目录

写在前面

无心插柳

笔者是从Slackware 13.1开始接触Slackware的，光盘中就附有Slackbook，一本很好的官方指南。但由于年代久远（第二版是2005年发行的，写作时是2012年），Slackware已经发生了很大的改变，其中的很多内容已不再适用。在网上发现2009年是就已经出了Slackbook 3.0的beta版，可正式版仍迟迟不出。一方面为了学习Slackware，另一方面为了为Slackware尽一点绵薄之力，遂决定翻译Slackbook。

笔者为Slackbook 3.0的beta版翻译了六章，由于beta版的内容实在过于简单，与Slackbook 2正式版实有天壤之别，最终决定放弃。但Slackbook 2为已经过时，实用价值大打折扣，遂决定在Slackbook 2基础之上，以翻译为主要途径，对其中过时的内容加以更新，提供一本好用的Slackware中文指南。

书中会尽量以笔者的使用经验对相关的内容进行陈述，尽量使用第一人称。另外，由于很多部分是翻译而来，其中会夹杂着以原作者为第一人称的描述。请见谅。另外，笔者水平有限，如有不恰当之处，敬请批评指正。

目标受众

Slackware Linux 操作系统作为一个功能强大的平台，是为基于Intel处理器的计算机设计的。它的设计目标是成为一个稳定，安全，实用的高端服务器以及功能强劲的工作站。

本书的初衷是带你走进Slackware Linux操作系统。这并不意味着本书涵盖了Slackware发行版的方方面面，而是为你展示它能做些什么，并且教会你使用该系统的基础知识。

如果你是Slackware的老手，那么我们希望本书可以作为参考手册使用。当然，我们也希望，如果有人向你问起Slackware，你可以向他们介绍介绍本书。Slackware，谁用谁知道。

当然，这本书不可能像小说一样吸引人，我们会尽我们所能地写得更有趣。写得好的话说不定还会有人请我们写剧本？当然，我们最希望的是你能从中学习到你认为有用的知识。

现在，准备被亮瞎吧！

为什么写本新的？

在现存的Linux发行版中，Slackware Linux算得上是爷爷辈的，但这并不意味着它与时代脱节。Slackware的确尝试过保持纯正的Unix风格，但还是没能逃过时代的进步的历程。子系统不一样了，窗口管理器变来变去，人们又发明了新的方法来管理复杂的现代操作系统。我们确实反对为了改变而改变，但当事物进化了，相应的文档就变得陈旧了，一切都是无法避免的——包括书籍。

本书中的一些约定

为了保持一致性，同时方便大家阅读，全书遵守一些约定。

印刷上的约定

Italic 斜体字*Italic*用于表示命令、强调、及技术名词第一次出现的地方。

`Monospace` 等宽字体`monospaced`用于表示错误信息、命令、环境变量、端口名、主机名、用户名、组名、设备名、变量及代码段。

Bold 粗体字用于表示例子中的用户输入。

用户输入

输入用粗体表示，用以区别其它的字符。至于键组合，我们用'+'来连接各个字符以表示要同时按下，例如：

```
Ctrl+Alt+Del
```

意味着我们必须同时按下*Ctrl*，*Alt*，以及*Del*键。

有时候是要先后键入的，我们用逗号来分隔它们，例如：

```
Ctrl+X, Ctrl+S
```

就代表我们要先同时输入*Ctrl*和*X*键，接着同时输入*Ctrl*键和*S*键。

例子

以`E:\>`开始的例子代表这是一个MS-DOS®命令。除非特别说明，这些命令都是在**Microsoft® Windows®**"命令提示符"之下运行的。

```
D:\> rawrite a: bare.i
```

以`#`开始的例子代表执行一个命令时必须使用Slackware中超级用户的权限执行。你可以直接以`root`登陆，也可以以普通用户登陆，之后再用`su(1)`来获得超级用户的权限。

```
# dd if=bare.i of=/dev/fd0
```

以%开始的例子代表执行命令时只须使用普通用户权限。除非特别说明，我们将使用C-shell语法来设置环境变量以及其它的shell命令。

```
% top
```

致谢

这个项目是由许多人花费几个月的贡献累积而成的。仅凭我一已之力，是不可能凭空创造出的。我们要感谢许许多多的人，感谢他们无私的奉献：Keith Keller 写了无线网络方面的内容；Joost Kremers单枪匹马完成了emacs这一章；Simon Williams完成了安全这章；Jurgen Phillippaerts完成基础网络命令的内容；Cibao Cu Ali G Colibri带来的启发以及在裤子上揣的一脚[1]。还有数不清的人为我们提出了建议及文章的修正。下面是一个完整的列表：Jacob Anhoej, John Yast, Sally Welch, Morgan Landry, 及Charlie Law。同时，我也要感谢Keith Keller为这个项目管理邮件列表，感谢Carl Inglis为我们的管理网页。最后但并非最不重要，我要感谢Patrick J. Volkerding创建了Slackware Linux，以及David Cantrell, Logan Johnson及Chris Lumens，是他们完成了Slackware Linux 精要的第一版。如果没有他们的初始框架，本书的一切都不可能发生。还有许多其它的人在本项目中的大大小小的方面做了贡献，但在此没有列出。希望他们能原谅我记忆不好。

Alan Hicks。2005年5月。

[1] 原文为a good kick in the pants. 译者对这些幽默不是很理解。

第一章　Slackware简介

1.1　什么是Linux？

Linux是一个操作系统的内核，是Linus Torvalds于1991年作为一个私人的项目完成的。最初他开始这个项目的目的只是为了不花钱就能得到一个基于Unix的操作系统。另外，他还想学习386处理器的细节。Linux是在遵守通用公共许可（见第1.3节及附录A中对该许可的介绍）下发行的，对于公众它是免费的，且任何人都能自由地学习并进行改善。现在，Linux已经成为操作系统市场上的一个重要的竞争者。人们已经把它移植到了许多架构上运行，包括HP/Compaq的Alpha，Sun公司的SPARC及UltraSPARC，以及Motorola的PowerPC芯片（在如苹果公司的Macintosh及IBM的RS/6000等计算机上运行）。当然世界上，即使没有上千，也有成百的程序员在开发Linux。在Linux 上可以运行如Sendmail、Apache及BIND等软件，这些都是当前流行的用来搭建英特网服务器的软件。要时刻记得，Linux这个词代表的是内核——一个操作系统的核心。这个核心的作用是控制计算机的处理器、内存、硬件驱动及外围设备。这也是Linux的所有功能：控制系统的运转并保证程序正常运行。一些公司或个人将Linux内核与一些程序绑定在一起构成一个操作系统。我们将每一个绑定称作一个Linux发行版。

1.1.1　GNU的世界

Linux 内核项目最早是由Linux Torvalds在1991年时独自努力实现的，但正如牛顿说过的“我看得远，是因为站在巨人的肩上”一样，正当Linus Torvalds准备开发内核时，自由软件基金会已经有了协作软件的想法了。他们将他们的想法命名为GNU，GNU是一个递归的缩写词，全称为“GNU's Not Unix”。GNU的软件从Linux元年1月1日[1]开始运行在Linux内核上。人们用他们编译器`gcc`来编译内核。直至今天，许多GNU工作仍是每个主流Linux 发行版的基础，从`gcc`到`gnutar`。因此，许多自由软件基金会的支持者都坚持认为他们所做的工作有着不小于Linux内核的功劳。他们强烈建议所有的Linux发行版都应该叫作GNU/Linux发行版。

在这个主题上引起了很多口水战，只输于vi与Emacs间的圣战了。本书的目的并不是扇动这个已经炙手可热的讨论的战火，而是为新手阐明术语。当你看到GNU/Linux时，它表示Linux发行版，但你看到Linux时，它可能指的是内核，也可能指的是一个发行版。这还是很

[1] 原文为‘day 1’

难区分的。特别的，因为GNU/Linux很饶舌，所以一般不用。

1.2 Slackware是什么

Slackware，由Patrick Volkerding于1992年末发起，并最早发布于1993年7年17 日，它是第一个得到广泛使用的Linux发行版。Volkerding最早认识Linux是在需要为一个项目找一个廉价的LISP解释器的时候。那时候可用的发行版中有一个叫SLS Linux，是Soft Landing Systems公司的产品。Volkerding用的是SLS Linux，并在找到bugs时进行修复。最终，他决定将所有修复的bug合并到他自己私人的发行版中，让他和他的朋友使用。这个私人发行版很快就小有名气，所以Volkerding 就决定将它命名为Slackware并对公众开放。在此过程中，Patrick为Slackware添加了一些新的东西：一个很友好的基于菜单系统安装程序，以及软件包管理的概念，包管理让用户在自己的系统上方便地添加、删除及更新软件包。

Slackware能成为现存的最古老的发行版，有诸多原因。例如它从不试图模仿Windows，它尽可能地保持近似Unix。它并不试图用绚丽的指点GUI（图形用户接口(Graphical User Interfaces)）来隐藏一些操作。相反地，它让用户看到底层的内容，以赐予用户最大的可控制性。它的开发并不是为了赶什么进度，新版本只在准备好的时候才出。

Slackware适用于那些喜欢学习，喜欢通过配置自己的系统来实现想做的事的那些人。Slackware的稳定性和简单性是多年来人们不断使用它的原因。Slackware目前享有的美名是——一个坚固的服务器，一个严肃的工作站。你会发现，Slacware 可以运行几乎所有的窗口管理器或桌面环境，也可以不运行其中的任意一个。Slackware提供了强大的服务，每个服务器能使用的地方，都能见到Slacware的身影。Slacware的用户是Linux用户中满意度最高的。废话，我们当然会这么说了。:^)

1.3 开源与自由软件

在Linux社区中，有两种主要的意识形态运动。自由软件运动（我们下面会说到）的目标是使所有的软件都有免费的知识产权。这项运动的追随者们认为，知识产权的限制阻碍了科技的发展并与社区的优点想违背。开源运动的目标与前者差不多，但走了一条更实际的路线。这项运动的追随者们的论点是以使源代码自由可得所带来的商业及技术的好处为基础，而不是一个精神与论理上的律条来推动自由软件运动。

这个运动的底端是一些想更精密控制他们软件的组织。

自由软件运行是由自由软件基金会带头的，而该基金会正是为GNU项目筹款的。自由软件远不止是一个意识形态。一个用烂的说法是“free[2]指的是演讲的自由，而不是啤酒的免费。”。本质上，自由软件是尝试同时保证用户与开发人员的一些权利。这些自由权利包括基于任何目的地运行程序的权利，自由修改源码的权利，重新发布源码的权利以及共享你所做的修改的权利。为了保证这些自由，他们创建了GNU通用公共许可（GPL）。简单地说，GPL的内容是：任何人在发布一个遵守GPL许可的编译后的程序时，也必须提供源代码，并且只要做出的修改也以源码的形式提供，就可以对原先的源代码进行任意的修改。这就保证了一旦一个程序

[2]free在英语中即指自由，也指免费

为社区“打开”了，那么除非得到其中每部分代码（包括所做的修改）的作者的许可，那么这个程序就不能被“关闭”。Linux下的程序绝大多数是遵守GPL许可的。

要注意一件事，GPL并没有说明有关价格的事。也许就像听起来很奇怪一样，你可以对自由[3]软件收费。许可中“自由”的部分是针对源代码说的，而不是针对软件的价格据说的。（然而，一旦有人告诉你或给你一个遵守GPL的编译后的软件，那么他也有义务给你软件的源码。）

另一个流行的许可是BSD许可，与GPL不同的是，BSD许可并不要求发布程序的源码。遵守BSD许可的软件只要满足几个条件就可以重新以源码或二进制的形式发布。程序作者的凭证并不作为程序某种形式上的广告。它也免除了作者任何因使用该软件而造成损失带来的责任。Slackware中包含的许多软件是以BSD许可发布的。

站在年轻的开源运动前线的，是称为Open Source Initiative的组织，它单独存在，以获取开源软件的支持，也就是，软件总是可以在得到可执行程序的同时得到它的源码。它们并不提供一个特定的许可，而是支持多种开源许可。

OSI背后的想法是通过让公司自己撰写自己的开源许可，并使许可通过OSI的认证来争取更多的公司参与开源运动。许多公司同意发布源码，但并不想使用GPL许可。由于不能改变GPL许可，所以OSI为它们提供了撰写自己的许可的机会，这个许可最后由该组织认证。

虽然自由软件基金会与Open Source Initiative的工作是互相帮助的，但却不是一个东西。自由软件基金会使用一个特殊的许可并提供该许可下的软件，而Open Source Initiative则寻求所有的开源许可，包括自由软件基金会的那个。谈到使人们自由获得源码的领域时通常分为两项运动，但这分立的两种不同意识形态追求的目标是一致的，因此要信任双方所做出的努力。

[3] free，有免费的意思

第二章　帮助

在配置一个程序或安装一个硬件时，我们时常需要查看某个命令的帮助。也可能是你想对一个命令更深入地了解，或者查看它别的选项。好在我们有很多方式来得到我们寻求的帮助。在安装Slacware时，其中的“F”系列就包含了FAQ和HOWTO文档，你可以选择安装它。一般程序都会包含关于选项、配置文件及使用方法的帮助。

2.1　系统帮助

2.1.1　man

`man`命令（“manual”的缩写）是Unix及Linux系统中最为传统的在线文档。“man手册页”由特定格式的文件构成，涵盖了绝大多数命令，并与软件本身一同发布。很自然地，执行命令`man somecommand`就显示指定命令的man手册页，本例中，则应该显示我们虚构的命令`somecommand`的手册页。

你可能立即会想到，man手册页的数量会急剧上升，最后连高级用户都会被弄得晕头转向的。因此，man手册被分为几个节。这个设定已经存在很久了；久到我们会经常看到一些命令、程序甚至是编程库函数在引用时都附上了man的节号。例如：

你可能会看到诸如`man(1)`的引用。其中的数字告诉我们“`man`”的文档在第一节（用户命令）；你也可以用命令`man 1 man`命令来为“man”指定查看手册的第一节。在一个同名的项有多个手册页时，指定man应查看的节号是很有用的。

节号	内容
第 1 节	用户命令（只含介绍）
第 2 节	系统调用
第 3 节	C库调用
第 4 节	设备及特殊文件
第 5 节	文件格式及协议（即，wtmp、`/etc/passwd`及nfs等）
第 6 节	游戏（只含介绍）
第 7 节	一些约定，宏包等（即nroff,ascii等）
第 8 节	系统管理（只含介绍）

表 2.1: man手册分节

除了man(1)，还可以使用其它命令：whatis(1)及apropos(1)。这些命令的目的是使我们更容易在man系统中找到有用信息。

whatis命令会为系统命令给出非常简洁的描述，就像放在口袋的小抄一样。

例如：

```
$ whatis whatis
whatis []            (1)  - search the whatis database for complete words
```

apropos命令的作用是：给定一个关键词，在man手册页中搜索含有该关键词的内容。

例如：

```
$ apropos wav
SDL_FreeWAV          (3)  - Frees previously opened WAV data
SDL_FreeWAV []       (3)  - Frees previously opened WAV data
SDL_LoadWAV          (3)  - Load a WAVE file
SDL_LoadWAV []       (3)  - Load a WAVE file
cdda2wav []          (1)  - a sampling utility that dumps CD audio
 data into wav sound files
cwaves []            (6)  - languid sinusoidal colors
fadeplot []          (6)  - draws a waving ribbon following a sinusoidal path
interference []      (6)  - decaying sinusoidal waves
oggdec []            (1)  - simple decoder, Ogg Vorbis file to PCM
 audio file (WAV or RAW)
pilot []             (1)  - wav - Decodes Palm Voice Memo files to
 wav files you can read on your desktop
wavelan []           (4)  - AT&T GIS WaveLAN ISA device driver
```

如果你想深入了解这些命令，请阅读它们的man手册以获取更多信息。:)

2.1.2　/usr/doc文件夹

我们构建的多数软件包含有一些文档：README文件、使用说明及许可文件等。源码中包含的任何文档都安装在系统的/usr/doc文件夹。每个文件都会（一般而言是的）在/usr/doc中创建自己的文件夹，并将文档放在该文件夹中，该文件夹的名字遵守如下约定：

```
/usr/doc/$program-$version
```

其中，$program表示你想查找的程序的名字，$version字段（很明显）表示在系统中安装的软件的版本号。

例如，想阅读关于命令man(1)的文档，你可能想用cd命令切换到目录：

```
$ cd /usr/doc/man-$version
```

如果对应的man手册页没有提供足够的信息，或者没有你想找的那些信息，那么接下来可以考虑阅读/usr/doc文件夹下的相关内容。

2.1.3　HOWTO及mini-HOWTO

归功于开源社区的最真挚的精神，我们才有了HOWTO/mini-HOWTO。这些文件跟名字一样，是一些如何完成某些任务的文档或指南。如果你选择了安装HOWTO文档，那么它们会被安装在/usr/doc/Linux-HOWTOs，mini-HOWTO会被安装在/usr/doc/Linux-mini-HOWTOs。

该软件包中还含有FAQ集，FAQ是**F**requently **A**sked **Q**uestions，中文含义为“常问问题”。这些文档是以问答的形式写成的。如果你只是想快速地解决一个问题，那么通常选择查看FAQ是极其有用的。如果在安装时选择安装FAQ，那么它们会被安装在/usr/doc/Linux-FAQs目录下。

在你不知道如何处理一些东西时，这些文档通常是值得阅读的。它们涵盖了相当大的范围，而且解决方法通常很详细，详细到不可思议。好东西是吧!

2.2　在线帮助

除了Slackware中可以提供安装的这些文档外，在网上还在大量的在线资源可供我们学习。

2.2.1　官网及论坛

Slackware官网[1]

Slackware的官网有时比较过时，但还是有关于最新的Slackware版本的一些信息。有一个时期官网上是有在线帮助的，但后来多了很多人来骚扰生事，导致维护这个论坛变得很不容易，于是Patrick就把论坛关了。如果想找的话，可以在http://www.userlocal.com/phorum/上找到之前数据的一个可搜索的归档。

在http://slackware.com上的论坛挂掉之后，一些其它的站点如雨后春笋般崛起以提供Slackware的论坛支持。经过再三思量，Pat决定将http://www.linuxquestions.org作为Slackware的官方论坛。

2.2.2　E-mail支持

据说只要购买官方的CD，就可以通过电子邮件得到开发人员的免费安装支持。话虽如此，请记住，我们Slackware的开发人员（及绝大多数的用户）是“老派[2]”。这意味着我们更倾向于帮助那些真的有兴趣的“自助者”。无论谁给我们发email问问题，我们都会尽我们所能帮助他们。然而，请在发email前查阅相关的文档及网站（尤其是FAQ及下列的一些论坛）。因此通过这些途径你可能更快得到答案，同时我们也可以少回一些email，显然，我们也才能够更快地帮助那些需要帮助的人们。

[1] www.slackware.com
[2] The Old School

技术支持的email是support@slackware.com[3]。其它的邮箱地址及联系信息都在官网上给出。

2.2.3　Slackware的邮件列表

我们有许多邮件列表，不论是摘要式的，还是正常格式的。请查阅如何订阅邮件列表的说明。

要订阅邮件列表，请发送邮件到

majordomo@slackware.com

邮件正文写上“subscribe [列表名]”。列表名的描述在下面（使用下面列出的名字中的任意一个）。

邮件列表的完整归档可以在Slakcware的官网上找到：http://slackware.com/lists/archive

slackware-announce

slackware-announce 邮件列表是关于新版本的声明，主要的一些更新及其它一般的信息

slackware-security

slackware-security 邮件列表是关于安全问题的声明。任何直接涉及到Slackware的任何攻击或者漏洞都会立即在这份列表上发布。

这些列表还有对应的摘要版本。这意味着你可以每天得到一条大的信息而不是每天得到一大堆的信息。由于Slackware的邮件列表并不允许用户发表信息，因此该列表的流量很小，所以多数用户认为摘要版没什么用。但如果你需要的话，只要订阅slackware-announce-digest或slackware-security-digest 即可。

2.2.4　非官方的网站及论坛

网站

Google (http://www.google.com)

搜索引擎的功夫大师。当你积极地、真心地想找到一个主题的每一个信息：它是不二之选。

Google:Linux(http://www.google.com/linux)

专门搜索Linux版块。

Google:BSD(http://www.google.com/bsd)

专门搜索BSD的内容。Slackware是一个类Unix的通用系统，通用到经常可以在这里找到一些与Slackware百分百相关的详细信息。很多时候，一个BSD内容的搜索会比一般的面向大众的Linux搜索得到更多的技术细节。

[3]作者没有给他们发过邮件，不知道该邮箱的有效性。

Google:Groups(http://groups.google.com)
搜索几十年来Usenet上的文章来发掘你的智慧之光。

http://userlocal.com
一个关于知识、好建议、第一手经验及有趣文章的宝库。通常我们会在这里听到Slackware世界的新进展。

网络资源

linuxquestions.org [4]
Slackware 官方认可的论坛。

LinuxISO.org Slackware论坛 [5]
“一个下载Linux和寻求帮助的地方”。

alt.os.linux.slackware FAQ [6]
另一个FAQ。

Usenet小组（NNTP）

Usenet一直以来都是geek们聚集并互相帮助的地方。其中有几个致力于Slackware的内容。但里面的人更多的是在行的人。

`alt.os.linux.slackware`

`alt.os.linux.slackware`，更为人所知的名字是aols（不要和AOL®混起来!），在遇到Slackware的问题时，是获取相关的技术帮助的最活跃的地方。就像所有的Usenet新闻组一样，一些不帮忙的参与者（“山精”们[7]）总是糟到大家的非议。学会无视那些山精们，认出那些真心帮助别人的人，对于使这个资源最大化利用很重要。

[4] http://www.linuxquestions.org/questions/forumdisplay.php?forumid=14
[5] 貌似和Linuxquestions.org合并了，新网址是iso.linuxquestions.org/slackware
[6] 貌似挂掉了。http://wombat.san-francisco.ca.us/perl/fom
[7] 斯堪的那维亚神话中的，邪恶的巨怪或顽皮的侏儒

第三章　安装

使用Slackware前，当然要先获取Slackware并进行安装。不论是购买Slackware 的CD或是从网上免费下载都是很容易的。只要对自己的电脑有一定的基础知识，并且想进一步学习，安装Slackware是非常容易的。安装程序本身就像是一步一步进行的。因此，我们很快就能熟悉这个过程并快速地运行它。事实上，Slackware鼓吹自己是所有功能完善的Linux发行版中安装时间最短的版本之一。

3.1　获取Slackware

3.1.1　官方CD及套装

官方的Slackware CD套装可以从Slackware公司得到。CD套装包含6个CD[1]，第一张CD包含基本安装所需要的所有软件，具体包括A/AP/D/E/L/N系列及可启动的安装器，内核，testing/，及Slackwbook。第二张CD包含一些文档及X系统，具体包括F/K/T/TCL/X/XAP/Y、L系列的源码及/testing kernel source。第三张CD含有KDE系列及A/AP/E/F/安装器的源码。第四张CD含有KDEI、/extra软件包及D系列的源码。第五张CD含有KDE/XAP系列的源码。第六张CD包含/pasture软件包、K/N/T/TCL/X/Y的源码及USB和PXE的安装器。

你也可以购买一个套装，其中含有6个CD及Slackbook的印刷版，还有一些简洁的Slackware工具，可供你显摆你作为geek的自尊。我们倾向于通过Slackware store来从网上购买Slackware商品。

http://store.slackware.com

你也可以通过打电话或发邮件来订购。

方式	详细联系信息
电话	1-(925)674-0783
网页	http://store.slackware.com
Email	orders@slackware.com
写信	114 Claremont Drive, Brentwood, CA 94513

表 3.1: Slackware Linux公司的联系信息

[1]这里以Slackware 13.37为例，参见http://slackware.com/getslack/torrents.php

3.1.2　通过互联网

Slackware可以从网上免费下载，你也可以发送email邮件问一些支持问题，但我们会优先考虑那些购买官方CD的人。为什么这么说呢？因为我们收到了一大堆的邮件，但我们的时间是有限的，所以在发送email之前，请先阅读第二章的帮助。

Slakcware官网为：

http://www.slackware.com

Slackware的FTP主站为：

ftp://ftp.slackware.com/pub/slackware

请记住，虽然我们的FTP站点是为公众开放的，但带宽有限。在下载时请优先考虑离你近一点的镜像站下载。你可以在我们的网址上找到一个镜像站列表：

http://www.slackware.com/getslack

3.2　系统要求

一个简单安装的Slackware的最小要求[2]如下：

硬件各类	具体要求
处理器	586架构以上
内存	32MB以上
硬盘空间	1GB以上
多媒体驱动器	4倍速的CD-ROM以上

表 3.2: 系统要求

如果你有可启动的CD盘，那么就不需要软盘驱动器了。当然，这表明如果你没有光盘驱动器，那么你就得有一个软盘驱动器来进行网络安装。如果采用NFS方式安装的话，还需要一张网卡。请参见NFS一节获取更多信息。另外，如果你并不是在全新的系统上安装Slackware，还可以采用从硬盘安装。

声称只要1G的硬盘有点狡猾了。因为对于最小安装而言1G空间是可以的，但如果你是采用完全安装，那么至少要有2GB的硬盘空间[3]，另外还得准备出存放私人文件的硬盘空间。多数用户并不采用完全安装，事实上，许多人在只有100MB硬盘空间的机器上运行Slackware。

Slackware可以安装到内存少，硬盘小或CPU落后的系统上，但如果这么做，就要付出点代价。如果你正面临这样的情况，那么可以查看CD上的`LOWMEM.TXT`文件，它包含了一些有用的信息。

3.2.1　软件包系列

由于要求简洁的原因，历史上Slackware被分为软件系列。那时候，人们要想连接到FTP服务器上，只能通过奇慢无比的波特率300的调制解调器，所以Slackware被拆分成不同的集合，

[2]这是Slackbook 2的内容，对于Slackware 13.37不知是否还适用。

[3]Slackware 13.37的完全安装需要5.6GB左右

而这些集合的大小适合存放于软盘上，所以用户只需要下载和安装他们感兴趣的软件集合。今天，Slackware中使用软件包系列的目的，主要是用来对软件包进行分类。用软盘安装的日子已不复存在。下面是对软件包系列的一个简要描述。

系列	内容
A	基本系统，其中包含的软件足以让我们启动并运行一个系统，且有一个文本编辑器及基本的通信软件
AP	一些不需要X Window系统的应用软件。
D	软件开发工具。包括编译器、调试器、解释器及man手册等。
E	GNU Emacs
F	FAQ、HOWTO及其它一些文档。
K	内核源码。
KDE	KDE桌面环境。一个外观类似MacOS及Windows的X桌面环境。其中还包含作为KDE依赖的Qt库。
KDEI	KDE桌面的国际化语言包
L	库文件。其它程序要用到的动态链接库。
T	teTex文档系统。
TCL	工具命令语言。包括Tk、TclX及TkDesk等。
X	基本的X Window系统
XAP	主要桌面环境中不包含的一些X应用程序（如Ghostscript及firefox 等）。
Y	BSD控制台游戏。

表 3.3: 软件包系列

3.2.2　安装方法

本节中介绍有一些方法相当古老，笔者只尝试过CD/DVD安装及硬盘安装，其它都没试过，只能照书翻译。

软盘

早先的时候是可以通过软盘的方法安装Slackware的，但随着软件包的增大（是的，一些单独的软件在增大），软盘的方法已经不得不被淘汰了。最后一个能用软盘安装的版本是Slackware 7.1的部分安装。A系列和N系列几乎可以完全安装，这就提供了安装该发行版其它软件的一个基础系统。如果你考虑使用软盘安装（尤其在老的硬件上），那么我们推荐用其它的方法安装或者用Slackware老的版本。Slackware 4.0和7.0在这种情况下还是很可靠的。

注意，如果你想采用CD安装，但却没有可启动的CD，那么还是需要软盘的，同样，对NFS安装也如此。

现在多数的电脑连软驱都没有了，更别说软盘了，如果你是一个新手，完全不用考虑这种方法，当然，老手也几乎不需要考虑。

光驱

可启动的CD可以在Slackware Linux公司的官网上获得（参见获取Slackware这一节）。采用基于CD/DVD的安装全更容易一些。如果你没有可启动CD的话，那就需要从软驱启动了。另外，如果你的硬件使用启动CD上的内核有问题的话，你也可能需要使用特制的软盘了。

对于Slackware 8.1版本，我们使用了一种新的方法制作启动CD，这种方法对于一些特定的BIOS芯片工作得不太好（相比于当前多数Linux CD遇到的问题而言，这个问题简直不值一提）。如果你遇到这种情况，我们还是建议你使用软盘启动。

第3.2.2节和第3.2.2节提供了关于选择及创建启动软盘的信息，也许会很有用。

NFS

NFS（the Network File System 网络文件系统）是一个远程机器上使用的文件系统。NFS安装使我们可以从网络上的另一台机器上安装Slackware。作为获取源的那台机器要为安装机导出Slackware发行版的目录树。当然，这需要关于NFS的一定知识，在第5.7节中有介绍。NFS可以通过PLIP（parallel port并行端口）、SLIP及PPP（一个调制解调器连接）进行安装。然而，我们建议最好通过网卡进行连接。毕竟通过打印机端口来安装一个操作系统是非常非常慢的。

启动盘

软盘已经几乎被淘汰，这里按slackbook 2进行翻译，以防某些万一需要的情况。Slackware 13.37的光盘已经明确写明，不再支持使用其中的内核制作启动盘了。

所谓启动盘指的是一张软盘，通过它我们真正启动并开始安装。它包括一个压缩的内核镜像，用来在安装过程中控制硬件。因此，这是必须要有的（除非你是从CD启动，就像在光驱这一节中讨论的一样）。启动盘镜像位于`bootdisks/`目录下。

Slackware中，可以选择不止一个启动盘（具体而言是16个。）。在文件`bootdisks/README.TXT`中有所有启动盘的完整列表及对应的描述。然而，多数用户可以使用`bare.i`（IDE设备使用）或`scsi.s`（SCSI设备用）的启动盘镜像。

参见第3.2.2节以获取关于从镜像制作启动盘的介绍。

启动之后，会有提示要求插入根磁盘。我们建议你按照启动盘的提示进行。

根磁盘

这节的内容也是被淘汰的。

根磁盘包含了setup程序及安装过程中使用的一个文件系统。这个也是必须的。根磁盘镜像位于`rootdisks/`文件夹下。我们需要从`install.1`及`install.2`两文件制作两个根硬盘。在这个目录下，你还可以找到`network.dsk`、`pcmcia.dsk`、`rescue.dsk`及`sbootmgr.dis`的磁盘镜像。

追加盘

如果你执行的是NFS安装或安装系统中带有PCMCIA设备，那么你需要一个追加盘。追

加盘的镜像也是在rootdsks/文件夹下，文件名为network.dsk及pcmcia.dsk。近来也追加了rescue.dsk及sbootmgr.dsk盘。急救盘是一个安装在软盘上的，可以在4MB内存上运行的一个根磁盘镜像。它包含了一些基本的网络工具及vi编辑器，我们可以用它在一个破坏了的系统上进行快速的修复。sbootmgr.dsk是用来启动其它设备的。

根磁盘在加载之后会指示你使用其它的追加盘的。

制作启动盘

选择好了启动盘镜像，就需要将它放到软盘中，根据使用的操作系统的不同，这个过程也有一些小差别。如果运行的是Linux（或其它类Unix的操作系统），你需要使用dd命令，假设使用的是bare.i镜像文件，软驱对应的设备文件是/dev/fd0，那么制作**bare.i**软盘的命令为

```
% dd if=bare.i of=/dev/fd0
```

如果运行的是Microsoft的操作系统，你需要使用RAWRITE.EXE程序，在我们的发行版中也包含了该程序，它与软盘镜像在同一个文件夹下。同样的，假设用的是bare.i镜像，软驱为A:，在DOS提示符下，输入下面的命令即可。

```
C:\ rawrite a: bare.i
```

3.3　启动安装器

启动安装器很简单，只要将Slackware安装盘插入你的CD或DVD驱动器，之后重启就行了。另外，你可能要先进入BIOS修改启动顺序，把光驱的启动顺序放在硬盘的启动顺序之前。有一些计算机可以在系统启动时，按下某些键来动态的改变启动顺序。由于每台电脑都是不同的，我们不可能为所有的电脑提供一个修改启动顺序的说明，但修改的方法在几乎所有的机器上都是很容易的。

一旦你的电脑从CD启动了，你就会被带到一个界面，要求你输入特定的一些内核参数，从这里开始，你就可以像使用急救盘那样使用这个安装器了。一些系统可能要求特定的内核参数才能启动。但这是极少数的情况，多数情况下，只要按下回车键，启动内核就可以了。

```
Welcome to Slackware version 13.37 (Linux kernel 2.6.37.6)!

If you need to pass extra parameters to the kernel, enter them at the prompt
below after the name of the kernel to boot (huge.s etc).

In a pinch, you can boot your system from here with a command like:

boot: huge.s root=/dev/sda1 rdinit= ro

In the example above, /dev/sda1 is the / Linux partition.
```

```
This prompt is just for entering extra parameters.  If you don't need to enter
any parameters, hit ENTER to boot the default kernel "huge.s" or press [F2]
for a listing of more kernel choices.
```

启动过程中，你应该会看到屏幕上刷了一排排了文字。不必惊慌，这是很平常的。这是内核启动过程中，对硬件进行检测并准备载入操作系统（现在载入的是安装器）时产生的信息。如果你感兴趣的话，之后可以用dmesg(1)来阅读这些信息。通常，如果你的硬件有问题，这些信息对于排除问题是十分重要的。一旦内核完成对硬件的检测，屏幕上的信息就会停下，让你选择对非标准美式键盘的支持。

```
<OPTION TO LOAD SUPPORT FOR NON-US KEYBOARD>

If you are not using a US keyboard, you may not load a different
keyboard map.  To select a different keyboard map, please enter 1
now.  To continue using the US map, just hit enter.

Enter 1 to select a keyboard map: _
```

输入**1**并输入**Enter（回车）**，就能看到一张键盘映射表。选择符合你的键盘类型的那一项就可以继续了。

```
Welcome to the Slackware Linux installation disk! (version 13.37)

######  IMPORTANT!  READ THE INFORMATION BELOW CAREFULLY.  ######

- You will need one or more partitions of type 'Linux' prepared.  It is also
  recommended that you create a swap partition (type 'Linux swap') prior
  to installation.  For more information, run 'setup' and read the help file.

- If you're having problems that you think might be related to low memory, you
  can try activating a swap partition before you run setup.  After making a
  swap partition (type 82) with cfdisk or fdisk, activate it like this:
    mkswap /dev/<partition> ; swapon /dev/<partition>

- Once you have prepared the disk partitions for Linux, type 'setup' to begin
  the installation process.

- If you do not have a color monitor, type:  TERM=vt100
  before you start 'setup'.
```

```
You may now login as 'root'.

slackware login: root
```

其它的一些发行版会直接启动进入一个专门制作的安装程序，与之不同，Slackware 的安装器会让你进入一个载入系统内存的轻量Linux发行版。之后，我们就可以使用这个轻量的Linux手工运行安装程序，当然，我们也能将它作为一个应急启动系统，在系统不能启动时对系统进行修复。现在，你已经用root登陆了（安装器中不需要密码），这时就应该设置硬盘了。在这个节点上，如果你希望得到一个加密的根分区，你可以安装支持RAID或LVM的软件，但这个主题已经超出了本书的范畴。如果你希望使用这些先进的工具，我强烈建议你查看CD上的README_RAID.TXT，README_LVM.TXT及README_CRYPT.TXT文件。大多数用户并没有这种需求，因此应该直接跳到下一步——分区。

3.4　分区

与其它发行版不同的是，Slackware的安装器中没有使用图形化的磁盘分区工具；而取而代之的是fdisk(8)及cfdisk(8)，二者都是命令行工具。cfdisk基于curses库[4]，fdisk则不然。不管你用哪一个区别都不是很大，本书中，我们只讨论fdisk 。

要对你的硬盘正确分区，首先要了解如何识别它们。在Linux中，所有的硬件都是用一个特殊的称为设备文件的文件来识别。这些文件都（不失一般性地）放在/dev文件夹下。硬盘，不论是古老的IDE（PATA）盘，还是串行的ATA（SATA）盘，内核都把它当作SCSI设备。因此，会为它们创建一个诸如/dev/sda的设备节点。如果你不知道你的硬盘的设备节点名是什么，fdisk可以帮你。

```
root@slackware:/# fdisk -l

Disk /dev/sda: 72.7 GB, 72725037056 bytes
255 heads, 63 sectors/track, 8841 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
```

这里可以看到，我的系统有个72.7G的硬盘，位置是/dev/sda。你也可以看到一些额外的信息（这个例子中，实际上有3个SCSI硬盘，由一个RAID控制器控制，所以看起来只有一个硬盘）。fdisk的参数[-l]用来告诉fdisk，同时显示硬盘本身及在硬盘上检测到的所有分区，但这并不会对硬盘做任何改变。要对我们的硬盘进行分区，我们需要告诉fdisk具体如何操作。

```
 root@slackware:/# fdisk /dev/sda

The number of cylinders for this disk is set to 8841.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
```

[4]为程序提供字符终端用户接口。通俗地说：为字符终端（命令行）提供更好的显示，可理解为终端下的GTK等。译者注。

```
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)

Command (m for help):
```

现在，我们告诉fdisk应该对哪个盘分区，fdisk在显示一串讨厌的警告信息后，进入了命令模式。1024个柱面的限制早已不成问题，Slackware引导装载程序完全可以从拥有更大柱面的磁盘中启动。键入[m]然后输入回车，就可以得到更多的信息来告诉我怎么做。

```
Command (m for help): m
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   o   create a new empty DOS partition table
   p   print the partition table
   q   quit without saving changes
   s   create a new empty Sun disklabel
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)
```

现在你就知道了什么样的命令会有什么样的行为，是时候开始为我们的磁盘分区了。最低的要求是必须要有一个单独的/分区，并且，你也应该创建一个交换分区。你也可能想单独分一个/home分区来存储用户文件（把所有的用户文件放在单独的一个分区中，会使之后对系统升级，或是全新安装一个不同的Linux操作系统更为容易）。因此，我们继续分三个区。创建新分区的命令是[n]（你可能已经从帮助中看到了）。

```
Command: (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4):1
```

```
First cylinder (1-8841, default 1): 1
Last cylinder or +size or +sizeM or +sizeK (1-8841, default 8841): +8G

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (975-8841, default 975): 975
Last cylinder or +size or +sizeM or +sizeK (975-8841, default 8841): +1G
```

这里，我们创建了两个分区。第一个的大小为8G，第二个只有1G。我们可以用[p]命令来查看当前分区情况。

```
 Command (m for help): p

Disk /dev/sda: 72.7 GB, 72725037056 bytes
255 heads, 63 sectors/track, 8841 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot       Start      End          Blocks   Id  System
/dev/sda1                1      974         7823623+  83  Linux
/dev/sda2              975     1097          987997+  83  Linux
```

这两个分区的类型都是“83”，也就是标准Linux文件系统。我们需要将/dev/sda2的类型设置为“82”，才能将其转变为交换分区。我们使用fdisk的[t]命令来完成这个工作。

```
Command (m for help): t
Partition number (1-4): 2
Hex code (type L to list codes): 82

Command (me for help): p

Disk /dev/sda: 72.7 GB, 72725037056 bytes
255 heads, 63 sectors/track, 8841 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot       Start      End          Blocks   Id  System
/dev/sda1                1      974         7823623+  83  Linux
/dev/sda2              975     1097          987997+  82  Linux swap
```

所谓的交换分区是一个特殊的分区，Linux内核用它来作为虚拟内存。如果你一不小心内存不够用了，内核会将内存中的一些东西移动到交换分区中，用以防止崩溃。交换分区的大小完全取决于你自己。对于交换分区的大小，大家都有争执，但一个黄金定律就是将交换分区的大小设为系统内存大小的两倍。由于我们机子内存大小为512MB，所以我决定将交换分区设置为1GB。你可能希望测试自己的交换分区大小，看它是不是能最好地工作，但一般情况下，有很大的交换分区并不会有什么坏处。有一种说法，如果你有＊很多＊的内存（也就是大于2GB），就没必要遵守所谓的黄金定律了。如果你想要使用睡眼功能（挂起到硬盘），要求的交换分区大小至少要和物理内存（RAM）的大小一样，要记住。

在这个节骨眼上不能就这么停下，保存好设置然后继续后续的操作。但我还想创建第三个分区并挂载到/home目录下。

```
Command: (me for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 3
First cylinder (1098-8841, default 1098): 1098
Last cylinder or +size or +sizeM or +sizeK (1098-8841, default 8841): 8841
```

最后，保存设置。

```
Command: (me for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
root@slackware:/#
```

现在，我们已经对磁盘完成了分区，执行安装程序的前期工作已经完成。然而，如果你分了其它的区，可以先重启一下，来保证内核能正确读取这些分区。

3.5　setup程序

现在我们分好了区，是时候运行setup程序，安装Slackware了。setup程序包括对分区格式化、安装包以及一步步地执行基本设置。只要在shell提示符下键入setup就可以了。参见图3.1。

3.5.1　帮助

如果之前你没有安装过Slackware，你可以阅读Slackware的帮助菜单，它对Slackware安装

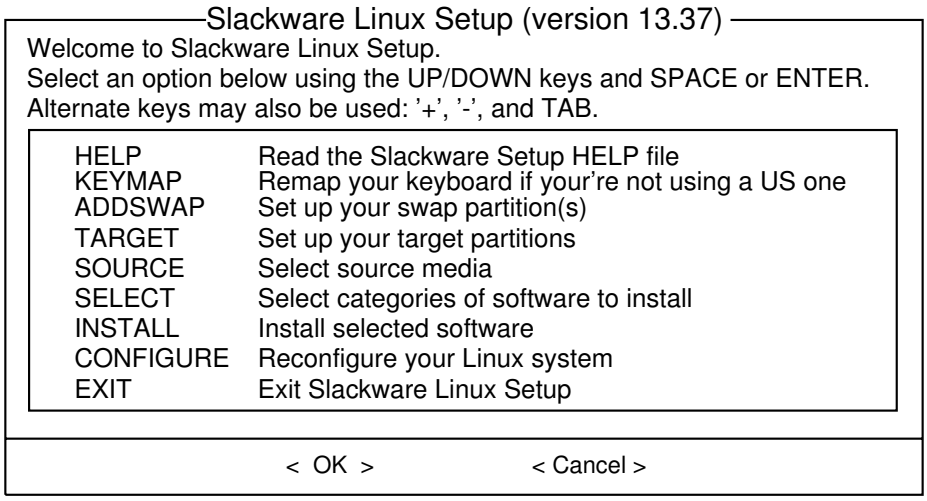

图 3.1: setup程序

器有一个基本的概述。其中的多数信息是关于对安装器的一些基本操作，这些操作也很直观。参见图3.2。

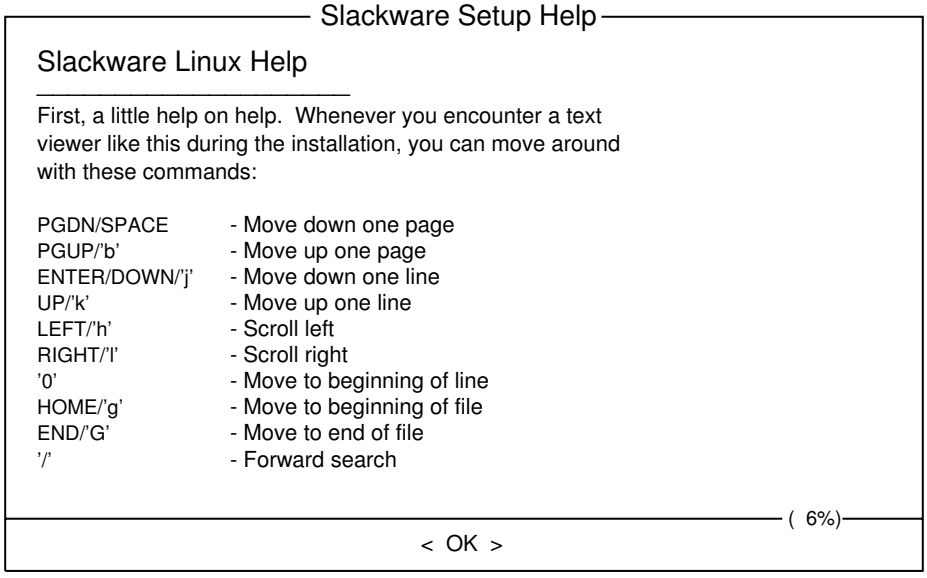

图 3.2: Slackware setup程序帮助

3.5.2　键映射

在继续之前，Slackware给了我们为键盘选择一个不同映射的机会。如果你用的是标准美式键盘，可以直接跳到下一步，但如果你用的是国际键盘，你应该在现在选择一个正确的键映射。这个步骤保证了我们从键盘输入的键与系统的理解是一致的。参见图3.3。

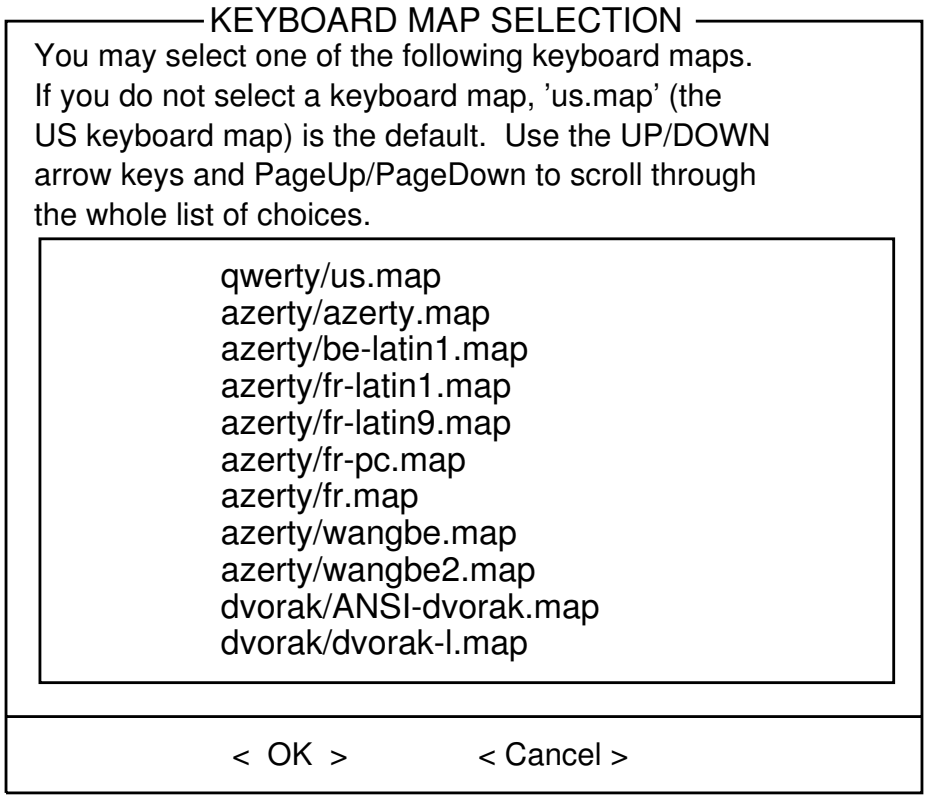

图 3.3: 键盘映射选择

3.5.3　添加交换分区

如果你创建了一个交换分区，那么这一步能让在你运行其它对内存敏感的活动，诸如安装软件包之前，启用这个交换分区。交换分区是一个磁盘分区（或者是一个文件，尽管Slackware的安装器不支持交换文件），当计算机的可用内存不足时，会将活动的系统内存拷贝到这个分区中。通过这个方法让计算机能够在活动内存中切换进切换出，从而能使用比计算机实际拥有的更多的内存。本步骤会将你的交换分区加入`/etc/fstab`中，使之在你的操作系统中生效。参见图3.4。

3.5.4　选择安装位置

下一个步骤是选择根分区的位置，以及Slackware要用到的其它分区。安装程序会提示我们选择是否使用某个分区及是否对要使用的分区格式化。如果你要安装到一个新的分区，那么就必须先格式化。如果你要装到的分区中有你不想删除的数据，那么不要进行格式化。例如，如果用户有一个`/home`分区，用以存放用户数据，那么切记在安装时不要对其进行格式化。那么新安装的Slackware就不需要对这些数据进行备份和还原了。根据使用的内核，我们可以选择不同的文件系统，包括reiserfs、ext2、ext3、ext4、jfs及xfs等，一般而言采用ext3或ext4即可。参见图3.5。

进入选择安装位置后第一次选择的是安装根（/）文件系统的位置。之后，你可以根据选择映射其它的分区到该文件系统中。（例如，你可能希望另一个分区，如`/dev/sda3`，作为你

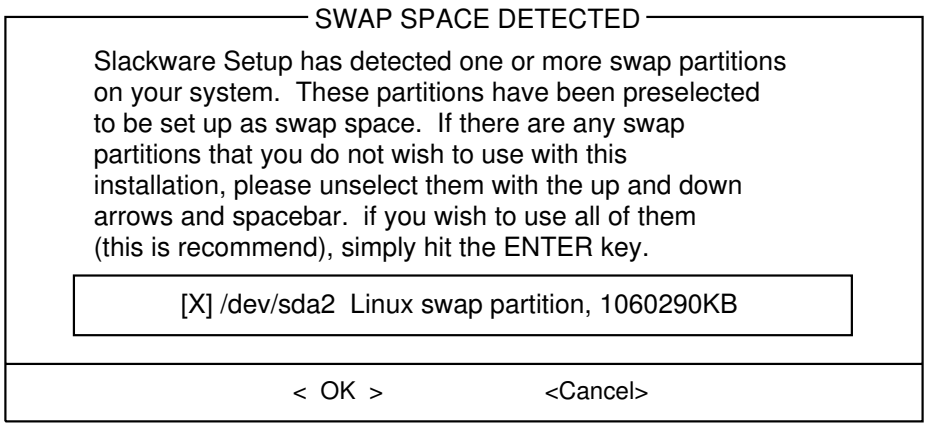

图 3.4: Slackware安装程序之添加交换分区

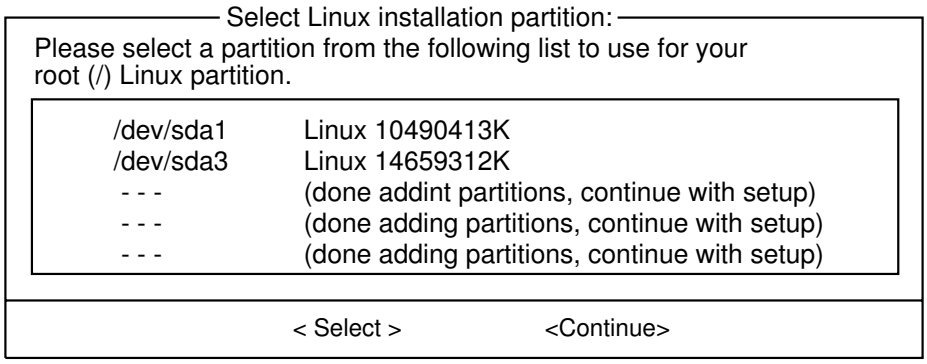

图 3.5: Slackware安装程序之选择安装位置

的Home文件系统。这只是一个例子，你可以按自己的想法进行映射。）

3.5.5　安装源

现在，你要告诉Slackware安装器，在哪可以找到Slackware的软件包。最常见的方法是用Slackware安装CD或DVD，但还可以用其它的一些方式。如果你之前已经把安装包放在之前你设置好的分区中，你可以选择从那个分区或是一个事先挂载的文件夹中安装。（你需要先使用mount(8) 命令挂载那个分区。参见第九章以获得更多信息。）另外，Slackware还提供了许多使基于网络的安装方法，如NFS共享、FTP、HTTP及Samba等方法。如果你选择网络安装，Slackware会先显示提示符，让你输入TCP/IP信息。这里，我们只讨论从DVD安装的情况，但其它的方法也很直接、简单。参见图3.6。

3.5.6　选择安装包

安装器会让你选择安装哪些集合。这些集合在第3.2.1节介绍过了。这个方法能让你方便地跳过那些你可能不想安装的软件包，如在服务器上你可能不想安装X或KDE，或者你压根就不想安装Emacs。注意“A”集合总是必须的。参见图3.7。

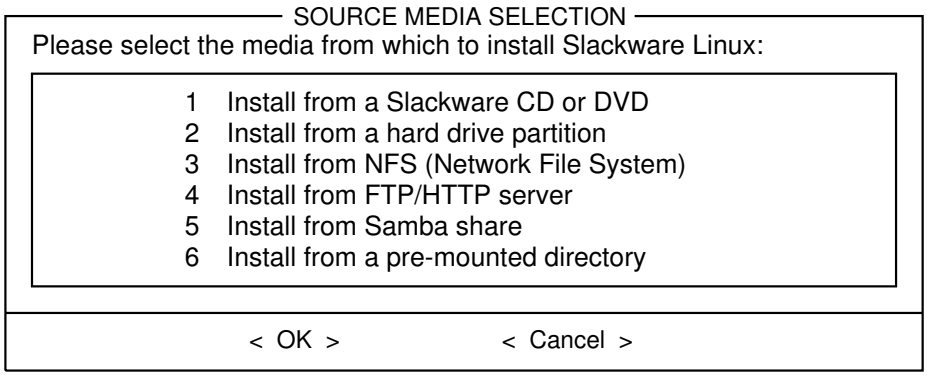

图 3.6: Slackware安装程序之选择安装源

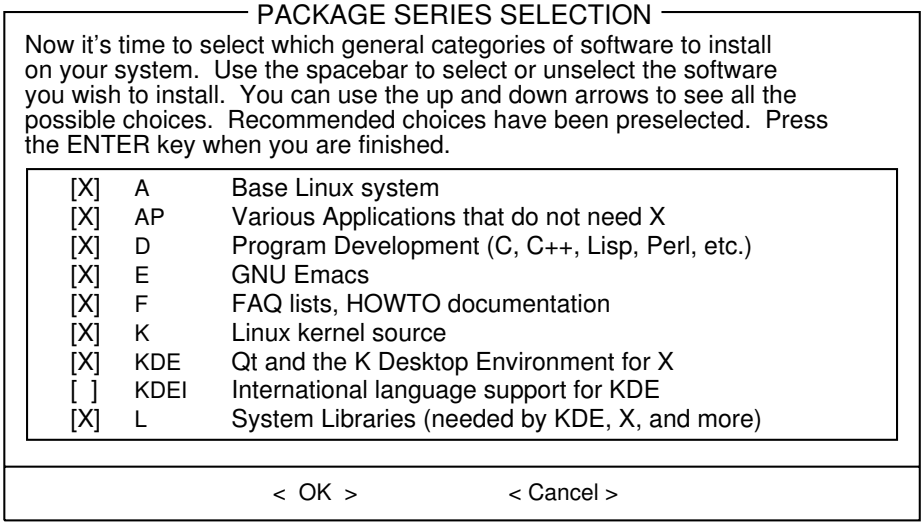

图 3.7: Slackware安装程序之选择软件包

3.5.7　安装

假设你已经完成了“target”，“source”及“select”等选项，接下来，“install”选项让我们在软件系列中选择具体要安装的软件包。如果前面的步骤未完成，它会提示你先完成之前的步骤。该选项可以让你从六个不同安装方法中进行选择:`full`——完全安装、`newbie`——新手安装、`menu`——菜单安装、`custom`——自定义安装及`tag path` 利用`TAG`安装。

`full`选项会安装选择的所有系列中的所有软件包。而不会提示其它信息。这是最容易的安装方法，因为我们不想思考到底要安装哪个软件包。当然，这个方法用到的磁盘空间也最多。

下一个选项是`newbie`。这个选项会安装选择的系列中所有必需的软件包。对于其它的软件包，它会给出提示符，让你选择“YES”、“NO”还是“SKIP”。YES表示安装，NO表示不安装，SKIP则跳到下一个系列。另外，你会看到软件包的一个描述及大小来帮助你决定是否要安装它。对于新用户来说，我们强烈推荐使用这个选项，因为它能保证我们安装了所有必需的软件包。然而，由于为每个软件包都显示提示符，所以速度慢。

`menu`选择是一个相对于`newbie`选项更快更高级的选项。对于每个系列，都显示一个菜单，

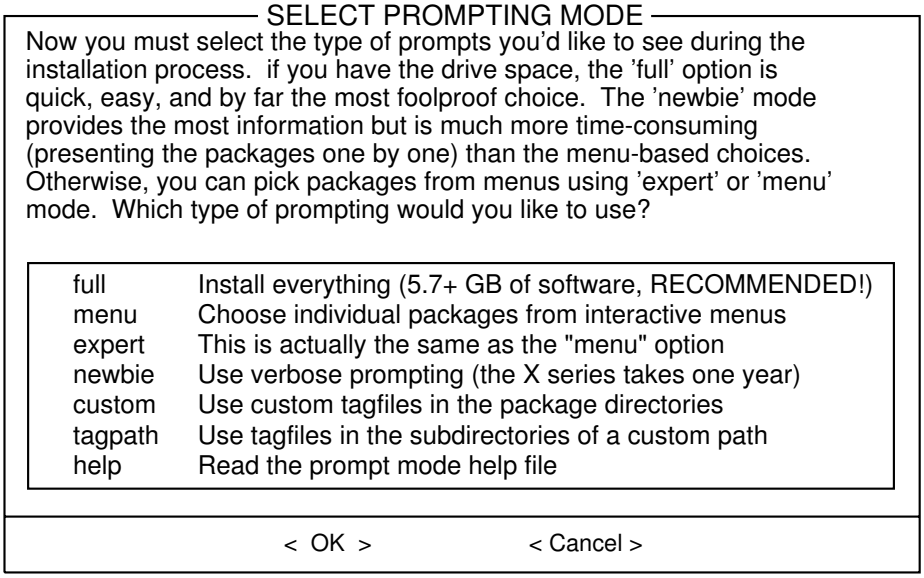

图 3.8: 选择提示方式

我们可以选择是否安装其它不是必需的软件包。必需的软件包不会显示在这个菜单上。

对于更高级的用户，install提供了`expert`选项，这个选项能让你完全控制想要安装的软件包。你可以不安装那些绝对需要的包，得到一个不能用的系统。相反的，你可以精确地控制系统中安装的东西。这个选项对于新手不建议使用，因为很可能会搬石头砸到自己的脚。

`custom`和`tag path`选项也是为高级用户而设的。这些选项让我们能基于预先制作的tag文件来安装系统。该功能对于批量安装Slackware很有帮助，关于使用tag文件的更多信息，请参见第//TODO:Section18.4//节。

选择了安装方法后，根据不同的方法，会有不同的响应。如果选择的是expert或menu，那么会出现一个菜单，让你选择要安装的软件包。如果选择的是full，会自动开始将软件包安装到目标位置中。如果选择的是newbie，在选择完可选的包后，才会开始安装软件包。

注意，安装时可能会出现磁盘空间不足，如果你选择了太多的包，而对应安装位置的剩余空间不足，就会出现问题。最安全的方法是先选择一些包，之后再添加其它的包。这可以通过使用slackware的包管理工具轻松完成。关于这些内容，参见第//TODO:Chapter18//章。

3.5.8　配置

配置这部分会对系统进行一个基本的配置。下面出现的画面很大部分依赖于我们安装的软件包。我们将以完全安装进行介绍。顺序可能与安装过程不太一致。

创建USB启动盘

很多年前，我们用软盘来创建启动盘，在其中存储启动数据。但现在软盘已经几乎被淘汰了，一些新的计算机甚至连软驱都没有，所以，Slackware采用USB设备来创建启动盘。基于安全的考虑，我们也建议你创建一个USB启动盘，防止之后Slackware不能启动。如果你的机

器支持从USB设备启动（现在的机器一般都支持），那么最好在这个步骤时就创建启动盘。切记，用来创建启动盘的U盘会被格式化，所以建议先对U盘中的数据进行备份。

另外，笔者尝试过创建USB启动盘，但4GB的U盘被格成了200MB左右，另外的空间反而不能使用了，虽然之后没有再尝试过，但请大家作好心理准备。

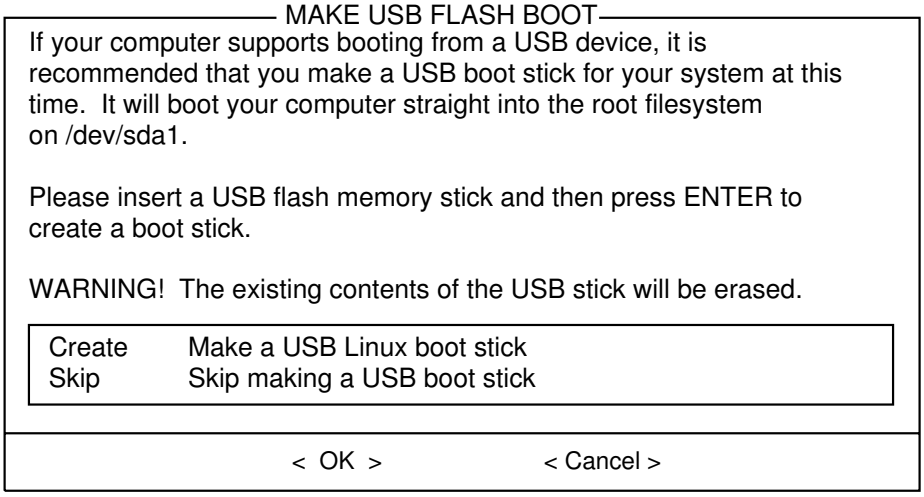

图 3.9: 创建USB启动盘

LILO

这里，会提示是否安装LILO（the LInux LOader；详情参见第//TODO:Section 7.1//节）如果系统中只安装了Slackware，那么选择`simple`就可以了。如果你安装的是双系统（或多系

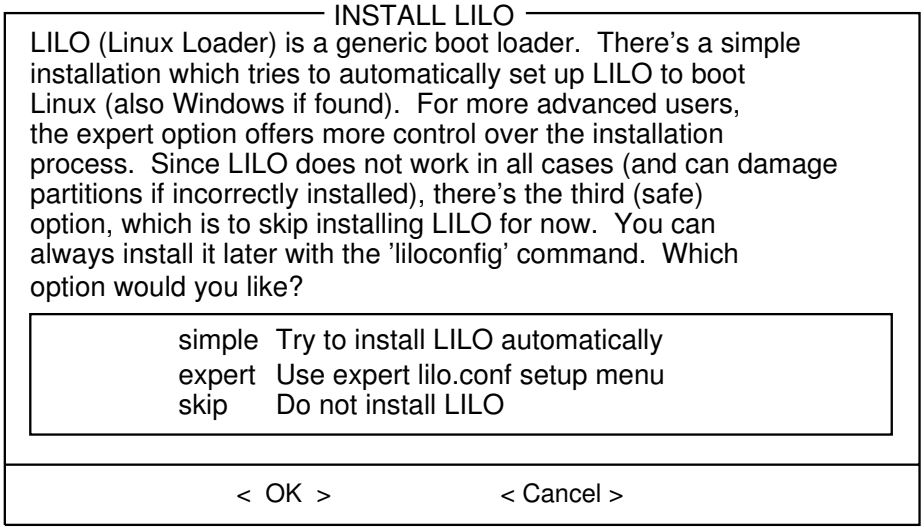

图 3.10: 安装LILO

统），那么使用`expert`选项会更好一些。关于双系统启动，请参见第//TODO:Section 7.3//节以获取更多信息。我们不推荐你使用第三个选项`do not install`，除非你知道自己在干什么，

知道自己为什么不需要安装LILO。如果选择的是expert选项，会有提示选择LILO的安装位置，通常可以选择将其安装在硬盘的MBR（主引导记录）中，Linux 根分区的superblock中，或是安装在软盘上。

是否使用UTF–8终端

从2.6.24内核起，就提供了一个标准的UTF–8终端，但由于常有一些问题，尽管你使用的是UTF8的locale，选择默认的非UTF8文本终端会更加安全一些。

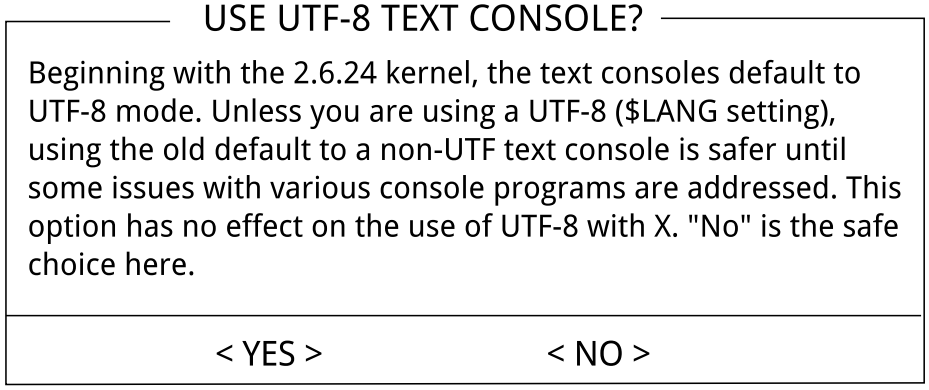

图 3.11: 是否使用UTF-8文本终端

鼠标设置

选择你的鼠标类型，一般选择ps/2类型或usb类型。之后会询问是否在启动时开启gpm(8)。选择了鼠标类型后，系统会创建链接/dev/mouse，并指向默认的鼠标设备。如果启动后鼠标无效，可以手动修改该文件。

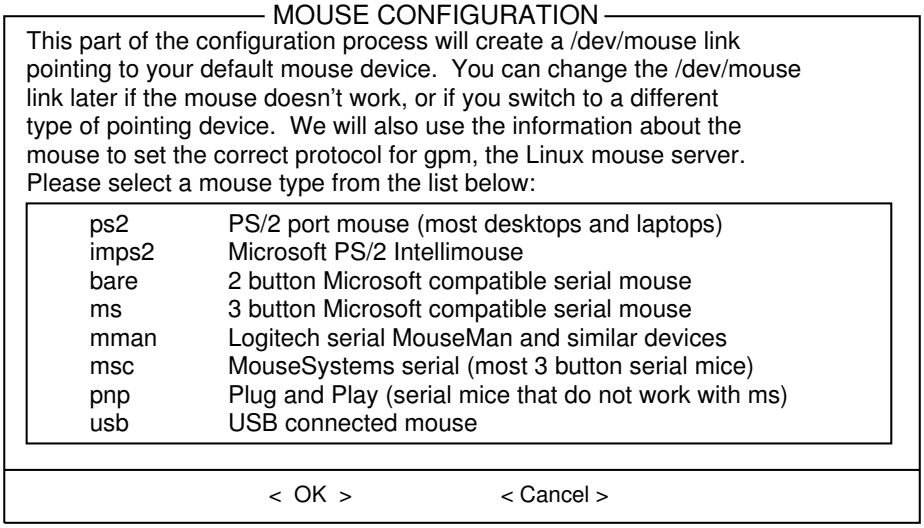

图 3.12: 鼠标设置

网络设置

网络设置实际上就是运行`netconfig`脚本。详情请参见第5.1节。

选择默认启动的服务

本节是选择启动时默认运行的服务。Slackware默认选择了几个服务，请根据自己是否需要该服务进行相应的选择。如果在服务器或桌面系统上安装Slackware，那么请关闭pcmcia服务。使用空格键进行选择及反选。请记住，开启越多的服务，系统的安全性就越低。

如果你是新手，那么不要考虑安全性问题，尽管选就是了。

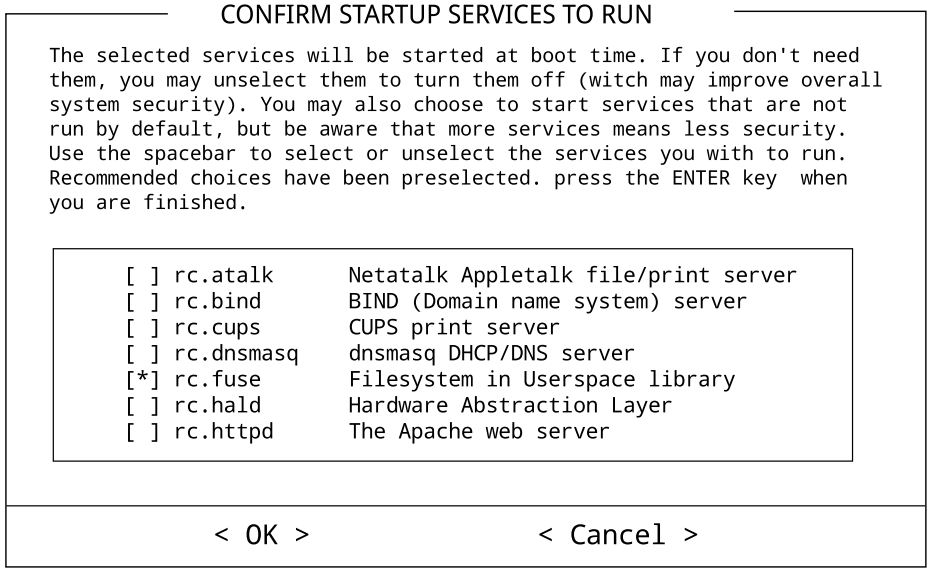

图 3.13: 选择默认启动的服务

终端字体选择

在该选项中可以选择终端中使用的字体。

个人建议选择“No”，因为一般而言用不到字符终端，另一方面，即使使用，使用默认字体就能很好地显示，其它一些字体还易出现乱码。

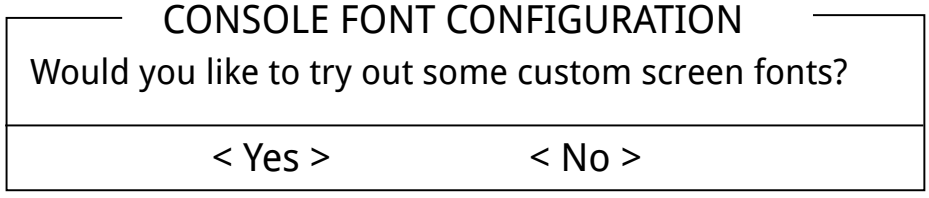

图 3.14: 文本终端字体选择

时区选择

这个步骤很直观，就是要你选择所在位置的时区。开始的选项是让我们选择是否使用UTC，一般而言，选择将时钟设置为本地时间（对应选项“No”）即可。之后选择所在的时区即可。

中国的同学们一般选择“Asia/Shanghai”，即上海时区即可。

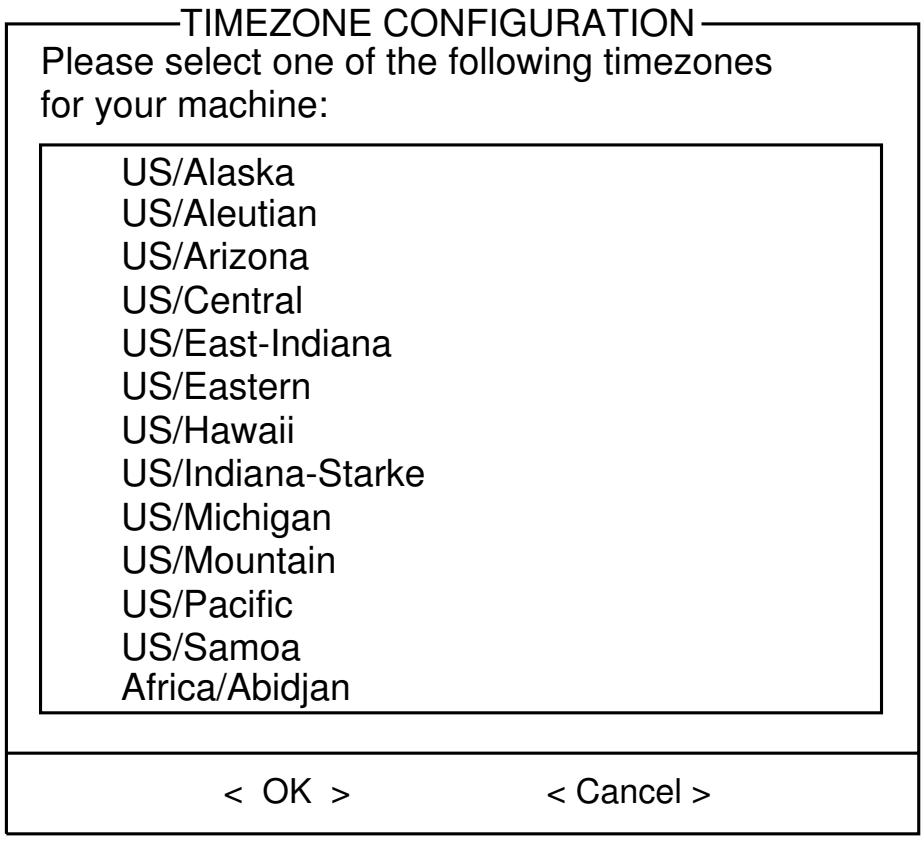

图 3.15: 时区选择

X 窗口管理器

通过本节，我们就可以选择默认的X窗口管理器，参见第//TODO:Chapter 6://章以获取关于X及窗口管理器的更多知识。

不管安装了什么软件包，最后一项配置是为`root`设置密码。出于安全考虑，设置`root`密码是不会错的，但是就如Slackware中的其它东西一样，设不设可以自己决定。

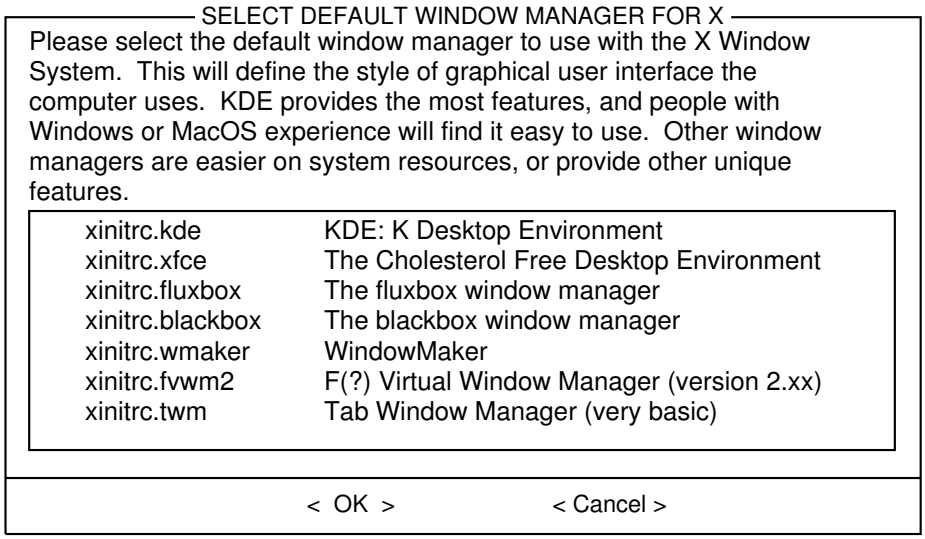

图 3.16: 选择默认的X窗口管理器

第四章　系统配置

在对系统进行进一步的配置之前，先学习关于系统的组织结构以及文件或程序的查找方法不失为一个不错的选择。我们还会介绍如何自己编译内核，如果你正打算这么做的话，本章应该很有帮助。本章的主要内容是让你了解系统的组织结构及相关的配置文件。之后我们就可以配置系统的更为高深的部分了。

4.1　系统一览

在对Linux分解认识之前，了解Linux系统的组成方式无疑是极其重要的。Linux 系统与DOS、Windows或是Macintosh（类Unix的Mac OS X例外）等系统有着显著的不同，我们会在下面的几节中帮助你了解Linux的布局，从而使你能够根据自己的需求来配置系统。

4.1.1　文件系统布局

Slackware Linux与DOS或者Windows的第一个显著差别就是文件系统。首先，我们并不使用驱动器的盘符来表示分区。在Linux下，有一个主目录，我们可以类比为DOS下的`C:`盘。并且系统中的每个分区都挂载到该主目录的某个子目录下。听起来有点像是一个可以无限扩展的硬盘。

我们将这个主目录称为`root`目录，并用一个斜杆（/）来表示。这个概念开始时可能不好理解，甚至有些诡异，但当我们需要增加更多的空间时，这个设计的优势就能很好地体现出来了。例如，我们假设有个硬盘，挂载到`/home`目录的某个子目录上，它的空间已经用完了。虽然大多数人在安装Slackware时都分配了一个较大的root分区。由于分区可以被挂载到任意目录下，所以我们可以随意到商店中买一块新的硬盘，并挂载到`/home`目录下，万事OK！现在，我们的系统中就添加了新的空间，整个过程异常简单。

下面，我们要对“/”目录下的主要顶级目录夹进行一个简要的介绍。

bin

该目录下存放一些必需的程序。它包含使用系统的最低要求下的一些程序。该文件夹下包括了如shell及一些文件系统的命令（`ls`、`cp` 等等）。在安装之后，一般就不再对`/bin`目录进行改动。即使进行改动，一般也是由官方的更新所做出的改动。

boot

该目录存在LILO所需的一些文件。Slackware的内核文件也存放于此。该目录在安装后可

能会做一些小的改动。

dev

在Linux中，一切皆文件，诸如串口、硬盘或扫描仪等设备文件也以文件的形式表示。所有的设备节点都存放于/dev目录下。在许多类Unix的系统中都采用了这种策略。

etc

该目录下存放系统的配置文件。从X的配置文件、用户数据库，到系统的启动脚本，都存放于此。随着使用Slackware时间的增长，系统管理员会对该目录越来越熟悉。

home

Linux是一个多用户系统，每个用户都有一个帐户和一个独立的文件夹来存放私人文件，这个文件夹就被称为用户的主目录，默认情况下，它就存放于/home目录下。

lib

该目录用于存放一些基本操作所依赖的库文件。包括C库、动态链接库、ncurse库及内核模块等等。

media

一些设备的默认挂载点。包括CDROM、DVD等的挂载点。

mnt

该目录的作用是作为硬盘或可移动设备的临时挂载目录。其中包括CDROM及软盘的挂载点。注意，该目录与/media的区别在于该目录一般用于临时挂载，而/media一般作为默认挂载点。

opt

该目录用于安装额外的一些软件包。其中的思想是：将软件包安装到/opt/software-package目录下，之后只需要将该目录删除即可卸载该软件包。一般而言，将一些占用空间较大的软件包安装到这里(如google-chrome、matlab、libreoffice及texlive等)。

proc

该目录是个特殊的目录。严格来说，它不属于文件系统的一部分，它是一个虚拟的文件系统，为我们提供内核的相关信息。内核需要让我们知道的信息会保存为文件的形式，并存放在/proc文件夹内。我们也可以通过修改这些文件来向内核传递信息。例如我们可以执行cat /proc/cupinfo来获取CPU相关的信息。

root

还记得系统管理员的用户名是什么吗？是的，root。与普通用户不同的是，root用户的主目录并不存放在/home/root中，而是放在/root下。这么做的原因很简单，试想，如果/home 与/在不同的分区下，那么当/home不能挂载时会发生什么情况？很自然地，我们会以root用户登陆并对系统进行修复，那么如果它的主目录就在损坏的文件系统上，那么甚至连登陆都成了问题。

sbin

在启动过程中，root用户需要用到的一些程序就存放在这个文件夹中。普通用户不会用到该目录下的程序。

tmp

该目录为临时存储的文件夹。所有用户都有对该目录的读写权限。

usr

该目录在Linux系统中是个相当大的目录。除了上面提到的东西外，其它的内容一般都放在该文件夹下，包括程序、文档、内核源码及X Window系统等。一般情况下，我们会将软件安装到该文件夹下。

var

该目录用于存放系统日志、缓冲的数据及程序锁文件等。该目录的内容最常改变。

现在你应该有些印象了，系统有什么文件夹，什么样的文件夹中放什么东西。如果你想更详细了解文件系统的布局，可以参见hier(7)的man 手册。下一节中，我们会讲解如何快速地查找文件，之后就不需要人工地查找了。

4.1.2 查找文件

现在我们知道了系统一些主要文件夹的功能，但这并不能真正帮我们找到一些特定的文件。是的，我们可以一个个文件夹地查找，但我们需要的是快速地查找。Slackware中有四个主要的文件搜索命令。

which

我们首先要介绍的就是which(1)命令。which一般用于快速定位某个程序的位置。它的作用是查找我们的环境变量PATH，并返回第一个搜索到的路径。例如：

```
% which bash
/bin/bash
```

从上例中我们可以看到，bash位于/bin目录中。对于搜索而言，which的功能有限，因为它只对PATH进行搜索。

whereis

接下来介绍命令whereis(1)。它与which类似，只是它除了搜索PATH外，还对man手册进行搜索，我们使用whereis对bash命令进行搜索时，它还返回如下结果：

```
% whereis bash
bash: /bin/bash /usr/bin/bash /usr/man/man1/bash.1.gz
```

whereis命令不仅告诉我们程序的实际位置，还告诉我们相应的文档的位置。但，一样的，该命令的功能有限。如果我们只想查找一个特定配置文件的位置呢？显然使用which或whereis命令都无法解决这个问题。

find

下面介绍find(1)命令，它是一个功能异常强大的命令。我们可以指定一定的搜索规则对文件系统进行搜索。例如可以使用文件名通配符、指定文件的创建或修改时间或都其它的一些规则。举个例子，如果我们想搜索系统中默认的xinitrc文件的位置，我们可以使用如下命令：

```
% find / -name xinitrc
/etc/X11/xinit/xinitrc
```

由于上述的find命令是对整个文件系统进行遍历搜索，所以要花上相当长的一段时间。并且，如果使用普通用户执行命令，在搜索那些只有root用户能看到的文件夹时，还会出现“权限不足”的错误，但最重要的是它的确找到了我们想要的文件。尽管如此，要是能再快一点就好了……

slocate

和find一样，slocate(1)也是对整个文件系统进行搜索，但它搜索的并不是真正的文件系统，而是一个事先生成的数据库。该数据库默认在每天凌晨时自动更新。但对于一些个人用户，凌晨时电脑一般处于关机状态，所以，我们可以手动执行updatedb(1)命令来更新数据库（执行该命令需要root权限，使用su命令切换到root用户或使用sudo即可获得root权限）。下面我们举个例子：

```
% slocate xinitrc # 这里并不需要root权限
/etc/X11/xinit/xinitrc
/etc/X11/xinit/xinitrc.fluxbox
/etc/X11/xinit/xinitrc.fvwm2
/etc/X11/xinit/xinitrc.twm
/etc/X11/xinit/xinitrc.xfce
/etc/X11/xinit/xinitrc.kde
...
```

得到的结果很长，但是运行速度很快。使用以上这些命令，我们就能够快速地找到我们想查找的文件了。

4.1.3 /etc/rc.d文件夹

/etc/rc.d文件夹用于存放系统的初始化文件。与System V采用init脚本的方法不同，Slacwkare为它的初始化文件采用了BSD风格的布局。init脚本的风格在不借助专门为此设计

的软件时，配置起来是很困难的。对于BSD风格的初始化脚本，每个运行级别都有一个单独的rc文件，而System V中，每个运行级别都有自己的一个目录，每个目录下又包含多个init脚本，采用这种设计维护进行比较方便。

初始化文件可以分为很多类别。有系统启动、运行级别、网络初始化及System V 兼容等。对于每个类别，我们都会将其它东西归为另一个类别。(As per tradition, we'll lump everything else into another category.)

系统启动

除了内核之外，Slackware运行的第一个程序是`init(8)`。该程序读取文件`/etc/inittab`中的配置告诉自己如何启动系统。在进行指定的运行级别之前，`init`执行`/etc/rc.d/rc.S`脚本，为系统作好准备。`rc.S`文件的作用是启用虚拟内存，挂载文件系统，清理一些特定的日志文件夹，初始化热插拔设备，载入内核模块，配置PCMCIA设备，设置串口，并执行System V的启动脚本（如果有的话）。很显然，`rc.S`做了很多工作，下面是一些`rc.S`调用的脚本，它们都位于`/etc/rc.d`目录下：

rc.S
: 系统真正的初始化脚本。

rc.modules
: 该脚本用于载入内核模块。如网卡、PPP设备及其它的一些模块。如果该脚本找到`rc.netdevice`文件，`rc.modules`也会执行它。

rc.pcmcia
: 查找并配置系统中的PCMCIA设备。这对于笔记本用户可能最为有用，因为它们的调制解调器或网卡可能就是PCMCIA的[1]。

rc.serial
: 执行适当的`setserial`命令对串口进行适当的配置。

rc.sysvinit
: 查找与运行级别相关的System V初始化脚本并运行。在下面会进行更详细的介绍。

运行级别初始化脚本

系统初始化结束后，`init`会继续作运行级别的初使化。所谓的运行级别是对系统将要运行的状态的描述。听起来很繁琐？好吧，运行级别就是告诉`init`，是支持多用户还是只支持单用户；要不要开启网络支持；是使用X Window还是使用`agetty(8)`来管理登陆。下面的文件定义了Slacwkare中的不同运行级别。

rc.0
: 关机（运行级别为0）。默认情况下，该文件是`rc.6`的一个软链接。

[1] 这里的描述与前面选择启动脚本的描述不符，笔者对PCMCIA没什么概念，弄懂了再改吧

rc.4

带有多用户支持启动（运行级别为4），并使用X11的KDM、GDM或XDM作为登陆管理器。

rc.6

重启系统（运行级别为6）。

rc.K

启动单用户模式（运行级别为1）。

rc.M

多用户模式（运行级别为2或3），使用传统的基于文本的登陆管理器，这也是Slacwkare默认的运行级别。

网络初始化

运行级别为2、3或4时都会开启网络支持。下面这些文件就是用于网络的初始化的：

rc.inet1

由`netconfig`命令生成，该文件用于配置实际使用的网络接口。

rc.inet2

在`rc.inet1`之后执行，并开启基本的网络服务。

rc.atalk

启动AppleTalk服务。

rc.http

开启Apache网络服务器。与其它脚本相同，只要向它传递stop、start或restart参数，就可以相应地停止、启动或重启Apache服务。

rc.news

启动新闻服务器。

System V兼容化

在Slackware 7.0时引入了System V初始化脚本的兼容措施。许多其它的Linux发行版都采用System V风格的初始化布局而不是BSD风格的。System V为每一个运行级别的初始化脚本准备了一个文件夹，而BSD风格则是为每个运行级别准备了一个初始化脚本文件。

脚本`rc.sysvinit`会搜索存放在`/etc/rc.d`文件夹下的System V初始化脚本文件，并执行相应运行级别的脚本。这个兼容性措施对一些安装System V初始化脚本的商业软件包而言是很重要的。

其它文件

下面介绍的脚本文件是其它的一些初始化脚本。它们一般会通过上面介绍的脚本的调用运行，所以我们只需要修改这些脚本的内容即可。

rc.gpm
启动文本终端下的鼠标服务。开启后就可以在文本终端下进行复制与粘贴。极少数情况下，gpm会引发在X Window下使用鼠标的一些问题。所以如果在使用X window时遇到一些鼠标问题，可以尝试去掉该脚本的执行权限并停止gpm服务器。

rc.font
为文本终端加载自定义字体。（在安装过程中可进行设置。）

rc.local
包含你系统独有的一些初始化命令。在安装后该文件是空的，因为它是为系统管理员准备的。所有的初始化步骤结束后，系统会执行该脚本。

要启用某个脚本，只需要使用`chmod`命令为其添加执行权限。相应的，要停用某个脚本，也只需要去除相应脚本的执行权限即可。关于`chmod`命令的更多信息，请参见第//TODO:Section 9.2//节。

4.2　选择内核

内核在一个操作系统中起着至关重要的作用，它提供了对硬件的访问、对进程的控制以及整个系统的控制。内核中包含了对硬件设备的支持，所以在安装的时候选择合适的内核是很重要的。

Slackware提供了不止一打的提前编译好的内核可供选择，每个内核都包含一个标准的驱动集合和一些特定的驱动。你可以选择直接使用提前编译的内核或从源码编译自己的内核。不管使用哪种方法，都要确保内核中包含系统所需的所有硬件支持。

4.2.1　Slackware CD-ROM上的/kernels文件夹

提前编译的内核可以在Slackware的CD或DVD上或FTP站点上的Slackware文件夹下的`/kernels`文件夹中找到。新的版本发布后，其中可用的内核也会发生改变，而该文件夹下的文档则是权威的。`/kernels`文件夹为每个内核准备了一个子文件夹，每个子文件夹的名字都与它们对应的启动盘名相同。在每个子文件夹中，你会找到以下文件：

文件名	作用
System.map	内核的系统映射文件
bzImage	内核本体
config	内核源码配置文件

表 4.1: kernel文件夹中的文件

要使用一个内核，将System.map及config文件拷贝到/bot目录中，并将内核镜像[2]复制到/boot文件夹下，重命名为vmlinuz，之后，运行/sbin/lilo(8)命令为新的内核安装LILO，重启即可。这就是安装一个新内核的步骤。

接下来的内容是过时的。以.i结尾的内核是为IDE设备准备的，这些内核不支持对SCSI。以.s结尾的内核为SCSI内核，支持IDE，同时支持SCSI。

4.2.2　从源码编译内核

新手们常问的一个问题是“我需要自己编译一个内核吗？”。答案是也也许吧。有一些情况下是需要为自己的系统编译内核的。多数用户只需使用编译好的内核，外加一些可加载的内核模块就可以构建一个可用的系统了。如果你想使用一个Slackware当前不提供的新的内核版本，或者你想对内核添加补丁以获取对一些当前内核不支持的设备的支持，那么你可能需要为自己的系统重新编译内核。如，一些有SMP功能的系统就会考虑编译一个带SMP支持的内核。许多用户还会自己编译内核来让机器跑得更快些。要是想针对某些特定的处理器做优化，编译内核也是有帮助的。

编译自己的内核其实并不难。首先是确保你的机器上安装了内核的源码。即安装是切记选择安装K系列下的软件包。当然，也要确保安装了D系列的软件包，我们需要的是D系列中的C编译器、GNU make、及GNU binutils。一般而言，如果做的工作与开发有关，那么把D系列的软件包全装上会是个不错的选择。你也可以从http://www.kernel.org/mirrors网站下载最新的内核源码。

4.2.3　2.4.x版本的内核编译

```
% su -
Password:
# cd /usr/src/linux
```

第一个步骤是让内核处于默认的状态。使用下面的命令（注意，该命令会删除.config文件，且没有任何提示，需要的话，可以先进行备份）：

```
# make mrproper
```

接下来，我们就可以根据自己的系统来配置内核了。当前的内核提供三种方式。第一种基于文本的问答系统，它会问用户一系列问题并根据用户的回答来创建配置文件。这个方法的一个问题就是如果你搞杂了，就得重新来过。多数人选择的是第二种方式——基于菜单的方式。第三种是基于X的配置工具。选择好你喜欢的方式，通过执行下面对应的命令即可：

```
# make config      （基于文本的问答系统）
# make menuconfig  （基于文本的菜单驱动系统）
# make xconfig     （基于X的版本，请先确保自己在使用X系统）
```

[2]kernel image，image不知道应该怎么翻才好。

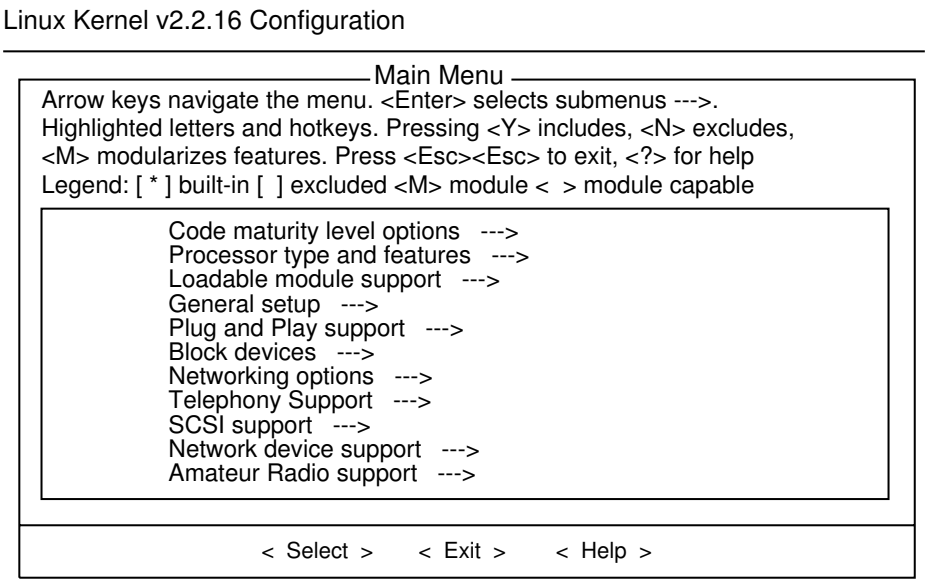

图 4.1: 内核配置菜单

对于新用户而言，menuconfig应该是最容易使用的方式。其中提供了帮助，解释了内核各部分的功能。配置完成后退出配置程序。它会创建一个必要的配置文件。之后我们就可以准备用来编译的源码树了：

```
# make dep
# make clean
```

下个步骤就是编译内核。首先尝试如下命令：

```
# make bzImage
```

根据你的CPU速度，这个命令会跑上一会。编译的过程中会显示一些编译器的信息。编译完成之后，我们还需要编译之前配置时标记为模块的部分。

```
# make modules
```

接下来就是安装刚编译好的内核了。要在Slackware系统上安装内核，可以使用如下的命令：

```
# mv /boot/vmlinuz /boot/vmlinuz.old
# cat arch/i386/boot/bzImage > /boot/vmlinuz
# mv /boot/System.map /boot/System.map.old
# cp System.map /boot/System.map
# make modules_install
```

之后是重新配置LILO加载新的内核。首先修改/etc/lilo.conf文件并为老的内核添加相应的启动项。完成后运行/sbin/lilo来安装新的启动块。最后重启并以新的内核启动即可。

4.2.4　2.6.x版本的内核编译

2.6版本的内核编译与2.4或2.2的内核只有小部分的区别，但在深究之前了解其中的差异还是很有必要的。编译2.6内核时，不再需要运行`make dep`及`make clean`了，并且在2.6系列内核中，编译时不再像之前的版本一亲显示很详细的信息。结果就是编译过程更容易理解了，当然，这样做也有一些短处。所以如果在编译过程中出了什么错误，我们强烈建议你选择显示详细信息。只要在编译时加上`V=1`选项即可。显示更为详细的信息有助于内核开发人员记录更多信息，在别的geek帮助你时也能更有效率。

```
# make bzImage V=1
```

4.2.5　使用内核模块

内核模块可以认为是设备驱动的一个别名，我们可以在内核运行时对其动态加载与卸载。通过内核模块，我们在不选择另一个内核或自己编译内核的情况下就能添加对新硬件的支持。

模块可以在任何时刻进行加载或卸载。这使得系统管理员在更新一些特定驱动时变得很容易。编译新的模块–¿ 卸载旧的模块–¿ 加载新的模块，连重启系统都不需要。

所有的模块都存储在`/lib/modules/kernel-version`目录下（kernel-version处根据你的系统而定）。它们可以通过`rc.modules` 文件在启动时就加载。这个文件的注释很很，并为主要的硬件提供了配置的实例。使用`lsmod(1)`可以显示当前活动的模块：

```
# lsmod
Module          Size    Used by
parport_pc      7220     0
parport         7844     0    [parport_pc]
```

就上面的例子而言，可以看到我只加载了并口的模块。要移除一个模块，使用`rmmod(1)`命令。加载模块可以使用`modprobe(1)`或`insmod(1)`命令。其中，`modprob`更为安全，因为它解决了模块间的依赖关系。

多数用户从来没有手工加载过模块。他们只使用内核自动加载器（the kernel autoloader）来管理模块。默认情况下，Slackware的内核中包括了`kmod`。`kmod`是一个内核选项，它使内核在需要时能自动加载所需的模块。想了解更多关于`kmod`及如何配置的内核，请参见`/usr/src/linux/Documentation/kmod.txt`[3]。再次，前提是你已经安装了内核源码软件包，或者可以从http://kernel.org网站上下载内核源码。

我们可以在上面涉及到的命令的相应man手册中找到更多信息，当然，还有`rc.modules`文件的相关内容。

[3] Slackware 13.37、内核2.6.37中已经找不到该文件。

第五章　网络配置

5.1　介绍：netconfig是我们的好朋友

在我们安装Slackware的时候，setup程序就调用了`netconfig`命令。`netconfig`为我们提供如下的功能：

- 它会询问我们电脑的主机名及域名。
- 它会给出可用的各种地址方案并作简要说明，告诉我们什么情况下应该使用什么方案，并询问根我们想使用什么方案来配置我们的网卡：
  - Static-IP 静态IP地址
  - DHCP 动态IP地址
  - Loopback 回环地址
- 之后会为我们检测网卡并提示我们如何配置。

如果我们使用`netconfig`来配置LAN网络连接，它会解决近80%的工作。我们强烈建议你在配置后再检查一下配置文件，原因如下：

1. 坚决不能信任一个安装程序，它不可能总是根据你的电脑做出合理的配置。如果你使用的是一个安装程序，那么切记自己检查配置文件。
2. 如果你还在学习Slackware或Linux管理，那么检查这些配置文件是很有帮助的，至少你会知道配置文件长什么样。这也会帮助你在日后出现错误配置时能及时解决问题。

5.2　网络硬件配置

如果你决定让你的Slackware连接到网络的话（这不是废话吗!），首先需要的是一张与Linux兼容的网卡。要小心确认网卡到底是不是与Linux兼容的（请查阅Linux文档计划[1]或内核文档，以获取当前准备使用的网卡支持情况）。一般而言，只要不是很老的内核，都会支持多得惊人的网卡。但就如上面所说，我们还是建议你在购买网卡之前，先查阅Linux硬件兼容清单（如GNU/Linux Beginners Group Hardware Compatiblity Links[2] 及Linux文档计划

[1] Linux Documentation Project

[2] http://www.eskimo.com/%7Elo/linux/hardwarelinks.html

的HOWTO文档[3]），这些文档在网上都能找到。如果因为网卡不兼容，而花上几天甚至几星期来解决这个问题，那购买之前花一点时间在搜索上就很值得了。

在你查阅相关文档时，最好同时记下支持某张网卡的对应模块。

5.2.1　加载网络模块

内核模块在是启动时由/etc/rc.d中的/etc/rc.modules加载的，也或者是通过/etc/rc.d/rc.hotplug自动加载的。默认的rc.modules文件中有一个小节是专门用于支持网络设备的。如果你手动查看该文件并阅读相关的代码，你会发现它首先检查/etc/rc.d/文件夹中的rc.netdevice文件是否存在并可执行。在setup程序安装系统时，如果自动检测到了你的网络设备，它就会自动创建该文件。

在那个“if”语句下面是一堆注释了的语句，注明了网络设备的类型和相应的modprobe语句。找到与我们的设备对应的modprobe行并取消注释，保存文件即可。之后以root权限运行rc.modules就可以加载我们的网络设备的驱动了（当然还有那些没有注释的其它模块）。注意，其中的一些模块（如ne2000驱动）是需要参数的，请确保选择了正确的行。

5.2.2　LAN(10/100/1000Base-T and Base-2)卡

这个标题涵盖了所有内置的PCI及ISA网卡。这些网卡是通过前一小节讲述的可加载的内核模块得到支持的。一般来说，/sbin/netconfig会自动检测你的网卡，并正确设置rc.netdevice文件。如果该步骤出错，最可能的情况就是为某张网卡加载的驱动是错误的（也不是没听说过，一些公司的同一个系列不同代的网卡需要不同的模块）。如果你确定网卡模块是对的，那么下个步骤最好是自己查看文档，看该模块在初始化时是否要加一些特定的参数。

笔者注：在笔者的Slackware 13.37上，找不到rc.netdekvice文件，并且rc.modules也相应的网卡加载行，但网络正常使用。查阅相关资料，发现现在将常用的一些驱动编译进了内核，因此会自动加载。

5.2.3　调制解调器

就像LAN卡一样，调制解调器也带有支持不同总线的选项。直到最近，多数的调制解调器还是8位或16们的ISA卡。由于Intel及motherboard公司不懈的努力，最终彻底消灭了ISA总线，现在我们能找到的调制解调器要么就是用串口或USB接口的外接调制解调器，要么就是内置的PCI调制解调器。如果你想为你的Linux配备调制解调器，那么事先考虑预算是很重要的。如果不算多数的话，那么可以说很多现在商店上的PCI调制解调器是WinModems。WinModems在自己的调制解调器卡上缺少一些基本的硬件：一些本应由这些硬件进行的运算就转移到了Windows上的调制解调器驱动及CPU上。这意味着在你尝试用它拨号到你的因特网服务提供商（internet Service Provider）时，PPPD会找不到标准的串口接口。

如果你想确保所购买的调制解调器一定能用在Linux上，那么就买一个外接的调制解调器，并通过串口接到你的PC上。这能保证它会更好地工作，并在安装及维护时碰到的麻烦会少一

[3] http://www.linux.org/docs/ldp/howto/hardware-HOWTO/

些。但它需要外接的电源，而且一般更贵。

有一些网站提供对基于WinModem设备的配置帮助。一些用户也说成功配置并安装了一些winmodem，包括Lucent、Conexant及Rockwell的芯片。由于支持这些设备的软件并不包含在Slackware中，并且不同的芯片要求不同的驱动，我们不对其进行详细描述。

5.2.4 PCMCIA

在Slackware安装过程中，我们可以选择安装pcmcia相关的软件包（在A系列中）。这些软件包提供了在Slackware下使用PCMCIA卡的一些软件及相关的设置文件。注意，pcmcia包只安装了一些使用PCMCIA卡的一般软件，并不安装任何驱动或模块。驱动和相应模块可以在`/lib/modules/‘uname-r’/pcmcia` 目录下找到。要找到适合我们网卡的PCMCIA模块可能要花上一些时间。

我们需要修改`/etc/pcmcia/network.opts`（对于以太网网卡）或`/etc/pcmcia/wireless.opts`（无线网卡）。和其它的Slackware配置文件一样，这两个文件的注释也很清晰，所以作任何的修改也很容易。

5.3 TCP/IP设置

到本节为止，我们的网上应该安装完毕，相应的内核模块也加载完成了。我们还是上不了网，但我们可以使用命令`ifconfig -a`来得到网络设备的信息。

```
darkstar:~# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 00:19:e3:45:90:44
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:122780 errors:0 dropped:0 overruns:0 frame:0
          TX packets:124347 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:60495452 (57.6 MiB)  TX bytes:17185220 (16.3 MiB)
          Interrupt:16

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:699 errors:0 dropped:0 overruns:0 frame:0
          TX packets:699 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:39518 (38.5 KiB)  TX bytes:39518 (38.5 KiB)

wlan0     Link encap:Ethernet  HWaddr 00:1c:b3:ba:ad:4c
          inet addr:192.168.1.198  Bcast:192.168.1.255  Mask:255.255.255.0
```

```
          inet6 addr: fe80::21c:b3ff:feba:ad4c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1630677 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1183224 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1627370207 (1.5 GiB)  TX bytes:163308463 (155.7 MiB)

wmaster0  Link encap:UNSPEC  HWaddr 00-1C-B3-BA-AD-4C-00-00-00-00-00-00-00-00-00-00
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

如果我们只输入/sbin/ifconfig，而不加-a参数，那么就看不到eth0这个接口，原因是该接口还没有一个有效IP地址或路由地址。

有许多不同的方法可以建立一个网络或加入一个子网络，但所有的方法可以大致归为两类：静态的和动态的。静态网络设置之后，每个网络节点（代表有IP地址的一些设备）都有固定的IP。动态网络设置之后，每个网络节点的IP地址都由一个叫作DHCP的服务器控制。

5.3.1 DHCP

DHCP（或称为动态主机设置协议(Dynamic Host Configuration Protocol)），是一种由一台计算机为其它计算机分配IP的方法。在DHCP客户端启动时，它会向本地局域网发送一个请求，查找是否存在DHCP服务器来为自己分配一个IP地址。DHCP服务器中存储了一个有效的IP地址池以及相应的过期时间。当一个已经分配了的IP地址到期之后，客户端必须重新联系服务器并重复协议过程。

该客户端在获取IP地址后会为对应的接口设置获取的IP地址。然而，DHCP客户端在与服务器协商IP时还会用个小“花招”，那就是它会记住之前使用的IP地址，并请求服务器为自己分配相同的IP地址。如果该IP可用，那么服务器会将该地址分配给客户端，若不可用，则会为其分配一个新的IP。所以协商过程示例如下：

客户端：在局域网中有可用的DHCP服务器吗？
服务器：是的，我就是。
客户端：我需要一个IP地址。
服务器：你可以使用192.168.10.10这个地址，时限为19200秒。
客户端：谢谢。

客户端：在局域网中有可用的DHCP服务器吗？
服务器：是的，我就是。
客户端：我需要一个IP，我们谈过的，之前我用的是192.168.10.10，我能再用这

个IP吗？
服务器：是的，可以。（或不行，但你可以使用192.168.10.12这个IP。）
客户端：谢谢。

Linux中的DHCP客户端程序是`/sbin/dhcpcd`。如果你用自己喜欢的文本编辑器打开`/etc/rc.d/rc.inet1` 文件，你会看到，该文中中间的部分调用了`/sbin/dhcpcd`命令。这会强制使用上面介绍的协商过程。`dhcpcd`还会记录当前IP剩余多少时间。并会在协议失效时自动重新发起连接，以获取新的IP地址。DHCP还能控制相关的信息，如使用哪个ntp服务器，使用哪个路由条目等。

在Slackware中设备DHCP是很简单的，只要运行`netconfig`命令并选择DHCP方式即可，如果你有不只一个NIC并且不想用DHCP对`eth0`进行设置，那么只要手工打开`/etc/rc.d/rc.inet1.conf` 文件，并将相应的代表NIC的变量改为“`YES`”即可。

5.3.2　静态IP

静态IP就是一个固定的地址，除非手工指定，这个地址不会改变。这种方法适用于管理员不希望IP信息变化的情况，如对于一个LAN（局域网）中的内部服务器，或所有连接到因特网的服务器，以及网络中的路由器。使用静态IP分配方法，只要为一台机器分配了IP，之后就可以不管了，其它的机器也会知道自己要一直使用这个IP，并且一直使用该IP地址与服务器联系。

5.3.3　手工配置

本节我们会介绍如何手工对网络进行配置。Slackware为我们提供了很多的工作，但首先我们要介绍的是功能强劲的`ifconfig(8)`命令，它的功能很多，不仅仅是为网卡设置设置IP地址，详细功能请参见它的man手册。

在第5.3节中，我们已经看到如何使用ifconfig查看当前的网卡。现在我们要使用ifconfig为网卡设置IP地址及子网掩码，当然，你可以自己指定参数。

```
darkstar:~# ifconfig eth0 192.168.1.1 netmask 255.255.255.0
darkstar:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:19:e3:45:90:44
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:122780 errors:0 dropped:0 overruns:0 frame:0
          TX packets:124347 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:60495452 (57.6 MiB)  TX bytes:17185220 (16.3 MiB)
          Interrupt:16
```

如果你仔细观察的话，会发现这张网卡的IP地址已经被设置成了192.168.1.1，子网掩码被设置成了255.255.255.0。现在我们已经做好了联网的最基本的设置，下面就是设置默认网关及DNS服务器了。要完成这些内容，我们就要介绍一些其它的工具。

我们的下一站是功能同样强大的**route(8)**。这个工具是用来修改Linux内核的路由表，而这会影响到网络的所有数据传输。路由表可以异常复杂，也可以相当简单直观。多数用户永远不需要手工设置默认网关，所以我们会演示如何操作。同样，如果你需要设置更多复杂的路由表，那么也请查阅它的man手册或查阅其它资料。现在，我们看看在刚设置好eth0后的路由表。

```
darkstar:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.0     *               255.255.255.0   U     0      0        0 eth0
loopback        *               255.0.0.0       U     0      0        0 lo
```

我们不会讲解每个细节，但如果你对网络熟悉的话，就会很快明白每一项的作用了。Destination项和Genmask一起匹配了某个网段的IP。如果指定了网关，那么相应IP地址发出的数据包就会被交给对应的网关进行转发。同时，最后的Iface 指定了要使用哪个网卡接口发送数据。现在，我们只能与IP在192.168.1.0和192.168.1.255之间的主机通信，还有就是通过回环设备lo和自己通信。为了和外界交流，我们需要设置默认的网关。

```
darkstar:~# route add default gw 192.168.1.254
darkstar:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.0     *               255.255.255.0   U     0      0        0 eth0
loopback        *               255.0.0.0       U     0      0        0 lo
default         192.168.1.254   0.0.0.0         UG    0      0        0 eth0
```

很快我们就注意到多了一条默认路由项。它表示如果上面所有的项都不匹配时，就采用默认的路由项。现在假设我们想与64.57.102.34通信，它就会被发送到192.168.1.254，而这个网关就会将我们的数据包进行转发。不幸的是，我们的方法还是不够完善，接下来就是设置DNS。请参见以下第5.3.5节resolv.conf的内容。

慢着，上面我们介绍了DHCP，那么DHCP又怎么配置呢？我们首先要介绍的是**dhcpcd(8)**，它是ISC的DHCP工具中的一部分。假设我们的电脑已经物理上连在网上，并且在我们的网上有个DHCP服务器，那么就可以直接使用下面命令来一次性为网卡配置完毕:

```
darkstar:~# dhcpcd eth0
```

如果一切顺利，那么我们的网卡就能配置好了，我们就能和网上的机器通信，自由地在因特网上翱翔了。但如果由于某些原因，dhcpcd出错了，那么可以试着使用**dhclient(8)**，dhclient是dhcpcd的修补，功能差不多。

```
darkstar:~# dhclient eth0
Listening on LPF/eth0/00:1c:b3:ba:ad:4c
Sending on   LPF/eth0/00:1c:b3:ba:ad:4c
Sending on   Socket/fallback
DHCPREQUEST on eth0 to 255.255.255.255 port 67
DHCPACK from 192.168.1.254
bound to 192.168.1.198 -- renewal in 8547 seconds.
```

那么Slackware为什么会有两个DHCP客户端呢？原因是一个特定的DHCP服务器有时会出错，导致对dhcpcd或dhclient的请求没有响应，这时我们就将希望寄托在另一个DHCP客户端是不是会有反应。一般而言，Slackware使用dhcpcd，并且多数情况下是能成功使用的，但在失效时，使用dhclient就变得很重要的。这两个客户端都是很优秀的，你想有哪个都行。

5.3.4　/etc/rc.d/rc.inet1.conf

如果想为新安装的Slackware分配IP地址，要么可以使用`netconfig`脚本，要么可以手动修改`/etc/rc.d/rc.inet1.conf`文件。在`/etc/rc.inet1.conf`文件中，你会看到如下的内容：

```
# Primary network interface card (eth0)
IPADDR[0]=""
NETMASK[0]=""
USE_DHCP[0]=""
DHCP_HOSTNAME[0]=""
```

之后在文件末尾中可以看到：

```
GATEWAY=""
```

本例中，我们所有的任务就是为这些变量赋值，即在引号中填上正确的值。这些变量会在启动时由`/etc/rc.d/rc.inet1`脚本调用来对网卡进行配置。对于每个NIC（network interface card——网络接口卡），只要输入正确的IP地址，或者在`USE_DHCP`变量栏中填上“YES”。Slackware会按照该文件中NIC 信息的位置来启动相应的NIC。

`DEFAULT_GW`变量用来为Slackware设置默认网关地址。在只有一个路由的情况下，所有本机与因特网的信息交流都是通过这个网关进行的。如果使用的是DHCP，那该行就不需要填写，因为DHCP会为你指定使用哪个网关。

5.3.5　/etc/resolv.conf

好了，现在我们已经设置好了IP地址，默认网关也设置了，你可能有几百万美元（可以考虑分我们一点），但还是不能将域名解析为IP地址。没有人想输入72.9.234.112来打开http://www.slackbook.org。毕竟，除了作者还有谁会记得这些IP地址呢？所以我们需要设置DNS，但应该怎么做呢？那就要用到`/etc/resolv.conf` 文件了。

有可能在你的/etc/resolv.conf 文件中已经有了正确的信息。如果你使用DHCP来设置你的网络，DHCP服务器会为你更新这个文件（技术上来说，它只是告诉dhcpcd应该在这个文件中填写什么内容，dhcpcd照做而已）。如果你想手工修改该文件，那么请参照下面这个例子：

```
# ca /etc/resolv.conf
nameserver 192.168.1.254
search lizella.net
```

第一行很简单。nameserver关键字只是告诉我们应该查询哪个DNS服务器。这行中填写的必须是IP地址。但你想写多少行就可以写多少行。Slackware会按顺序逐个查询，直到找到一个匹配的为止。

第二行就有点意思了。search 关键词的作用是列出一个域名列表，在我们搜索DNS服务器时，会假定是这些域名下的子域名。这就使得我们只使用一个域名的FQDN（完整的符合条件的域名）的前面部分就能访问一个域名。例如，如果“slackware.com”在搜索列表中，我们可以只输入http://store，就可以访问http://store.slackware.com

```
# ping -c 1 store
PING store.slackware.com (69.50.233.153): 56 data bytes
64 bytes from 69.50.233.153 : icmp_seq=0 ttl=64 time=0.251 ms
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.251/0.251/0.251 ms
```

5.3.6　/etc/hosts

现在我们的DNS也正常工作了。如果我们想跳过DNS服务器，或是为一个不存在于DNS服务器上的机器添加一个DNS条目，应该怎么做？Slackware 中包含了一个倍受喜爱的文件/etc/hosts，它相关于本地的DNS服务器，其中包含了一个列表，每个列表都由域名和相应的IP地址构成。

```
# cat /etc/hosts
127.0.0.1               localhost locahost.localdomain
192.168.1.101           redtail
172.14.66.32            foobar.slackware.com
```

本例中，我们可以看到域名localhost对应着IP地址127.0.0.1（该IP总是为localhost保留），域名redtail对应IP192.168.1.101，而域名foobar.slackware.com对应的IP则为172.14.66.32。

5.4　PPP

一些人仍然使用某种拨号连接来连网。尽管偶尔也有用SLIP的，最常用的方法就是PPP。配置PPP来连接远程服务器是相当容易的。我们可以使用下面几种程序来帮助我们进行设置。

请注意：PPP与现在常用的PPPoE是两个不同的概念。这里暂时不打算详细介绍其中的区别。

5.4.1 pppsetup

Slackware提供了一个命名pppsetup的程序来对系统的拨号帐号进行设置，它的外观与netconfig很像。运行该程序需要root权限。之后只需要输入pppsetup即可。

这个程序会提出一系列问题，我们需要正确回答每一个问题。如我们的调制解调器设备，调制解调器初始化字串及ISP的电话号码。其中的一些问题有默认的答案，这些问题通常选择默认即可。

这个程序运行结束后，会创建ppp-go及ppp-off程序。它们的作用分别是启动和结束PPP连接。这两个程序位于/usr/bin文件夹内，并且需要root权限才能运行。

5.4.2 /etc/ppp

对于绝大多数用户来说，只要运行pppsetup就足够了。然而，在一些情况下我们可能想设置一些PPP守护进程用到的值。所有的配置信息都保存在/etc/ppp文件夹下。下面是该文件夹中的一些文件及相应的功能：

ip-down PPP连接结束后pppd执行的脚本。

ip-up 当ppp连接成功后，ppp执行的脚本。可以把想在建立连接后运行的命令写在该文件中。

options 包含一般的pppd的配置选项。

options.demand 在pppd运行在请求模式下的一般配置选项。

pppscript 包含发送给调制解调器的命令。

pppsetup.txt 一个日志文件，记录了运行pppsetup时的所有输入。

注意，这些文件在没有运行pppsetup之前可能是不存在的。

5.5 PPPoE

PPPoE（全称Point-to-Ppoint Protocol over Ethernet，直译为以太网上的点对点协议）可以使以太网的主机通过一个简单的桥接设备连接到一个远端的集中器上。通过PPPoE协议，远端接入设备能实现对每个用户的控制和计费。由于PPPoE具有较高的性价比，它在包括小区组网建设等一系列应用中被广泛采用。目录流行的宽带接入方式ADSL就使用了PPPoE协议。如果您是使用路由连接ADSL 的，那么路由一般自带拨号功能，就不需要在主机上设置了，反之则需要设置PPPoE来进行连网。

Slackware中对PPPoE的支持是由rp-pppoe软件包提供的。它提供了三个主要的命令：pppoe-setup用来设置拨号信息，pppoe-start用来虚拟拨号，而相应地pppoe-stop的作用就是断开连接。

5.5.1 pppoe-setup

pppoe-setup是一个脚本文件，运行之后会向用户提出一些问题，之后会将用户的答案转换成配置文件，保存在/etc/ppp/pppoe.conf文件中。运行pppoe-setup命令的显示如下：

```
# pppoe-setup
Welcome to the Roaring Penguin PPPoE client setup.  First, I will run
some checks on your system to make sure the PPPoE client is installed
properly...

Looks good!  Now, please enter some information:

USER NAME

>>> Enter your PPPoE user name (default bxxxnxnx@sympatico.ca):
```

首先是填写我们的上网帐号。

```
INTERFACE

>>> Enter the Ethernet interface connected to the DSL modem
For Solaris, this is likely to be something like /dev/hme0.
For Linux, it will be ethn, where 'n' is a number.
(default eth0):
```

这个步骤是选择连接到DSL的网卡，Linux下，网卡的形式为“ethn”，其中n为网卡的编号（可以通过ifconfig查看）。默认为第一张网卡。

```
Do you want the link to come up on demand, or stay up continuously?
If you want it to come up on demand, enter the idle time in seconds
after which the link should be dropped.  If you want the link to
stay up permanently, enter 'no' (two letters, lower-case.)
NOTE: Demand-activated links do not interact well with dynamic IP
addresses.  You may have some problems with demand-activated links.
>>> Enter the demand value (default no):
```

接下来的内容是选择是否只在需要时进行连接，如果选择“no”，代表连接后一直保持连接，如果希望只在需要时进行连接，则以秒为单位输入断开连接的时间，例如输入5s，则如果5s内没有用到连接，则自动将连接断开。默认为“no”，即一直保持连接。

```
DNS
```

```
Please enter the IP address of your ISP's primary DNS server.
If your ISP claims that 'the server will provide DNS addresses',
enter 'server' (all lower-case) here.
If you just press enter, I will assume you know what you are
doing and not modify your DNS setup.
>>> Enter the DNS information here:server
```

本步骤是填写ISP（Internet Service Provider，直译为因特网服务提供商）的DNS，如果ISP声称由服务器提供DNS，则此处填写server即可。

```
PASSWORD

>>> Please enter your PPPoE password:
>>> Please re-enter your PPPoE password:
```

此步骤输入密码。

```
FIREWALLING

Please choose the firewall rules to use.  Note that these rules are
very basic.  You are strongly encouraged to use a more sophisticated
firewall setup; however, these will provide basic security.  If you
are running any servers on your machine, you must choose 'NONE' and
set up firewalling yourself.  Otherwise, the firewall rules will deny
access to all standard servers like Web, e-mail, ftp, etc.  If you
are using SSH, the rules will block outgoing SSH connections which
allocate a privileged source port.

The firewall choices are:
0 - NONE: This script will not set any firewall rules.  You are responsible
          for ensuring the security of your machine.  You are STRONGLY
          recommended to use some kind of firewall rules.
1 - STANDALONE: Appropriate for a basic stand-alone web-surfing workstation
2 - MASQUERADE: Appropriate for a machine acting as an Internet gateway
                for a LAN
>>> Choose a type of firewall (0-2):1
```

选择防火墙的规则，0表示不使用防火墙，规则1则适合于一台机器上网的情况，规则2则适用于作为网关连接上网的机器。

```
** Summary of what you entered **
```

```
Ethernet Interface: eth0
User name:          bxxxnxnx@sympatico.ca
Activate-on-demand: No
DNS addresses:      Supplied by ISP's server
Firewalling:        NONE

>>> Accept these settings and adjust configuration files (y/n)?y
```

最后是对刚才回答的信息的一个总结，并寻问是否保存，保存即可。

至此，对PPPoE的配置就完成了。之后只需要使用pppoe-start就可拨号上网，使用pppoe-stop即可断开连接。

```
# pppoe-start
. Connected!
# pppoe-stop
Killing pppd (27230)
Killing pppoe-connect (27213)
```

5.6　无线网络

在Slackbook 2.0的年代，无线网络还是相对较新的东西，但随着笔记本的流行，越来越多人使用无线网络，且由于无线网络不需要传统的双绞线等有线设备，不会占用太多的物理空间，因而越来越流行，现在，无线网络几乎是无所不在了。不幸的是，Linux对于无线网的支持并不像传统有线网那么强劲。

我们将先介绍无线网络配置的三个步骤，之后再对无线网络的相关内容进行更深入的介绍。如果你只是想快速使用无线网络，可以考虑先看第5.6.2节中wicd的相关内容。

5.6.1　无线网络配置步骤

配置一块802.11的无线以太网网卡需要三个基本步骤：

1. 为无线网卡添加硬件支持。
2. 将无线网卡连接到一个无线接入点。
3. 配置该无线网络。

硬件支持

无线网卡的支持是通过内核实现的，要么编译成一个模块，要么直接编译进入内核。一般地，最新的以太网网卡都是通过内核模块支持的，所以我们需要决定使用哪个内核模块并通过/etc/rc.d/rc.modules加载。netconfig可能检测不了无线网卡，所以你可能需要自己

对无线网卡进行设置。请参见http://www.hpl.hp.com/personal/Jean_Torrilhes/Linux/，其中包含了对无线网卡的驱动支持的相关信息。

配置无线网卡

大部分的工作都是通过iwconfig命令完成的，所以如果你想了解更多的信息，可以查阅iwconfig的man手册页。

首先，我们需要设置网卡的无线接入点。就术语“无线接入点”而言，它本身的含义就有很大的不同，而在对无线接入点的配置上也有一些不同，所以我们需要对网卡有一定的了解才能做出合适的配置。一般而言，至少需要知道下面的信息：

- 区域ID，或称为准备接入的网络名称（iwconfig中的ESSID）。
- WAP使用的信道（channel）。
- 加密设置，包括可能用到的密码（最好是十六进制的）。

> **注意：**一个关于WEP的注记。WEP缺陷很大，只要有它的足够的包就可以破解出密码。但也聊胜于无。如果你希望无线网络有更强的安全性，那么你应该看看VPN或IPSec等知识，它们都超出了本书的范围。你也可以设置无线网，让它不对外广播它的ESSID。对无线网管理方针的全面讨论超出了本节的范围，你可以用Google搜索相关内容。
>
> 笔者注：WEP由于安全性问题已经不再推荐使用，现在的无线网一般使用WPA进行加密，以达到较好的效果。

一旦得到足够的信息，并假设已经使用modprobe为无线网卡载入了正确的驱动，就可以开始修改rc.wireless.conf文件添加我们的设置了。rc.wireless.conf文件有些杂乱无章，我们最少要对其中的ESSID、KEY、及CHANNEL（如果网卡需要的话）项进行修改。（可以尝试不设置CHANNEL，如果网卡正常工作，那么好吧；如果不能工作，那么设置合适的CHANNEL即可。）如果你够胆，可以只设置需要乃至的变量。rc.wireless.conf文件中的变量名对应了iwconfig的各项参数。rc.wireless会读取该文件并使用这些参数正确调用iwconfig命令。

如果你的密钥是十六进制的，那是最好的，因为这样方便使用WAP及iwconfig直接使用。如果你只有一串字符串，那么就不能确定WAP是如何将其转换成十六进制的密钥了。这种情况下就要猜测WAP的加密方式了（或者直接得到十六进制的密钥）。

在修改完rc.wireless.conf文件后，以root权限运行rc.wireless，之后再以root权限运行rc.inet1就可以了。我们可以使用一些传统的工具如ping来测试无线网络，当然，也可以用iwconfig进行测试。在测试无线网络时，如果你同时还有有线网卡，那么应该用ifconfig把它们关了，防止干扰。我们也可以通过重启系统来测试我们的修改是否成功。

现在我们已经知道如何通过修改/etc/rc.d/rc.wireless来配置默认的网络，现在让我们仔细研究iwconfig的工作方式。我们能学会一种快速建立无线网络的方法，这种方法我们不推荐，但如果我们在诸如网吧、咖啡厅或其它一些有无线热点覆盖的地方时，这种方法能帮助我们快速连网。

第一个步骤是告诉我们的无线网卡该连哪个网。记得我们的例子里采用的是“wlan0”这个网卡，你在使用时要用自己的网卡来替换。并将“mynetwork”换成你想连到的网络的ESSID。好吗，我知道你知道……之后要做的就是输入无线网的密码（如果设置了的话）。最后指定要使用的信道（如果需要的话）。

```
# iwconfig wlan0 essid "mynetwork"
# iwconfig wlan0 key XXXXXXXXXXXXXXXXXXXXXXXXXX
# iwconfig wlan0 channel n
```

这应该就是无线网络涉及到的内容。

设置网络

这个步骤就是设置如DNS或host一类的事，请参见之前第5.3.2节中有线网络的相关内容。

5.6.2　深入了解无线网络

上一节中，我们对如何配置无线网络有了一个简要的介绍，知道这些知识还不够，下面我们会对无线网络常用到的知识进行介绍。

首先更详细介绍命令`iwconfig`的用法，接着介绍WEP和WPA加密算法，之后再讨论`rc.inet1.conf`在配置无线网络中的使用，最后再介绍一个常用的无线网络配置工具——wicd。

iwconfig

无线网络比传统的有线网络更为复杂，因此需要额外的配置工具。Slackware中包含了多种多样的无线网络工具集，使用它们我们就能对无线网络接口卡（WNIC，简称无线网卡）在最基础的层次上进行配置了。这里并不涵盖所有的内容，而是教会你一些基础知识，让你能进行快速地配置。这里要介绍的就是`iwconfig(1)`命令。在不加任何参数的情况下，它默认显示我们计算机上所有网卡的无线信息。

```
darkstar:~# iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wmaster0  no wireless extensions.

wlan0     IEEE 802.11abgn  ESSID:"nest"
          Mode:Managed  Frequency:2.432 GHz  Access Point:
00:13:10:EA:4E:BD
          Bit Rate=54 Mb/s   Tx-Power=17 dBm
          Retry min limit:7   RTS thr:off   Fragment thr=2352 B
          Encryption key:off
```

```
          Power Management:off
          Link Quality=100/100  Signal level:-42 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0

tun0      no wireless extensions.
```

与有线网络不同的是，无线网络是“模糊的”。网络的边界是不确定的，不同的网络可能互相覆盖。为了防止产生混乱，每个无线网卡（还好）有唯一的标识符。最基本的两个标识符就是ESSID（全称为Extended Service Set Identifier，直译为扩展服务集标识符），以及信道或称作无线传输频率。ESSID就是用来标识我们要连接到的无线网的名字；我们也称为网络名或一些类似的名称。典型的无线网络是在11个不同频率下工作的。即使我们想连接到最基本的无线网络，在诸如设置无线网卡的IP地址之前，我们需要知道这两个信息，当然，可能还需要其它信息。

在上面的例子中，可以看到，我们ESSID设置为“nest”，而我们笔记本是在2.432GHz频率下进行无线传输的。只是连接到一个未加密的无线局域网的话，只要知道这两个信息就足够了。（如果你想来我家使用我家的未加密的无线网，要知道，在接入点允许你与我们LAN通信之前，你要破解一个2048位的SSL密钥。-_-!）

```
darkstar:~# iwconfig wlan0 essid nest freq 2.432G
```

[freq]参数和[channel]本质上是一个东西。只要设置其中的一个就可以了。如果你不知道应该设置哪个频率或信道，Slackware一般能为你自动设置。

```
darkstar:~# iwconfig wlan0 essid nest channel auto
```

使用了auto参数，无论ESSID为“nest”的网络使用哪个频率，Slackware都会尝试连接到信号最强的那个频率。

`iwconfig`的另外一些参数会在下面几个小节中介绍。

WEP

很显然，无线网络的安全性不如有线网络，将数据在空中传输，使得数据容易被第三方拦截。所以过了这么多年，我们开发了很多方法来增强无线网络的安全性。第一种方法名为Wired Equivilant Protection（直译为等价有线保护），或简称为WEP，但该方法与目标相去甚远。如果你现在还在使用WEP，那么我们建议你使用WPA2或其它保护能力更强的措施。对WEP的攻击很平常，一般只要几分钟就能完成。不幸的是，现在还有一些接入点是须采用WEP加密的。

我们有时也会需要连接到这样的网络。连接到一个以WEP加密的接入点是很容易的，尤其是如果知道加密后的十六进制密钥。如果知道十六进制的密钥，只要直接使用[key]参数加上密钥即可，如果只知道ASCII码的密钥，那么输入密钥时要加上前置的“s:”；下面就是几个例子。一般来说，人们倾向于用十六进制的。

```
darkstar:~# iwconfig wlan0 key cf80baf8bf01a160de540bfb1c
darkstar:~# iwconfig wlan0 key s:thisisapassword
```

WPA

Wifi Protected Access（缩写为WPA，直译为wifi访问保护）是WEP的继承人，它的目标是修复无线加密中存在的几个问题。不幸的是，WPA本身也有一些缺陷。更新后的WPA2提供更强大的保护措施。就当前而言，几乎所有的网卡及接入点都支持WPA2，但仍有一些老的设备只支持WEP。如果你需要对你的网络通信进行加密，那么WPA2只能说提供了最基本的保护。不幸的是，iwconfig本身无法设置WPA2。因此，我们需要一个助手程序`wpa_supplicant(8)`。

更悲剧的是，只能手工配置一个用WPA2进行保护的网络；我们必需直接在文本编辑中编辑`/etc/wpa\_supplicant.conf`文件。这里我们只讨论最简单的WPA2保护类型，Pre-Shared Key（直译为预先分享密钥）或简称为PSK。如果需要配置Slackware以连接到更复杂的WPA2加密网络，请参见`wpa\_supplicant.conf`的man手册。

```
# /etc/wpa_supplicant.conf
# ========================
# This line enables the use of wpa_cli which is used by rc.wireless
# if possible (to check for successful association)
ctrl_interface=/var/run/wpa_supplicant
# By default, only root (group 0) may use wpa_cli
ctrl_interface_group=0
eapol_version=1
ap_scan=1
fast_reauth=1
#country=US

# WPA protected network, supply your own ESSID and WPAPSK here:
network=  scan_ssid=1  ssid="nest"  key_mgmt=WPA-PSK  psk="secret passphrase"
```

我们感兴趣的是network括进来的这一块。本例中，我们要连接到的ESSID为“nest”，同时PSK使用“secret passphrase”。设置就完成了。现在我们可以运行wpa_supplicant并使用DHCP获取IP或手动设置静态IP。当然，这还是很麻烦，应该要用个更简单的方法。

再读rc.inet1.conf

欢迎重回`rc.inet1.conf`。记得之前我们就是通过这个文件来设置有线网络的。现在，我们要用该文件来设置wifi网络。然而，如果你使用的是WPA2，还是需要先正确设置`wpa_supplicant.conf`文件。

还记得吗？每张网卡都有与其对应的标识符变量。对于wifi网卡也是一样的，只不过因为无线网络更复杂，所以变量也更多。

```
# rc.inet1.conf (excert)
# ======================
## Example config information for wlan0.  Uncomment the lines you need and fill
## in your info.  (You may not need all of these for your wireless network)
IFNAME[4]="wlan0"
IPADDR[4]=""
NETMASK[4]=""
USE_DHCP[4]="yes"
#DHCP_HOSTNAME[4]="icculus-wireless"
#DHCP_KEEPRESOLV[4]="yes"
#DHCP_KEEPNTP[4]="yes"
#DHCP_KEEPGW[4]="yes"
#DHCP_IPADDR[4]=""
WLAN_ESSID[4]="nest"
#WLAN_MODE[4]=Managed
#WLAN_RATE[4]="54M auto"
#WLAN_CHANNEL[4]="auto"
#WLAN_KEY[4]="D5AD1F04ACF048EC2D0B1C80C7"
#WLAN_IWPRIV[4]="set AuthMode=WPAPSK | \
#   set EncrypType=TKIP | \
#   set WPAPSK=96389dc66eaf7e6efd5b5523ae43c7925ff4df2f8b7099495192d44a774fda16"
WLAN_WPA[4]="wpa_supplicant"
#WLAN_WPADRIVER[4]="ndiswrapper"
```

我们讨论有线网络的时候，用ethn表示网卡，每个n对应一张网卡。这里就不一样了。注意到变量IFNAME[4]的值为“wlan0”。无线网卡的名字可能不是“ethn”类型的，这里就是一个例子。当启动脚本读取`rc.inet1.conf`时，Slackware就知道将这些选项应用到“wlan0”这张无线网卡上，而不是eth4这张有线网卡（你机子上大概也不存在这张网卡）。别的一些选项和有线的配置类似。IP地址信息和有线网络的设置一模一样。但还有些变量需要解释一下的。

第一个就是WLAN_ESSID[n]及WLAN_CHANNEL[n]变量，当然，阅读完之前的内容后只要看名字就知道干什么用的了，它们代表了要连接的ESSID和相应的信道。WLAN_MODE[n]的值可以是“managed”或“ad-hoc”。如果是连接到一个接入点（可以理解为要连到其它网）的话，就选择managed模式，如果使用的是WEP加密的话，WLAN_KEY[n]是要使用的WEP密钥。WLAN_IWPRIV[n]是个很复杂的变量，它在变量的值中设置其它变量。WLAN_IWPRIV[n]是用于WPA2网络中的。在这个变量中，要告诉Slackware用什么样的认证模式、加密类型及WPA2的密钥。请注意，WLAN_KEY[n]变量与WLAN_IWPRIV[n]变量是相互排斥的，我们不能在同一个接口上同时使用两个变量。如果你正确地配置完这些变量后，Slackware就是会

系统启动时尝试为你连接到无线网络上。

但等等，还是好麻烦啊！而且如果我想连到不止一个无线网络上呢？例如我会带笔记本到去学校，回家，我需要在不同范围的时候自动连上不同的网。用这里讲的方法是不是太麻烦了啊！是的，的确如此。

wicd

接下来介绍`wicd(8)`，它是那些抱着笔记本到处跑的人们配置有线或无线网络的首选管理器。正确的发音为“wicked”。wicd能保存任意数量的无线网络连接信息，且在指点之间或只运行一个简单的命令就是能连接到对应的网络。Slackware默认不安装wicd。因此它多少会对正常的配置方法有影响。但你可以在Slacwkare DVD或FTP上的`/extra`目录下找到wicd的安装包。wicd不仅是一个网络连接守护进程，同时提供了一个配置网络的图形界面。当然，也提供了在文本终端下的配置界面，`wicd-curses(8)`几乎和传统的GTK前端一样强大。要使用wicd，我们首先要把在`rc.inet1.conf`中的所有设置清空。

```
# rc.inet1.conf
# =============
# Config information for eth0:
IPADDR[0]=""
NETMASK[0]=""
USE_DHCP[0]="no"
DHCP_HOSTNAME[0]=""
# Default gateway IP address:
GATEWAY=""
```

现在我们可以安装wicd了，设置守护进程开机启动，之后就可以用这个更友好的程序了。

```
darkstar:~# installpkg /path/to/extra/wicd/wicd-1.6.2.1-1.txz
darkstar:~# chmod +x /etc/rc.d/rc.wicd
darkstar:~# /etc/rc.d/rc.wicd start
```

如果你主要使用的是文本终端，那么可以从终端中运行wicd-curses。相反，如果你一般使用的是X提供的图形桌面，那么可以从KDE或XFCE菜单中启动wicd的图形前端。还有一种方法，就是在虚拟终端或run对话框中运行`wicd-client(1)`。

5.7　网络文件系统

现在，我们的网络能正常工作了。我们可以ping到内网的主机，并且如果网关设置正确的话，我们也能ping到因特网上的计算机了。正如我们知道的一样，将一台电脑连接到网络，目的就是能通过网络访问该电脑，当然也有人连网只是觉得好玩，但大多数人的目的是共享文件或打印机。他们希望能访问因特网上的文件或是玩一些连网的游戏。安装TCP/IP并使它正常工作是达成这一目标的一个途径，但这只是最基础的步骤。要想共享文件，我们还需要使

用FTP或SCP进行来回传输。在新安装的Slackware上，我们并不能像使用Windows时，点击“网上邻居”就可以的。我们希望与其它的Unix机器无缝地共享文件。

幸运的是，我们可以使用网络文件系统技术，这样访问其它机器上的文件也是透明的了。在使用这些软件访问其它机器上的文件时，我们并不需要知道一个文件存在哪个机器上，只要知道它存在及如何访问它就可以了。之后管理如何通过可用的文件系统及网络文件系统来使用户访问该文件，就归操作系统管了。最常用的两种网络文件系统是SMB（通过Samba实现）及NFS。

5.7.1 SMB/Samba/CIFS

SMB（全称为Server Message Block，直译为服务器信息块）是一个古老的NetBIOS协议，最早由IBM为其局域网管理工具使用。微软一直对NetBIOS及其衍生物感兴趣（NetBEUI，SMB及CIFS）。Samba项目早在1991年就存在了，当时有用来将运行NetBIOS的IBM PC连接到一个Unix服务器上。而现在，由于Windows的支持，几乎整个文明世界的人们都倾向于使用SMB来在网络上共享文件及打印服务。

Samba的配置文件是是`/etc/samba/smb.conf`；该文件的注释良好，文档齐全。并且系统已经创建了一个样例，我们可以通过查看和修改该样例来创建一个满足我们需要的配置文件。如果你想要更为精细的控制，那么smb.conf的man 手册是不可获缺的。由于Samba的文档是如此之好，并且可以在我们上面提到的位置找到，所以我们不再详述文档，取而代之的是，我们将对基础的一些方面，做一个快速而简单的介绍。

`smb.conf`文件分为好几节：一个共享内容占一小节，加上一个全局的小节用来设置一些全局使用的选项。有一些选项只能用在全局小节中；而另一些变量只能在非全局小节中使用；而全局的选项会被每个内容小节中的选项覆盖。记得自己翻翻man手册。

我们希望通过修改`smb.conf`文件来在局域网中共享文件，我们建议你修改下面列出的选项。

```
[global]
# workgroup = NT-Domain-Name or Workgroup-Name, eg: LINUX2
workgroup = MYGROUP
```

修改工作组名，只要填上你在本地使用的工作组或域名即可。

```
# server string is the equivalent of the NT Description field.
server string = Samba Server
```

这个选项代表我们的Slackware的机器名，这个机器名就是在Windows的“网上邻居”中显示的机器名。

```
# Security mode. Defines in which mode Samba will operate. Possible
# values are share, user, server, domain and ads. Most people will want
# user level security. See the Samba-HOWTO-Collection for details.
   security = user
```

安全模式，我们一般选择user级别就可以了。

```
# You may wish to use password encryption. Please read
# ENCRYPTION.txt, Win95.txt and WinNT.txt in the Samba
# documentation.
# Do not enable this option unless you have read those documents
encrypt passwords = yes
```

如果encrypt passowrds选项没有开启，那么就不能使用Samba与NT4.0、Win2k、WinXP及Win2003等平台共享。更早的Windows版本则不需要加密就能共享文件。

注：Slackware 13.37中的samba-3.5.10的sample文件中是没有这个选项的。原因是已经默认开启了encrypt password，且由于现在NT4.0以前的版本几乎没人用，所以也没有必要将该选项关闭。

SMB是一个认证式的协议，也就是在使用该服务时要提供正确的用户名及密码。我们将使用smbpasswd命令来告诉samba服务器哪些用户名及密码是有效的。smbpasswd有一些选项来告诉自己是添加一个传统意义上的用户还是添加一个机器用户（SMB在添加机器用户时，要求使用该计算机的NETBIOS名作为用户名，以此来限制用户能认证的计算机）。

```
Adding a user to the /etc/samba/private/smbpasswd file
添加一个普通用户，用户名为user，保存在/etc/samba/private/smbpasswd文件中。
# smbpasswd -a user
Adding a machine name to the /etc/samba/private/smbpasswd file
添加一个机器用户，用户名为machine，保存在/etc/samba/private/smbpasswd文件中。
# smbpasswd -a -m machine
```

要注意的是，所添加的用户名或机器用户名必须事先就存在于/etc/passwd文件中。当然，可以通过adduser命令添加该用户。注意，在使用adduser添加机器名时，机器名的末尾要加上美元符（“$”），但在使用smbpasswd的时候不要加美元符，因为smbpasswd会自动加上。如果使用adduser时不这么命名的话，为samba添加机器名时会出现错误。

```
# adduser machine$
```

5.7.2　网络文件系统（NFS）

NFS（全称为Network File System，直译为网络文件系统）最早是由SUN公司为其旗下的Solaris系统（另一个类Unix系统）开发的。与SMB对比，NFS的设置及运行极为简单，但安全性比SMB略逊一筹。NFS的不安全性体现在我们可以根据一台机器的的用户名及组ID猜到另一台机器的信息。NFS不是一个认证式协议。NFS 协议在之后的版本被重新修订以增加安全性，但在本书写作的时候，NFS已经不怎么流行了。

NFS的配置是由/etc/exports文件管理的。当我们用文本编辑器打开该文件时，只能看到一个空的文件，只是最上面的几行注释。我们需要为每一个想导出的目录在该文件中加上一

条导出条目，在条目上还在加上允许共享的客户端列表。还是举个例子说明吧。假设我们要导出/home/foo目录，并允许工作站Bar共享，我们只要在/etc/exports中添加下面这行：

```
/home/foo Bar(rw)
```

下面，我们将展示在exports的man手册中展示的例子：

```
# sample /etc/exports file
/               master(rw) trusty(rw,no_root_squash)
/projects       proj*.local.domain(rw)
/usr            *.local.domain(ro) @trusted(rw)
/home/joe       pc001(rw,all_squash,anonuid=150,anongid=100)
/pub            (ro,insecure,all_squash)
```

从例子中可以看到，我们可以使用很多的选项，这些选项的意思在例子中也可以清楚地看出（不懂英语的同学查查单词吧）。

NFS工作时作出如下假设：在一个网络中，一台机器上的用户即使在其它机器上想要访问服务器上的文件，使用的也是同一个用户ID。而在NFS客户端对NFS服务器进行访问时，要将发起请求的用户ID连同请求一起发送给服务器。而该用户ID 的权限与该用户在本机上的读写权限是一致的。正如我们看到的一样，如果另一个用户使用一个已知用户的ID来访问服务器，那他就可以在ID的主人不知情的情况下做一些坏事。作为防止这种事件的一个手段，每个文件夹在挂载的时候使用了root_squash选项。使用该选项会将那些声称自己是root的ID映射到一个不同的UID上，这就使从外部访问这些导出目录的用户无法使用root权限。root_squash貌似是默认开启的，但我们还是建议你在/etc/exports文件中手动设置一下。

当然，我们也可以通过使用exports命令来添加导出目录。例如：

```
# exports -o rw,no_root_squash Bar:/home/foo
```

本行命令的作用是：导出/home/foo目录使名为“Bar”的机器访问，相应的权限为读写权限。另外，NFS服务器不会使用**root_squash**权限，这意味着在“Bar”机器上的任何UID为“0”（root用户的UID）的用户都会拥有与服务器上的root用户相同的权限。该命令的语法看起来比较奇怪（一般而言，使用诸如computer:/directory/file这样的语法时，我们指的是一台机器上的某个文件）。

关于exports文件的更多信息，请参阅它的man手册。

第六章　X的相关配置

6.1　什么是X

千万年前……计算机的终端只包含一个屏幕和键盘，就没有其它东西了。那时候，鼠标是个稀少的东西，终端下的操作都是菜单驱动的。上帝说，要有图形接口，于是就有了图形用户接口（GUI），之后，世界发生了巨大的变化。当今的用户都习惯于在屏幕上移移鼠标，点点图标，运行的程序也带有漂亮的图片和动画。但UNIX是在GUI之前诞生的，所以UNIX中的GUI有点事后诸葛的味道。在很长一段时期，人们都在毫无图形界面支持的环境下工作，但现在的Linux用户恐怕没有人不是工作在一个美仑美奂的可指点的GUI环境了吧，这些GUI都运行在*X(7)*之下。

那么，什么是X？它是一个带有图标的桌面吗？还是说是菜单？难道是窗口管理器？它会为一些点作标记吗[1]？天空中一记响亮的回声：不！GUI由很多部分组成，X只是其中最基本的部分。X是一个从鼠标、键盘或是其它设备中获取输入的软件。X是告诉显卡应该怎么操作的软件。简单地说，X就是一个以图形为目的与系统硬件交流的软件；而其它所有的图形软件只和X进行对话。

我们先停一下来讨论X的命名，X只是那一打名字中的一个。它还被叫作X11、xorg、the X Window System、X Window、X11R6，X Version 11以及其它的一些名字。无论你听到的是哪个，只要把它理解为X就行了。

从Slackware-10.0开始，Slackware中的X Window环境就是由Xorg提供的。所谓的X，就是为用户提供图形接口，它与操作系统是独立的，在这点上与Windows或MacOS不同。

X Window System 是由许多运行在用户空间的程序组成的。最主要的两大部分就是X服务器和窗口管理器。X服务器提供了与视频硬件相互作用的函数，所以是专门针对系统设计的。窗口管理器位于X服务器之上，为用户提供用户接口。采用这个架构的优点就在于我们可以拥有多种用户接口，而这只要通过选择不同的窗口管理器就能实现。

6.2　设置X服务器

曾经有这么一段时间，那时配置X是困难并且痛苦的过程，成百上千的显示器都因此“冒烟”了。而现在，X已经很人性化了。事实上，多数用户根据不要需要去配置X，Slackware会

[1] Does it mark the spot，不懂怎么翻译。

自动检测并做出正确的设置。然而，还是有一些计算机不会被正确地识别，因此需要一些额外的手工设置。

之前（起码在Slackbook 2.0的年代），X的配置文件为/etc/X11/xorg.conf文件，并且如果你创建了这个文件，X会尊重你在该文件中做出的任何配置，即采用这些配置。幸运的是，从XOrg 1.6.3开始，X不需要/etc/X11/xorg.conf文件就能正常地工作。如果因为某个不为人知的原因，你需要为X做一些设置，我们建议你也不要使用这个文件，这个文件已经较为过时，并且它的灵活性很差。现在我们采用的是在/etc/X11/xorg.conf.d/文件夹来对X进行配置。X在启动时会读取该文件夹中的配置文件，这就使得我们可以将对X的配置分为几个部分，从而方便管理。注意，/etc/X11/xorg.conf.d/文件夹中的配置文件的文件名要以.conf结尾。例如，下面是我的笔记本上的/etc/X11/xorg.conf.d/synaptics.conf文件的内容。

```
darkstar:~$ cat /etc/X11/xorg.conf.d/synaptics.conf
Section "InputDevice"
    Identifier      "Synaptics Touchpad"
    Driver          "synaptics"
    Option          "SendCoreEvents"        "true"
    Option          "Device"                "/dev/psaux"
    Option          "Protocol"              "auto-dev"
    Option          "SHMConfig"             "on"
    Option          "LeftEdge"              "100"
    Option          "RightEdge"             "1120"
    Option          "TopEdge"               "50"
    Option          "BottomEdge"            "310"
    Option          "FingerLow"             "25"
    Option          "FingerHigh"            "30"
    Option          "VertScrollDelta"       "20"
    Option          "HorizScrollDelta"      "50"
    Option          "MinSpeed"              "0.79"
    Option          "MaxSpeed"              "0.88"
    Option          "AccelFactor"           "0.0015"
    Option          "TapButton1"            "1"
    Option          "TapButton2"            "2"
    Option          "TapButton3"            "3"
    Option          "MaxTapMove"            "100"
    Option          "HorizScrollDelta"      "0"
    Option          "HorizEdgeScroll"       "0"
    Option          "VertEdgeScroll"        "1"
    Option          "VertTwoFingerScroll"   "0"
EndSection
```

把对X配置的每个小节者单独保存为一个文件，能更好地对X的配置文件进行管理。你也可以查看/usr/share/X11/xorg.conf.d，查看Slackware对X的一些默认配置文件。

6.2.1　生成xorg.conf

尽管上面我们已经介绍了如何对X进行配置，但主要还是介绍X的配置文件的布局，且体的配置方法还没说明。首先，关于X配置文件的语法请参见Xorg的man手册，此处不再介绍，这里要介绍的是Slackware中提供的两个自动生成配置文件的工具：Xorg与xorgsetup。

运行命令

```
darkstar~# Xorg -configure # 或
darkstar~# X-configure
```

则Xorg会自动检测当前机器的硬件，并在/root目录下生成一个xorg.conf.new文件，我们可以先用如下命令对该文件进行测试：

```
darkstar~# Xorg -config /root/xorg.conf.new -retro
```

如果之后屏幕上出现一个“X”的鼠标图标，并能正常移动，那么一般说足以说明该xorg.conf文件是正确的，之后可以将其复制为/etc/X11/xorg.conf，采用古老的方法配置X，或者将该文件中的需要的某些部分复制为新的文件，放入/etc/X11/xorg.conf.d文件夹中，会用同样的效果。当然，就是上面提到的，现在建议采用第二种形式。

这是一个很方便的方法，但在Slackware中，还保留着一个脚本配置工具，它就是xorgsetup，它是一个脚本，以root权限运行，会有一些选项让用户选择，再根据对硬件的检测生成正确的配置文件。对于新手进行配置也是很有帮助的，生成配置文件后的用法就不再赘述。

6.3　选择窗口管理器

Slackware中包含了许多不同的窗口管理器及桌面环境。窗口管理器是一种用于在屏幕上画出应用程序窗口、改变窗口大小及类似工作的一个程序。而桌面环境除了包含一个窗口管理器外，还包含任务栏、菜单、图标及一些其它的东西。Slackware中包含了KDE及XFCE桌面环境以及其它的一些窗口管理器。使用哪一个完全取决于你的喜好，但一般而言，窗口管理器比桌面环境要快，所以更适合于那些缺内存、CPU慢的老机器。桌面环境对习惯使用Windows的用户更为友好。

6.3.1　xinitrc

xinit(1)是真正启动X的程序；它由startx(1)调用，所以你可能不会注意到它的存在（也不需要注意到）。但它的配置文件决定了X启动时，运行哪个程序（尤其包括了使用哪个窗口管理器）。xinit首先检查主目录下是否有.xinitrc文件，如果有，就使用它进行启动；否则就使用/etc/X11/xinit/xinitrc（系统默认）文件进行启动。下面是xinitrc文件的一个例子：

```
#!/bin/sh
# $XConsortium: xinitrc.cpp,v 1.4 91/08/22 11:41:34 rws Exp $
userresources=$HOME/.Xresources
usermodmap=$HOME/.Xmodmap
sysresources=/usr/X11R6/lib/X11/xinit/.Xresources
sysmodmap=/usr/X11R6/lib/X11/xinit/.Xmodmap

# merge in defaults and keymaps
if [ -f $sysresources ]; then
xrdb -merge $sysresources
fi
if [ -f $sysmodmap ]; then
xmodmap $sysmodmap
fi
if [ -f $userresources ]; then
xrdb -merge $userresources
fi
if [ -f $usermodmap ]; then
xmodmap $usermodmap
fi
# start some nice programs
twm &
xclock -geometry 50x50-1+1 &
xterm -geometry 80x50+494+51 &
xterm -geometry 80x20+494-0 &
exec xterm -geometry 80x66+0+0 -name login
```

上面所有的“if”语句的作用是整合在其它配置文件中的配置信息。该文件有趣的地方是文件末尾，这里我们运行了很多程序。X会话会开启**twm(1)**窗口管理器、一个时钟及三个虚拟终端。注意到在最后一个xterm前的exec，它的作用是用最后一行的**xterm(1)**命令替换当前运行的shell（即当前执行xinitrc脚本的shell），当用户从那个xterm中退出时，整个X会话就结束了。

要定制自己的X启动脚本，只要将/etc/X11/xinit/xinitrc文件复制到~/.xinitrc文件并进行修改，随意替换那些运行程序的行。例如我们xinitrc文件最后几行就是：

```
# Start the window manager
exec startkde
```

注意，在/etc/X11/xinit文件夹中的那些xinitrc.*文件对应启动了相应的窗口管理器及GUI。我们也可以直接拿来使用。

6.3.2 xwmconfig

由于Slackware中安装的窗口管理器和桌面环境比较多，配置起来也相对麻烦一些，所以Slackware中包含了一个名为`xwmconfig`的程序，用来选择窗口管理器。运行如下：

```
darkstar~% xwmconfig
```

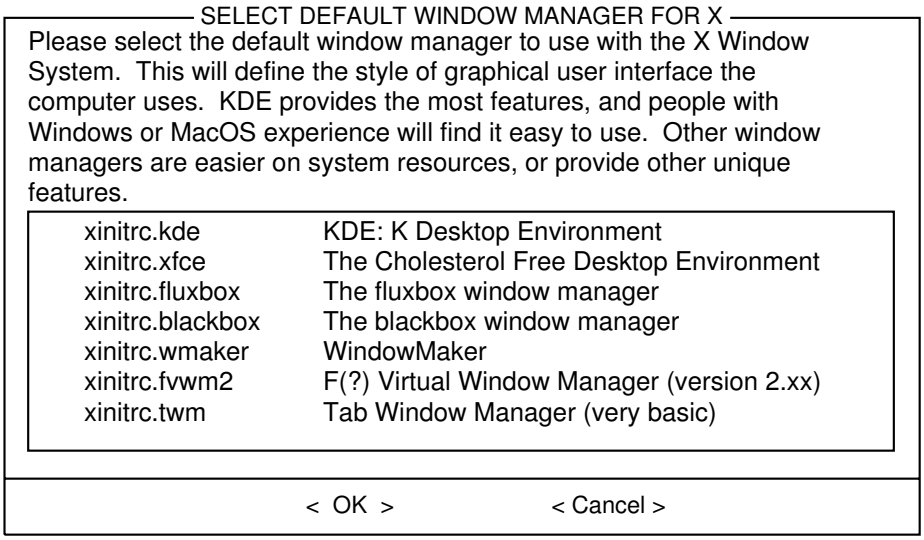

图 6.1: 使用xwmconfig选择桌面

运行之后会显示当前安装的所有窗口管理器的列表，选择自己想用的即可。安装过程中也是使用此程序选择窗口管理器。因为每个人的风格不同，也没有必要都选择默认的。

之后启动X就万事OK了。

6.4 启动进入图形界面

默认情况下，Slackware启动后会显示一个虚拟终端，并提示登陆。这对于大多数用户已经足够满足要求了。如果你要运行的是命令行程序，那么登陆后直接在终端中运行即可，如果你想运行X，那么登陆后运行`startx`就可以启动X了。但如果我们的工作几乎是在图形界面下的呢？这种情况下，直接登陆GUI界面会不会更方便一点呢？幸运的是，这种想法很容易就能达成了。

Slackware使用System V的init系统，这就使得管理员能启动进入或临时改变为某个运行级别。事实上，关机也只是改变运行级别而已。运行级别的内容也比较复杂，这里就不深入讨论了。

运行级别是通过`inittab(5)`配置的。最常用的是第3级（Slackware默认的运行级别）以及第4级（GUI）。如果想让Slackware启动后进入GUI界面，那么只要用文本编辑器（可以考虑先看看介绍vi或emacs的章节）打开`/etc/inittab`文件进行修改即可。在文件中找到这些行：

```
# These are the default runlevels in Slackware:
#   0 = halt
#   1 = single user mode
#   2 = unused (but configured the same as runlevel 3)
#   3 = multiuser mode (default Slackware runlevel)
#   4 = X11 with KDM/GDM/XDM (session managers)
#   5 = unused (but configured the same as runlevel 3)
#   6 = reboot

# Default runlevel. (Do not set to 0 or 6)
id:3:initdefault:
```

该文件中（大多数配置文件也是如些），所有在#号之后的内容都是注释，`init(8)`不会解释这些内容。即使你对initab不理解也没关系，因为很多老手也不理解。我们感兴趣的只有最后一行，把其中的3改成4就可以了，之后重启就完成任务了。

```
# These are the default runlevels in Slackware:
#   0 = halt
#   1 = single user mode
#   2 = unused (but configured the same as runlevel 3)
#   3 = multiuser mode (default Slackware runlevel)
#   4 = X11 with KDM/GDM/XDM (session managers)
#   5 = unused (but configured the same as runlevel 3)
#   6 = reboot

# Default runlevel. (Do not set to 0 or 6)
id:4:initdefault:
```

改完之后，Slackware就会启动进入第4个运行级别，并且运行`/etc/rc.d/rc.4`文件（之前的第4.1.3节文件系统布局中介绍过了）。该文件会启动X并调用我们选择的登陆管理器。那么登陆管理器在哪选择的呢？有好几种方法，我们会在看完`rc.4`的内容后进行介绍。

```
# Try to use GNOME's gdm session manager:
if [ -x /usr/bin/gdm ]; then
exec /usr/bin/gdm -nodaemon
fi
# Not there? OK, try to use KDE's kdm session manager:
if [ -x /opt/kde/bin/kdm ]; then
exec /opt/kde/bin/kdm -nodaemon
fi
```

```
# If all you have is XDM, I guess it will have to do:
if [ -x /usr/X11R6/bin/xdm ]; then
exec /usr/X11R6/bin/xdm -nodaemon
fi
```

正如我们看到的，rc.4先检查gdm的存在与否及是否可执行，如果存在且可执行则执行gdm，之后检查kdm，最后是xdm。选择登陆管理器的一个方法就是用removepkg把不想要的管理器删除。关于removepkg的内容请参见第章。

当然，我们也可以把不想使用的管理器的可执行权限取消了。chmod命令会在第章中介绍。

```
darkstar~# chmod -x /usr/bin/gdm
```

最后，可以把不想使用的窗口管理器在rc.4文件中注释了。

```
# Try to use GNOME's gdm session manager:
# if [ -x /usr/bin/gdm ]; then
# exec /usr/bin/gdm -nodaemon
# fi

# Not there? OK, try to use KDE's kdm session manager:
if [ -x /opt/kde/bin/kdm ]; then
    exec /opt/kde/bin/kdm -nodaemon
fi

# If all you have is XDM, I guess it will have to do:
if [ -x /usr/X11R6/bin/xdm ]; then
    exec /usr/X11R6/bin/xdm -nodaemon
fi
```

上面，我们把gdm的内容加了注释，即使gdm存在并可执行，shell也不会对它进行检测。

第七章　启动

好的，我们已经在系统上安装了Slackware，现在我们要学的不是别的，就是控制你机器启动顺序的东西，以及防止有一天不能用了，还要学习如何修复它。如果你用Linux的时间够长，范个错误导致不能启动是尽早的事。幸运的是，不需要重装系统我们就能修复它。许多操作系统都隐藏底层如何工作的细节，与之相反，Linux（这里尤指Slackware）让我们能完全控制启动过程。只要简单地修改一两个配置文件，重新运行引导装载程序的安装程序，就是简单快速地对系统进行修改（或者破坏）。Slackware甚至使得多操作系统启动变得很容易，如和其它的Linux发行版或Microsoft Windows一起启动。

7.1　mkinitrd

进行其它步骤之前，快速地讨论一下Linux内核是有必要的。Slackware Linux包括了至少两个，有时更多的不同内核。它们都是从相同的源码编译而来，也因此是“一致”的，但它们不是一模一样的。根据你的系统架构及Slackware的版本，安装器可能已经用一好几个内核来引导你的系统。一些内核是为单处理处系统准备的，另一些是为多处理器系统准备的（在32位的Slackware中）。在古老的年代，有许许多多用于安装在不同磁盘控制器下的内核。对于我们的讨论，重要的是“huge”内核和“generic”内核。

如果你查看你的`/boot`文件夹，你会看到你的系统里安装了好几个内核。

```
darkstar:~# ls -1 /boot/vmlinuz*
/boot/vmlinuz-huge-2.6.29.4
/boot/vmlinuz-generic-2.6.29.4
```

这里，可以看到，我安装了两个内核,`vmlinuz-huge-2.6.29.4`及`vmlinuz-generic-2.6.29.4`。Slackware的每个发行版都包含了好几个不同内核版本，有时甚至取了不同的名字，所以如果在你的机子上看到的内容跟我不同，不要担心。

Huge内核就像你感觉的一样，很大。然而，这并不意味着这个内核里包含了所有能有的驱动，也不意味着所有的驱动都编译到了内核中。相反，这些内核是为了从Slackware支持的每种你能想到的电脑（当然，也有一些电脑不能通过它们启动）启动而编译的。这些内核理所当然地含有一些支持你的机器没有的（也可能永远不会有的）硬件的驱动，但这并没有关系。Slackware包含这些内核是有道理的，其中最重要的一条就是Slackware的安装器要用到它——Slackware的安装盘就是在这上面运行的。如果你是让安装器自动选择内核进行启动的话，很有可能选择的就是这些内核，主要原因就是它们支持很多的硬件。相反，不使用外部模块的

放，generic内核支持的硬件就很少。如果你想用generic内核的话，你就必须用到叫作initrd的东西，它是用mkinitrd(8)这个工具做的。

那么，为什么还会有generic内核呢？当前，Slackware开发团队有一些推荐使用generic内核的理由。最明显的一个就是大小。在解压、载入内存之前，huge内核的大小的几乎是generic内核的两倍。如果你是在一台很老的机子上运行，拥有的内存很小的话，那么generic节省的空间还是很可观的。与之不同的是，我们的理由考虑的是质量问题。huge内核中的驱动会时不时地冲突，而且一般来说，huge内核的表现不如generic内核。并且，使用generic内核的话，我们可以单独地为特定驱动指定一些特别的参数，而不是必须通过内核命令行来传递。比起将驱动编译进内核，Slackware中的一些工具，在驱动编译成模块的情况下运行得更好。如果还是不好理解的话，只要想着“huge内核＝好，generic内核＝更好”。

不幸的是，generic用起来不像huge内核那样直接。要使用generic内核启动系统，你还必须使用一个包含基本模块的initrd。所谓的模块，是指从内核源码编译而得的一些部件，这些部件在内核运行时能动态地加载或卸载（使用modprobe(8)是一个理想的选择）。以增加一点复杂性为代价，这种方法使系统更为灵活。你可以把模块看作是设备驱动。至少在本节里能这么干。一个典型的例子是，无论你的root分区用的是什么文件系统[1]，你都要加入支持相应文件系统的模块。如果你的分区是在一块SCSI或RAID硬盘上，你还要加入支持它们模块。最后，如果你用到软件模拟的RAID、磁盘加密或LVM，那么不管你用不用generic内核，都要创建一个initrd。

所谓的initrd，是一个压缩的cpio(1)归档文件。所以要创建的话不是很直接。庆幸吧，Slackware已经包含了能容易创建它的工具——mkinitrd。对它的全面讨论已经超出了本书的范围，但我们会讨论一些重点概念。想看完整的解释，要么去看它的man手册，要么运行带参数[--help]的mkinitrd。

```
darkstar:~# mkinitrd --help
mkinitrd creates an initial ramdisk (actually an initramfs cpio+gzip
archive) used to load kernel modules that are needed to mount the
root filesystem, or other modules that might be needed before the
root filesystem is available.  Other binaries may be added to the
initrd, and the script is easy to modify.  Be creative.  :-)
.... many more lines deleted ....
```

使用mkinitrd的时候，你必须了解一些信息：你的root分区、root文件系统、你用到的硬盘控制器[2]以及你是否用到了LVM、软件RAID或磁盘加密。如果你正在使用某些SCSI控制器（并且root分区就放在这个控制器上），那么你只要知道root文件系统及分区类型。假设你是用huge内核启动进入Slackware安装界面的，那么你可以很容易地用mount命令来找到这些信息，当然，也可以通过查看/proc/mounts中的内容得到。

```
darkstar:~# mount
/dev/sda1 on / type ext4 (rw,barrier=1,data=ordered)
```

[1] 如ext2,ext3,ext4,xfs等。译者注。

[2] 简单的理解就是硬盘类型，如PATA、SATA等。译者注。

```
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
usbfs on /proc/bus/usb type usbfs (rw)
/dev/sda2 on /home type jfs (rw)
tmpfs on /dev/shm type tmpfs (rw)
```

在上面这个例子中，可以看到，root分区位于/dev/sda1中，类型是ext4。如果想为这个系统创建一个initrd，我们就可以告诉initrd这些信息。

```
darkstar:~# mkinitrd -f ext4 -r /dev/sda1
```

大多数情况下，mkinitrd能自动检测到这些信息，但手动指定这些信息也不会让你少块肉。现在我们创建了我们自己的initrd，接下来只要告诉LILO去哪找到这个文件就行了。我们会在下一节中集中介绍这个内容。

然而，如果去查看mkinitrd的所有不同选项是很痛苦的，如果想记住它们就更糟糕了，尤其是在你想一口气试用我们不同的内核。对于Slackware开发团队而言这是很无趣但又痛苦的事，所以他们想到编写一个简单的配置文件：mkinitrd.conf(5)。在/etcmkinitrd.conf.sample文件夹下可以找到一个文件，使用这个文件就可以为很容易地为我们的系统做配置。下面晒晒我的。

```
darkstar:~# >/prompt>cat /etc/mkinitrd.conf.sample
# See "man mkinitrd.conf" for details on the syntax of this file
#
SOURCE_TREE="/boot/initrd-tree"
CLEAR_TREE="0"
OUTPUT_IMAGE="/boot/initrd.gz"
KERNEL_VERSION="$(uname -r)"
#KEYMAP="us"
MODULE_LIST="ext3:ext4:jfs"
#LUKSDEV="/dev/hda1"
ROOTDEV="/dev/sda1
ROOTFS="ext4"
#RESUMEDEV="/dev/hda2"
#RAID="0"
LVM="1"
#WAIT="1"
```

至于每行都有什么作用，并自行查看**mkinitrd.conf** 的man手册。将该样例文件复制到**/etc/mkinitrd.conf** 并作出合适的修改。一旦进行了正确的设置，我们只需要以参数[-F]运行mkinitrd即可。无需记住这些晦涩的参数，mkinitrd就会自动创建一个正确的initrd文件，并且自动安装。

如果你不确定在配置文件或命令行中应指定什么样的选项，还有一个终级的必杀。Slackware 中有一个帅气的工具，能告诉我们用现在运行的内核，创建一个initrd要使用什么参

数，这个工具是/usr/share/mkinitrd/mkinitrd_command_generator.sh。运行这个脚本，它就会生成一个适用于现在操作系统的mkinitrd的参数，但你最好还是检查一遍。

```
darkstar:~# /usr/share/mkinitrd/mkinitrd_command_generator.sh
mkinitrd -c -k 2.6.33.4 -f ext3 -r /dev/sda3 -m \
  usbhid:ehci-hcd:uhci-hcd:ext3 -o /boot/initrd.gz
```

7.2 LILO

LILO是Linux Loader的缩写，是Slackware Linux的默认引导装载程序。如果你用过其它版本的话，你可能对GRUB更熟悉。如果你更喜欢GRUB的话，可以在Slackware的CD上找到extra/文件夹，GRUB的安装包就在里面。由于LILO是Slackware默认的引导装载程序，我们只对LILO进行讨论。

LILO的设置比较麻烦，可能会吓到一些新手，所以Slackware自带了一个特殊的设置程序，叫作liloconfig。一般情况下，第一次运行liloconfig的是Slackware安装器，但你也可以自己在终端中运行。

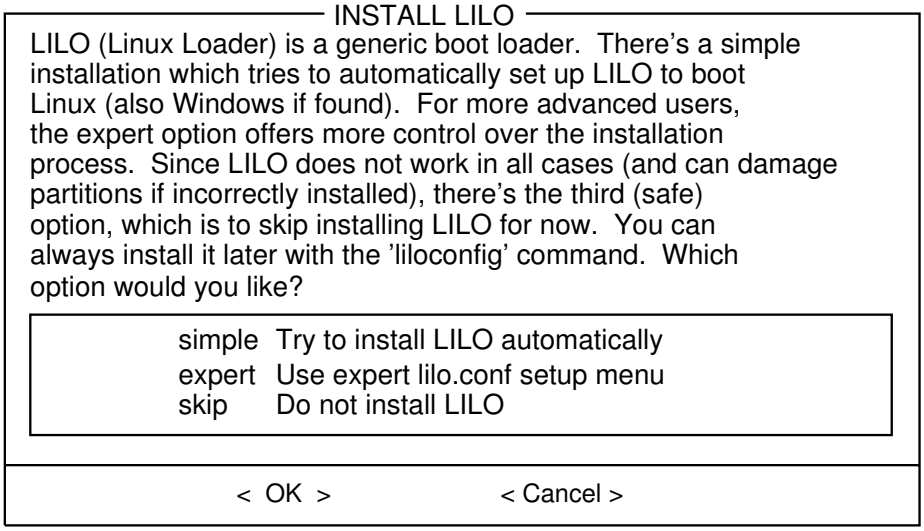

图 7.1: liloconfig

运行liloconfig有两个模式：简单级和专家级。“简单级”可以自动为你完成设置。如果Slackware是你机子上唯一的系统，那么使用“简单级”就可以了，通常情况下它都能快速准确地完成任务。同时，它还擅长于检测机子上安装的Windows，并将它的启动项加入/etc/lilo.conf，那么在启动的时候，你就能选择启动哪个系统了。

要使用“专家级”的话，你就要对Slackware的root分区有所了解。如果知道其它Linux系统的root分区在哪的话，你也可以添加它们的启动项，但这个过程并不像你想像中的那样。liloconfig会尝试使用Slackware的内核来启动每个Linux 系统，而这大概不是你想要的行为。庆幸的是，专家级中对Windows分区的设置是一致的。使用专家级的一个提示：记得把LILO装到主引导记录（MBR）中。我们曾建议把引导装载程序装到root分区中并将该分区设置为可引

导的。现在，LILO已经足够成熟，装到MBR上也是很安全的。事实上，安装到MBR上反而更不会遇到问题。

liloconfig对于快速设置引导装载程序而言是绝佳的选择，但如果你想了解它的原理，那么你应该查看LILO的配置文件：位于/etc下的lilo.conf(5)。/etc/lilo.conf被分为几个部分。在顶部，能看见一个“全局”的部分，在这里可以指定LILO安装的位置（一般是MBR）、启动时要显示的图像或屏幕的设置，以及LILO启动默认项的等待时间。下面是我的lilo.conf文件的全局部分，为了显示已经分成不同的部分。

```
1  # LILO configuration file

2  boot = /dev/sda
3    bitmap = /boot/slack.bmp
4    bmp-colors = 255,0,255,0,255,0
5    bmp-table = 60,6,1,16
6    bmp-timer = 65,27,0,255

7  append=" vt.default_utf8=0"
8  prompt
9  timeout = 50

10 # VESA framebuffer console @ 1024x768x256
11 vga = 773
12 .... many more lines ommitted ....
```

想了解LILO的所有选项，你可以查看lilo.conf的man手册。我们在本文档中只讨论最常见的选项。

第一件要注意的事是”boot”行。该行决定了LILO将要安装到什么位置。要将它安装到硬盘的主引导记录（MBR）中，只要在该行填上硬盘的设备条目。本例中，我用的是SATA硬盘，但是以一个SCSI设备名字显示的，即/dev/sda。要将LILO安装到一个分区的引导块中，就要在该行填上该分区的设备条目。例如，如果你要安装到你电脑中唯一的SATA硬盘中的第一个分区，相应的设备条目为/dev/sda1。

“prompt”选项只是为了告诉LILO，给出一个提示，让你选择要启动的系统。要启动的系统在文件的后面会列出，一个系统占一个段。我们过一会再说明。“timeout”选项告诉LILO在启动默认OS之前，应该等待多长时间（以十分之一秒计算）。我的例子中，等待时间为5秒。有些系统要花很长的时间才能显示启动屏幕，所以你应该设置比我更长的等待时间。这就是为什么简单级的LILO安装设置了一很长的等待时间（大概是整整两分钟）。“append”行是由liloconfig设置的。你在查看（最好看一下）你自己的/etc/lilo.conf时，应该能看到类似的东西。对这行我们不深入讨论，所以你只要相信存在即合理。

我们已经浏览了全局部分，现在应该看看操作系统部分。每个Linux操作系统段都以一个“image”行开始。Microsoft Windows操作系统用一个“other”行来指定。让我们看看一个即

有Slackware也有Microsoft Windows的/etc/lilo.conf的样本吧。

```
# LILO configuration file
... global section ommitted ....
# Linux bootable partition config begins
image = /boot/vmlinuz-generic-2.6.29.4
  root = /dev/sda1
  initrd = /boot/initrd.gz
  label = Slackware64
  read-only
# Linux bootable partition config ends
# Windows bootable partition config begins
other = /dev/sda3
  label = Windows
  table = /dev/sda
# Windows bootable partition config ends
```

对于像Slackware的Linux操作系统，“image”行用于指定采用哪个内核启动。本例中，我们使用/boot/vmlinuz-generic-2.6.29.4这个内核。剩下的段看看字面就知道意思了。它们告诉LILO从哪去找root文件系统，用哪个（如果有的话）initrd，以及开始挂载root文件系统时用只读权限。“initrd”行对于用generic内核的用户，或是用到LVM或软件RAID的用户而言是十分重要的。因为它告诉LILO（及内核）在哪可以找到用mkinitrd生成的initrd文件。

一旦设置好了/etc/lilo.conf，只要运行lilo(8)就可以安装了。不像GRUB及其它的引导装载程序，只要对配置文件作了改变，LILO就要求重新执行lilo命令，否则新的（改变后的）引导装载程序镜像是不会安装的，也因此所做的修改不会得以表现。

```
darkstar:~# lilo
Warning: LBA32 addressing assumed
Added Slackware *
Added Backup
6 warnings were issued.
```

在执行lilo时你会看到很多警告信息，不要害怕。只要看到的不是一个致命错误（fatal error），那都没事。特别的，如果看到一个LBA32地址警告是很正常的。

第八章　Shell

在图形环境下，程序包含了窗口、滚动条、菜单等等，这些程序为用户提供了图形接口。在命令行环境下，用户接口则是由shell提供的。shell的功能就是解释命令，一般地，有了shell，一些东西就变得可用了。在登陆后（本章中涵盖了该内容），用户会进入一个shell，并可以在shell中完成他们的工作。本章会对shell作一个介绍，并介绍Linux用户中使用最多的一个shell—the Bourne Again Shell（bash）。有关本章内容的更多细节，请参见见`bash(1)`的man手册。

8.1　用户

8.1.1　登陆

那么，我们启动了系统，我们会看到诸如下面的内容：

```
Welcome to Linux 2.4.18
Last login: Wed Jan
1 15:59:14 -0500 2005 on tty6.
darkstar login:
```

嗯……貌似没人说过应该怎么登陆。还有，darkstar是什么东东？darkstar只是我们计算机的名字，而darkstar就是默认的名字。如果在安装过程中我们对计算机名进行了设置（在netconfig中设置的），那么我们看到的就不是darkstar而是我们自己设置的名字了。

至于登陆嘛……如果这是第一次启动，那么我们应该使用`root`登陆，输入用户名后会提示输入密码，如果在安装过程中设置了`root`用户的密码，那么这时候只要输入那个密码就可以了。如果当时没有设置密码，那么输入回车即可。这就OK了—我们登陆了。

8.1.2　root:超级用户

OK，那么谁是或者说什么是`root`？在我们的系统中它又和用户帐号有什么联系呢？

好的，在Unix及一些类似的操作系统（如Linux）的世界中，先有的是用户，然后还是用户。之后我们会详细介绍，现在我们要知道`root`是一个用户，且是在其它所有用户之前就存在的一个用户；`root`是全知的全能的，没有用户可以违抗`root`。这是不允许发生的情况。`root`被称为超级用户，它就是这么牛B。但最重要的是，`root`就是你自己。

是不是很嚣张？

不知道？那我们告诉你，是的，它就是很嚣张。但要注意的是，root的设计使它能破坏一切它想破坏的东西。你可能希望先跳到第节，学习如何创建一个普通用户，并用普通用户登陆及工作。传统的用法是，只在绝对需要的情况下才切换到超级用户，以此来保证破坏系统的最小可能性。

顺带一提，如果你先用其它用户登陆了，但想临时切换到root用户也是没问题的。只要使用su(1)命令即可。它会要求你输入root用户的密码，正确的话，它会让你拥有root权限，直到你用exit或logout退出为止。当然，使用su命令可以让你临时以任何用户登陆，只要知道密码即可。

注意：root用户可以su到任意用户，而且不需要输入密码。

8.2 命令行

8.2.1 运行程序

不运行程序，几乎就做不成任何事，你可能会用你的电脑支撑一些东西；或着用来挡着门，好让门一直开着；一些电脑在运行时还会发出可爱的嗡嗡响，但这并不是我们所指的“做成事情”。我们可以相信，把计算机当作嗡嗡响的制门器并不是计算机倍受喜爱的原因。

so，还记得Linux中几乎所有的东西都是用文件表示吗？好吧，这个规则对于程序也适用。每个我们运行的（非shell内置的）程序都指向存在于某个地方的文件。只要指定该文件的完整路径即可运行对应的程序。

例如，还记得上节中介绍的su命令吗？它实际位于/bin目录中：输入/bin/su就可以正确运行它。

既然要输入完整路径，那为什么我们只输入su就能运行呢？毕竟我们没有说它就在/bin目录下，它也可能在/usr/local/share目录下对吧？那么shell是如何知道它的位置呢？这个问题的根源就在于环境变量PATH中；多数的shell要么有环境变量PATH，要么有一此和PATH功能类似的变量。它包含一个列表，列表中的每一项就是我们可能运行程序的目录。所以我们执行su命令时，shell会逐一搜索PATH中的目录列表，在每个列表中寻找一个名为su的可执行文件，遇到第一个su后，就运行它。在没有指定程序的完整路径时，shell就会按如上的方法进行搜索，所以如果你碰到“Command not found”错误，那就意味着你运行的程序所有的目录不在PATH中（当然，也可能是这个程序压根就不存在）。我们会在第//TODO: ref Section 8.3.1节中对环境变量进行深入介绍。

还记得“.”代表当前目录吗？所以如果你恰好就在/bin目录下，那么输入./su就代表了完整的路径，所以也能正常运行。

8.2.2 通配符匹配

几乎所有的shell都会识别一些特定的字符，用来表示可以使用任意内容来替换当前位置的内容。这些字符就被称作通配符（wildcard），最常用的两个就是“*”和“?”。根据约定，?通常代表能匹配任意单个字符。例如，假设我们所在的目录下有三个文件：ex1.txt、

ex2.txt及ext3.txt，我们想把所有的文件复制到另一个文件夹下（使用在将在第//TODO: ref Section 10.5.1节中介绍的cp命令），假设是/tmp。那么一个直观的方法是使用命令

```
cp ex1.txt ex2.txt ex3.txt /tmp
```

但这个命令要输入的东西太多了。我们可以使用上面介绍的?通配符，那么输入

```
cp ex?.txt /tmp
```

就可以了。上面例子中，?会匹配字符“1”、“2”、“3”，这三个字符会轮流替换上述命令中的?。

现在呢？是不是觉得要输的东西还是太多呢？是的。这是很令人震惊的，我们已经有劳动法来保护我们，却还要做这么我的工作。幸运的是，我们还有一个通配符——“*”。正如我们已经说过的，*可以匹配“任意多个任意的字符”。包括0 个字符。所以如果上面提到的三个文件是文件夹下仅有的三个文件，那么我们可以使用如下命令就能一次性搞定。

```
cp * /tmp
```

然而，如果文件夹下还有一个ex.txt文件和一个hejaz.txt文件，我们希望复制ex.txt文件但不希望复制hejaz.txt，那么只要使用命令cp ex* /tmp即可。

而cp ex?.txt /tmp命令只会匹配最先提到的那三个文件，而ex.txt中并没有额外的字符可以匹配?，所以不会被匹配。

另一个常用的通配符是中括号对“[ ]”。在进行匹配时，中括号中出现的任意字符都可以替换到中括号所在的位置。听起来比较乱，举个例子：现在我们有一个文件夹，文件夹中有下面8个文件：a1,a2,a3,a4,aA,aB,aC,aD，现在我们要列出以数字结尾的文件，那么使用“[ ]”就可以完成这个任务。

```
darkstar~% ls a[1-4]
a1 a2 a3 a4
```

但如果我们想要的只是a1,a2和a4呢？前面的例子中，我们用-来表示从1到4的数字，我们也可以使用逗号来分隔单独的项。

```
darkstar~% ls a[1,2,4]
a1 a2 a4
```

我猜现在你在想的是，“如何匹配字符呢？”。Linux中是区分大小写的，这意味着a与A是不同的字符，除了在我们脑袋之外，计算机是不认为它们之间有什么关联的。要注意的是大写的字母在字母表中位于小写字母的前面。所以A和B出现在a各b的前面。继续之前的例子，如果我们想显示的文件名是a1及A1，也可以快速地用“[ ]”实现。

```
darkstar~% ls [A,a]1
A1 a1
```

注意，如果我们不是用逗号而是用横杠（‘-’），那么得到的就是错误的结果了

```
darkstar~% ls [A-a]1
A1 B1 C1 D1 a1
```

我们也可以结合使用横杠和逗号来表示一个串。

```
darkstar~% ls [A,a-d]
A1 a1 b1 c1 d1
```

8.2.3 重定向输入输出及管道

（接下来的东西就相当有意思了）

```
darkstar~% ps > blargh
```

知道这是什么吗？这是我在运行ps命令来查看当前运行的进程。ps命令会在第//TODO:Section 11.3节中进行介绍，这不是有意思的部分，有意思的是> blargh，它的意思是将ps命令的输出写到一个名为blargh文件中。但先等等，下面的东西更有意思。

```
darkstar~% ps | less
```

这个命令将ps命令的输出通过管道送给less，这样我就可以在我想滚动的时候滚动屏幕了。

```
darkstar~% ps >> blargh
```

这是第三个最常用的重定向，它的作用与“>”一致，只是，当blargh文件存在时，“>>”会将ps命令的输出追加到blargh文件中；如果不存在，它的作用与“>”一致，会先创建文件，再将内容写入文件。（“>”会擦除blargh文件的所有内容。）

还有一个重定向符：“¡”，它的作用是从该符号之后指定的位置读取输入，相比之下用得不多。

```
darkstar~% fromdos < dosfile.txt > unixfile.txt
```

如果我们学会使用管道，很多东西就变得很有意思：

```
darkstar~% ps | tac >> blargh
```

上面这个命令会先运行ps，将输出的每一行反转后追加到文件blargh中。你可以按自己的需要将一些命令组合进来，只要记得它们是从左往右解释的就行了。

关于重定向的更多知识，请参见bash(1)的man手册。

8.3 The Bourn Again Shell (bash)

8.3.1 环境变量

Linux系统是个复杂的畜牲，在运行的时候需要记录很多东西，在与其它程序正常交互的时候需要用到很多的细节（有些你可能从来都没意识到）。没有人想在运行程序的时候为程序传递一堆参数，告诉程序要用什么样的终端、计算机的主机名是什么，要使用什么样的提示信息……

就好比是一个自动复制的机器一样，我们把它叫作环境变量。所有的Shell都用环境变量来记录信息，来方便用户的使用。所谓的环境变量只是一堆信息的缩写，用户可能希望记录这些信息以便之后使用。例如，环境变量PS1告诉Shell如何显示提示符，另外一些变量可能是用来告诉程序如何运行的。这里，我们会对bash 提供的处理环境变量的一些命令进行简要介绍。

`set`在单独使用时会显示当前定义的所有环境变量。就像多数bash内置的命令一样，它还能完成一些其它的任务（加参数），关于这些内容，请自己查阅`bash(1)`的相关文档。下面的例子中就是在作者机子上单独运行`set`使得得到结果的一个摘录。注意到我们之前提到的`PATH`环境变量。在`PATH`变量的目录中的程序只要输入文件名就可以运行了，而不需要完整路径。

```
darkstar~% set
PATH=/usr/local/lib/qt/bin:/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:
/usr/openwin/bin:/usr/games:.:/usr/local/ssh2/bin:/usr/local/ssh1/bin:
/usr/share/texmf/bin:/usr/local/sbin:/usr/sbin:/home/logan/bin
PIPESTATUS=([0]="0")
PPID=4978
PS1='\h:\w\$ '
PS2='> '
PS4='+ '
PWD=/home/logan
QTDIR=/usr/local/lib/qt
REMOTEHOST=ninja.tdn
SHELL=/bin/bash
```

```
darkstar~% unset VARIABLE
```

`unset`会移除环境变量，包括环境变量的值及环境变量本身，`bash`会忘记该变量的存在（不要害怕，除非是你在某个shell会话中单独定义的变量，否则其它会话里变量还是会重新定义的，那就是默认的一些变量）。

```
darkstar~% export VARIABLE=some_value
```

使用`export`真的很方便。它可以为某个环境变量`VARIABLE`赋值为“some_value”，如果环境变量`VARIABLE`之前不存在，那么执行完上述命令后它就存在了。如果环境变量`VARIABLE`之前

有其它值，那么现在也被删除，替换为新的值。当然，如果我们只是想在PATH中加入一个新的文件夹，那么这个特性就不太适合了，这个情况下，我们可能会采用如下命令：

```
darkstar~% export PATH=$PATH:/some/new/directory
```

注意$PATH这里的用法：当我们希望bash解释一个变量时（即得到它原有的值），在变量名前加上$符即可。例如echo \$PATH就会将PATH中的值显示出来，如我的例子中：

```
darkstar~% echo $PATH
/usr/local/lib/qt/bin:/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:
/usr/openwin/bin:/usr/games:.:/usr/local/ssh2/bin:/usr/local/ssh1/bin:
/usr/share/texmf/bin:/usr/local/sbin:/usr/sbin:/home/logan/bin
```

8.3.2 Tab补全

（这个东西又是很有意思的）

1. 命令行接口意味着要输入很多内容。
2. 输入是一项工作
3. 没人喜欢工作

由第3点和第2点，我们可以得到第4点：没人喜欢输入。幸运的是，bash作了一些设计，防止第5点（没有喜欢命令行接口）的发生。

那么我们会问：bash是怎么完成这项壮举的呢？除了之前我们介绍的通配符匹配，bash还有一个特性：tab补全。

Tab补全的工作原理如下：我们输入一个文件名，它可能在你的PATH中，也可能由我们手工输入完整路径。不论是哪种情况，我们要做的只是输入足够的内容使我们能唯一定位某个文件，之后按下tab键就可以了。bash会为我们完成剩下的输入。

当然，还得用例子来解释。假设/usr/src目录下有两个子文件夹：/usr/src/linux和/usr/src/sendmail。现在我们想知道/usr/src/linux目录下有什么，所以我只要输入ls /usr/src/l，之后按TAB键，bash就会补全得到ls /usr/src/linux。

现在，假设我们有两个目录/usr/src/linux和/usr/src/linux-old，如果我们输入/usr/src/l后键入TAB，bash会为我们补全尽可能多的内容，因此我们会得到/usr/src/linux，当然，到这点我就可以停下了，或者我也可以再敲下TAB键，bash会显示一张列表，列出与我当前输入内容匹配的所有目录的列表。

从今往后，我们就不要输入这么多内容了（也因此，人们喜爱命令行接口）。是吧，我说过这个东西很有意思的。

8.4 虚拟终端

想像这样一种情况，你在做着一件事，但突然决定要先做另一件事。现在我们必须先停下

当前的工作，并切换到准备完成的任务。但还记得Linux是多用户系统吗？也就是我们想同时登陆多少次都可以的对吗？所以为什么我们只能一次做一件事呢？

的确没必要，我们不可能都为同一个机器配备好几个键盘、鼠标，更关键的是我们中的多数人也不想要这么多东西。这就很明了了，增加硬件并不是我们要的答案。那么就剩下软件了，Linux已经在这方面做了一定的工作，为我们提供了“虚拟终端(virtual terminals)”或简称“VTs”。

按下Alt及功能键（F1～F12），就能在虚拟终端中进行切换，每个功能键对应一个虚拟终端。Slackware默认提供了6个VT，按下Alt+F2就可以切换到第二个虚拟终端中，Alt+F3对应第三个，以此类推。

剩下的功能键为X会话保留，每个X会话使用自己的VT，从第7个（Alt+F7）往上。在X中，Alt+Fn键会被捕捉的，因此，在X中使用Ctrl+Alt+Fn进行切换。所以如果我们在X中，想切换回文本终端（而不退出X会话），那么如Ctrl+Alt+F3会切换到第三个终端中。（如果使用X会话，那么Alt+F7会再切回X。）

8.4.1 Screen

但如果在没有虚拟终端的情况下呢？怎么办？幸运的是，Slackware还包括了一个很好的屏幕管理器，而名字就叫作`screen`。`screen`是一个终端模拟器，有着与虚拟终端近似的功能。执行`screen`会闪出一个简短的介绍，之后就进入一个终端。与传统虚拟终端不同的是，`screen`有自己的命令。所有的`screen`命令都要先按下Ctrl+A组合键。例如，Ctrl+A+C的作用是创建一个新的终端会话，Ctrl+A+N的作用就是切换到下一个终端，对应的Ctrl+A+P会切换到上一个终端中。

`screen`还提供了从`screen`会话中分离及重新附加到某一会话中的功能，这对于通过`ssh`及`telnet`（之后会介绍）进行的远程连接来说异常有用。执行`screen -r`命令会列出当前运行的可以重新附加到的会话。

```
darkstar~% screen -r
There are several suitable screens on:
     1212.pts-1.redtail (Detached)
     1195.pts-1.redtail (Detached)
     1225.pts-1.redtail (Detached)
     17146.pts-1.sanctuary (Dead ???)
Remove dead screens with ' screen -wipe' .
Type "screen [-d] -r [pid.]tty.host" to resume one of them.
```

运行`screen -r 1212`会重新附加到表中的第一个screen。我之前提到过它对于远程连接的优势。如果我们使用`ssh`登陆到一个远程的Slackware服务器上，现在我们的连接可能由于本地机器断电等原因而断开，那么无论我当前在做什么工作，都会瞬间毁灭，这可能对服务器也很伤。如果使用`screen`，那么在连接断开时，它会将当前的会话从screen中分离开，一旦连接恢复，就可以重新附加到之前的screen会话中，并从断开处继续工作。

第九章　文件系统结构

我们已经讨论过Slackware的目录结构。现在，我们已经能够找到我们想找的文件或目录。但文件系统的内容不仅仅是目录结构。

Linux是一个多用户的操作系统，系统的每个部分都是多用户的，即使是文件系统也是如此。系统存储了诸如文件的拥有者、能读取文件的用户等等信息。还有其它一些关于文件系统的独立部分，如链接及挂载NFS。本节中会对上述内容以及文件系统的其它方面进行解释。

9.1　所有权

文件系统为系统中的每个文件及目录保存了所有权信息。它包括哪个用户及用户组拥有一个特定的文件。查看这些信息的最简单的方法是通过`ls`命令：

```
darkstar~% ls -l /usr/bin/wc
-rwxr-xr-x 1 root bin 7368 Jul 30 1999 /usr/bin/wc
```

现在我们感兴趣的是第三和第四列。它们包含了拥有该文件的用户及用户组的信息。我们可以看到，用户“`root`”及用户组“`bin`”拥有该文件。

使用`chown(1)`命令（它代表“change owner”——改变所有者）及`chgrp(1)`命令（它代表“change group”——改变所有者），我们可以很容易地改变文件的所有者和所有的用户组。要将文件的所有者改为`daemon`，我们可以使用`chown`：

```
# chown daemon /usr/bin/wc
```

要将文件的所属用户组改为“`root`”，我们可以使用**chgrp**：

```
# chgrp root /usr/bin/wc
```

我们也可以同时用`chown`指定文件的用户及用户组：

```
# chown daemon:root /usr/bin/wc
```

上面的例子中，用户也可以使用点号（‘.’）来代替冒号（‘:’），得到的结果是一样的，但我们认为冒号是更好的形式。现在已经不再鼓励使用点号，在今后的`chown`版本中可能会移除

这种形式，以支持带点号的用户名。带点号的用户名在Windows Exchange Server上非常流行，在email地址上也较常见，如mr.jones@example.com。在Slackware中，管理员一般避免使用这样的用户名，因为有一些脚本可能还是使用点号来分隔文件或目录的用户名和用户组。在我们的例子中，email地址就会被chown解释为用户名为“mr”，用户组为“johns”。

文件的所有权是使用Linux系统中极为重要的一部分，即使你是使用这个系统的唯一用户。有时我们需要修复文件或设备节点的所有权。

9.2　权限

多用户文件系统的另一个重要方面就是文件或目录的权限。有了权限的概念，我们就可以控制哪个用户可以读取、写入、执行某个文件。

文件的权限信息是用四个八进制位保存的，每个八进制位指定了一个不同的权限集合，集合包括：所有者权限、所有组权限及其它用户权限。第四个八进制位是用来保存特殊的信息，如SetUID位、SetGID位及粘着位（sticky bit）等。要修改文件或目录的权限，对应的权限模式如下（可以用数字指定也可以用字符指定，这些字符与ls的输出相同，也可以使用于chmod中。）：

权限类型	八进制值	对应字母
“sticky” bit （粘着位）	1	t
SetUID 位	4	s
SetGID 位	2	s
读权限（read）	4	r
写权限（write）	2	w
执行权限（execute）	1	x

表 9.1: 八进制权限值

对于每个权限组，都可以使用八进制值指定权限。例如，如果你希望一个权限组的权限为可读写，那么，可以将该组的权限设置为“6”。

bash的默认权限为：

```
darkstar~% ls -l /bin/bash
-rwxr-xr-x 1 root bin 477692 Mar 21 19:57 /bin/bash
```

注意到该行最前面的一位为‘-’，代表普通文件，如果是目录，则第一位为‘d’。之后列出的是三个用户权限组（所有者、所有组及其它用户组）。我们可以看到，所有者的权限为可读、可写、可执行（rwx）。所有组权限的权限为可读、可执行（r-x）。其它用户的权限为可读、可执行（r-x）。

那么如何设置一个文件的权限，让它与bash的权限一致呢？首先，我们创建一个文件example：

```
darkstar~% touch /tmp/example
darkstar~% ls -l /tmp/example
-rw-rw-r--- 1 david users 0 Apr 19 11:21 /tmp/example
```

我们使用chmod(1)命令（意思是“change mode”，即改变模式）来设置文件example的权限。将目标权限的八进制数字赋给文件即可。如我们希望所有者的权限为可读、可写、可执行，我们得到$4+2+1=7$，所有组及其它用户的权限为可读、或执行，我们得到$4+1=5$将这些数字结合起来传递给chmod，运行命令如下：

```
darkstar~% chmod 755 /tmp/example
darkstar~% ls -l /tmp/example
-rwxr-xr-x 1 david users 0 Apr 19 11:21 /tmp/example
```

现在你可能会想，为什么不在创建的时候就把文件的权限设置为现在这样呢？其实解决方法也很简单。**bash**内置了一个很漂亮的小程序，叫作umask。在多数Unix Shell中也包含这个命令。要适应umask 需要一小段时间。它与chmod的工作方式相似，但是以相反的方式完成的。你可以使用umask为新创建的文件设置那些你不想让文件拥有的权限。默认权限为0022。以此为例，0022代表不希望赋予文件所有组的可写权限及其它用户的可写权限，还记得之前bash的权限为755，完整权限是777，那么umask为$777-555=022$.

```
% umask
0022
% umask 0077
% touch tempfile
% ls -l tempfile
-rw-------- 1 david users 0 Apr 19 11:21 tempfile
```

参见**bash**的man手册以获取更多信息。

要使用chmod设置一些特殊的权限，在第一列中加上权限数值。例如，要设置SetUID位及SetGID位，就在权限数值的第一列处加上6：

```
% chmod 6755 /tmp/example
% ls -l /tmp/example
-rwsr-sr-x 1 david users 0 Apr 19 11:21 /tmp/example
```

如果你被这些八进制数搞晕了，chmod还允许使用字母来表示权限，权限表示如下：

下面我们使用字母来完成上面的任务：

```
% chmod a+rx /tmp/example
% chmod u+w /tmp/example
% chmod ug+s /tmp/example
```

权限组	对应字母
Owner（所有者）	u
Group（所有组）	g
World（其它用户）	o
All of the above（以上所有）	a

表 9.2: 字母表示的权限组

一些用户更喜欢使用字母来设置权限。两种方法的结果都是一致的。

八进制的方法更为迅速，我们经常在shell脚本中看到的也是这种模式。然而有时字母模式更为强大。例如，如果想只改变其中一个权限组的权限而保持其它组不变，使用八进制数是很困难的。而使用字母模式就很平常。

```
% ls -l /tmp/
-rwxr-xr-x 1 alan users 0 Apr 19 11:21 /tmp/example0
-rwxr-x--- 1 alan users 0 Apr 19 11:21 /tmp/example1
----r-xr-x 1 alan users 0 Apr 19 11:21 /tmp/example2
% chmod g-rwx /tmp/example?
-rwx---r-x 1 alan users 0 Apr 19 11:21 /tmp/example0
-rwx------ 1 alan users 0 Apr 19 11:21 /tmp/example1
-------r-x 1 alan users 0 Apr 19 11:21 /tmp/example2
```

我们在上面提到好几次SetUID及SetGID权限，你可能在想这究竟是什么东西。一般来说，在运行一个程序时，系统使用的是你的用户帐号，也就是程序的权限与用户所有的权限一致。至于用户组也是一样，在运行一个程序时，使用的是当前的用户组。但如果设置了SetUID权限，你就可以强制总是以程序的所有者（如“root”用户）的权限运行。SetGID也是一样的，只是应用到用户组而已。

要小心使用这个功能，SetUID和SetGID可能会在你的系统的安全之墙上打开一个大洞。如果你将一个root用户的程序设置了SetUID权限，那么就使所有用户能以root运行该程序。由于root在系统中是不受限制的，我们可以看出，经常为root用户的程序设置SetUID及SetGID对系统的安全会造成威胁。简单地说，SetUID和SetGID是好东西，但只按常理使用。

9.3　链接

链接是一种将不同文件名指向同一个文件的方法。使用ln(1)程序，我们就能用多个文件名来引用同一个文件。其中的一个文件并不是另一个文件的复本，它们是同一个文件，只是名字不同罢了。要彻底删除这个文件，就必须删除它所有的引用（rm和其它与之类似的软件实际的原理就是如此。它们只不过是将文件引用删除了，释放那部分空间，而不是删除文件的内容。ln会为一个文件创建一个新的引用或称为“链接(link)”）。

```
darkstar:~$ ln /etc/slackware-version foo
darkstar:~$ cat foo
Slackware 12.0.0
darkstar:~$ ls -l /etc/slackware-version foo
-rw-r--r-- 1 root root 17 2007-06-10 02:23 /etc/slackware-version
-rw-r--r-- 1 root root 17 2007-06-10 02:23 foo
```

还有另一种链接类型，称为符号链接。与传统链接不同的是，符号链接并不是另一个文件的一个引用，它本身就是一个特殊类型的文件。符号链接可以指向一个文件或文件夹。符号链接的主要优势在于它不仅能指向文件，还能指向文件夹，同时还能跨文件系统操作。生成符号链接使用的参数是[-s]。

```
darkstar:~$ ln -s /etc/slackware-version foo
darkstar:~$ cat foo
Slackware 12.0.0
darkstar:~$ ls -l /etc/slackware-version foo
-rw-r--r-- 1 root root 17 2007-06-10 02:23 /etc/slackware-version
lrwxrwxrwx 1 root root 22 2008-01-25 04:16 foo -> /etc/slackware-version
```

使用符号链接时要记得，如果链接指向的文件被删除了，那么这个符号链接也就失效了；它就只是指向一个早已经不存在的文件。

使用符号链接时要格外注意。有一次，我在一台机器上工作，但这台机器每次备份到磁带时都失败，原因就是我的系统上有两个符号链接，分别在另一个符号链接目录下。备份软件就递归地创建两个符号链接，死循环。现在，一般软件在操作时都会对这种情况进行检查，但我们例子显然就是例外。

9.4　挂载设备

正如之前第4.1.1节中介绍的，计算机中的所有驱动及设备都是一个大的文件系统。不同的硬盘分区、CDROM、U盘及软盘等都存在于同一个目录树下。要将这些设备附加到文件系统以使我们能对它进行访问，我们可以使用`mount(1)`及`umount(1)`命令。

在计算机启动时，一些设备已经被自动挂载了。这些设备在`/etc/fstab` 文件中列了出来。如果你想对任何设备进行自动挂载，都可以在该文件中加入新的条目。而另一些设备，则需要在每次想使用时都运行一次挂载命令。

9.4.1　fstab

首先先看看`/etc/fstab`文件中的内容：

```
darkstar:~# cat /etc/fstab
/dev/hda1       /               ext3        defaults              1   1
```

```
/dev/hda2       /home           ext3       defaults               1   2
/dev/hda3       swap            swap       defaults               0   0
/dev/cdrom      /mnt/cdrom      auto       noauto,owner,ro,users  0   0
/dev/fd0        /mnt/floppy     auto       noauto,owner           0   0
devpts          /dev/pts        devpts     gid=5,mode=620         0   0
proc            /proc           proc       defaults               0   0
```

第一列代表设备名。本例中，是一个分为三个区的硬盘，一个CDROM，一个软驱、及两个特殊的设备。第二列是这些设备的挂载点，这一列除了swap外，其它都必须是目录名。第三列代表设备使用的文件系统类型。对于正常的Linux文件系统，这项为ext3（第三代扩展文件系统）。CDROM的类型则为iso9660，而基于Windows的设备类型为vfat或ntfs-3g。第四列是应用到文件系统的一些选项。不管是什么设备，一般使用默认选项就可以了。然而，只读类型的设备应该使用ro标志。我们可以使用的选项有很多，请参见fstab(5)的man手册以获取更多信息。最后两列的内容是由fsck或其它使用该设备的软件使用的。这部分内容也请参阅man手册。

安装Slackware时，setup程序会自动创建fstab的大部分内容。

9.4.2　mount和umount

将一个新的设备附加到文件系统的过程是很容易的，我们只要使用mount命令和几个选项就行了。如果这个设备已经在/etc/fstab文件中有了条目，那使用mount会更加容易。例如，我希望挂载CDROM，而我的fstab文件如上节所示，那么我只要按如下方式调用mount命令即可：

```
# mount /mnt/cdrom
```

由于在fstab文件中已经有了该挂载点的一个条目，mount就知道如何挂载这个设备。如果我的fstab文件中没有这个设备的条目，那么就要手工为mount指定一些选项：

```
# mount -t iso9660 -o ro /dev/cdrom /mnt/cdrom
```

这个命令的结果与上面fstab的例子相同，但我们还是要对这个命令进行详细解释。-t iso9660选项指定了挂载设备的文件系统类型，本例中，文件类型是iso9660，也就是CDROM一般使用的类型。-o ro告诉mount命令将设备挂载为只读权限。而/dev/cdrom则代表要挂载的设备名称，对应的/mnt/cdrom则为设备的挂载点。

在拔除如软盘、CDROM或其它当前挂载的设备之前，要先卸载它（类似Windows下的“删除硬件”）。这个工作通过umount命令完成。

不要问字母‘n’跑哪去了，我们不会告诉你的。在卸载设备时，umount 的参数既可以是设备名，也可以是挂载点。例如，如果我们想卸载之前例子中的CDROM，那们下面两个命令都能正确工作：

```
# umount /dev/cdrom
# umount /mnt/cdrom
```

9.5　挂载NFS

NFS的全称是网络文件系统（the Network Filesystem）。它并不是实际文件系统中的一部分，但可以用来为挂载的文件系统添加一些部分。

大型Unix环境中通常需要共享软件、主目录文件夹及邮件卷。让不同机器得到同一个复本的问题可以通过NFS得到解决。我们可以使用NFS在所有工作站中共享主目录集合。工作站可以挂载一个NFS共享，就像这些内容就在自己机器上一样。

请参见第5.7.2节及man手册中的内容以得到关于`exports(5)`、`nfsd(5)`及`mountd(8)`的更多内容。

第十章　操作文件及目录

Linux的目标是尽可能地类Unix。传统上，Unix操作系统是面向命令行的。Slackware中的确有图形的用户接口，但命令行仍是控制系统的主要方法。因此，理解一些基本文件管理命令是很重要的。

接下来的几节中，我们会解释常见的文件管理命令并举一些使用它们的例子。还有许多其它的命令，但我们介绍的命令能让你逐渐上手。另外，我们只对这些命令进行简要介绍，要了解这些命令的细节，请查阅命令附带的man手册。

10.1　导航命令：ls、cd及pwd

10.1.1　ls

这个命令是“list”的缩写，它的作用是列出一个文件夹中的文件。如果你熟悉Windows及DOS，那么ls就像是其中的dir命令。如果不带任何参数，ls(1)会列出当前目录下的文件。要查看根目录下的文件，可以使用如下命令：

```
darkstar~% cd /
darkstar~% ls
bin   dev  home  lost+found  mnt  proc  sbin  sys  usr
boot  etc  lib   media       opt  root  srv   tmp  var
```

对于这个输出，多数人的问题是没办法轻易区分哪个是目录，哪个是文件。一些用户喜欢让ls在每个列表项的末尾加上一个类型识别符，如：

```
darkstar~% cd /
darkstar~% ls -FC
bin/   dev/  home/  lost/+found/  mnt/  proc/  sbin/  sys/  usr/
boot/  etc/  lib/   media/       opt/  root/  srv/   tmp/  var/
```

目录名之后会加上一个反斜杠（‘/’），可执行文件名之后会加上一个星号（‘*’）等等。

ls也可以用来显示其它的文件数据信息。例如，要显示一个文件的创建日期、所有者及权限等，可以使用ls显示长列表：

```
darkstar~%ls -l
# ls -l
total 88
drwxr-xr-x   2 root root  4096 Apr 30  2007 bin/
drwxr-xr-x   2 root root  4096 Jun 19 13:56 boot/
drwxr-xr-x  15 root root  5180 Jun 21 21:37 dev/
drwxr-xr-x  78 root root 12288 Jun 21 12:03 etc/
drwxr-xr-x   4 root root  4096 Feb 28  2011 home/
drwxr-xr-x   6 root root  4096 Mar 19  2011 lib/
drwx------   2 root root 16384 Jun 15 21:35 lost+found/
drwxr-xr-x  16 root root  4096 Jun 21 12:04 media/
drwxr-xr-x  10 root root  4096 Sep 26  2006 mnt/
drwxr-xr-x   3 root root  4096 Jun 20 08:25 opt/
dr-xr-xr-x 110 root root     0 Jun 21 20:03 proc/
drwx--x---  11 root root  4096 Jun 23 15:28 root/
drwxr-xr-x   2 root root 12288 Jun 20 08:29 sbin/
drwxr-xr-x   2 root root  4096 Jun 15 22:12 srv/
drwxr-xr-x  13 root root     0 Jun 21 20:03 sys/
drwxrwxrwt   4 root root  4096 Jun 21 09:40 tmp/
drwxr-xr-x  15 root root  4096 Apr  5  2011 usr/
drwxr-xr-x  17 root root  4096 Mar 19  2011 var/
```

假设你想得到一个包含隐藏文件的当前目录的列表，可以使用如下命令：

```
darkstar~% ls -a
./   bin/   dev/  home/  lost+found/  mnt/  proc/  sbin/  sys/  usr/
../  boot/  etc/  lib/   media/       opt/  root/  srv/   tmp/  var/
```

以点号开头的文件（也称为点文件）在运行ls命令时会隐藏，只有在以-a为参数运行ls才可以看到。

在ls的man手册上可以看到它还有许许多多的选项。运行ls的时候别忘了还可以加上选项。

10.1.2　cd

cd命令的作用是改变当前的工作目录。只要输入cd命令，以及想要切换到的目录即可。下面是一些例子：

```
darkstar:~$ cd /bin
darkstar:/bin$ cd usr
bash: cd: usr: No such file or directory
darkstar:/bin$ cd /usr
```

```
darkstar:/usr$ ls
bin
darkstar:/usr$ cd bin
darkstar:/usr/bin$
```

注意到，如果没有前置的斜杠（‘/’），则切换目录时是相对于当前目录进行的。另外，如果运行cd命令是不带任何参数，则会切换到用户的主目录中。

cd命令与其它命令不同，因为它是一个shell内置的命令。这个特点现在对你可能没什么影响，一般而言，这意味着这个命令没有man手册。相反的，你可以使用shell的帮助，如下：

```
darkstar~% help cd
```

它会打印出**cd**命令可用的选项。

10.1.3　pwd

pwd的全称是打印工作目录（print working directory）。它的作用就是显示当前位置。使用pwd命令只要输入**pwd**即可。例子如下：

```
darkstar~% cd /bin
darkstar~% pwd
/bin
darkstar~% cd /usr
darkstar~% cd bin
darkstar~% pwd
/usr/bin
```

10.2　分页程序：more、less及most

10.2.1　more

more(1)命令被称为分页工具。有时，一个命令的输出太大了，一个屏幕不足以显示。而单独的命令并不知道如何控制输出使它适合在一个屏幕上显示。它们将这个任务留给了分页工具。

more命令将其它命令的输出分成几个单独的屏幕，一次输出一个屏幕，直到用户按下空格键才输出下一屏，按回车键则向前显示一行。下面是个很好的例子：

```
darkstar~% cd /usr/bin
darkstar~% ls -l
```

这个命令的输出会在屏幕上滚动一会。要将该输出一屏一屏的输出，只要用管道将它传递给more命令即可：

```
darkstar~% ls -l | more
```

中间的符号为管道符（‘—’），上述管道的含义是将ls的输出作为more的输入。除了ls，还可以用管道将其它几乎所有的东西传递给more命令。管道在第8.2.3节中有介绍。

10.2.2　less

more命令很方便，只是有个致命的缺陷，就是如果翻到下一页就再也不能回到上一页。试想如果你在查看一个很大的文件，不小心错过了某一行，那就不得不重头查找，急煞人也。好在人们开发了less(1)命令，来解决这个问题。它的工作方式与more一样，所以上面的例子在这里也适用。所以less的功能远不止more。Joost Kremers 是这样说的：

> less is more, but more more than more is, so more is less less, so use more less if you want less more.
>
> less与more类似，但比more的功能更强大，所以more的功能反而更少，因此，如果你想更少使用more，那么请更多地使用less。

10.2.3　most

more和less已经停止开发了，而most(1)继承了它们的衣钵。如果说less是more的加强版，那么most就是less的加强版。其它的分页程序一次只能查看一个文件，而most能同时查看任意数量的文件，前提是每个文件至少两行以上。most提供了很多选项，同样，也请自己查阅man手册。

10.3　简单输出：cat及echo

10.3.1　cat

cat(1) 是“concatenate（连接）”的缩写，它最初的设计目的就是将多个文件合并成一个文件，但它同时被用于其它的许多目的。

要想将两个或更多的文件合并成一个，只要将想合并文件的文件名传递给cat命令，并将输出重定向到另一个文件。cat还能与标准输入标准输出协同工作，所以我们需要使用shell的重定向符。照例举个例子：

```
% cat file1 file2 file3 > bigfile
```

这个命令将file1、file2及file3的内容合并在一起，最后的输出传递给标准输出，由于使用了重定向，标准输出的内容最终存储到bigfile文件中。

我们可以使用cat命令来显示文件的内容。许多人都使用cat 将文件输出到more或less命令中进行查看，如：

```
% cat file1 | more
```

这个命令会显示file1文件的内容，并通过管道将它传递给more命令，因此我们就可以一屏一屏地查看该文件。

cat的另一个用法是用来复制文件。例如：

```
% cat /bin/bash > ~/mybash
```

执行该命令后，/bin/bash程序就会被复制到我们的主目录下，并命名为mybash。

cat还有许多用法，这里介绍的只是很小的一部分，由于cat 利用了标准输入及标准输出，因此，它适用于shell脚本中，作为其它复杂命令的一部分。

10.3.2　echo

echo(1)命令的作用是在屏幕上显示特定的文本。我们可以在echo命令之后指定要显示的字符串。默认情况下，echo会显示我们给出的字符串并在之后打印一个换行符。当然，我们也可以为其传递参数-n来告诉echo不打印末尾的新行。-e选项则告诉echo解释字符串中的转义符。

10.4　新建命令：touch及mkdir

10.4.1　touch

touch(1)命令的初衷是修改一个文件的时间戳，使用touch 命令，我们就可以修改一个文件的访问时间戳及修改时间戳。而如果一个文件之前并不存在，则touch会以给定的文件名创建一个空的文件。要将一个文件标记为当前的系统时间，只要使用如下命令即可：

```
% ls -al file1
-rw-r--r-- 1 root root 0 Jun 25 10:42 file1
% touch file1
% ls -al file1
-rw-r--r-- 1 root root 0 Jun 25 10:43 file1
```

touch命令有很多参数，包括选择修改哪个时间戳、修改成什么时间及一些其它的选项。man手册中有详细的介绍。

10.4.2　mkdir

mkdir(1)的作用是创建一个新的目录，使用方法是在mkdir 后加上想要创建的目录名。下面的例子将会在当前目录下创建一个新的名为hejaz的目录：

```
% mkdir hejaz
```

当然，我们也可以指定一个路径，如：

```
% mkdir /usr/local/hejaz
```

-p选项会告诉mkdir创建目录时一同创建它的父目录。例如，上面的例子中，如果/usr/local目录不存在，则这条命令就会执行失败，而加上-p选项，则意味着先创建/usr/local，之后再创建/usr/local/hejaz目录。

```
% mkdir -p /usr/local/hejaz
```

10.5　复制与移动

10.5.1　cp

cp命令（copy的缩写）的作用就是复制文件。DOS用户会发现它与DOS 的copy命令类似。cp命令的选项很多，所以在使用之前还是建议查看它的man手册。这里会介绍一些最常用的选项。

首先，cp最常用的方法就是将一个文件复制到另一个位置。如下例：

```
% cp hejaz /tmp
```

这个命令就是将当前目录下的hejaz文件复制到/tmp目录下。

许多用户在复制时还希望保留文件的时间戳，如下面的例子：

```
% cp -a hejaz /tmp
```

这个命令就保证了复本的时间戳与原文件一致。

如果想复制整个目录，则使用下面的命令：

```
% cp -R mydir /tmp
```

该命令的效果是将mydir目录复制到/tmp目录下。

另外，像前面的例子一样，如果在复制文件或目录的过程中，同时希望保持原文件的时间戳，那么使用cp -p命令。

```
% ls -l file
-rw-r--r-- 1 root vlad 4 Jan 1 15:27 file
% cp -p file /tmp
% ls -l /tmp/file
-rw-r--r-- 1 root vlad 4 Jan 1 15:27 file
```

cp还有许多其它的选项，请自行参阅man手册。

10.5.2　mv

mv命令的作用是将文件从一个位置移动到另一个位置，听起来是不是很简单？

```
% mv oldfile /tmp/newfile
```

mv有一些很有用的参数，在man手册中有详细的描述，但实际使用中一般几乎不怎么用到。

注意，在Linux，并没有重命名的命令，因此，所有的重命令都是通过mv命令实现的。如下面的例子：

```
% mv oldfile newfile
```

该命令的作用就是将当前目录下名为oldfile的文件重命名为newfile。

10.6　删除命令：rm及rmdir

10.6.1　rm

rm(1)的作用是删除文件及目录树。DOS用户会发现它的功能与DOS下的del及deltree命令类似。如果不小心使用，rm可能造成意想不到的后果。尽管有一些方法能还原最近删除的文件，但是方法很复杂（而且代价昂贵），并且不在本书的范围内。

删除一个单独的文件，只要将文件名传递给rm即可：

```
% rm file1
```

如果一个文件没有可写权限，则执行rm时会得到一个提示，说该文件是写保护，是否强制删除。如果想跳过这个提示信息，在执行rm时加上-f选项，如：

```
% rm -rf file1
```

如果想删除整个文件夹，可以联合使用-r及-f选项。下面是一个例子，教你如何删除整个硬盘，**请不要在自己的机器上尝试**。下面贴出代码：

```
# rm -rf /
```

使用rm时一定要小心，不然很可能走火，它也有一些命令行参数，和以往一样，在man手册中有详细介绍。

10.6.2　rmdir

rmdir(1)的作用是删除目录，但在删除之前要确保该目录是空的。该命令的语法如下：

```
% rmdir <directory>
```

下面这个例子的作用是删除当前目录下的hejaz子目录：

```
% rmdir hejaz
```

如果要删除的目录是不存在的，rmdir会提醒你。你也可以手工指定要删除目录的完整路径，如下目录：

```
% rmdir /tmp/hejaz
```

这个命令的作用是将/tmp目录下的hejaz目录删除。

还可以通过传递-p选项，在删除一个目录的同时，删除它的父目录。

```
% rmdir -p /tmp/hejaz
```

这个命令首先删除/tmp下的hejaz目录，如果成功删除，则rmdir会尝试删除/tmp目录，rmdir会递归地执行此操作直到遇到错误或成功删除整个目录树。

10.7　文件别名：ln

ln(1)（link的缩写）的作用是创建文件间的链接。链接分为硬链接或软链接（又称为符号链接）。两种链接的区别在第9.3节中有介绍。例如我们想创建一个到/var/media/mp3的链接，并将其放在主目录下，那么可以执行如下命令：

```
% ln -s /var/media/mp3 ~/mp3
```

-s选项告诉ln要创建符号链接。下一个参数代表链接源的位置，最后一个参数则代表这个链接的名字是什么。这个例子中，我们在主目录下创建了一个名为mp3的文件，指向/var/media/mp3，通过改变第三个参数，你可以为该链接指定任意的名字。

创建一个硬链接则更为容易，只要去掉-s选项即可。但硬链接不允许指向目录，或跨越文件系统（即分区）指定，如创建一个名为/usr/bin/email的指向/usr/bin/mutt的链接，可以输入以下命令：

```
# ln /usr/bin/mutt /usr/bin/email
```

第十一章　进程控制

每个运行的程序(program)都叫作一个进程(process)。从诸如X Window系统到计算机启动时开启的系统程序（守护进程）等属于进程的范畴。每个进行都以一个特定的用户运行。计算机启动时运行的进程一般以root用户或是nobody用户运行。由我们启动的进行以我们自己为用户运行，而别的用户启动的进程则以那些用户运行。

对于那些我们自己启动的进程，我们有着完全的控制。另外，root用户控制着整个系统的所有进程，包括所有由别的用户开启的进程。进行的控制及监视可以通过一些程序及一些shell命令完成。

11.1　后台运行

从命令行运行的程序是从前端运行的。这使得我们可以看见程序的所有输出并与之交互。然而，另外一些情况下，我们可能希望程序在运行时不控制我们的终端。这种行为就叫作后台运行，有好几种方法可以实现后台运行。

第一种方法就是当我们从命令行中启动程序时，在程序之后加上一个‘与’号（‘&’）。例如，我们希望手动执行updatedb来更新slocate的数据库，但由于更新时间较长，我们希望在更新的同时利用同一个终端做一些其它的工作，那么，可以执行下面的命令将该命令后台执行：

```
# updatedb &
```

这个程序会照常运行，当运行结束后正常退出。

另一个后台进行一个进程的方法是在进程运行的过程中对它进行处理。首先，运行某个进程，在它运行的过程中，键入Ctrl+Z。这个组合键会挂起当前进程，所谓挂起就是暂停了进程，只是暂时停止了运行，之后还能再次启动该进程。一旦我们挂起了一个进程，我们就可以通过以下命令来将一个进程转到后台运行：

```
% bg
```

现在挂起的进程就会转入后台运行了。

11.2　前台运行

如果我们需要和一个后台运行的进程进行交互，那么就需要将它带到前台来运行。如果我们只有一个后台运行的进程，那么使用下列命令就能将它带到前台运行：

```
% fb
```

如果这个程序的运行尚未结束，那么它会取得我们终端的控制权，而相应的，我们就进入了与之交互的界面而暂时得不到shell提示符界面。有时，一个程序在后台运行的过程中就结束退出了，这种情况下，我们会得到类似如下的信息：

```
[1]+ Done /bin/ls $LS_OPTIONS
```

这个命令告诉我们在后台运行的进程（本例中为ls——没什么意思的进程）已经结束了。

我们可以同时拥有几个后台进行的进程。在这种情况下，我们就必需知道想带回前台运行的是哪一个进程。只输入fg而不带其它参数只会将最近一个进入后台运行的进程带回前台。那么如果我们在后台运行了一大排进程怎么办？好在bash为我们提供了一个列出当前所有进行的命令，我们称为jobs，它会输出类似下面的信息：

```
% jobs
[1] Stopped          vim
[2]- Stopped         amp
[3]+ Stopped         man ps
```

这个命令显示了后台运行的进程的一个列表。正如我们看到的，后台的所有进程的状态都是stopped。这意味着所有的进程都被挂起了。最前面的数字代表后台运行进程的某种ID，ID后面带‘+’号（本例中为man ps）的意味着如果不带任何参数运行fg，则对应的进程将会被带回前台。

如果我们想将vim带回前台，那么对应上述例子，只需输入：

```
% fg 1
```

vim就会被带回前台终端运行。如果我们只有一个通过拨号连接的终端，那么后台运行的进程就十分有用。通过后台运行，我们就在一个终端中同时运行几个进程，并不时地在它们之间进行切换。

11.3　ps

现在我们已经知道如何在从命令行中启动的进程中来回切换。也正如我们知道的那样，系统中总是有进程自始至终在运行。所以，怎么列出这些程序呢？好的，我们要用到的是ps(1)命令。这个命令有非常多的选项，我们只介绍最常用的一部分。其它的选项请查阅ps的man文件。

只输入ps会得到在当前终端中运行的程序的一个列表。这其中包括当前前台运行的进程（意味着包括我们在使用的shell，及ps本身）。当然还包括当前后台在运行的进程。很多情况下，我们只会得到如下的非常简短的列表：

```
% ps
  PID TTY          TIME CMD
13995 pts/5    00:00:00 bash
16262 pts/5    00:00:00 ps
```

尽管这个输出包含的进行不多，但显示的信息是很典型的。不论当前运行的进程有多少，得到的输出的列是相同的，那么这些列的含义是什么呢？

PID就是process ID，即进程ID。所有运行中的进程都用一个独一无二的标识符来标识，它的范围是1到32767.每个进程都被赋予下个可用的PID。而当一个进程退出（或被杀死时，杀死进程在下节中介绍），它会交出自己的PID。当达到最大的PID时，会返回，选择从0开始的最小可用PID。

TTY列代表该进程所在的终端。不带任何参数的ps只会列出当前终端中运行的程序，所以TTY这一列显示的信息是一致的。正如我们看到的，两个进程都运行在pts/5，这意味着它运行在X中的第5个虚拟终端中。

TIME并不代表进程运行的时间，而是指进程运行过程中占用的CPU时间。还记得Linux是个多任务的操作系统吗？系统中总是有一堆进程在运行着，每个进程都获取一小部分的处理器时间。所以TIME列代表每个进程比它所需的CPU时间少用了多少。如果我们看到CPU这列中的数值为好几分钟，那么意味着什么地方出错了。

最后，CMD列表示我们实际执行的命令。它只列出了程序的名字，而不包括调用的参数及类似的信息，我们过会儿会讨论。

只查看我们的进程没什么意思，所有让我们用[-e]参数来看看系统中运行的所有程序。

```
darkstar:~$ ps -e
  PID TTY          TIME CMD
    1 ?        00:00:00 init
    2 ?        00:00:00 kthreadd
    3 ?        00:00:00 migration/0
    4 ?        00:00:00 ksoftirqd/0
    7 ?        00:00:11 events/0
    9 ?        00:00:01 work_on_cpu/0
   11 ?        00:00:00 khelper
  102 ?        00:00:02 kblockd/0
  105 ?        00:01:19 kacpid
  106 ?        00:00:01 kacpi_notify
... many more lines omitted ...
```

上面的例子中使用了标准的ps语法，但是如果我们用BSD语法的话，可以看到到多的信息。只要使用[aux]参数就能做到。

注意 [aux]参数与[-aux]是不同的，但多数情况下两个参数是等价的。这是N年前的遗留问题了。请查看ps的man手册以了解更多内容。

```
darkstar:~$ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   3928   632 ?        Ss   Apr05   0:00 init [3]
root         2  0.0  0.0      0     0 ?        S    Apr05   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    Apr05   0:00 [migration/0]
root         4  0.0  0.0      0     0 ?        S    Apr05   0:00 [ksoftirqd/0]
root         7  0.0  0.0      0     0 ?        S    Apr05   0:11 [events/0]
root         9  0.0  0.0      0     0 ?        S    Apr05   0:01 [work_on_cpu/0]
root        11  0.0  0.0      0     0 ?        S    Apr05   0:00 [khelper]
... many more lines omitted ....
```

正如你所看到的，BSD语法提供了更多的信息，包括哪个用户在使用某个进程，及在ps运行的时候，进程占用的内存及CPU的百分比。

其中，有一些进程的tty为“?”，这代表它们并没有附加到特定的终端中，这对于一些守护进程而言最为常见，守护进程在进行时并不附加到特定的终端中，常见的一些守护进程有sendmail、BIND、apache及NFS等。典型地，它们从一个客户端中监听某些请求，并在接到请求时返回相应的信息。

另外，可以看到一个新列：STAT。它代表进程的状态。其中S代表进程处理睡眠状态：这个进程在等待某些事件发生。Z 代表进程为僵尸进程。僵尸进程指的是一个进程的父进程已经死亡，而子进行仍活着。当然这不是什么好事。D代表进行已经进入了不可打断的睡眠状态中。一般而言，这些进程即使使用SIGKILL信号都无法杀死。下一节对kill的介绍中有关于SIGKILL的更多内容。W代表分页。一个死去的进程被标记为X。而一个标记为T的进程代表正在被追踪或已经停止了。R则代表访进程是可运行的。

最后，ps还能生成进程树。进程树可以显示什么进程有子进程。结束一个父进程的同时也会结束相应的子进程。我们用[-ejH]参数来查看。

```
darkstar:~$ ps -ejH
... many lines omitted ...
 3660  3660  3660 tty1     00:00:00   bash
29947 29947  3660 tty1     00:00:00     startx
29963 29947  3660 tty1     00:00:00       xinit
29964 29964 29964 tty7     00:27:11         X
29972 29972  3660 tty1     00:00:00         sh
29977 29972  3660 tty1     00:00:05           xscreensaver
29988 29972  3660 tty1     00:00:04           xfce4-session
```

```
29997 29972  3660 tty1     00:00:16             xfwm4
29999 29972  3660 tty1     00:00:02             Thunar
... many more lines omitted ...
```

正如你看到的，不管是查看系统中当前活动的进程，还是了解进程的更多信息而言，ps都是一个极其强大的工具。

11.4　kill

有时候，程序的行为与我们的预期不符，这时候我们就要去矫正它。用于这方面管理任务的程序名为kill(1)，它能以许多方式控制进程的行为，最明显的一种方式当然就是杀死一个进程。例如一个程序已经失控、用尽了系统资源或者就是不想让它运行时，我们就需要将进程杀死。

要杀死一个进程，就必需知道进程的PID或进程的名字。要知道进程的PID，可以使用我们上一节讨论过的ps命令。例如，要杀死进程号为4747的进程，可以使用如下命令：

```
% kill 4747
```

注意，要杀死一个进程，前提就是我们要是这个进程的所有者。这是一个安全性特征。如果我们允许杀死其它用户的进程，那么人们就可以做许多恶意的事。当然了，root用户可以杀死系统中的任意进程。

kill命令还有另一个变种，名为killall(1)。这个命令的行为和名字一致：作用是杀死给定名称的所有运行中的进程。例如，想要杀死所有名为vim的运行，只需要输入下面的命令：

```
% killall vim
```

所有我们所拥有的vim进程都会停止。而以root用户运行这个命令则会杀死所有用户运行的vim进程。这让我们想到了一种将其它所有用户（包括我们自己）踢出系统的方法：

```
# killall bash
```

有时，一个通常的kill命令并不能完成任务。有些进程只是用简单的kill并不会退出。这时候我们需要一种更强有力的形式来消灭它。假如现在PID为4747的进程就是一个顽固的小强，那么我们可以使用如下的命令：

```
% kill -9 4747
```

绝大多数情况下，这个命令就可以让PID为4747的进程滚蛋了。当然，killall命令也可以这么做。这个命令的独特之处在于它为进程传递了一个不同的信号。通常的kill命令传递的是SIGTERM（终止）信号，这个信号告诉进程：赶快结束手头的工作，清理后就退出吧。而kill -9传递的则是SIGKILL（杀死）信号，直接杀死进程。当收到SIGKILL信号时，进程是不允许进行清理的，有时这个信号有一些副作用，如数据冲突等。有一系列的信号可供我们使用。我们可以使用如下命令得到可用信号的一张列表：

```
% kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

列表中的数字是调用kill时使用的参数，而后面的名字去掉“SIG”后可以用在killall命令中。下面是另一个例子：

```
% killall -KILL vim
```

最后介绍一个重启进程的方法。传递SIGHUP信号会导致多数程序重新读取它们的配置文件。这对于修改系统配置文件后，告诉系统进程重新读取配置文件的情况十分有用。

11.5　top

最后要介绍的命令与之前的不同，它是一个监视系统进程的程序，名为top，启动的方式也很简单：

```
% top
```

这个命令会占用整个屏幕，显示当前系统中的运行的进程的信息，以及系统的统计信息，包括系统平均负载、进程数量、CPU状态、空闲的内存及进程的详细信息等。而进程的详细信息又包括进程的PID、所属用户、优先级、CPU使用信息及内在使用信息、运行时间及程序名。

```
darkstar:~$ top
top - 16:44:15 up 26 days,  5:53,  5 users,  load average: 0.08, 0.03, 0.03
Tasks: 122 total,   1 running, 119 sleeping,   0 stopped,   2 zombie
Cpu(s):  3.4%us,  0.7%sy,  0.0%ni, 95.5%id,  0.1%wa,  0.0%hi,  0.2%si, 0.0%st
Mem:   3058360k total,  2853780k used,   204580k free,   154956k buffers
Swap:        0k total,        0k used,        0k free,  2082652k cached
```

```
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
    1 root      20   0  3928  632  544 S    0  0.0   0:00.99 init
    2 root      15  -5     0    0    0 S    0  0.0   0:00.00 kthreadd
    3 root      RT  -5     0    0    0 S    0  0.0   0:00.82 migration/0
    4 root      15  -5     0    0    0 S    0  0.0   0:00.01 ksoftirqd/0
    7 root      15  -5     0    0    0 S    0  0.0   0:11.22 events/0
    9 root      15  -5     0    0    0 S    0  0.0   0:01.19 work_on_cpu/0
   11 root      15  -5     0    0    0 S    0  0.0   0:00.01 khelper
  102 root      15  -5     0    0    0 S    0  0.0   0:02.04 kblockd/0
  105 root      15  -5     0    0    0 S    0  0.0   1:20.08 kacpid
  106 root      15  -5     0    0    0 S    0  0.0   0:01.92 kacpi_notify
  175 root      15  -5     0    0    0 S    0  0.0   0:00.00 ata/0
  177 root      15  -5     0    0    0 S    0  0.0   0:00.00 ata_aux
  178 root      15  -5     0    0    0 S    0  0.0   0:00.00 ksuspend_usbd
  184 root      15  -5     0    0    0 S    0  0.0   0:00.02 khubd
  187 root      15  -5     0    0    0 S    0  0.0   0:00.00 kseriod
  242 root      20   0     0    0    0 S    0  0.0   0:03.37 pdflush
  243 root      15  -5     0    0    0 S    0  0.0   0:02.65 kswapd0
```

它的名字为top的原因是占用CPU最多的程序会在显示在最上面。注意，由于top对CPU的占用，在一些当前不活动的系统（及一些当前活动的系统）中，top可能会被显示在最前面。然而，top对于查看哪些程序失控及需要被杀死是十分有用的。

但如果我们只想查看自己所有的进程，或者其它用户拥有的进程。我们的进程可能并不在占用CPU的前几名，可能找不到。所以可以使用-u选项来指定要查看的的用户名或用户ID，而只查看相应UID所拥有的所有进程。

```
% top -u ice
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
16843 ice       20   0  303m  51m  23m S    6  2.5  40:34.26 chrome
13877 ice       20   0  454m 361m 317m S    3 18.0  14:16.20 VirtualBox
 2200 ice       20   0 20364 5880 3516 S    1  0.3   0:25.01 wmfs
 2211 ice       20   0  9760 7020 3052 S    0  0.3   0:04.72 python
 2397 ice       20   0  3716 1892 1172 S    0  0.1   0:19.16 tmux
 7783 ice       20   0  2660 1132  836 R    0  0.1   0:00.02 top
 8142 ice       20   0  152m  37m  22m S    0  1.9   1:04.52 okular
 2032 ice       20   0  8484 6660 1340 S    0  0.3   0:00.58 bash
 2167 ice       20   0  3276 1452 1120 S    0  0.1   0:00.00 startx
 2183 ice       20   0  3124  660  564 S    0  0.0   0:00.00 xinit
 2188 ice       20   0  3672  732  616 S    0  0.0   0:00.00 ck-launch-sessi
 2206 ice       20   0 15412 2916 2320 S    0  0.1   1:36.46 ibus-daemon
```

```
2207 ice       20   0  3252 1208  908 S    0  0.1   0:00.00 goagent.sh
2221 ice       20   0 16908 3644 3056 S    0  0.2   0:00.02 ibus-gconf
2227 ice       20   0  127m  24m  14m S    0  1.2   4:13.21 python
2229 ice       20   0 15980 5144 4280 S    0  0.3   0:55.54 ibus-x11
2239 ice       20   0  3636  596  372 S    0  0.0   0:00.00 dbus-launch
2246 ice       20   0  2940  884  664 S    0  0.0   0:00.04 dbus-daemon
2257 ice       20   0  6928 3532 2312 S    0  0.2   0:00.12 gconfd-2
2276 ice       20   0 37252  31m 4440 S    0  1.6   5:27.14 python
2346 ice       20   0 16300 4692 3900 S    0  0.2   0:01.27 xfce4-notifyd
2348 ice       20   0  4000 1800 1588 S    0  0.1   0:00.00 xfconfd
2371 ice       20   0 15524 8928 4192 S    0  0.4   0:02.51 xterm
2377 ice       20   0  5828 4108 1388 S    0  0.2   0:00.25 bash
2395 ice       20   0  3104  972  796 S    0  0.0   0:00.00 tmux
```

可以看到，我运行了top、xterm及一些其它的与X有关的程序，而运行的chrome浏览器是当前最占用CPU的程序。可见，top对于监控系统的资源是十分有效的。

top还支持根据进程的PID、是否空闲、是否僵尸进程及其它许多选项来监视进程，掌握top的最好方法是查看它的man手册页。

11.6　cron

好的，现在我们学会几种查看系统中活动进程的方法，以及如何向它们发送信号。但如果我们想隔一段时间运行一个进程呢？好在Slackware无所不包，crond(8) 就是用来干这事的。cron按用户计划好的内容为每个用户运行程序。这对于需要隔一段时间运行，但又没必要弄成一个守护进程的程序而言十分有用，如备份脚本就是这种情况。每个用户都有自己在cron数据库中的条目，所以普通用户也能以一定间隔执行某个进程。

要使用cron运行进程，首先需要使用crontab(1)。它的man手册页列出了使用它的N种方法，但最常用的方法是使用[-e]参数。这个参数会锁定cron数据库中该用户的条目（防止被其它程序覆盖），之后用环境变量VISUAL中指定的编辑器打开该条目。在Slackware系统上，一般用的是vi编辑器，在继续操作前你可能想先查看一下vi这一章。

cron数据库中的条目看起来可能有点复古，但它们的可定制性很高。除了注释的行外，cron会处理每一行，如果所有的时间条件都满足，cron就会执行相应的命令。

```
darkstar:~$ crontab -e
# Keep current with slackware
30 02 * * * /usr/local/bin/rsync-slackware64.sh 1>/dev/null 2>&1
```

就像之前提到的一样，咋一看，cron条目的语法有点难理解，所以我们一部分一部分看。从左到右看，每部分依次代表：分钟、小时、日期、月份、周几及命令。星号*代表

匹配了每一分钟，每一小时，每一天，每月份等等。所以上面的例子中，要执行的命令为“/usr/local/bin/rsync-slackware64.sh 1>/dev/null 2>&1”，且在每天上午的2:30分时运行。

crond也能通过e-mail将命令产生的任何输出发送给本地用户。因此，许多任务都将它们的输出重定向到了/dev/null，这是一个特殊的设备文件，它会立即抛弃所有接收到的东西。为了让你更容易记住这些规则，你可能希望将下面这条注释后的规则放到你自己的cron条目中。

```
# Redirect everything to /dev/null with:
#   1>/dev/null 2>&1
#
# MIN HOUR DAY MONTH WEEKDAY COMMAND
```

默认情况下，Slackware为root用户的crontab添加了一些条目及相应的注释。它根据运行程序的频率在/etc中创建了几个文件夹，而这些条目让我能更方便地设置一些需要同期性运行的任务。放到这些文件夹中的脚本会以每小时、每天、每周或每个月的频率运行。这些文件夹的名字也很容易看名知意：/etc/cron.hourly, /etc/cron.daily, /etc/cron.weekly以及/etc/cron.monthly。

附录 A　GNU通用公共许可证

待完成