The Debian GNU/Linux FAQ

translator: etony C.F.AN <etony@tom.com>
Debian FAQ Authors

version 5.0.1, 17 March 2012

摘要

本文档解答一些关于 Debian GNU/Linux 的常见问题.

版权声明

Contents

Chapter 1

定义与概述

1.1 什么是 Debian GNU/Linux?

Debian GNU/Linux 是指一个 linux 操作系统发行版和在它上运转的许多的包的集合.

事实上, 用户可以通过互联网获取 Linux 内核, 进行编译. 通过同样的方法获取应用程序的源码, 进行编译. 然后装到自己的系统上. 对于那些复杂的程序来说, 这个过程是费时的易错的. 因此, 用户通常通过发行版来获取操作系统和应用程序. 发行版是通过软件, 协议, 包管理机制, 以及安装维护工具, 文档和其它服务来区分的.

Debian GNU/Linux 是志愿者建立一个免费, 高质量 Unix 兼容操作系统的努力的结果. 建立自由的类 UNIX 操作系统的想法源于 GNU 计划, 组成 Debian GNU/Linux 许多应用程序也是由 GNU 项目开发的.

Debian 的免费与 GNU 是一致的（详见 Debian 自由软件指南 (http://www.debian.org/social_contract#guidelines)）. 我们所说的 Free 是指其自由, 而不是价格. 免费软件是指你可以自由分发其拷贝, 如果你愿意, 你可以得到其源码, 可以对其修改或使用, 并且你知道你能做这些事情.

Debian 项目是 Ian Murdock 在 1993 年创建的, 最初是在自由软件基金会的 GNU 计划下发起的. 现在, Debian 开发者认为这是 GNU 计划的一个分支.

Debian GNU/Linux 是:

- **灵活性**: Debian 目前有超过 29000 个软件包. Debian 为用户提供了选择软件包安装的工具. 在任何 Debian 镜像站点 (http://www.debian.org/distrib/ftplist)都可以找到关于当前软件包的列表和描述.
- **自由使用和分发**: 使用和分发无需任何费用. Debian GNU/Linux 的所有正式软件都是遵循 GNU 的通用公共许可证的. Debian FTP 包含大概 187 个受限制可分发的软件包 (在 `non-free` 和 `contrib` 部分).
- **动态**: 大约有 880 位志愿者经常开发新的或改进代码，Debian 更新非常快，每几个月都有新的发行计划, FTP每天都更新.

尽管 Debian GNU/Linux 本身是免费软件，仍然可以在它的基础上构建 Linux 商业版本，详见‘可以在 Debian GNU/Linux 上开发我的 Linux 版本吗?’ on page 43 .

1.2 OK, 现在我知道Debian 是... Linux是什么?!

简而言之, Linux 是一个类 UNIX 操作系统的内核. 最初是为 386(或者更高)PC 设计的, 现在包括多处理器在内的其它系统下开发. Linux 是由 Linus Torvalds 和全世界很多计算机科学家编写.

除了内核, “Linux” 还包括:

- 符合 Liunx 标准的一个文件系统 http://www.pathname.com/fhs/.
- 大量的 Unix 实用程序, 其中许多是由 GNU 计划和自由软件基金会开发的.

是 Linux 内核, 文件系统, GNU 和 FSF 应用软件, 和其它符合 POSIX(IEEE 1003.1) 标准的应用软件的结合体. 详见 ‘Debian 源码与其他 Unix 兼容性如何?’ on page 5.

更多的关于 Linux 的信息请参阅 Michael K. Johnson 的 Linux Information Sheet (ftp://ibiblio.org/pub/Linux/docs/HOWTO/INFO-SHEET) 和 Meta-FAQ (ftp://ibiblio.org/pub/Linux/docs/HOWTO/META-FAQ).

1.3 什么是 “Hurd”?

Hurd 是运行在 GNU Mach 微内核上的一套服务器, 是为GNU开发的.

目前仅有Debian GNU/Linux, 不过我们也正在开发 Debian GNU/Hurd 服务器与桌面, 现在还没有官方发行版, 不过不会太久了.

更多 GNU/Hurd 信息参见http://www.gnu.org/software/hurd/ , Debian GNU/Hurd 参见http://www.debian.org/ports/hurd/.

1.4 Debian GNU/Linux 与其他 Linux 发行版有什么不同? 为什么要选择Debian GNU/Linux?

与其它发行版的主要区别:

The Debian package maintenance system: 整个系统, 或其一部分可以在不需重新设置, 不丢失配置文件, 多数情况不需重起的情况下升级. 现有的许多Linux发行版都有自己的软件包管理系统; Debian 的软件包管理系统是独一无二的. (参阅 ‘Debian 的包管理系统’ on page 17)

Open development: 尽管其它的 Linux 发行版是由独立的, 小型的, 封闭的或商业组织开发的, Debian 是唯一一个由全世界范围内的软件工作者通过互联网开发的 Linux 发行版.

全世界超过 880 位志愿者包维护人员维护着超过 29000 个包, 并且不断改进 Debian GNU/Linux. Debian 开发者不是通过撰写报告, 而是通过根据项目标准封装现有软件, 提交错误报告和提供用户支持来为项目贡献自己的力量. 怎样成为一位捐助者在里的附加信息见 ‘如何成为一个 Debian 软件开发者?’ on page 41.

The Bug Tracking System: 开发者地理上的分散需要一个成熟的工具和快速的通讯用于错误提交和错误修复, 以加速系统的开发. 鼓励用户使用正式的格式通过 WWW 或 e-mail 来提交错误. 更多信息详见 ‘已知错误的记录?’ on page 38.

The Debian Policy: Debian 有关于软件标准和 Debian 策略的详细的说明. 这文档定义了维护管理包的质量与标准.

其它信息详见我们的网页 选择 Debian 的理由 (http://www.debian.org/intro/why_debian).

1.5 Debian 计划与自由软件基金会的GNU计划

Debian 是通过 自由软件基金会 (http://www.gnu.org/) 特别是 Richard Stallman (http://www.stallman.org/) 的理想构建的. FSF 强有力的系统开发工具, 工具和应用程序也是一个 Debian 系统的关键部分.

Debian 项目是完全同 FSF 项目分离的, 但是保持着经常的联系, 并进行许多项目的协作. FSF 明确要求我们称我们的系统为“Debian GNU/Linux”, 并且我们乐于遵循这样的要求.

FSF 的长期目标是基于Hurd (http://www.gnu.org/software/hurd/) 开发一个称做 GNU 的新的操作系统. Debian 是在此系统中对 FSF 的使用, 称做 Debian GNU/Hurd (http://www.debian.org/ports/hurd/).

1.6 Debian 的发音及含义?

Debian 的发音是 Deb’-ee-en, 重音在第一个音节, 是 Debian 的项目创始人 Ian Murdock 和他的妻子 Debra 的名字缩写.(很多字典中好象对 Ian 发音并不明确(!), Ian 倾向与 ee’-en.)

Chapter 2

Debian GNU/Linux 的获取与安装

2.1 Debian 的最新版本是?

目前 Debian GNU/Linux 有三个版本:

***release 6.0*, 即. *'stable'* 版** 这是通过良好测试的稳定的软件, 仅当出现重大安全问题或修补时才更新.

***'testing'* 版** 下一个 "stable" 版;是经过测试，但是还不足以发行的 unetable 包, 比stable 更新快, 但比 unstable 稍慢.

***'unstable'* 版** 开发中的版本, 频繁更新. 可以随时从 Debian 的 Ftp 上的 'unstable' 区下载, 来更新你的系统, 但是你不能期望系统象以前一样稳定可用 - 这就是称作 '**unstable**' 的原因!

详见 '在 `dists` 目录有哪些版本?' on page 13 .

2.2 如何得到 Debian 的安装盘?

可以到 Debian 镜像站点 (`http://www.debian.org/mirror/list`).

根据硬件系统的不同分别放置在 `dists/stable/main` 目录的形如 `disks-`*`arch`*(*arch* 为 "i386", "sparc", 等, 从站点获取精确列表)的子目录下. 这些目录下每个发行版又分作一个目录. 最新版在 'current' 目录(一个符号连接).

详见目录下的`README.txt` .

2.3 如何从光驱安装 Debian?

Linux支持 ISO 9660 (CD-ROM) 文件系统, 一些商家 (`http://www.debian.org/CD/vendors/`) 提供这种格式的Debian GNU/Linux.

警告: 从光驱安装 Debian 时, 选择 dselect 方式不是个好主意，会很慢. `mountable` 和 `apt` 方式则要好的多(详见 'dpkg-mountable' on page 28 和 'APT' on page 27).

2.4 我有刻录机, 可以获取 Debian 影像吗?

是的. 为了使 CD 商家更容易提供高质量的磁盘, 我们提供 官方 CD 影像 (`http://cdimage.debian.org/`).

2.5 可以软盘安装吗?

首先, 警告: 整个 Debian GNU/Linux 太大了, 不适于类似于标准 1.44 MB 软盘这样的小介质安装方式, 你会发现从软盘安装不是一件愉快的事情.

把 Debian 包复制到软盘上, "DOS" 格式, "ext2" 格式，"minix" 格式都可以, 然后用mount 命令挂接软盘.

使用软盘比较复杂:

- 短的 MS-DOS 文件名: 如果你把包复制到了 MS－DOS 格式的软盘上, 你会发现它们的名字太长了, 不符合 MS-DOS 8.3 的命名格式, 因此你必须使用支持长文件名的 VFAT 格式的软盘.
- 大文件: 一些软件包大于 1.4MB, 不能复制在一张软盘上, 可以使用 dpkg－split 来解决这类问题(详见'dpkg-split' on page 25), 可以从Debian镜像 (http://www.debian.org/mirror/list)站点的 tools 目录下载这个工具.

你必须在内核中支持软盘, 这样才能读写软盘；现在的许多内核都包含支持软盘的驱动.

把软盘挂接到 /floppy (应该在安装过程中创建的一个目录), 使用:

- ```
  mount -t msdos /dev/fd0 /floppy/
  ```
  如果软盘在A驱, MS-DOS 格式,
- ```
  mount -t msdos /dev/fd1 /floppy/
  ```
  如果软盘在B驱, MS-DOS 格式,
- ```
  mount -t ext2 /dev/fd0 /floppy/
  ```
  如果软盘在A驱, ext2 格式 (即, 通常的 Linux) .

2.6 可以网络安装吗?

是的, 你可以使用从 Debian 的 FTP 和其镜像下载的安装系统引导.

可以下载一个小的 CD 影像文件, 制作可引导 CD, 用于安装基本的系统, 其它从网络安装, 详见 http://www.debian.org/CD/netinst/.

你甚至可以下载更小的软盘影像文件, 用它们创建可引导软盘, 然后通过网络开始安装 Debian. 详见http://www.debian.org/distrib/floppyinst.
  ```

Chapter 3

兼容性问题

3.1 可以在什么样的硬件系统上运行?

Debian GNU/Linux 包含所有程序的完整源代码, 因此可以在所有Linux内核支持的硬件系统上运行; 详见 Linux FAQ (http://en.tldp.org/FAQ/Linux-FAQ/intro.html#DOES-LINUX-RUN-ON-MY-COMPUTER).

Debian GNU/Linux 现在的版本是 6.0, 包括在以下硬件系统上运行的完整的源代码和二进制程序:

i386: 指基于 Intel 和兼容处理器的 PC 机,包括Intel的 386, 486, Pentium, Pentium Pro, Pentium II (Klamath 和 Celeron), 和 Pentium III, 以及 AMD, Cyrix 等制造的兼容处理器.

m68k: 指基于Motorola 680x0 的Amiga 和 ATARI系列.

alpha: 指 Compaq/Digital 的 Alpha 系统.

sparc: 指Sun的 SPARC 和大部分的 UltraSPARC 系统.

powerpc: 指 IBM/Motorola PowerPC, 包括 CHRP, PowerMac 和 PReP.

arm: 指ARM 和 StrongARM.

mips: 指 SGI 的 big-endian MIPS 系统, Indy 和 Indigo2; *mipsel*: 指 little-endian MIPS, Digital DECstations.

hppa: 指 Hewlett-Packard 的 PA-RISC(712, C3000, L2000, A500).

ia64: 指Intel 的 IA-64 (“Itanium”) 计算机.

s390: IBM 的 S/390 系统.

基于 Sparc64(UltraSPARC native) 的 Debian 的二进制版本正在开发阶段.

更多的具体硬件支持详见用户安装手册 http://www.debian.org/releases/stable/installmanual.

3.2 与其他的linux发行版兼容行如何?

Debian 发者努力与其他 Linux 发行版沟通, 以保持软件的兼容性. 大多数的软件都可以象在他们的开发环境下一样运行的很好.

Debian GNU/Linux 遵循 Linux 文件系统层次标准)Linux Filesystem Hierarchy Standard) (http://www.pathname.com/fhs/). 但是, 在规则解释上存在一定的回旋余地因此某些细节上可能与其它发行版有所不同.

3.3 Debian 源码与其他 Unix 兼容性如何?

大多数Linux程序的源码是和其他 Unix 系统相兼容的. 它几乎支持 System V Unix 系统和自由的和商业的 BSD 系统中的所有程序的源码. 但是说法无法证明, 因此对于 UNIX 没有什么价值. 在软件开发中需要的是完全兼容, 而不是大部分兼容. 因此出现了今天的 POSIX.1 (IEEE Standard 1003.1-1990), 类 UNIX 系统源码兼容性的主要标准之一.

Linux 原本要基于 POSIX.1 的, 但是 POSIX 不是免费的, 而且 POSIX.1 证书相当昂贵. 这使得 Linux 基于 POSIX 开发相当困难. 证书费用使得 Debian 获得官方兼容性证明相当困难, 即使已经完全通过确认条款(为了让更多的人在 POSIX 标准上工作, 这些确认条款可以免费获得).

Unifix 股份有限公司(Braunschweig, 德国) 开发了一个获得了 FIPS 151-2 证书的 Linux 系统. 这种技术用于 Unifix 的发行版 Unifix Linux 2.0 和 Lasermoon 的 Linux-FT.

3.4 我可以在 RedHat/Slackware/... 上使用 Debian 的包(".deb"文件)吗? 我可以在 Debian 上使用 RedHat 的 rpm 包吗?

不同的发行版使用不同的软件包格式和软件包管理程序.

你或许能: 通过一个程序可以把一个 Debian 包安装基于'其它'发行版的 Linux 系统中, 通常可以正常运行, 也可以把一个RedHat 或 Slackware 格式的包转换成 Debian GNU/Linux 格式的包. 这得益于 Linux 文件系统的层次标准. Alien (http://packages.debian.org/alien) 程序用于不同格式的包的转换.

你或许不想: 有些人在安装文件时, 自己来写安装控制文件, 通常这些文件是不标准的. 因此在 '其它' 系统上安装一个 Debian 包, 对于包管理系统可能产生不可预知的影响. 同样一个其它系统上的程序也许可以成功的安装到 Debian 系统中, 但是, 可能会导致 Debian 包管理系统不能完成一些包升级或删除, 甚至不能报告系统上安装了哪些包.

一个比较好的方法: Linux文件系统标准建议 /usr/local/ 下的目录完全由用户使用, 因此可以将 'foreign' 软件安装到这个目录下,进行配置,升级,或删除.

3.5 Debian 可以运行 "a.out" 程序吗 ?

你确实还有这样的程序吗? :-)

执行 一个类似于 a.out 格式的程序(即, QMAGIC 或 ZMAGIC),

- 确定内核支持, 要么内建支持(CONFIG_BINFMT_AOUT=y), 要么动态模块支持(CONFIG_BINFMT_AOUT=m). (Debian 的内核影像含有一个 binfmt_aout 模块)

  如果你的内核是动态模块支持, 那么确保 binfmt_aout 模块已加载. 你可以修改 /etc/modules 文件, 使得 binfmt_aout 模块启动时加载. 也可以执行 insmod DIRNAME/binfmt_aout.o 命令来完成. DIRNAME 指和内核版本有关的路径名, 在 2.2.17 内核的系统中 DIRNAME 有可能是 /lib/modules/2.2.17/fs/.

- 安装可以在 2.0 以前版本找到的 libc4 包（因为 2.0 开始删除了那个包). 可以在老版的 Debian 光盘（Debian 1.3.1 仍然含有这个包）或者这里 ftp://archive.debian.org/debian-archive/ 找到

- 如果你执行的程序是个a.out图形客户端, 安装 xcompat 包.(参阅前面的获取方法).

如果你有 a.out 格式的商业程序, 这是要求商家发给你一个 ELF 升级版的好机会.

3.6 IDebian 可以运行 libc5 程序吗?

是的,只需要从 oldlibs 区域(与老程序兼容所需包)安装 libc5 库.

3.7 Debian 可以编译 libc5 程序吗?

是的. 从 oldlibs 目录安装 libc5-altdev 和 altgcc 软件包. 你会在 /usr/i486-linuxlibc1/bin 目录下发现所需的 gcc 和 g++, 把它放入你的 $PATH 变量, 使得make 和其他程序首先执行它.

编译客户端图形窗口系统(X clients), 需要安装 xlib6 和 xlib6-altdev 包.

注意我们的其他软件包对 libc5 环境不是支持的太好.

3.8 如何安装非 Debian 格式程序?

/usr/local/ 目录下的文件不在 Debian 包管理系统控制范围之内. 因此把你的程序的源代码放到 /usr/local/src/ 目录下是个不错的习惯. 例如你可以把一个名为 "foo.tar" 的包解压到 /usr/local/src/foo 目录, 编译后, 可执行程序放到 /usr/local/bin/, 库文件放到 /usr/local/lib/, 配置文件放到/usr/local/etc/.

如果你的程序必须放到其它目录, 你仍可以把它们放到这个目录, 在需要的目录建立一个符号连接指向 /usr/local/ 目录下它的位置. 如

```
ln -s /usr/local/bin/foo /usr/bin/foo
```

如果你获得一个可再分发的软件, 可以把它做成 Debian 格式的包, 然后把它加载到 Debian 系统中, 在用户手册中有关于Debian 格式包制作的介绍(详见'Debian 系统的其他文档?' on page 37).

3.9 我运行 `foo` 时为什么提示 " 无法找到libX11.so.6"?

这个错误表明此程序使用的 X11 的库是 `libc5`版本, 这就意味着你需要从 `oldlibs` 安装`xlib6` This error message could mean that the program is linked against the `libc5` version of the X11 libraries. In this case you need to install the `xlib6` package, from the `oldlibs` section.

你也可能碰到关于 libXpm.so.4 的错误信息, 这就需要从 `xpm4.7` 包中安装libc5 版本的 XPM 库.

3.10 为什么我不能编译需要 libtermcap 的程序?

Debian 使用 `terminfo` 数据库和 `ncurses` 库文件而不是 `termcap` 数据库和`termcap` 库文件. 编译这些软件时应该用 `libncurses` 替换 `libtermcap`, 并且用户应该有一定的关于终端接口的知识.

为了运行已经和 `termcap` 库相连并且你没有源代码的程序, Debian 提供了一个称作 `termcap-compat` 的包, 它提供了 `libtermcap.so.2` 和 `/etc/termcap`, 安装这个包可以解决程序运行提示无法加载'libtermcap.so.2'库或缺少 `/etc/termcap` 文件的问题.

3.11 什么无法安装 AccelX?

AccelX 安装时会用到 `termcap` 库, 详见'为什么我不能编译需要 libtermcap 的程序?' on this page.

3.12 为什么我的 XFree2.1Motif 崩溃了?

你需要安装 `motifnls` 包, 他提供了在 XFree-3.1下运行基于 XFree-2.1 编译的 Motif 程序的XFree-2.1 配置文件.

没有这些文件, 一些 Motif 程序在做拷贝和粘贴操作时有可能会崩溃, 也可能会出现其他的问题.

Chapter 4

Debian 的软件系统

4.1 Debian GNU/Linux 上有那些应用程序与开发软件?

和大多数发行版一样, Debian GNU/Linux提供:

- 用于软件开发, 文档管理, 文字处理的主要的 GUN 应用程序, 包括gcc, g++, make, texinfo, Emacs, Bash shell 和众多的改进版 UNIX 程序,
- Perl, Python, Tcl/Tk 以及各种的相关软件, 模块, 库文件,
- TeX (LaTeX) 和 Lyx, dvips, Ghostscript,
- X 窗口管理程序, 为 Linux 提供了互联的图形用户界面, 和包括 GNOME 在内的 X 图形应用程序,
- 一整套网络应用程序, 包括用于互联网协议的服务器如 HTTP (WWW), FTP, NNTP (news), SMTP 和 POP (mail)，name server; 也包括网页浏览器和开发工具.

包含超过 28200 个包, 从新闻服务器到读者语音支持, 传真程序 ,数据库与电子表格, 图形处理, 通信, 网络和邮件工具, 网页服务器, 甚至 ham-radio 程序.另外的 187 组 Debian 软件包, 因为许可证的原因, 没有成为 Debian 的正式组成部.

4.2 谁写的这些软件?

每个软件的作者的名字都在/ `/usr/doc/PACKAGE/copyright` 文件中, 这里 PACKAGE 指软件包的名称.

系统的每个软件的维护者都被写在和软件包在一起的控制文件(参阅 'Debian 的控制文件是什么?' on page 18)中.

4.3 如何得到Debian的当前已开发软件列表?

有两种方式获取一个完整的列表:

任何一个 Debian 镜像 (`http://www.debian.org/distrib/ftplist`) 的 `indices/Maintainers` 文件中都有一个可分发包的列表, 文件包括包名及其维护者的名字, e-mail.

任何一个 url name="Debian non-US 镜像" id="http://www.debian.org/mirror/list-non-US"> 的 `indices-non-US/Maintainers` 文件中都有一个美国禁止出口包的列表, 包括包名及其维护者的名字,e-mail.

Debian包的www查询页面大概20类The Debian 包的 WWW 查询页面 (`http://packages.debian.org/`) 大概20类.

4.4 Debian GNU/Linux 缺少什么?

开发中和预期开发表 (`http://www.debian.org/devel/wnpp/`), 列举了所需的软件包.

更多信息参见 '如何成为一个 Debian 软件开发者?' on page 41.

4.5 我编译程序时为什么会有 “ld: cannot find -lfoo” 提示?Debian 的库文件里怎么没有?

Debian 规则要求符号链接(类似于 libfoo.so.x.y.z)位于不同的包中, 这些包通常命名为libfoo-dev 或 libfooX-dev(假设库包是 libfooX, X 是个整数).

4.6 Debian 支持 Java 吗?

因为 Sun 官方的 JVAV 开发工具是非自由软件, 因此不能加入 Debian. 但是可以获得Debian 软件包格式的 JAVA 的 JDK 和几个 *free* 的工具, 你可以使用 Debian 来开发, 调试, 运行 JAVA 程序.

运行 JAVA 小程序(applet), 需要 web 浏览器有识别执行它们的能力, Debian 的一些浏览器如 Mozilla, Konqueror 都支持运行 JAVA 所需的插件, 也能找到 non-free 的 Debian 格式 Netscape 软件包, 它也能运行 JAVA 小程序(applet).

更多信息参见 Debian Java FAQ (http://www.debian.org/doc/manuals/debian-java-faq/).

4.7 怎么确定我正在使用的是 Debian 系统, 怎么检查它的版本?

通过检查是否存在 `/etc/debian_version` 文件来确认你的系统是不是 Debian, 这个文件中包含了一行显示你的版本号的文字, 这是由 `base-files` 包给定的.

`dpkg` 程序的存在使得在你的系统上安装 Debian 包成为可能, 但是这个程序移植到其他系统或构架上后, 不再是一个可靠的鉴别方法.

用户应该知道, Debian 由很多部分构成, 每一部分(几乎)都可以单独升级, 每个版本都有定义好的固定不变的内容. 分步更新是可以的, 使用 `dpkg --list foo` 命令可得到 `foo`, 包的安装状态. 查看所有包的版本, 运行

```
dpkg -l
```

更多信息:

```
dpkg --status foo
```

4.8 对其他语言(非英语)支持的怎么样?

- Debian GNU/Linux 发行版包含很多键盘的键盘映射(keymaps)表, 并提供工具(`kbd` 软件包中)安装, 查看, 修改这写表格.

  安装时会提示用户选择正确的键盘.

- 大多数的软件支持在非 US-ASCII 字符的其他拉丁语系(如 ISO-8859-1 或 ISO-8859-2)下使用, 很多程序支持如中文或日文的多字节语言.

- 现在, `manpages-LANG`(LANG 是两位的 ISO 国家代码)软件包提供德, 西班牙, 芬兰, 法, 匈牙利, 意大利, 日, 朝鲜, 和波兰语的用户手册. 要使用一个 NLS 手册, 必须正确的设置 shell 的 LC_MESSAGES 变量.

  例如, 要使用意大利语的手册应将 LC_MESSAGES 设为 ’italian’, man 程序就会到 `/usr/share/man/it/` 下寻找意大利语的手册.

4.9 关于 US 的出口限制?

美国法律中限制密码软件的出口. PGP, ssh 等都在此列.

为了避免不必要的法律问题, 某些 Debian GNU/Linux 软件包, 仅在 ftp://non-US.debian.org/debian-non-US/ 提供. 此类镜像站点的完整列表见 ftp://non-US.debian.org/debian-non-US/README.non-US.

4.10 如何得到 pine?

因为许可证的限制, pine 被放在非自由(non-free)区域, 而且, 许可证甚至不允许分发修改后的二进制程序, 因此你必须自己编译源代码和 Debian 补丁.

源码包名是 `pine`. 你可以用`pine-tracker` 获取更新通知.

注意有很多 pine 和 pico 的替代品, 如 main 区的m`mutt` 和 `nano`.

Chapter 5

Debian 的 FTP

5.1 Debian 的 FTP 上有哪些目录?

可以从 Debian 镜像站点的目录树下获取打了包的软件.

`dists` 目录包含“发行版”(distributions), 此处是获得 Debian 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 packages.gz 文件仍在其中.

`pool` 目录为软件包的物理地址. 详见‘`pool` 目录下是什么 ?’ on page 15.

还有一些目录:

/tools/: 用于创建启动盘, 磁盘分区, 压缩/解压文件, 启动 Linux 的 DOS 下的小工.

/doc/: 基本的 Debian 文档, 如 FAQ, 错误报告系统指导等..

/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.

5.2 在 `dists` 目录有哪些版本?

通常有三个 Debian 发行版本, 它们是 “stable” 发行版, “testing” 发行版和 “unstable” 发行版. 有时还有一个 “frozen” 发行版, 详见(see “‘frozen” 是什么?’ on page 15).

5.3 象 slink, potato, 等等, 是什么意思?

它们只是一些版本代号(codenames). 处于开发阶段的发行版只有版本代号, 没有版本号, 使用版本代号的目的在于简化建立 Debian 发行版镜像的工作(如果真实目录例如 `unstable` 突然改名为 `stable`, 许多文件都没必要再次下载).

当前, `stable` 是一个指向 `woody`(即Debian GNU/Linux 6.0)的符号链接, `testing` 是指向 `sarge` 的符号链接. 也就是说 `woody` 是当前的 `stable` 发行版, `sarge` 是当前的`testing` 发行版 .

`unstable` 发行版是指向 `sid` 的永久符号链接, 即 unstable 发行版总称为`sid`(参见 “‘sid” 是什么?’ on the next page).

5.3.1 以前用过哪些代号名?

已使用过的发行版代号有: `buzz` for release 1.1, `rex` for release 1.2, `bo` for releases 1.3.x, `hamm` for release 2.0, `slink` for release 2.1 和 `potato` for release 2.2.

5.3.2 它们源自何处?

到目前为止它们均出自 Pixar 的电影“玩具总动员(Toy Story)” .

- *buzz* (Buzz Lightyear) 是个宇航员,
- *rex* 是只暴龙,
- *bo* (Bo Peep) 是个放羊的女孩,
- *hamm* 是个小猪攒钱罐,
- *slink* (Slinky Dog) 是只玩具狗,
- *potato* 当然就是 Potato Head 先生,
- *woody* was the cowboy.
- *sarge* 是位绿色塑料玩具士兵首领.
- *etch* 是玩具黑板.
- *sid* 是隔壁的男孩, 那个玩具终结者.

5.4 “sid” 是什么?

sid 或 *unstable* 是大多数软件最初上载的地方, 这些软件首先要进入 *testing*, 然后在 *stable* 里发行. sid 里的软件可能被发行, 也可能不被发行.

“sid” 来自于电影 “玩具总动员(Toy Story)” 里的动画形象: Sid 是隔壁的男孩，那个玩具终结者 :-)

[1]

5.5 stable 目录的内容?

- stable/main/: 目录包含的软件包均是最新 Debian GNU/Linux 系统发布版的正式组成部分.

  这些软件包均遵循Debian 自由软件指南 (http://www.debian.org/social_contract#guidelines).

- stable/non-free/: 本目录包含的软件包受到一定限制, 发行者需遵循特殊版权要求.

  例如, 有些软件包的许可证禁止其用于商业发行的. 有些虽可以再发行, 但本身是共享软件而非自由软件. 以任何方式再发行这些软件包时(例如写入光盘), 必须认真阅读有关的许可证或与所有者协商.

- stable/contrib/: 本目录包含的软件包均遵循 DFSG-free 原则，本身也是自由发布的, 但这些软件包的关联包不具有自由发行的属性, 它们位于 non-free 目录.

5.6 testing 目录的内容?

处于 unstable 版本的 testing 通过级别测试后登记到 ’testing’ 目录.

这些软件包必须可同时运行于所有架构, 并且没有关联性问题影响到其卸载; 比起在 unstable 中的相应版本, 它们有更少的 release-critical 错误. 我们将 ’testing’ 作为更佳发布候选版本.

有关 “testing” 版本的更多信息见于 http://www.debian.org/devel/testing

[1]过去 sid 并不存在, FTP 站点结构有个缺点: 假设当前 unstable 发行版中创建了某个软件开发项目, 当 unstable 成为新的 stable 版之时, 便是它的发布之日. 由于软件包一但发布就需要移动到新的 stable 目录, 当众多软件开发项目移动目录时大量带宽会被吞噬掉, 这个流程就显得很不切实际, 因而许多软件开发项目并没有按这个方法行事. 经过几年的研究摸索, 文档管理员提出一个方案, 将未获准发布的二进制文档存入名为 sid 的特定目录. 由于这些软件尚未发布, 从那时起, 它们就被加入到 unstable 目录树. 当它们首次发布时, 将会建立一个从当前 stable 指向 sid 的链接. 这个方案用户听起来的确有些晕头. 有了软件包储藏池(pool)(详见‘pool 目录下是什么 ?’ on the facing page), 二进制软件包均按一定规范存放于 pool 目录, 而与发行版无直接关系, 当发布新版本时, 就不会再出现大量带宽被消耗的问题. (不过, 大量带宽还是被开发进程消耗了).

5.6.1 “frozen” 是什么?

当 “testing” 发行版足够成熟了, 它就会被 ’冻结’(freezing), 通常不再作宣传, 确保尽可能少的新 “unstable” bug 进入 “testing”.

一段时间以后这个 “testing” 发行版就成为真正的 ’frozen’ 了, 表示这个版本不再加入新代码, 只进行除错工作. “testing” 发行版还要经过称之为 “循环测试” 的深度冻结.

我们将 “testing” 发行版中可能延迟软件包或整个版本发布的错误都记录在案. 详见 current testing release information (http://www.debian.org/releases/testing/).

一但错误总数低于可接受的最大值, 冻结的 “testing” 发行版就晋升成 “stable” 并分派一个版本号.

新版本发布了, 先前发布的版本成为过期版(obsolete). 详见 Debian archive (http://www.debian.org/distrib/archive).

5.7 unstable 目录的内容?

‘unstable’ 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 unstable 发行版的好处就是你可以获得 Debian 项目的最新更新——不过新东西也会出新问题, 你得好坏兼收 :-)

在 ‘unstable’ 下同样有 main, contrib 和 non-free 子目录, 它们的作用与 ‘stable’ 中的一样.

5.8 dists/stable/main的内容?

在每个主目录树下[2] 包含三个包含索引文件的三个子目录.

包含一组型如 `binary-`*`something`* 的子目录, 其包含各种计算机平台下, 二进制软件包的索引文件, 例如, `binary-i386` 为运行于 Intel x386 PC 机上的软件包, `binary-sparc` 则是运行于 Sun SPARCStations 上的软件包.

每个发行版的完整的平台列表参阅 the release’s web page (http://www.debian.org/releases/). 当前发行版, 参阅 ‘可以在什么样的硬件系统上运行?’ on page 5.

binary-* 下的索引文件称做 Packages(.gz), 其包括这个发行版中所有二进制软件包的列表. 软件包的物理位置则为上一级 `pool` 目录.

另外, 还有一个称为 source/ 的子目录, 其包含本发行版的所有源代码包的索引文件. 索引文件称做 Sources(.gz).

最后, 但不是全部, 还有一组用于安装系统索引文件的子目录. 在 *woody* 中, 称做 `disks-`*`architecture`*; 在 *sarge* 中, 称做 `debian-installer/binary-`*`architecture`*.

5.9 在哪里可以获取源代码?

Debian 系统中的一切程序都有源代码. 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, 或向商家索取源代码.

通常源代码分散于 `pool` 目录, 同时处于多个架构目录(详见‘`pool` 目录下是什么 ?’ on this page). 用户不必非常熟悉 FTP 目录结构, 想获得源代码可以试试 `apt-get source mypackagename` 命令.

有些软件包, 如著名的`pine`, 由于许可证限制, 只提供源码包. 详见 ‘如何得到 pine?’ on page 11.

“contrib” 和 “non-free” 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 Debian 系统S.

5.10 pool 目录下是什么 ?

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”, “contrib” 和 “non-free”), 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/` 目录中.

[2] `dists/stable/main`, `dists/stable/contrib`, `dists/stable/non-free`, 和 `dists/unstable/main/`, 等.

另外, 由于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/.
[3]

5.11 什么是 “incoming”?

上载的软件包在检查它的真实性和容许进入 FTP 以前首先存放于 “incoming” 目录.

通常没有人需要从这个目录安装软件. 然而, 在某些在紧急情况下, 你可以直接从 incoming 目录(http://incoming.debian.org/)手工下载软件, 检查 GPG 签名, .changes 和 .dsc 文件中的校检码 MD5sums, 然后安装.

[3]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, 例如当镜像站点进行新版本发布时大量带宽被消耗. 这个问题通过引入软件包 pool 得到了解决. dists 诸如apt等命令访问的索引文件仍位于dists目录中，直到本文写作之时，旧发行版的软件包还没转到pool目录，所以你将看到路径的 “Filename” 域中包含有发行版名称如 dists/potato 或 dists/woody.

Chapter 6

Debian 的包管理系统

6.1 什么是 Debian 包?

软件包一般包括实现一系列命令或特殊功能所必须的所有文件. 有两种类型的 Debian 软件包:

- 二进制包, 包含可执行文件, 配置文件, man/info 手册, 版权信息, 以及其它文档. 以一种 Debian 特有的格式分发(详见'Debian 软件包的格式?' on this page), 通常以 '.deb' 作为后缀. 可以使用 Debian 的 `dpkg` 工具解包(安装); 详见联机手册.
- 源代码包, 包括一个描述源代码包的 `.dsc` 文件, 一个包含 gzip-tar 归档压缩格式的未经修改源码的 `.orig.tar.gz` 文件, 一个包含对源代码作Debian特有修改的 `.diff.gz` 文件. 可以使用 `dpkg-source` 打包和解压 debian 源码文档. 详见联机手册.

软件包体系使用包维护者特制的 "依赖关系(dependencies)" 来安装软件. 这些依赖关系被写在每个包的 控制(`control`) 文件里. 例如: 安装依赖于 `binutils` 软件包的 `gcc` 时,如果没有预先安装 `binutils`, 包管理系统 (dpkg) 就会停止安装 `gcc`, 并返回需要 `binutils` 的错误信息.(解决这类问题,见`dpkg(8)`). 详见'包的关联' on page 19.

Debian 的打包工具可用于:

- 维护和管理软件包或部分软件包,
- 用于大软件包的切割, 如, 需要使用小容量软盘来传输,
- 帮助开发者构建软件包, 并且
- 帮助用户进行远程(FTP)安装.

6.2 Debian 软件包的格式?

一个 Debian "软件包", 或 Debian 归档文件, 包含可执行文件, 库文件, 附属文档. 名字通常以 `.deb` 为后缀.

Debian 二进制软件包内部格式描述见`deb(5)`联机手册. 由于这种内部格式会改变的(特别对于Debian GNU/Linux 的主发行版), 所以通常使用 `dpkg-deb(1)` 操作.deb文件.

6.3 为什么 Debian 软件包名字这么长?

Debian 二进制软件包的命名格式: <foo>_<版本号>-<Debian修订号>.deb

注意, `foo` 是假定的软件包名. 作为检验, 你可以通过下面的方法之一了解和软件包名称对应的一个 debian 软件包(.deb 文件):

- 检查 Debian FTP 站点下的 "Packages" 文件, 文件中包含对于每个软件包的描述段, 每个描述段的第一个字段就是正式包名.
- 使用命令 `dpkg --info foo_VVV-RRR.deb` (这里 VVV 和 RRR 是被查询包的版本和修订版本). 显示的内容中将包含软件包的对应名称.

VVV 是指该软件开发者制定的版本号, 没有什么标准格式, 可能像 “19990513” 和 “1.3.8pre1” 一样有所不同..

RRR 是 Debian 的修订版本号, 由 Debian 开发者(或者创建 Debian 包的用户自己)指定, 反映了 Debian 软件包的修正层次, 一个新的修正版通常在 Debian Makefile (`debian/rules`)文件 Debian 控制文件, 安装, 删除脚本(`debian/p*`), 或软件包的配置文件中作了修改.

6.4 Debian 的控制文件是什么?

关于控制文件的详细内容参见 Debian 打包手册, 第 4 章, ‘Debian 系统的其他文档?’ on page 37.

下边是一个 Debian 软件包 hello 的简单配置文件的主要内容:

```
包名: hello
优先级: optional
类别: devel
安装大小: 45
维护者: Adam Heath <doogie@debian.org>
平台: i386
版本: 1.3-16
依赖: libc6 (>= 2.1)
描述: The classic greeting, and a good example
 The GNU hello program produces a familiar, friendly greeting.  It
 allows nonprogrammers to use a classic computer science tool which
 would otherwise be unavailable to them.
 .
 Seriously, though: this is an example of how to do a Debian package.
 It is the Debian version of the GNU Project's 'hello world' program
 (which is itself an example for the GNU Project).
```

包名(Packahe)字段给出软件包的名称, 这是软件包工具用以识别这个包的名称, 通常(单不是必须)和这个 Debian 软件包的名称的第一个字符串相似.

版本(Version)字段给出上游开发者的版本号和修正版本号,详见‘为什么 Debian 软件包名字这么长?’ on the previous page.

平台(Architecture)字段指定这个二进制包的编译硬件平台.

依赖(Depends)字段给出所依赖的包的列表.

安装大小(Installed-Size)字段说明安装这个包所需磁盘空间, 用于安装前端显示是否有足够的空间安装此程序..

类别(Section)行给出此包在Debian FTP上的存储位置,上存储此包的目录名(详见‘Debian 的 FTP 上有哪些目录?’ on page 13).

优先级(Priority), 对应安装来说的重要程度, 象 dselect 和 console-apt 一类的半智能软件可以据此对软件安装分类,详见 ‘包的优先级?’ on the next page.

维护者(Maintainer)字段给出当前维护此包负责人的电子信箱..

描述(Description)字段此软件包的简要说明.

更多内容参阅 Debian 打包手册, 第4章, “控制文件及其字段”.

6.5 Debian 的配置文件

配置文件(Conffiles)是一个配置文件列表(通常在 `/etc` 下), 软件升级时不会被覆盖, 以确保所含文件的本地配置不会被破坏, 使得可以在系统运行状态下升级.

运行:

```
dpkg --status package
```

查看 “Conffiles” 段来确定升级时哪些文件被保护.

6.6 Debian 的 preinst, postinst, prerm, 和 postrm 脚本?

这些是软件包安装前后自动运行的可执行脚本. 统称为控制文件, 是 Deian 软件包的“控制”部分..

它们是:

preinst Debian软件包(".deb")解压前执行的脚本, 为正在被升级的包停止相关服务,直到升级或安装完成.(成功后执行 'postinst' 脚本).

postinst 主要完成软件包(".deb")安装完成后所需的配置工作. 通常, postinst 脚本要求用户输入, 和/或警告用户如果接受默认值, 应该记得按要求返回重新配置这个软件. 一个软件包安装或升级完成后, postinst 脚本驱动命令, 启动或重起相应的服务.

prerm 停止一个软件包的相关进程, 要卸载软件包的相关文件前执行.T.

postrm 修改相关文件或连接, 和/或卸载软件包所创建的文件(见 '什么是虚拟包?' on this page.)

当前的所有配置文件都可在 `/var/lib/dpkg/info` 目录下找到, 与 `foo` 软件包相关的命名以 "foo" 开头,以 "preinst", "postinst", 等为扩展.这个目录下的 `foo.list` 文件列出了软件包安装的所有文件.(注意这些文件的位置是dpkg确定的; 可能会因Debina版本而异)

6.7 包的优先级?

每个软件包都有一个维护者指定的优先级, 用于包管理系统. 这些优先级是:

- **必须的(Required)**: 系统运转所必须的软件包.
  包括修复系统缺陷所必须的所有工具. 不能删除这些软件包, 否则系统可能会崩溃, 且甚至有可能无法用 dpkg 恢复. 仅有这类包的系统是不可用的, 但是它为系统管理员启动系统安装其它软件提供足够的功能.
- **重要的(Important)**: 在任何类 Unix 系统上均安装有该级别软件包.
  没有这类包, 其它的包无法在系统上正常运转或使用, Emacs , X11 , TeX 等大型应用程序不在此列. 此类包构成基本系统.
- **一般的(Standard)**: Linux 系统里的一般软件包, 构成小型字符系统.
  这是用户什么也不选也会默认安装的软件包. 不包括大型软件, 但是 Emacs(与其说它是一个应用软件,不如说它是基础构件)一小部分 TeX 和 LaTeX(不支持X)除外.
- **可选的(Optional)**: 软件包包含了所有的你想要安装的文件, 如果你一开始不知道它是什么. 或者没有特殊的需要.
  这包括 X11, 所有的 TeX 和许多应用程序.
- **额外的(Extra)**: 这类包不是与其它高优先级的软件冲突, 只有知道它的用途才可能对你有用, 就是因为特别的原因而不能进入"可选"优先级.

6.8 什么是虚拟包?

是指一组具有近似功能的软件的统称, 例如 `tin` 和 `trn` 都是新闻阅读程序, 为系统中其它需要新闻阅读的程序提供支持. 因此可以说它们都提供了"新闻阅读 (`news-reader`)"的虚拟包.

同样, `smail` 和 `sendmail` 都提供了邮件传输代理的功能. 因此说它们提供了"邮件传输代理(mail transport agent)"的虚拟包,两者安装都可以满足其它程序对于"邮件传输代理 (`mail transport agent`)"虚拟包的需求.

Debian 提供这样一种, 如果系统中提供同一虚拟包的软件了安装了多个, 系统管理员可以指定一个为首选. 相关的命令是 `update-alternatives`, 更多描述详见'Debian 对不同喜好的支持?' on page 35.

6.9 包的关联

Debian 的软件包管理系统有一套包"依赖性"概念, 用以标示(一个标志符号)系统中程序 A 对于现存程序 B 的依赖程度:

- 软件包 A *depends* 软件包 B, 指运行 A 必须安装 B, 某些情况下 A 不仅依赖于 B, 还依赖于它的版本. 这种情况通常有最低版限制, A 更依赖于 B 的最新版而不是特定版.
- 软件包 A *recommends* 软件包 B, 如果软件维护者认为用户更喜欢 B 提供功能支持的 A.
- 软件包 A *suggests* 软件包 B, 如果 B 的某些软件与 A 的功能有关(通常是增强).
- 软件包 A 与软件包 B *conflicts* 指如果系统中有 B 则 A 不能运行. 通常是因为 A 包含了 B 中文件的改进而出现冲突, "Conflicts" "replaces" 经常同时出现.

- 软件包 A *replaces* 软件包 B, 指 A 安装时, B 中的文件会被 A 的删除和覆盖.
- 软件包 A *provides* 软件包 B, 指 A 会提供 B 所有的功能和文件, 这种机制为那些磁盘空间受限用户提供了一个方法, 即只安装 A 中需要的部分.

以上条目的更详细信息参阅打包手册和策略手册

6.10 Pre-Depends 什么意思?

"Pre-Depends" 是一种特别的依赖关系, 很多软件, 不管系统中它的依赖包是否存在, `dpkg` 都会将其(即, `.deb` 文件)解包, 解包通常是指释放出包中的安装文件, 如果系统中不存在依赖的包, `dpkg` 将会拒绝完成安装(执行它的"配置"动作).

但是, 对于某些包, 在依赖性问题解决前, `dpkg` 甚至拒绝解包, 这就称作, 这种包对某些包有 "Pre-depend" 关系. Debian 项目提供这种机制是为了系统由 `a.out` 格式安全升级到 `ELF` 格式, 这种情况对于解包要求非常严格. 对于其它的重要升级这种方法也非常有用, 比如对那些 "required" 级并有 LibC 关联的软件包.

更多信息,详见 打包手册.

6.11 包的状态(*unknown, install, remove purge* 和 *hold*)?

这些 "want" 标志位描述了用户打算如何操作一个软件包(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 - 用户并没指出他想对软件包进行的操作
- 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, 但不想删除它的配置文件.
- 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状态.

6.12 如何锁定一个包?

有两种办法锁定软件包, 使用dpkg 或 dselect.

使用dpkg, 仅需要导出软件包选择列表:

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

然后编辑 `selections.txt` 文件, 修改你要锁定的包的所在行, 例如 `libc6`将:

```
libc6                                           install
```

改为:

```
libc6                                           hold
```

然后存盘再把它导入 dpkg 数据库:

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

使用dselect, 仅需要进入 [S]elect 屏幕, 找到要锁定的(软件)包,按下 '=' 键 (或 'H'). 离开 [S]elect 屏幕后,改动马上生效.

6.13 如何安装一个 source 包?

Debian 源代码包实际上不能“安装”, 只是解包到你欲创建二进制包的目录.

大多数二进制软件包的镜像站点都提供源代码包, 如果在你的 APT 的 `sources.list(5)` 文件中写入了相应的 “deb-src” 源, 通过运行

```
apt-get source foo
```

来下载源代码包

Debian 源代码包提供了所谓的构造-依赖机制, 即源代码包的维护者提供了一个创建包所依赖的包的列表, 创建二进制包前运行

```
apt-get build-dep foo
```

你就知道它的用处了.

6.14 如何从源码创建二进制包?

编译源码, 你需要所有的ffoo_*.dsc, foo_*.tar.gz 和 foo_*.diff.gz (注意, 对于由 Debian 开发的软件包, 没有 .diff.gz 文件)(注: 指对于 foo 软件包).

完成后(‘如何安装一个 source 包?’ on this page), 如果你已经安装了 dpkg-dev (软件)包,运行一下命令:

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

将释放包到 `foo-version` 目录.

如果仅想编译这个包,进入 `foo-version` 目录, 执行命令

```
dpkg-buildpackage -rfakeroot -b
```

创建包(注意, 需要 `fakeroot` package 软件包), 然后

```
dpkg -i ../foo_version-revision_arch.deb
```

来安装新创建的包.

6.15 如何自己制作 Debian 包?

更多细节, 阅读 新维护者指南, 该文档在 `maint-guide` 包中, 或浏览 http://www.debian.org/doc/devel-manuals#maint-guide.

Chapter 7

Debian 的包管理工具

7.1 提供了哪些管理工具?

7.1.1 dpkg

这个主要的软件包管理工具, 有很多参数, 常用的有:

- 参数查找: `dpkg --help`.
- 输出指定软件包的控制文件(和其它信息): `dpkg --info foo_VVV-RRR.deb`
- 安装软件包(包括解包和配置): `dpkg --install foo_VVV-RRR.deb`.
- 解包(但不配置): `dpkg --unpack foo_VVV-RRR.deb`. 注意, 此操作解开的包处于不可用状态, 要正常运行, 一些文件还需要进一步配置. 这个命令会删除这个程序的已安装版本. 并运行相关联的 preinst 脚本(详见'Debian 的 preinst, postinst, prerm, 和 postrm 脚本?' on page 18).
- 配置一个解开的包: `dpkg --configure foo`. 这个操作会运行相关联的 postinst(详见'Debian 的 preinst, postinst, prerm, 和 postrm 脚本?' on page 18)脚本, 并升级 `conffiles` 中列举的文件. 注意,'配置'(configure)操作使用软件包名(如 foo), 而不是Debian文档文件名(如, foo_VVV-RRR.deb).
- 从 Debian 包中释放一个名为 "blurf" 文件(或一组名为 "blurf" 的文件): `dpkg --fsys-tarfile foo_VVV-RRR.deb | tar -xf - blurf*`
- 删除软件包(不包括它的配置文件): `dpkg --remove foo`.
- 删除软件包(包括它的配置文件): `dpkg --purge foo`.
- 列出包含 "foo*" 字符串的软件包的状态: `dpkg --list 'foo*'`.

7.1.2 dselect

Debian 包管理系统的菜单界面. 对第一次安装和大范围升级特别有用.

`dselect` 可以:

- 引导用户选择安装或删除软件包时, 确保要安装的包不与其它包冲突, 使得每个(要安装的软件)包正常运行的所需的所有软件包都被安装.;
- 对用户所作选择的不一致和不兼容做出警告;
- 确定必须安装的软件包的安装顺序;
- 自动完成安装或删除; 并
- 引导用户完成每个软件包的配置\.

进入 dselect 时, 有七个菜单项, 每项完成特定的功能, 用户可以用上下键移动亮度条, 然后按 <回车> (*<enter>*) 键选择加亮显示的项.

接下来的显示和用户的选择有关. 如果选的不是 Access 或 Select, dselect 就会继续执行指定的动作: 如, 选 Remove, dselect就会删除用户在 Select 中的最后选择的所有软件包.

Access 和 Select 菜单项下有更多的菜单选项, 两种选择都会出现分屏, 上部给出选项的滚动列表, 部是对应选项的简要解释(“info”).

任何时候都可按下 “?” 键, 都可呼出帮助信息, 使用在线帮助.

通常按照第一个界面里菜单排列顺序操作来完成软件包安装, 但, 用户也可以根据自己的需要来做出选择(但这和用户的具体选择有关)..

- 通过选择一个 **Access Method** 开始. 这是指用户访问 Debian 软件包的方法; 如有些用户从 CD-ROM 上获取 Debian 软件包, 有些打算通过匿名 FTP 获取. dselect 退出后所选的 “Access Method” 会被存储下来, 因此如不再调用这个选项, 访问方式不会改变.
- 接着 **Update** 可用软件列表. dselect 读取位于 Debian 软件包存储目录的顶层的 “Packages.gz” 文件(如果不存在, dselect 会试图生成一个).
- **Select** 欲安装的软件包, 选取此菜单项后, 首先显示的是满屏的帮助信息(除非使用了 ‘–expert’ 命令行参数), 退出帮助, 就会出现选择软件包的分屏菜单..

  相对较窄的上半屏是 29000 个 Debian 软件包的滚动列表; 下半屏是对应软件包或包组的描述T.

  用高亮条来选定软件包名或者包组, 然后选择操作:

  安装: 按下“+”键.

  删除: 有两种办法删除:

  - 删除: 删除软件包的大部分相关文件, 但是不包括配置文件(参阅 ‘Debian 的配置文件’ on page 18)内列举的需要保护的文件和软件包的配置信息, 使用 ‘-’ 键..
  - 清除: 删除软件包的所有文件, 使用 ‘_’ 键.

  注意，不可能删除 “所有” 的软件包. 如果试图这样做, 系统就会回到最初的基本系统状态.

  锁定 使用 ‘=’ 键, 告诉 dselect 即使这个包即使不是最新版本也不要升级.

  可以通过 ‘:’ 键来取消锁定, 这是默认设置.

  可以使用不同的排序方式显示软件包列表, 使用 ‘o’ 键在不同的排序方式间切换. 默认的排序方式是优先级排序, 同一优先级内, 按存储目录排序, 这种排序方式, 可能先显示 A 目录的软件包, 然后是B, 接着是低一个优先级的 A.

  你也可以使用 ‘v’ (verbose) 键来展开屏幕顶部标签的解释. 这样就可以在右边显示更多的内容, 通过左右箭头来进行左右移动.

  如果选了安装或删除一个软件包, 如foo.deb, 这个软件包又依赖于(需要)另一个(软件)包, 如 blurf.deb, dselect 就会在下一屏显示出来, 可以对系统建议(安装或不安装)做出选择, 接受或拒绝. 按 Shift-D 键稍后操作, Shift-U 返回上一菜单. 任何时候都可以按 Shift-Q 保存选择,返回主菜单.

- 返回主菜单, 选择 “Install” 菜单进行对选择的软件包解包和配置. 或者选择 “Remove” 菜单来删除. 选择 “Quit” 退出 dselect, 所作选择会被 dselect 保存下来.

7.1.3 dpkg-deb

用于 Debian 格式 (.deb) 文件的操作. 常见操作:

- 查看全部选项: dpkg-deb --help.
- 查看 Debian 格式软件包内的文件: dpkg-deb --contents foo_VVV-RRR.deb)
- 释放 Debian 格式软件包内的文件到指定目录: dpkg-deb --extract foo_VVV-RRR.deb tmp 释放 foo_VVV-RRR.deb 内的全部文件到 tmp/ 目录. 这是不需要安装, 在本地目录测试(软件)包内容的简单方法.

注意, 仅仅执行 dpkg-deb --extract 并不能正确的安装软件包, 应该执行 dpkg --install.

详细参见手册 dpkg-deb(1).

7.1.4 apt-get

`apt-get` 提供一个简单的命令行安装软件包的方法. 和 `dpkg` 不同, `apt-get` 不能识别 .deb 文件, 它使用软件包原来的名称通过 `/etc/apt/sources.list` 指定的安装源进行安装.

更多信息, 安装 `apt` 软件包参阅 `apt-get(8)`, `sources.list(5)` 和 `/usr/share/doc/apt/guide.html/index.html`.

7.1.5 dpkg-split

这个程序用来将大软件包分割成小文件(如, 写到软盘上), 和将分割的文件合并. 这个程序只能在 Debian 系统上使用(或含 `dpkg` 包的系统), 因为它需要调用 `dpkg-deb` 程序分析这个 Debian 包的组成.

例如, 把一个大 .deb 分割成 N 部分,

- 执行命令 `dpkg-split --split foo.deb`. 就会在当前目录出现分割出N个大小为 460KB 的文件.
- 把这 N 个文件考到软盘上.
- 把软盘上的内容考到其他机器上.
- 使用命令 `dpkg-split --join "foo*"` 合并.

7.2 Debian 可以对一个运行中的程序进行升级, 如何做到的?

Debian GNU/Linux 系统的内核支持运行中替换文件.

我们另外提供可一个称作 `start-stop-daemon` 的程序, 用于启动时驱动进程或内核运行级别发生变化时停掉进程(如, 由多用户到单用户模式或到关机模式). 包含某个进程的软件包安装时, 安装脚本停止和重起进程调用用的也是这个程序.

7.3 我的 Debian 系统上装了哪些软件包?

要查看 Debian 系统上安装的所有软件包的状态, 运行

```
dpkg --list
```

输出每个软件包的一行简单介绍, 2字符的状态标志, 包名, 所安装版本, 和简要描述.

查看以 "foo" 开头的软件包的状态, 执行:

```
dpkg --list 'foo*'
```

要得到某个软件包的更详细信息, 执行:

```
dpkg --status packagename
```

7.4 如何找出一个文件的归属包?

要查找出包含文件 `foo` 的软件包, 执行:

- `dpkg --search filename`
  在已安装软件包中搜寻 `filename`.(等同于搜索 `/var/lib/dpkg/info/` 目录下扩展名为 `.list` 的文件, 并输出所有包含此文件的软件包名和版本号).
- `zgrep foo Contents-ARCH.gz`
  通过绝对路径来搜寻含 `foo` 字符串的文件, `Contents-ARCH.gz` 文件(ARCH 指要查询的平台)在 Debian FTP 的主软件包目录(main, non-free, contrib)下, 一个 `Contents` 文件只包含同一目录下的软件包,因此用户查找含 `foo` 文件的软件包，需要搜寻多个 `Contents` 文件.
  相对于 `dpkg --search` 这种方法的优点是，它不仅仅搜寻系统已安装软件包.

Chapter 8

更新系统

Debian 的设计目标之一就是提供一致的升级途径和安全的升级过程. 我们一直尽力于平滑升级过程的实现. 如果升级过程中软件包将会对重要的注意事项警告用户, 并提供一个可能的解决方法.

你也应该阅读 Debian CD 上的发行记录, 该文档对升级作了详细描述,也可从http://www.debian.org/releases/stable/releasenotes处获得该文档.

8.1 把基于 libc5 的 Debian1.3.1(或更低)升级到基于 libc6 的2.0版(或更高)?

有几种升级的方法:

- 使用一个叫做 autoup.sh 的简单的 shell 脚本进行大部分重要软件包的升级, autoup.sh 升级完成后, 再使用 dselect 进行其他软件包的安装. 这是推荐使用的,但不是唯一的方法.

  最新版的 autoup.sh 可以在一下地址找到:

  - http://www.debian.org/releases/2.0/autoup/
  - http://www.taz.net.au/autoup/
  - http://debian.vicnet.net.au/autoup/
- 下边的方法与 Debian libc5 to libc6 Mini-HOWTO (http://debian.vicnet.net.au/autoup/HOWTO/libc5-libc6-Mini-HOWTO.html) 方法很接近, 手动升级大部分的重要软件包. autoup.sh 就是基于 Mini-HOWTO 的, 因此这种方法和使用 autoup.sh 差不多.
- 使用基于 libc5 的 apt. APT 是一个有可能替代 dselect 的软件包工具. 现在的 APT 与 dselec 不同, 是命令行界面你可以在 Debian 的 dists/slink/main/upgrade-older-i386 目录下找到基于 libc5 的版本.
- 在没有手动升级任何软件包的情况下, 使用 dselect. 如果可以不使用, 建议用户尽量不要使用这种方法 .因为 dselect 并不是使用优化的顺序来安装软件包, APT 要安全的多.

8.2 更新我的系统?

可以简单的匿名登录到 Debian 的 FTP, 找到自己想要的文件包, 把它下载下来,然后用 dpkg 安装. 注意, dpkg 会进行升级安装, 即使这个软件正在运行. 有时候安装修正包需要另一个包的修正版本. 这样的话, 安装就会停止, 直到另一个包被安装.

很多人认为这种方法过于浪费时间. 因为 Debian 升级太快, 每周都有很多新软件上载. 在一个新版本发行前会更多. 因此, 许多用户希望使用更加自动的方法. 有几个软件包可以实现这种目的:

8.2.1 APT

APT 是 Debian 文件系统的高级界面. apt-get 是处理软件包的命令行工具, APT dselect 是 dselect 的 APT 接口, 提供了一个简单的, 安全的安装和升级软件包的方法.

APT 的特性包括: 定制式安装, 多安装源支持, 还有其它一些特点. 见用户指南 /usr/share/doc/apt/guide.html/index.html.

首先安装 apt 软件包, 编辑 /etc/apt/sources.list 并使之生效, 如果你想升级到 Debian 最新稳定版, 可以使用类似这样的安装源:

```
http://http.us.debian.org/debian stable main contrib non-free
```

可以用其它你附近较快的 Debian 镜像来替换 http.us.debian.org 更多信息见 http://www.debian.org/misc/README.mirrors.

更多细节参见 apt-get(8) 和 sources.list(8) 联机手册, 以及前边提到的/usr/share/doc/apt/guide.html/index.html处的用户指南.

然后运行

```
apt-get update
```

接着

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

回答每一个可能出现的问题, 完成系统升级.

在 dselect中使用APT, 在 dselect 的方法选择屏幕选择APT存取方法. 然后指定要用的安装源, 配置文件是 /etc/apt/sources.list, 其格式在 sources.list(5) 联机手册有详细描述.

果你要从 CD 安装软件包, 可以使用 apt-cdrom. 更多细节见发行备忘录的"本地镜像升级的设定"章节.

注意安装完成后, 你下载用于安装的包仍然在你的 /var 目录下, 要释放空间, 记得用apt-get clean 和 apt-get autoclean 将它们删除或移到别的地方(提示:使用 apt-move).

8.2.2 dpkg-ftp

这是 dselect 中较老的一个方法. 可以从 dselect 中调用, 因此允许用户使用它直接下载安装软件, 在dselect 选择 ftp 存取模式指定远程主机名和目录, 那么 dpkg-ftp 就会自动下载选中的软件包.

注意, 和 mirror 程序不同, dpkg-ftp 不会抓取镜像站点的所有东西, 而是仅仅下载你(启动时)选择的软件包, 然后将它们升级.

dpkg-ftp is somewhat obsolete. You should use the APT access method with ftp:// URLs in sources.list instead.

8.2.3 mirror

一个 Perl 脚本, 和其称作 mirror-master 的管理程序, 用来通过匿名 FTP 从指定的主机上下载目录树中用户指定的部分.

mirror 对于下载大量软件包非常有用. 从站点上下载的软件被保存成一个称作 .mirrorinfo 的文件, 存在本地. mirror 自动跟踪远端文件系统的变化, 并与这个文件比较, 并下载不同部分.

mirror 程序对于升级远端目录树在本地的拷贝非常有用, 下载的文件不一定是 Debian 格式文件. (mirror是一个 Perl 脚本, 因此也可以在非 UNIX 系统上运行). 尽管 mirror 程序提供了排除文件名中和用户指定字符串匹配的机制, 这个程序相对于选择性下载, 对于下载整个目录树更有用.

8.2.4 dpkg-mountable

dpkg-mountable 为 dselect 增加了一个称作 'mountable' 的存取方法. 允许你从任何一个在 /etc/fstab 指定的文件系统安装, 例如, 文档系统可以是普通的硬盘分区或 NFS 服务器, 可以必要时自动挂接或卸掉.

还有一些特性并不能在标准 dselect 方法中找到, 比如提供本地文件树(可以同主发行版并行或者分开), 和仅下载需要的软件包, 而不是费时的对整个目录反复扫描, 和记录所有安装软件包的作用.

8.3 升级软件必须是单用户模式吗?

不. 即使在运行状态的软件包也可以升级. Debian 有一个 start-stop-daemon 程序, 升级过程中必要时, 用于停止, 启动运行的进程.

8.4 需要在硬盘上保留所有的 .deb 吗?

不. 如果你把文件下载到了你的硬盘上(不是必须的,详见 dpkg-ftp 的描述), 安装完成后, 可以把它们删除.

8.5 添加软件日志?

dpkg 保留一个已经解包, 设置, 删除 和/或 完全删除的包的记录, 但当一个包被处理时不保存当前终端的记录.
最简单的办法就是让它记录 dpkg/dselect/apt-get/ 的在 script(1) 里的所有会话.

Chapter 9

Debian 与内核

9.1 可以不考虑 Debian 因素编译内核吗?

可以.

需要注意的是: Debian 的 C 库文件是在最新的 *stable* 发行版的 **kernel** headers 基础上构建的, 如果你碰巧需要使用比 *stable* 发行版的 kernel headers 更高版本编译一个程序, 那么要么升级包含headers的软件包(`libc6-dev`), 要么从新版的 kernel 中解压出 headera 来使用, 如果 kernel 源文件在 `/usr/src/linux` 目录下, 那么编译时需要在命令行加入 `-I/usr/src/linux/include/`.

9.2 Debian 的编译内核工具

如果用户想(或必须)定制内核, 建议下载使用 `kernel-package` 软件包, 其包含构建 kernel软件包的脚本, 并提供了创建 Debian kernel-image 软件包, 在 kernel 源文件的最上层目录运行命令:

```
make-kpkg kernel_image
```

要获取帮助, 运行

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

, 或者查询 `make-kpkg(1)`.

如果没有现成的 kernel-source-version 软件包(这里 "version" 指 kernel 版本号), 则用户必须自己到 Linx 站点上下载新版的 kernel(或者需要的那个版本)..

在 `/usr/share/doc/kernel-package/README.gz` 处有 `kernel-package` 的详细使用说明. 主要步骤:

- 将 Kernel 源代码解包, 切换到新建目录.
- 用下面(任一)命令修改 kernel 配置:
  - `make config` (命令行界面).
  - `make menuconfig` (一个基于 ncurses 的菜单界面). 注意必须安装了 `libncurses5-dev` 软件包.
  - `make xconfig` (X11 界面). 需要安装相关的 X 和 Tcl/TK 软件包.

  置完成后, 就会在 kernel 源文件的最上层目录生成一个 `.config` 文件.
- 执行命令: `make-kpkg -rev Custom.N kernel_image`, N 指用户指定的版本数字. 就会生成一个修正版为 Custom.1 的新 Debian 包, 例如 Linux 2.2.14 内核，就会是`kernel-image-2.2.14_Custom.1_i386.deb`.
- 安装生成的软件包.
  - 运行 `dpkg --install /usr/src/kernel-image-VVV_Custom.N.deb` 安装内核. 安装脚本会:
    - 运行启动加载程序, LILO(如果安装了的话),
    - 安装放在 boot/vmlinuz_VVV-Custom.N 下定制的 kernel, 并生成相应的符号连接.
    - 提示用户制作启动软盘, 启动盘仅包含基本内核. 见'如何制作启动软盘?' on the following page.
  - 如果使用第三方启动加载程序, 如 `grub` 或 `loadlin`, 把这个影像考到相应位置(如 /boot/grub 或 `MS-DOS` 分区).

9.3 如何制作启动软盘?

制作启动盘需要用到 Debian 的 `boot-floppies` 工具, 该软件包位于 Debian FTP 的 `admin` 目录下. 该软件包的脚本会生成一个 SYSLINUX 格式的启动盘, 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 linxux(或在 syslinux.cfg 中定义的其它系统). 这个包中的其它脚本还可以制作应急盘, 甚至重建基本系统.

安装 `boot-floppies` 后可以在 `/usr/doc/boot-floppies/README` 中找到更详细信息..

9.4 Debian 下的模块管理?

Debian 的 `modconf` 软件包提供了一个 shell 脚本(`/usr/sbin/modconf`)用于完成模块的配置.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. package provides a shell script (`/usr/sbin/modconf`) which can be used to customize the configuration of modules. This script presents a menu-based interface, prompting the user for particulars on the loadable device drivers in his system. The responses are used to customize the file `/etc/modules.conf` (which lists aliases, and other arguments that must be used in conjunction with various modules) through files in `/etc/modutils/`, and `/etc/modules` (which lists the modules that must be loaded at boot time).

新版的配置帮助文件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/lib/modules_help/`), 告诉你如何对模块设定合适的参数.

9.5 我可以删除旧内核吗, 如果可以, 怎么做?

是的. `kernel-image-NNN.prerm` 脚本检查当前运行 kernel 是否与你要删掉的相同. 因此用下边的命令删除你不想要的内核影像:

```
dpkg --purge --force-remove-essential kernel-image-NNN
```

("NNN" 当然要用你的内核版本和修订号替换)

Chapter 10

定制 Debian GNU/Linux 的安装

10.1 如何确定所有的程序使用的是相同的页面尺寸(paper size)?

安装 libpaperg 软件包时, 会询问整个系统的默认页面尺寸, 设定会保存在 /etc/papersize文件里.

用户可以不用理会使用 PAPERSIZE 环境变量时的页面尺寸设置, 详见 papersize(5) 联机手册.

10.2 访问硬件设备的安全问题

/dev 目录下的许多设备文件属于预先设定的组, 例如 /dev/fd0属于 floppy 组, /dev/dsp 属于 audio 组.

如果要某个用户对设备有存取权限, 只要将他加入设备所属组就可以了, 即:

```
adduser user group
```

这样就不需要修改设备权限了.

10.3 如何启动Debian时加载控制台字体?

kbd 和 console-tools 软件包支持这种操作, 编辑/etc/kbd/config 或 /etc/console-tools/config 文件.

10.4 如何配置一个 X11 程序的默认值?

Debian 的 X 程序配置数据位于 /etc/X11/app-defaults/ 目录下, 如果你要定制一个 X 应用程序, 把你的配置数据放在那些文件里, 这样在升级时才不会被破坏.

10.5 好像每 个linux 发行版都有不同的启动方式, 告诉我 Debian 的方式.

同所有的 Unix 一样, Debian 启动时要执行 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. 该脚本执行 /etc/rcS.d/ 目录中各脚本的扩展名指定或衍生进程完成诸如检查并挂载文件系, 装载内核模块, 启动网络服务, 设定时钟等系统初始化工作. 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), 目录中的脚本通常是供系统管理员使用的, 用于有兼容性问题的软件包.

完成系统启动进程后, init 执行默认运行级别(该运行级别由 /etc/inittab 中的 id 给出)指定的所有的启动脚本. 同大多数 System V 兼 容Unix 一样, Linux 有 7 个运行级别:

- 0 (关闭系统),
- 1 (单用户模式),
- 2 到 5 (各种多用户模式), 以及

- 6 (重启系统).

Debian 系统运行 id=2, 它表示进入多用户模式时默认运行级别为 '2', 所以将运行 `/etc/rc2.d/` 中的脚本.

实际上, 任意目录 `/etc/rcN.d/` 中的脚本都是指向 `/etc/init.d/` 的符号链接. 然而, 每个 `/etc/rcN.d/` 目录中文件的名称用来指定 /etc/init.d相应脚本的运行方式. 特别是, 在进入任何运行级别之前, 所有名称以 'K' 打头的脚本均被运行, 这些脚本的工作是中止进程. 然后, 所有名称以 'S' 打头的脚本被运行, 这些脚本的工作是启动进程. 名称中跟在 'K' 或 'S' 后的两位数规定了脚本运行的先后次序, 数字小的脚本先运行.

采用这种工作方式是因为 `/etc/init.d/` 中的脚本均有一个参数规定脚本'start', 'stop', 'reload', 'restart' 或 'force-reload', 脚本按各自参数的赋值执行任务. 这些脚本甚至可以在系统启动后, 用来控制各种进程.

例如, 使用带 'reload' 参数的命令

```
/etc/init.d/sendmail reload
```

发给 sendmail daemon 进程一个信号, 要它重读配置文件.

10.6 好像 Debian 不使用 `rc.local` 定制启动过程; 那么提供了什么工具?

假设系统要在启动时运行 `foo` 脚本, 或进入指定的 (System V) 运行级别. 那系统管理员可以这样:

- 将 `foo` 脚本加入 `/etc/init.d/` 目录.
- 使用合适的参数运行 Debian 命令 `update-rc.d`, 这样就在(命令行指定的)rc?.d目录和 `/etc/init.d/foo` 之间建立了链接, 这里? 是 0 到 6 中的一个数字, 对应于 System V 的各个运行级别.
- 重启系统.

`update-rc.d` 命令会建立rc?.d目录中文件与 `/etc/init.d/` 目录中脚本的链接, 每个链接名会以 'S' 或 'K' 打头, 接下来是一个数字, 再就是脚本名. `/etc/rcN.d/`中以 'S' 打头的脚本在系统进入运行级别 `N` 时被执行. 以 'K' 打头的脚本在离开运行级别 `N` 时被执行.

还可以这样来做, 将脚本 `foo` 放在 `/etc/init.d/` 下然后使用 `update-rc.d foo defaults 19` 建立链接, 让 `foo` 脚本在系统启动期间执行. 参数 'defaults' 指默认运行级别, 它可以是 2 到 5 中某个值. 参数 '19' 确保 `foo` 在其它参数大于 20 的脚本之前执行.

10.7 软件包管理工具怎样处理非 Debian 格式的包?

有些用户可能想建立一个由 Debian 格式的包和非 Debian 格式的包混合组成的服务器, 通常这不是一个好主意, 因为 `dpkg` 无法了解非 Debian 格式包的配置文件, 因此在软件包升级时可能会出现冲突.

可以创建一个非 Debian 格式的包, 修改其配置文件所属组为 Debian 软件包所属组, 这样 `dpkg` 和其它的软件包管理系统就能识别本地系统管理员对于这个文件所作的修改,并且在升级时不会覆盖调它们.

10.8 不同版本软件包的文件的替代

假设系统管理员或本地用户想使用 "login-local" 而不是 Debian 提供的登录程序 `login`.

不 要这样做:

- 用 `login-local` 将 `/bin/login` 覆盖掉.

包管理系统并不知道这个变化, 当 `login`(或其它依赖 `/bin/login` 的软件包)安装或升级时, 你定制的 `/bin/login` 就会被覆盖掉.

应该是

- 运行:

```
dpkg-divert --divert /bin/login.debian /bin/login
```

使将来Debian的 `login` 包安装时, 写入 `/bin/login.debian` 而不是`/bin/login`.

- 然后:

```
cp login-local /bin/login
```

将你定制的程序移到相应位置.

Debian 提供了 `dpkg-divert(8)` 连接手册.

10.9 如何让 Debian 的包管理系统管理非 Debian 格式软件包?

执行:

```
dpkg-scanpackages BIN_DIR OVERRIDE_FILE [PATHPREFIX] > my_Packages
```

其中:

- BIN-DIR 指 Debian 格式软件包(通常扩展名为 ".deb")的存放路径i.
- OVERRIDE_FILE 这个文件由发行版的维护者编辑, 对于 "main" 发行版中的软件包, 通常保存在 Debian FTP 文档的 `indices/override.main.gz` 中. 对于非 Debian 格式软件包, 可以忽略这个文件.
- PATHPREFIX 是一个 可选 字符串, 在制作 `my_Packages` 文件时可以预先考虑.

一旦你构建了 `my_Packages` 文件, 使用以下命令告诉软件包管理系统:

```
dpkg --merge-avail my_Packages
```

如果你使用 APT, 也可以把本地源加入你的 `sources.list(5)` 文件.

10.10 Debian 对不同喜好的支持?

有这样一些情况, 两个不同的软件包提供了相同的基本功能, 一些用户因为习惯问题, 或者感觉界面比较友好选择其中一个, 而使用同一系统的另一些用户却有不同的选择.

Debian 使用虚拟软件包系统, 当有两个或两个以上包提供相同的基本功能时, 而没有指定特殊依赖时, 使得系统管理员(或用户)可以选择他们喜欢的工具.

例如, 系统中可能存在两个不同版本的新闻阅读器. 新闻服务器可能需要系统中新闻阅读器的支持 ,但是选 `tin` 还是`trn`就有用户决定了, 因为 `tin` 和 `trn` 提供的 `news-reader` 虚拟包都能满足要求, 通过指向所选文件(如 `/usr/bin/trn`)的称作 `/etc/alternatives/news-reader` 虚拟包的一个连接来调用该程序.

单个的连接并不能满足所有程序的要求, 通常系统中提供同一虚拟包的软件了安装了多个, Perl 脚本 `update-alternatives` 提供了一个方法指定一个软件包作为系统的默认A.

例如检查可用的 'x-window-manager', 运行:

```
update-alternatives --display x-window-manager
```

要更改,则:

```
update-alternatives --config x-window-manager
```

然后屏幕等待指令(敲入一个数字, 选择你喜欢的).

如果以为自身的原因, 没有注册为窗口管理器(比如存在一个错误), 或你使用了 /usr/locale 目录下的窗口管理器, 屏幕上可能没有你要的选择, 可以使用如下命令:

```
update-alternatives --install /usr/bin/x-window-manager \
  x-window-manager /usr/local/bin/wmaker-cvs 50
```

‘–install’ 选项的第一个参数是指向 /etc/alternatives/NAME 的一个符号连接, NAME 是第二个参数, 第三个参数是 /etc/alternatives/NAME 指向的程序, 第四个参数是优先级(值越大运行的可能性越大).

运行:

```
update-alternatives --remove x-window-manager /usr/local/bin/wmaker-cvs
```

来删除你的添加.

Chapter 11

获取 Debian GNU/Linux 的支持

11.1 Debian 系统的其他文档?

- I当前版本的安装说明: 参阅 http://www.debian.org/releases/stable/installmanual.
- 策略手册(Policy manual)文档列举了发行版的策略要求, 即, Debian FTP 的结构, 目录, 操作系统的几个设计问题, 等等. 还包括分发包的安全性技术要求, Debian 二进制包和源码包的基本技术.

  可从 debian-policy 软件包或在 http://www.debian.org/doc/devel-manuals#policy 处获得此文档.
- 已安装 Debian 包的文档: 大多数包的文档都解压在 /usr/doc/PACKAGE.
- Linux 项目的文档: doc-linux 软件包包含了来自 Linux 文档项目 (http://www.tldp.org/) 的大部分最新 HWOTO 和 miniHOWTO.
- 类 UNIX 联机手册: 很多命令都有一个类 UNIX 的联机手册. 它们参考了所在目录的 'man' 文件, 例如 foo(3) 参照了 /usr/share/man/man3/ 下的联机手册, 并且可以用 man 3 foo 命令将其呼出, 如果是 foo 的第一手册页可以直接用 man foo 呼出.

  可以通过 man -w foo 来学习一个特殊的联机手册 /usr/share/man/ 的内容.

  Debian 的新用户应该注意, 许多常用系统命令的只有安装了以下软件包才可以得到:

  - man-db, 包含了 man 程序本身, 以及操作手册页的其它命令w.
  - manpages, 包含系统手册页. (参阅 '对其他语言(非英语)支持的怎么样?' on page 10).
- 类 GNU 信息页: 许多命令的用户文档, 特别是 GNU 工具的, 是用 GNU 工具 info 读取的信息页. 在 GNU Emacs 或其它信息页浏览器中运行 M-x info 查看.

  手册页的主要特点是, 它是一个超文本系统. 不需要 WWW 的支持. info 可在纯文本控制台下运行. 它是先于 WWW 由 Richard Stallman 设计的.

注意你可以使用 WWW 浏览器在你的系统中访问很多文档, 这些文档可以通过 'dwww' 或 'dhelp' 命令从各自的软件包中获得.

11.2 有哪些讨论 Debian 的在线资源事实上 Debian 提供的获得技术支持的主要方法就是使用 email.

11.2.1 邮件列表

有很多 Debian 相关邮件列表 (http://www.debian.org/MailingLists/).

如果系统里安装了 doc-debian 软件包, 可以从 /usr/share/doc/debian/mailing-lists.txt 文件中获得完整的邮件列表.

Debian 的邮件列表以 debian-*list-subject* 的格式命名, 例如 debian-announce, debian-user, debian-news. 发送一个主题为 "subscribe" 的邮件到 debian-*list-subject*-request@lists.debian.org 就可完成对 debian-*list-subject* 列表的订阅. 用这种方法订阅或退订时记得注清请求.

你也可以使用WWW 表单 (http://www.debian.org/MailingLists/subscribe) 来订阅邮件列表. 也可以使用WWW 表单 (http://www.debian.org/MailingLists/unsubscribe) 退订.

如果你有什么麻烦也可以给邮件列表管理者发信 <listmaster@lists.debian.org>.

Debian FTP 的邮件列表可以通过网页在 http://lists.debian.org/ 处获得.

邮件列表的规则?

使用邮件列表时, 请遵守以下规则:

- 不要寄 spam(垃圾邮件). 参阅 Debian 邮件列表广告策略 (http://www.debian.org/MailingLists/#ads).
- 请不要故意滥用信息; 那是不礼貌的, Debian 的开发者都是志愿者, 贡献他们的时间, 精力, 和金钱来一起开发 Debian 项目.
- 不要使用恶劣的语言; 此外, 有些人是通过无线电包通讯来接收列表的, 而咒骂在这里是非法的.
- 确认您使用正确的列表. 尤其, 不要发送与用户相关的问题到与开发者相关的邮件列表[1]
- 提交错误见 '如何提交一个 Debian 中的错误?' on the facing page 章节.

11.2.2 维护人员

用户可以使用 email 向某个软件包的维护者提问, 如使用 *xyz@packages.debian.org* 联系 xyz 软件包的维护者.

11.2.3 新闻组

用户可以在以 comp.os.linux.* 或 linux.* 命名的 Linux 新闻组张贴非 Debian 性的 linux 问题, 网上有很多 Linux 新闻组和相关资源, 如在Linux Online (http://www.linux.org/docs/usenet.html) 和 LinuxJournal (http://www.linuxjournal.com/helpdesk.php) 站点.

11.3 寻找 Debian GNU/Linux 相关资料的快速方法?

有很多方法快速查询 Debian 相关服务文档:

- Debian 的搜索引擎 (http://search.debian.org/).
- Google Groups (http://groups.google.com/): 一个新闻组搜索引擎.

  例如, 查找别人搜寻 Debian 下 Promise 控制器驱动的经验,可以使用 `Promise Linux driver` 字段搜索. 你就会得到所有包含这些字符的帖子, 即讨论此主题的地方. 如果再加上 `Debian` 字段, 就会得到与 `Debian` 相关的主题.
- 任何一个常用的搜索引擎, 如 AltaVista (http://www.altavista.com/) 或 url id="http://www.google.com/" name="Google">, 使用正确的搜索字段.

  例如, 搜索 "cgi-perl" 就会得到关于这个软件包比它的控制文件里更详细的解释.

11.4 已知错误的记录?

Debian GNU/Linux 有一个错误跟踪系统(BTS)文件, 详细列举了用户和开发者报告的错误, 每个错误都配有一个识别码, 并持续保留到该错误被标示为处理完成为止.

可以在 http://www.debian.org/Bugs/ 处获得这个文件的拷贝.

可以给 request@bugs.debian.org 发一封正文为 "help" 的 e-mail 从邮件服务器获取错误跟踪系统的数据库.

[1] 应当使用 debian-*list-subject*-REQUEST@lists.debian.org 地址.

11.5 如何提交一个 Debian 中的错误?

如果你在 Debian 中发现一个错误, 请阅读报告 Debian 中错误的说明, 这个说明可以通过以下途径获得:

- 匿名 FTP. Debian 镜像站点的 `doc/bug-reporting.txt` 文件中含有这个说明.
- 通过 WWW. 在 http://www.debian.org/Bugs/Reporting 处有一个这个说明的拷贝.
- 任何一个安装了 `doc-debian` 包的 Debian 系统.这个说明在 `/usr/doc/debian/bug-reporting.txt` 文件中.

你可以使用 `bug` 或 `reportbug` 程序引导你完成错误报告, 并提交到正确的地址, 此过程会自动添加一些关于你系统的信息.

如果你要使用 email 程序发送信息到 <submit@bugs.debian.org>. 这个信息的第一应该类似于

```
Package: package-name
```

(将 *package-name* 替换成您要报告包含错误的软件包名称)第二行应该与这个包的版本号相关. 类似于:

```
Version: version-number
```

软件的版本号可以用下边的命令获得

```
dpkg -s package-name
```

这些仿真标头(pseudo-header)区域应该在一行的最前面. 下边的部分应该包含确切而完整的错误信息, 你使用的 Debian 的版本, 任何其他相依于这个问题软件包的软件包版本. 可以使用下边的命令查看 Debian 的版本

```
cat /etc/debian_version
```

然后等待你的错误提交报告的自动确认通知, 还会自动分配一个错误跟踪号. 加进错误记录和debian-bugs-dist 邮件列表.

您一次要报告很多个相似错误时, 请改把您的报告送到 <maintonly@bugs.debian.org> 去, 而不要送到 submit@... 在发送许多相似的错误报告之前, 您应该也送一份摘要到 debian-bugs-dist.

另外, 还有一个叫作 Lintian (http://www.debian.org/lintian/) 的 Debian 软件包检查器, 用于机械的检查 Debian 软件包的原则性错误和一般的打包错误. 因此, 如果你发现一个包中的错误, 可能在别的包中也出现, 最好和 Lintian的维护者<lintian-maint@debian.org> 取得联系, 重新生成 Lintian 检测报告而不是直接报告错误, 这样可以有效的阻止这样的错误在下一个版本或者发行版的其它包中出现.

你也可以使用 <quiet@bugs.debian.org> 只把错误报告提交给 BTS, 而不向debian-bugs-dist 或其维护者提交. 这个 'quiet' 地址很少使用, 例如错误已经送到维护者手上, 这只是把它归档到系统之中.

Chapter 12

为 Debian 项目捐赠

可以通过捐赠时间(开发新的软件包, 维护现有软件包, 或为用户提供技术支持), 资源(提供镜像空间), 或资金(提供测试的硬件平台)来资助这个项目.

12.1 如何成为一个 Debian 软件开发者?

Debian 的开发是全面开放的, 熟练的或好学的用户可以去维护被前任维护者 “遗弃” 的软件包, 开发新软件包, 或为用户提供技术支持i.

详见 Debian 站点的New Maintainer’s Corner New Maintainer’s Corner (http://www.debian.org/devel/join/newmaint).

12.2 如何向 Debian 项目捐赠资源?

因为 Debian 项目的目标是在全世界范围内建立可以快速方便存取的软件体系, 因此迫切需要镜像站点. 将所有的 Debian 格式软件制作镜像并不是必须的, 但是非常可取的办法.请访问url id=“http://www.debian.org/mirror/size” name=“Debian mirror size”>页面来获取更多的关于镜像所需硬盘空间的信息.

许多镜像站点完全由脚本自动控制, 但是偶尔的误操作或系统变动仍然需要人的干预.

如果你有高速的网络接入, 有用于制作整个或部分发行版镜像的资源, 愿意(或能找到人)对系统进行正常的维护, 请与 <debian-admin@lists.debian.org> 取得联系..

12.3 如何为Debian项目捐资?

可以单独向一两个 Debian 项目的关键组织捐赠O.

12.3.1 SPI组织

SPI 是 FSF 撤销对 Debian 项目资助时成立的非盈利性组织 .其目标是开发和分发自由软件.

目标与 FSF 非常相似. 鼓励程序员使用 GNU 通用公共许可证发布其软件. 但是在构建和发布一个 Linux 系统的许多技术细节上与 FSF 的 GNU 计划有所不同. 我们与 FSF 保持着联系, 合作修改 GNU 软件, 要求用户对 FSF 和 GNU 项目捐赠.

可以通过 http://www.spi-inc.org/ 访问 SPI.

12.3.2 自由软件基金会(FSF)

此刻 Debian 与 FSF 之间还没有正式的联系. 但是 FSF 对于一些构成 Debian 非常重要的软件负责, 包括 GNU C 编译器. GNU Emacs, 和系统中软件运行用到的许多 C 运行库, FSF 为今天许多的自由软件开辟了道路: 他们编写了许多 Debian 软件使用的通用公共许可证. 发起了 GNU 计划, 创建完全自由的 Unix 系统. Debian可 以认为是 GNU 系统的一个分支.

可以通过FS http://www.fsf.org/ 访问 FSF.

Chapter 13

作为商品销售 Debian GNU/Linux

13.1 我可以制作并销售 Debian CD 吗??

没问题, 对于我们发行版中的任何东西, 不需要得到我们的授权, 因此, 只要我们的 beta 测试完成后你就可以制作 CD. 不必付给我们任何费用. 当然, 所有 CD 制造商必须诚实的遵循 Debian 的软件许可证. 例如, 许多软件是遵循 GPL 发行的. 这就需要你发布他们的源代码.

同时, 我们将在网上公布一个为 Debian 项目捐赠资金, 软件, 和时间的 CD 制造商的名单. 鼓励用户从这些制造商处购买 CD, 即捐赠也是不错的广告.

13.2 可以包含非免费软件吗?

是的, Debian 主要组成是自由软件, 但是我们也提供了一个非自由目录放置非自由再分发的程序.

CD 制造商可以根据许可证条款或软件作者的个人声明来分发非自由目录下的软件. CD 制造商也可以在同一张 CD 上分发其他途径获取的非自由软件. 这已经不是什么新鲜事: 现在很多制造商都在同一张 CD 发布自由软件和商业软件. 当然, 我们仍然鼓励软件开发者以自由软件的形式发布他们的程序.

13.3 可以在 Debian GNU/Linux 上开发我的 Linux 版本吗?

是的. 例如, 有人制作了 “Linux for Hams”(业余无线电类的 Linux)发行版, 含有为无线电爱好者开发的程序, 并且这个版本是在 Debian 的基础上开发的, 添加了发射控制程序, 卫星跟踪程序, 等. 所有的程序都是用 Debian 格式添加的. 这样他的用户可以很容易的从从他以后发行的CD上升级.

市场上还有其他发行版的 Debian 如 Corel Linux 和 Storm Linux, 它们面向不同的用户群,是在 Debian GNU/Linux 的基础上加入了许多它们的产品.

Debian 还提供了允许开发者和系统管理员添加本地(非 Debian 格式)版本文件的机制, 这样其他软件升级时它们不会被覆盖. 这个问题的更多讨论见 ‘不同版本软件包的文件的替代’ on page 34.

13.4 可以我的商业程序做成 Debian 包吗?

当然可以. 软件包工具是自由软件, 可用于安装自由或非自由软件.

Chapter 14

对下一个 Debian 发行版的一些展望

14.1 增强安全性

Debian 从 1.3 版开始支持影子密码(shadow passwords). 另外, 可插入认证模块(Pluggable Authentication Modules)的linux库(见 libpam (http://www.kernel.org/pub/linux/libs/pam/);)允许系统管理员选择认证模式,最初的认证设定是通过影子密码完成的.

所支持的高级认证模式如 Kerberos, RSBAC 等, 在进一步的改进中.

14.2 增强对非英语用户的支持

已经有了一些非英语用户的支持, 详见 '对其他语言(非英语)支持的怎么样?' on page 10.

我们希望更多的人为更多的语言提供支持和翻译. 一些程序已经支持国际化了, 所以我们需要翻译信息目录. 许多程序还没有实现完全国际化.

GNU 翻译项目 ftp://ftp.gnu.org/pub/gnu/ABOUT-NLS 从事于GNU程序的国际化.

14.3 更多的体系结构

基于 SPARC64, SuperH 平台的 Debian 系统正在开发中.

14.4 更多内核

除了 Debian GNU/Hurd, Debian 正在致力于源于 NetBSD, FreeBSD 和 OpenBSD的DSB 内核的开发.

Chapter 15

关于这篇 FAQ 的一些资料

15.1 作者

本 FAQ 的第一版是由 J.H.M. Dassen (Ray) 和 Chuck Stickelman 制作维护的. Susan G. Kleinmann 和 Sven Rudolph对Debian GNU/Linux FAQ 作了修正, 以后是由 Santiago Vila 维护的, 现在的维护者是 Josip Rodin.

资料来源:

- The Debian-1.1 release announcement, Bruce Perens (http://www.perens.com/).
- The Linux FAQ, Ian Jackson (http://www.chiark.greenend.org.uk/~ijackson/).
- Debian Mailing Lists Archives (http://lists.debian.org/),
- dpkg 程序员手册 和 Debian策略手册 (参阅 'Debian 系统的其他文档?' on page 37)
- 许多开发者, 志愿者和测试者
- 作者的记忆片断. :-)

作者在此感谢所有在文档写作过程中曾给予帮助的人.

15.2 反馈

欢迎对本文档提出意见和建议. 请发送e-mail到 <doc-debian@packages.debian.org>, 或用 doc-debian (http://bugs.debian.org/doc-debian) 软件提交错误报告.

15.3 获取

可以通过 Debian 的网页 http://www.debian.org/doc/FAQ/ 查看本文档的最新版本.

也可以从 http://www.debian.org/doc/user-manuals#faq 下载纯文本, HTML, PostScript 和 PDF 格式的文档, 另外那里还有一些译本.

本文档使用 SGML 完成. SGML 可以从 doc-debian 的源代码包或:pserver:anonymous@cvs.debian.org:/cvs/debian-d 中获得.

15.4 文档格式

本文档使用 DebianDoc SGML DTD（由 LinuxDoc SGML 改进而来）完成的. 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.