BTS)，它将来自用户和开发人员的错误报告的详细内容进行归档，每个错误都有一个编号，错误报告将一直存在于数据库中，直到获得错误已更正的标记。

在发送错误报告之前请先检查一下别人是否以提交了相同的错误报告。BTS 的网站 或 其它地方均可找到当前未更正错误列表。参阅 检测程序错误寻求帮助，第 6.3.1 节。

许多严重错误的报告标记为 **FTBFS**。意思是“Fails To Build From Source”。

http://www.debian.org/Bugs/Reporting 中描述了 bug 报告方法。

15.5 邮件列表

阅读最新的“debian-devel-announce”（英文、只读、低流量）与 Debian 保持同步。

邮件列表“debian-user”（英文、开放、高流量）和“debian-user-language”（需向其它语言的用户）中有 Debian 用户最兴趣的内容。

想了解这些邮件列表的详细信息以及如何订阅，参阅 http://lists.debian.org/。发问前请先搜索答案并注意遵守列表有关礼仪和规则。

如果你不希望在回复至邮件列表时收到复本的话，你应该用 Mail-Followup-To: 作为邮件的标头，这是很有效的。这是邮件列表中非官方的习惯，在 http://cr.yp.to/proto/replyto.html 有些说明。

15.6 Internet Relay Chat (IRC)

IRC (Internet Relay Chat) 可让你与世界各地的人进行实时聊天。在 freenode 上可找到与 Debian 相关的 IRC 频道。 连接它们需要一个 IRC 客户端，最流行的客户端有 XChat、BitchX、ircII、irssi、epic4 以及 KSirc，任何一个都能在 Debian 软件包中找到。安装好客户端后，需要告诉程序连接到服务器，对于绝大多数客户端，可输入：

```
/server irc.debian.org
```

一但连接上服务器，可输入下面的命令加入 #debian 频道

```
/join #debian
```

可输入下面的命令离开 #debian 频道

```
/part #debian
```

你可输入下面的命令退出 irc 客户端

```
/quit
```

要给 foo 发送一条内容为 "Hello Mr. Foo"的私人消息，请输入

/msg foo Hello Mr. Foo

要注意的是，你所输入的任何信息，只要不是以 / 打头，就会被当作是聊天信息而发送至频道上。

注意：类似 XChat 的客户端它们用于连接服务器/频道的图形用户界面会稍有不同。

15.7 搜索引擎

许多搜索引擎提供有关 Debian 的文档：

- Debian WWW 搜索站点.
- Google: 将“site:debian.org”包含于搜索字段。
- Google Groups: 新闻组搜索引擎。将“group:linux.debian.*”包含于搜索字段。
- AltaVista

例如，搜索字符串“cgi-perl”会得到更多有关该软件包的详细说明而非其 control 文件中的简要描述。参阅 检测程序错误寻求帮助, 第 6.3.1 节 获得相关建议。

15.8 网站

下列是我收集的一些特别专题讨论的零散网页地址。

- IBM developerWorks: Linux
- Adrian Bunk's packages to run kernel 2.4.x on potato
- Linux on Laptops
- Xterm FAQ
- EXT3 File System mini-HOWTO
- Large File Support in Linux
- Window Managers for X
- Linux USB Project
- SuSE pages for CJK
- LNX-BBC (Business-card-sized boot CD project)
- Linux info by Karsten Self (partitioning, backup, browsers...)
- Backup info HOWTO by Alvin Oga
- Security info HOWTO by Alvin Oga
- Various UNOFFICIAL sources for APT
- Laptop Ethernet Configuration

附录 A - 附录

A.1 作者

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

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

本文档的编辑、翻译、扩充工作由下列 QRFF 组员参与：

- 英文版初稿及《Quick Reference...》的初稿
    - Osamu Aoki osamu#at#debian.org (leader: all contents)
- 英文版校对和其它贡献
    - David Sewell dsewell#at#virginia.edu (extensive work: en style)
    - Thomas Hood jdthood#at#yahoo.co.uk (network related)
    - Brian Nelson nelson#at#bignachos.com (especially X related)
    - Jan Michael C Alonzo jmalonzo#at#spaceants.net
    - Daniel Webb webb#at#robust.colorado.edu
    - 所有翻译者反馈
- 法文版翻译
    - Guillaume Erbs gerbs#at#free.fr (leader: fr)
    - Renald Casagraude rcasagraude#at#interfaces.fr
    - Jean-Pierre Delange delange#at#imaginet.fr
    - Daniel Desages daniel#at#desages.com
- 意大利文版翻译
    - Davide Di Lazzaro mc0315#at#mclink.it (leader: it)
- 葡萄牙文（巴西）版翻译
    - Paulo Rogerio Ormenese pormenese#at#uol.com.br (leader: pt-br)
    - Andre Luis Lopes andrelop#at#ig.com.br
    - Marcio Roberto Teixeira marciotex#at#pop.com.br
    - Rildo Taveira de Oliveira to_rei#at#yahoo.com
    - Raphael Bittencourt Simoes Costa raphael-bsc#at#bol.com.br
    - Gustavo Noronha Silva kov#at#debian.org (coordinator)
- 西班牙文版翻译
    - Walter Echarri wecharri#at#infovia.com.ar (leader: es)
    - Jose Carreiro ffx#at#urbanet.ch
- 德文版翻译
    - Jens Seidel tux-master#at#web.de (leader: de)
    - Willi Dyck wdyck#at#gmx.net
    - Stefan Schroeder stefan#at#fkp.uni-hannover.de
    - Agon S. Buchholz asb#at#kefk.net

- 波兰版翻译 — PDDP 的下列成员：
    - Marcin Andruszkiewicz
    - Mariusz Centka mariusz.centka#at#debian.linux.org.pl
    - Bartosz Feński fenio#at#debian.linux.org.pl (leader: pl)
    - Rados aw Grzanka radekg#at#debian.linux.org.pl
    - Bartosz 'Xebord' Janowski
    - Jacek Lachowicz
    - Rafa Michaluk
    - Leonard Milcin, Jr.
    - Tomasz Z. Napiera a zen#at#debian.linux.org.pl
    - Oskar Ostafin cx#at#debian.linux.org.pl
    - Tomasz Pi ko
    - Jacek Politowski
    - Mateusz Prichacz mateusz#at#debian.linux.org.pl
    - Marcin Rogowski
    - Pawe Ró ański
    - Mariusz Strzelecki
    - Krzysztof cierski
    - Przemys aw Adam miejek tristan#at#debian.linux.org.pl
    - Mateusz Tryka uszek#at#debian.linux.org.pl
    - Cezary Uchto
    - Krzysztof Witkowski tjup#at#debian.linux.org.pl
    - Bartosz Zapa owski zapal#at#debian.linux.org.pl
- 简体中文版翻译
    - 刘浩(LYOO) iamlyoo#at#163.net
    - Ming Hua minghua#at#rice.edu
    - 肖盛文 atzlinux#at#163.com (leader: zh-cn)
    - Haifeng Chen optical.dlz#at#gmail.com
    - 解彦博 xieyanbo#at#gmail.com
    - easthero easthero#at#gmail.com
- 繁体中文版翻译
    - Asho Yeh asho#at#debian.org.tw (leader: zh-tw)
    - Tang Wei Ching wctang#at#csie.nctu.edu.tw (ex-leader: zh-tw)
- 日文版翻译
    - Shinichi Tsunoda tsuno#at#ngy.1st.ne.jp (leader: ja)
    - Osamu Aoki osamu#at#debian.org

QREF 是文档原始标题的缩写,“Quick Reference...”也是在 qref.sourceforge.net 上的项目名称。

Debian 系统上的许多手册页和信息页被用来作为写作该文档的主要参考。 在 Osamu Aoki 认为不超出合理引用的范围之内， 这些手册页和信息页的许多部分，特别是命令定义，在经过仔细的编辑后， 才使它们成为符合本文写作风格和目的组成部分。

Debian 基础, 第 2 章 中的大部分内容来源于《The Debian FAQ》(2002 年三月):

- 5. The Debian FTP archives: ftparchives.sgml (整章)
- 6. Basics of the Debian Package Management System: pkg_basics.sgml (整章)
- 7. The Debian Package Management Tools: pkgtools.sgml (整章)
- 8. Keeping Your Debian System Up To Date: uptodate.sgml (整章)
- 9. Debian and the kernel: kernel.sgml (整章)
- 10. Customizing your installation of Debian GNU/Linux: customizing.sgml (部分内容)

在本文档中某些来源于《The Debian FAQ》的段落进行了较大的重组,以反映 Debian 系统的变化。故本文档的内容更新。

最初《Debian FAQ》由 J.H.M. Dassen (Ray) 和 Chuck Stickelman 创建和维护。《Debian FAQ》的修订版作者是 Susan G. Kleinmann 和 Sven Rudolph。在他们之后,《The Debian FAQ》由 Santiago Vila 来维护。当前的维护者是 Josip Rodin。

有关《The Debian FAQ》的信息部分来源于:

- The Debian-1.1 release announcement, by Bruce Perens.
- The Linux FAQ, by Ian Jackson.
- Debian Mailing List Archives,
- the dpkg programmers' manual and the Debian Policy manual (参见 参考资料, 第 15.1 节)
- 众多开发人员、自愿者、beta 版测试人员
- 以及其作者的片断记忆。 :-)

“教程”章的部分内容来源于

- Havoc Pennington, Oliver Elphick, Ole Tetlie, James Treacy, Craig Sawyer 和 Ivan E. Moore II 写的“ Debian 教程” (该文档来源于 Larry Greenfield “Linux 用户手册"。)
- John Goerzen 和 Ossama Othman 的“Debian GNU/Linux:安装和使用指导”

作者在此感谢所有在文档写作过程中曾给予帮助的人。

A.2 保证

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

本文档不能替代任何权威指南。

本文档不承诺任何保证。所有的商标均归属于其各自的拥有者。

A.3 反馈

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

尽管我生活在美国，但并不是一名地道的英语使用者。欢迎任何有关语法方面的指正。

最好的反馈莫过于一个 SGML 版的 diff,不过 text 版的 diff 也很欢迎。参阅 官方文档, 第 1.1 节 了解官方文档站点。

本文档原始的 SGML 文件位于 CVS： :pserver:anonymous@cvs.sf.net/cvsroot/qref 或 http://qref.sourceforge.net/Debian/qref.tar.gz。

A.4 文档格式

本文档使用 DebianDoc SGML DTD（由 LinuxDoc SGML 改进而来）写作。DebianDoc SGML 系统允许一个源文件输出多种格式的文档，例如，本文档可以使用 HTML、纯文本、TeX DVI、PostScript、PDF 或 GNU info 方式阅读。

DebianDoc SGML 的转换工具位于 debiandoc-sgml 软件包中。

A.5 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。配置打印机就是个好例子。

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。

为了在 Linux 系统配置的迷宫里找到出路，我开始用 text 格式写下简单的备忘录。当这些备忘录越积越多的时候我学会了 debaindoc。于是就有了 *Debian 参考手册*。

A.6 Debian 引言

这儿有一些来自于 Debian 邮件列表的有趣引言。

- “这就是 Unix。它给你足够的绞索来吊死你自己。”—Miquel van Smoorenburg miquels@cistron.nl
- “Unix 对用户来说是友好的…… 但它仅仅选择它的用户是谁。”—Tollef Fog Heen tollef@add.no

来源

Debian 参考手册

CVS, 星期日 三月 12 12:53:54 UTC 2006

Osamu Aoki osamu#at#debian.org

译者：

Hao "Lyoo" Liu iamlyoo#at#163.net

Ming Hua minghua#at#rice.edu

肖盛文 atzlinux#at#163.com

Haifeng Chen optical.dlz#at#gmail.com

解彦博 xieyanbo#at#gmail.com

easthero easthero#at#gmail.com

作者, 第 A.1 节

取自"http://wiki.ubuntu.org.cn/index.php?title=UbuntuManual&variant=zh-cn"