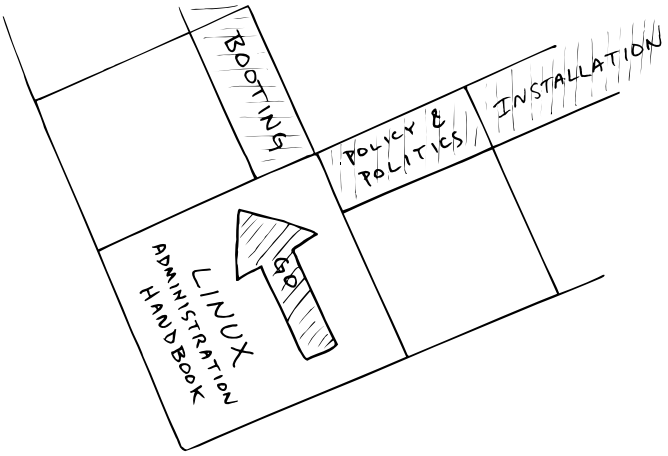
1 Where to Start



We set out to write a book that could be a system administrator's trusty companion, affording the practical advice and basic system administration theory that you can't get from reading manual pages. As a result, this book is designed to complement—not replace—the existing body of Linux documentation.

This book helps you in five ways:

- It reviews the major administrative systems, identifying the different pieces of each and explaining how they work together.
- It introduces general administrative techniques that we have found, through experience, to be efficient and beneficial.
- It helps you choose solutions that continue to work well as your site grows in size and complexity.
- It helps you sort good ideas from bad and educates you about assorted atrocities of taste committed by distributors.
- It summarizes common procedures so that you don't have to dig through the excessive detail of the manuals to accomplish simple tasks.

It's impossible to perform these functions with perfect objectivity, but we think we've made our biases fairly clear throughout the text. One of the interesting things about system administration is that reasonable people can have dramatically different notions of what constitute the most appropriate policies and procedures. We offer our subjective opinions to you as raw data. You'll have to decide for yourself how much to accept and to what extent our comments apply to your environment.

1.1 SUGGESTED BACKGROUND

We assume in this book that you have a certain amount of Linux or UNIX experience. In particular, you should have a general concept of how Linux looks and feels from the user's perspective before jumping into administration. Several good books can get you up to speed; see the reading list on page 19.

You perform most administrative tasks by editing configuration files and writing scripts, so you must be familiar with a text editor. To the dismay of many, using Microsoft Word as one's only text editor is a significant impediment to effective system administration.

We strongly recommend that you learn **vi** (which is seen most commonly on Linux systems in its rewritten form, **vim**). It is standard on all UNIX and Linux systems, and though it may appear a bit pallid when compared with glitzier offerings such as **emacs**, it is powerful and complete. We also like **pico**, which is a simple and low-impact "starter editor" that's good for new sysadmins. It's included in many distributions. Be wary of nonstandard editors; if you become addicted to one, you may soon tire of dragging it along with you to install on every new system.

One of the mainstays of administration (and a theme that runs throughout this book) is the use of scripts to automate administrative tasks. To be an effective administrator, you must be able to read and modify Perl and **sh** scripts (which in the Linux world are really **bash** scripts). Scripts that you write from scratch can be written in the shell or scripting language of your choice.

See cpan.org for a complete selection of useful Perl software.

For new scripting projects, we recommend Perl or Python. As a programming language, Perl is a little strange (OK, more than a little). However, it does include many features that are indispensable for administrators. The O'Reilly book *Programming Perl* by Larry Wall et al. is the standard text; it's also a model of good technical writing. A full citation is given on page 20.

Many administrators prefer Python to Perl, and we know of sites that are making a concerted effort to convert from Perl to Python. Python is a more elegant language than Perl, and Python scripts are generally more readable and easier to maintain. A useful set of links that compare Python to other scripting languages (including Perl) can be found at

www.python.org/doc/Comparisons.html

We also recommend that you learn **expect**, which is not a programming language so much as a front end for driving interactive programs. You will most likely pick up **expect** quite rapidly.

1.2 LINUX'S RELATIONSHIP TO UNIX

Using the names Linux and UNIX together in one sentence is like stepping into a political minefield, or perhaps like blundering into a large patch of quicksand. Here is our short version of the facts, stated as clearly and objectively as we can make them.

Linux is a reimplementation and elaboration of UNIX. It conforms to the POSIX standard, runs on several hardware platforms, and is compatible with most existing UNIX software. It differs from most other variants of UNIX in that it is free, open source, and cooperatively developed, with contributions having come from thousands of different individuals and organizations. Linux incorporates technical refinements that did not exist in the original versions of UNIX, so it is more than just a UNIX clone. It is also a legally distinct entity and cannot be properly be referred to as "UNIX."

It's worth noting that Linux is not the only free UNIX-like operating system in the world. FreeBSD, NetBSD, and OpenBSD, all offshoots of the Berkeley Software Distribution from UC Berkeley, have ardent followers of their own. These OSes are generally comparable to Linux in their features and reliability, although they enjoy somewhat less support from third-party software vendors.

Linux software is UNIX software. Thanks largely to the GNU Project, most of the important software that gives UNIX systems their value has been developed under some form of open source model. The same code runs on Linux and non-Linux systems. The Apache web server, for example, doesn't really care whether it's running on Linux or HP-UX. From the standpoint of applications, Linux is simply one of the best-supported varieties of UNIX.

UNIX and Linux systems have been used in production environments for many years. This book, unlike most others on Linux administration, focuses on the effective use of Linux in a production environment—not just as a single-user desktop.[1]

1.3 LINUX IN HISTORICAL CONTEXT

Linux originated in 1991 as a personal project of Linus Torvalds, a Finnish graduate student. He originally conceived the project as a modest offshoot of Minix, a model operating system written by Andrew S. Tanenbaum. However, Linux generated substantial interest in the world at large, and the kernel soon took on a life of its own. By exploiting the power of cooperative development, Linus was able to tackle a much more ambitious agenda. Kernel version 1.0 was released in 1994; as of this writing (September 2006), the most recent stable version of the Linux kernel is 2.6.17.

Because Linux owes much to its UNIX ancestors, it's not quite fair to locate the dawn of the Linux era in 1991. The history of UNIX goes back several decades to 1969, when UNIX originated as a research project at AT&T Bell Labs. In 1976, UNIX was made available at no charge to universities and thus became the basis of many operating systems classes and academic research projects.

Berkeley UNIX began in 1977 when the Computer Systems Research Group (CSRG) at the University of California, Berkeley, licensed code from AT&T. Berkeley's releases

1. A "production" environment is one that an organization relies on to accomplish real work (as opposed to testing, research, or development).

(called BSD, for Berkeley Software Distribution) started with 1BSD for the PDP-11 and culminated in 1993 with 4.4BSD.

As UNIX gained commercial acceptance, the price of source licenses rose rapidly. Eventually, Berkeley set the long-term goal of removing AT&T's code from BSD, a tedious and time-consuming process. Before the work could be completed, Berkeley lost funding for operating systems research and the CSRG was disbanded.

Before disbanding, the CSRG released its final collection of AT&T-free code, known as 4.4BSD-Lite. Most current versions of BSD UNIX (including FreeBSD, NetBSD, Mac OS X,[2] and OpenBSD) claim the 4.4BSD-Lite package as their grandparent.

Most other major versions of UNIX (including HP-UX and Solaris) are descendants of the original AT&T lineage. Linux doesn't share code with the AT&T or BSD flavors of UNIX, but from a functional perspective it falls somewhere between the two.

1.4 LINUX DISTRIBUTIONS

See the section starting on page 962 for additional comments on distributions.

Linux differs from other variants of UNIX in that the core kernel project defines only an OS kernel. The kernel must be packaged together with commands, daemons, and other software to form a usable and complete operating system—in Linux terms, a "distribution." All Linux distributions share the same kernel lineage, but the ancillary materials that go along with that kernel can vary quite a bit among distributions.

Those "ancillary materials" consist of a vast collection of software developed over the last 30 years by thousands of individuals. It has been argued, with some justification, that the act of referring to the completed operating system simply as "Linux" fails to acknowledge the contributions of those developers and the historical context in which they worked. Unfortunately, the most commonly suggested alternative, "GNU/Linux," has its own political baggage and has been officially endorsed only by the Debian distribution. The Wikipedia entry for "GNU/Linux naming controversy" outlines the arguments on both sides.

Distributions vary in their focus, support, and popularity. Table 1.1 lists the most popular general-purpose distributions. Distributions are listed in alphabetic order, not in order of preference or popularity.

Many smaller distributions are not listed in Table 1.1, and many unlisted special-purpose distributions are targeted at groups with specialized needs (such as embedded system developers).

One useful distribution not found in Table 1.1 is Knoppix (www.knoppix.com), a version of Linux that lives on a bootable CD-ROM. Its primary value lies in its utility as a recovery CD for a Linux system rendered unbootable by a security compromise or technical problem. The bootable CD concept has proved so popular that most major distributions are moving in that direction. Now that Ubuntu can boot from

2. Strictly speaking, the Mac OS X kernel is a variant of Mach, a hybrid system that includes both BSD sections and parts that are rather non-UNIX in flavor.

Table 1.1 Most popular general-purpose Linux distributions

Distribution	Web site	Comments
CentOS	www.centos.org	Free analog of Red Hat Enterprise Linux
Debian	www.debian.org	A popular noncommercial distribution
Fedora	fedora.redhat.com	De-corporatized Red Hat Linux
Gentoo	www.gentoo.org	Source-code based distribution
Mandriva[a]	www.mandriva.com	One of the most user-friendly distros
openSUSE	www.opensuse.org	Free analog of SUSE Linux Enterprise
Red Hat Enterprise	www.redhat.com	Super-corporatized Red Hat Linux
Slackware	www.slackware.com	Stable, basic, bare-bones distribution
SUSE Linux Enterprise	www.novell.com/linux	Strong in Europe, multilingual
TurboLinux	www.turbolinux.com	Strong in Asia, multilingual
Ubuntu	www.ubuntu.com	Cleaned-up version of Debian

a. Formerly Mandrakelinux

the distribution CD, Knoppix is becoming less important. An updated list of bootable Linux distributions can be found at www.frozentech.com/content/livecd.php.

Red Hat has been a dominant force in the Linux world for most of the last decade, and its distributions are predominant in North America. In 2003, the original Red Hat Linux distribution was split into a production-centered line called Red Hat Enterprise Linux (which we sometimes refer to as RHEL in this book) and a community-based development project called Fedora. The split was motivated by a variety of technical, economic, logistic, and legal reasons, but so far the distributions have remained similar. RHEL offers great support and stability but is effectively impossible to use without paying licensing fees to Red Hat.

The CentOS Project (www.centos.org) collects source code that Red Hat is obliged to release under various licensing agreements (most notably, the GNU public license) and assembles it into a complete distribution that is eerily similar to Red Hat Enterprise Linux, but free of charge. The distribution lacks Red Hat's branding and a few proprietary tools, but is in other respects equivalent. CentOS aspires to full binary and bug-for-bug compatibility with RHEL.

CentOS is an excellent choice for sites that want to deploy a production-oriented distribution without paying tithes to Red Hat. A hybrid approach is also feasible: front-line servers can run Red Hat Enterprise Linux and avail themselves of Red Hat's excellent support, while desktops run CentOS. This arrangement covers the important bases in terms of risk and support while also minimizing cost and administrative complexity.

SUSE, now part of Novell, has recently taken the path of Red Hat and forked into two related distributions: one (openSUSE) that contains only free software; and another (SUSE Linux Enterprise) that costs money, includes a formal support path, and offers a few extra trinkets. In the past there seemed to be an effort to hide the existence of

the free version of SUSE, but Novell has been more up front about this edition than SUSE's previous owners. Now, you can go right to www.opensuse.org for the latest information. Nothing in this book is specific to one SUSE distribution or the other, so we simply refer to them collectively as "SUSE."

The Debian and Ubuntu distributions maintain an ideological commitment to community development and open access, so there's never any question about which parts of the distribution are free or redistributable. Debian survives on the zeal and goodwill of the GNU community, while Ubuntu currently enjoys philanthropic funding from South African entrepreneur Mark Shuttleworth. Ubuntu will even send you free CDs in the mail, no postage required.

So what's the best distribution?

A quick search on the net will reveal that this is one of the most frequently asked—and least frequently answered—Linux questions. The right answer for you depends on how you intend to use the system, the varieties of UNIX that you're familiar with, your political sympathies, and your support needs.

Most Linux distributions can do everything you might ever want to do with a Linux system. Some of them may require the installation of additional software to be fully functional, and some may facilitate certain tasks; however, the differences among them are not cosmically significant. In fact, it is something of a mystery why there are so many different distributions, each claiming "easy installation" and "a massive software library" as its distinguishing feature. It's hard to avoid the conclusion that people just like to make new Linux distributions.

On the other hand, since our focus in this book is the management of large-scale Linux installations, we're partial to distributions such as Red Hat Enterprise Linux that take into account the management of networks of machines. Some distributions are designed with production environments in mind, and others are not. The extra crumbs of assistance that the production-oriented systems toss out can make a significant difference in ease of administration.

When you adopt a distribution, you are making an investment in a particular vendor's way of doing things. Instead of looking only at the features of the installed software, it's wise to consider how your organization and that vendor are going to work with each other in the years to come. Some important questions to ask are:

- Is this distribution going to be around in five years?
- Is this distribution going to stay on top of the latest security patches?
- Is this distribution going to release updated software promptly?
- If I have problems, will the vendor talk to me?

Viewed in this light, some of the more interesting, offbeat little distributions don't sound quite so appealing. On the other hand, the most viable distributions are not necessarily the most corporate. For example, we expect Debian (OK, OK, Debian GNU/Linux!) to remain viable for quite a while despite the fact that Debian is not a company, doesn't sell anything, and offers no formal, on-demand support.

A comprehensive list of distributions, including many non-English distributions, can be found at www.linux.org/dist, lwn.net/Distributions, or distrowatch.com.

In this book, we use five popular distributions as our examples: Red Hat Enterprise Linux 4.3 ES, Fedora Core 5, SUSE Linux Enterprise 10.2, Ubuntu 6.06 ("Dapper Drake"), and the current (as of September 2006) testing release of Debian GNU/Linux 3.2 ("Etch"). These systems represent a cross-section of the enterprise Linux market and account collectively for a majority of the installations in use at large sites today.

Distribution-specific administration tools

Many distributions include visually oriented tools (such as the Red Hat Network Administration Tool or SUSE's YaST2) that help you configure or administer selected aspects of the system. These tools can be very useful, especially for novice administrators, but they do tend to obscure the details of what's actually going on when you make changes. In this book, we cover the underlying mechanisms that the visual tools refer to rather than the tools themselves, for several reasons.

For one, the visual tools tend to be proprietary, or at least distribution-specific—they introduce variation into processes that may actually be quite consistent among distributions at a lower level. Second, we believe that it's important for administrators to have an accurate understanding of how their systems work. When the system breaks, the visual tools are usually not helpful in tracking down and fixing problems. Finally, manual configuration is often just plain better: it's faster, more flexible, more reliable, and easier to script.

1.5 NOTATION AND TYPOGRAPHICAL CONVENTIONS

In this book, filenames, commands, and literal arguments to commands are shown in boldface. Placeholders (e.g., command arguments that should not be taken literally) are in italics. For example, in the command

cp *file directory*

you're supposed to replace *file* and *directory* with the names of an actual file and an actual directory.

Excerpts from configuration files and terminal sessions are shown in a fixed-width font.[3] Sometimes, we annotate interactive sessions with italic text. For example:

```
$ grep Bob /pub/phonelist          /* Look up Bob's phone # */
Bob Knowles 555-2834
Bob Smith 555-2311
```

Outside of these specific cases, we have tried to keep special fonts and formatting conventions to a minimum as long as we could do so without compromising intelligibility. For example, we often talk about entities such as the Linux group named daemon and the printer anchor-lw with no special formatting at all.

3. Actually, it's not really a fixed-width font, but it looks like one. We liked it better than the real fixed-width fonts that we tried. That's why the columns in some examples may not all line up perfectly.

In general, we use the same conventions as the manual pages for indicating the syntax of commands:

- Anything between square brackets ("[" and "]") is optional.
- Anything followed by an ellipsis ("…") can be repeated.
- Curly braces ("{" and "}") mean that you should select one of the items separated by vertical bars ("|").

For example, the specification

bork [**-x**] { **on** | **off** } *filename …*

would match any of the following commands:

```
bork on /etc/passwd
bork -x off /etc/passwd /etc/termcap
bork off /usr/lib/tmac
```

We use shell-style globbing characters for pattern matching:

- A star (*) matches zero or more characters.
- A question mark (?) matches one character.
- A tilde or "twiddle" (~) means the home directory of the current user.
- ~*user* means the home directory of *user*.

For example, we might refer to the Debian startup script directories **/etc/rc0.d**, **/etc/rc1.d**, and so on with the shorthand pattern **/etc/rc*.d**.

Text within quotation marks often has a precise technical meaning. In these cases, we ignore the normal rules of English and put punctuation outside the quotation marks so that there can be no confusion about what's included and what's not.

System-specific information

Information in this book generally applies to all of our example distributions unless a specific attribution is given. Details particular to one distribution are marked with the vendor's logo:


Red Hat® Enterprise Linux® 4.3 ES


Fedora™ Core 5


SUSE® Linux Enterprise 10.2


Ubuntu® 6.06 "Dapper Drake"


Debian® GNU/Linux 3.2 "Etch" (testing release of 9/06)

These logos are used with the kind permission of their respective owners. However, the distributors have neither reviewed nor endorsed the contents of this book.

1.6 WHERE TO GO FOR INFORMATION

Linux documentation is spread over a number of sources, some of which you will find installed on your system and some of which live out on the net. The biggies are

- Manual pages (man pages), read with the **man** command
- Texinfo documents, read with the **info** command
- HOWTOs, short notes on various subjects (www.tldp.org)
- Guides, longer treatises on various subjects (www.tldp.org)
- Distribution-specific documentation
- Web pages associated with specific software projects

The man pages and Texinfo documents constitute the traditional "on-line" documentation (though, of course, all the documentation is on-line in some form or another). These docs are typically installed with the system; program-specific man pages usually come along for the ride whenever you install a new package.

Man pages are concise descriptions of individual commands, drivers, file formats, or library routines. They do not address more general topics such as "How do I install a new device?" or "Why is my system so slow?" For those questions, consult the HOWTOs.

Texinfo documents were invented long ago by the GNU folks in reaction to the fact that the **nroff** command to format man pages was proprietary to AT&T. These days we have GNU's own **groff** to do this job for us, and the **nroff** issue is no longer important. Unfortunately, many GNU packages persist in documenting themselves with Texinfo files rather than man pages. In addition to defining an unnecessary second standard for documentation, Texinfo proves to be a rather labyrinthine little hypertext system in its own right.

To escape from Texinfo hell, pipe **info**'s output through the **less** command to evade **info**'s built-in navigation system. As a side effect, this procedure also lets you take advantage of the searching features built into **less**.

Fortunately, packages that are documented with Texinfo usually install man page stubs that tell you to use the **info** command to read about those particular packages. You can safely stick to the **man** command for doing manual searches and delve into **info** land only when instructed to do so. **info info** initiates you into the dark mysteries of Texinfo.

HOWTOs and guides are maintained by The Linux Documentation Project, reachable on-line at www.tldp.org. The LDP is a central repository for all sorts of useful Linux information. It also centralizes efforts to translate Linux-related documents into additional languages.

Some free, on-line LDP guides of particular relevance to system administrators are *The Linux System Administrators' Guide* by Lars Wirzenius, Joanna Oja, Stephen Stafford, and Alex Weeks; the *Advanced Bash-Scripting Guide* by Mendel Cooper;

The Linux Network Administrator's Guide, Second Edition, by Olaf Kirch and Terry Dawson; and *Linux System Administration Made Easy* by Steve Frampton.

Unfortunately, many of the LDP documents are not assiduously maintained. Since Linux-years are a lot like dog-years in their relation to real time, untended documents are apt to quickly go out of date. Always check the time stamp on a HOWTO or guide and weigh its credibility accordingly.

Many of the most important parts of the Linux software base are maintained by neutral third parties such as the Internet Systems Consortium and the Apache Software Foundation. These groups typically generate adequate documentation for the packages they distribute. Distributions sometimes package up the software but skimp on the documentation, so it's often useful to check back with the original source to see if additional materials are available.

Another useful source of information about the design of many Linux software packages is the "Request for Comments" document series, which describes the protocols and procedures used on the Internet. See page 274 for more information.

Organization of the man pages

The Linux man pages are typically divided into nine sections as shown in Table 1.2.

Table 1.2 Sections of the Linux man pages

Section	Contents
1	User-level commands and applications
2	System calls and kernel error codes
3	Library calls
4	Device drivers and network protocols
5	Standard file formats
6	Games and demonstrations
7	Miscellaneous files and documents
8	System administration commands
9	Obscure kernel specs and interfaces

Some sections are further subdivided. For example, section 3M contains man pages for the system's math library. Sections 6 and 9 are typically empty. Many systems have a section of the manuals called "l" for local man pages. Another common convention is section "n" for software-specific subcommands (such as **bash** built-ins).

nroff input for man pages is usually kept in the directories **/usr/share/man/man***X*, where *X* is a digit **1** through **9**, or **l** or **n**. The pages are normally compressed with **gzip** to save space. (The **man** command knows how to uncompress them on the fly.) Formatted versions of the manuals are kept in **/var/cache/man/cat***X*. The **man** command formats man pages as they are needed; if the **cat** directories are writable,

man also deposits the formatted pages as they are created, generating a cache of commonly read man pages.

The **man** command actually searches a number of different directories to find the manual pages you request. You can determine the search path with the **manpath** command. This path (from Fedora) is typical:

```
$ manpath
/usr/kerberos/man:/usr/local/share/man:/usr/share/man/en:/usr/share/man
```

If necessary, you can set your MANPATH environment variable to override the default path. You can also set the system-wide default in **/etc/man.config** (RHEL and Fedora) or **/etc/manpath.config** (SUSE, Debian, and Ubuntu).

man: read manual pages

man *title* formats a specific manual page and sends it to your terminal with **less** (or whatever program is specified in your PAGER environment variable). *title* is usually a command, device, or filename. The sections of the manual are searched in roughly numeric order, although sections that describe commands (sections 1, 8, and 6) are usually searched first.

The form **man** *section title* gets you a man page from a particular section. Thus, **man tty** gets you the man page for the **tty** command, and **man 4 tty** gets you the man page for the controlling terminal driver.

man -k *keyword* prints a list of man pages that have *keyword* in their one-line synopses. For example:

```
$ man -k translate
objcopy (1)          - copy and translate object files
dcgettext (3)        - translate message
tr (1)               - translate or delete characters
snmptranslate (1) - translate SNMP OID values into more useful information
tr (1p)              - translate characters
gettext (1)          - translate message
ngettext (1)         - translate message and choose plural form
...
```

Other sources of Linux information

There's a great big Linux-lovin' world out there. We couldn't possibly mention every useful collection of Linux information, or even just the major ones, but a few significant sources of information are shown in Table 1.3 on the next page.

Don't be shy about accessing general UNIX resources, either—most information is directly applicable to Linux. A wealth of information about system administration is available on the net, in many forms. For example, you can type sysadmin questions into any of the popular search engines, such as Google, Yahoo!, or Ask. A list of other "starter" resources can be found in Chapter 30, *Management, Policy, and Politics*.

Table 1.3 Linux resources on the web

Web site	Description
linux.slashdot.org	Linux-specific arm of tech news giant Slashdot
lwn.net	Linux and open source news aggregator
www.freshmeat.net	Large index of Linux and UNIX software
www.kernel.org	Official Linux kernel site
www.linux.com	Linux information clearing house (unofficial)
www.linux.org	Another Linux information clearing house (unofficial)
www.linuxhq.com	Compilation of kernel-related info and patches
www.linuxworld.com	On-line magazine from the Computerworld folks
www.tldp.org	The Linux Documentation Project
www.tucows.com	Multiplatform software archive with Linux content

Many sites cater directly to the needs of system administrators. Here are a few that we especially like:

- www.ugu.com – the UNIX Guru Universe; lots of stuff for sysadmins
- www.stokely.com – a good collection of links to sysadmin resources
- www.tucows.com – Windows and Mac software, filtered for quality
- slashdot.org – "the place" for geek news
- www.cpan.org – a central source for Perl scripts and libraries
- securityfocus.com – security info; huge, searchable vulnerability database

Another fun and useful resource is Bruce Hamilton's "Rosetta Stone" page at

bhami.com/rosetta.html

It contains pointers to the commands and tools used for various system administration tasks on many different operating systems.

1.7 HOW TO FIND AND INSTALL SOFTWARE

Linux distributions divide their software into packages that can be installed independently of one another. When you install Linux on a new computer, you typically select a range of "starter" packages to be copied onto the new system.

This architecture simplifies many aspects of system configuration and is one of Linux's key advantages over traditional versions of UNIX. Unfortunately, this design also complicates the task of writing about these distributions because it's never really clear which packages are "part of" a given distribution. Is a package "included" if it's on the installation CDs but isn't part of the default installation? Only if it's on every computer running that distribution? If it's on the "bonus" CDs that come only with the supersize version of the distribution?

In this book, we generally describe the default installation of each of our example distributions. When we say that a particular package isn't included in the default installation, it doesn't necessarily mean that the package won't be on *your* system or

that it isn't supported by your distribution. Here's how to find out if you've got it, and if not, how to get it.

First, use the shell's **which** command to find out if a relevant command is already in your search path. For example, the following command reveals that the GNU C compiler has already been installed on this machine in **/usr/bin**:

```
$ which gcc
/usr/bin/gcc
```

If **which** can't find the command you're looking for, try **whereis**; it searches a broader range of system directories and is independent of your shell's search path. Be aware also that some systems' **which** command does not show you files that you do not have permission to execute. For example:

```
$ which ipppd
/usr/bin/which: no ipppd in (/bin:/usr/bin:/sbin:/usr/sbin)
$ whereis ipppd
ipppd: /usr/sbin/ipppd
$ ls -l /usr/sbin/ipppd
-rwx------    1 root     root       124924 Aug  3  2000 /usr/sbin/ipppd
```

Another alternative is the incredibly useful **locate** command, which consults a precompiled index of the filesystem to locate filenames that match a particular pattern. It is not specific to commands or packages but can find any type of file. For example, if you weren't sure where to find the **signal.h** include file (which is the authoritative source for Linux signal definitions), you could try

```
$ locate signal.h
/usr/include/asm/signal.h
/usr/include/linux/signal.h
/usr/include/signal.h
/usr/include/sys/signal.h
```

locate's database is usually regenerated every night by the **updatedb** command, which runs out of **cron**. Therefore, the results of a **locate** don't always reflect recent changes to the filesystem.

If you know the name of a package you're looking for, you can also use your system's packaging utilities to check directly for the package's presence. For example, on a Red Hat, Fedora, or SUSE system, the following command checks for the presence of the Python scripting language:

```
$ rpm -q python
python-1.5.2-27
```

See Chapter 11, *Software and Configuration Management*, for more information about our example distributions' packaging commands.

If the package you're interested in doesn't seem to be installed, the first place to look for it is your distribution's automatic package management system. Every distribution supports some form of Internet-based system for updating old packages and

finding new ones. The most common systems are **yum** and APT, both of which are described in the section *High-level package management systems*, which starts on page 237.

For example, on a Debian system, which uses APT, the following command could be used to obtain and install the most recent version of Python:

```
# apt-get install python
```

Most Linux software is developed by independent groups that release the software in the form of source code. Linux distributors then pick up the source code, compile it appropriately for the conventions in use on their particular system, and package the resulting binaries. It's usually easier to install a distribution-specific binary package than to fetch and compile the original source code. However, distributors are sometimes a release or two behind the current version.

The fact that two distributions use the same packaging system doesn't necessarily mean that packages for the two systems are interchangeable. Red Hat and SUSE both use RPM, for example, but their filesystem layouts are somewhat different. It's always best to use packages designed for your particular distribution if they are available.

If all else fails, try looking for the package at a download site such as freshmeat.net or doing a Google search on the name of the package.

1.8 ESSENTIAL TASKS OF THE SYSTEM ADMINISTRATOR

The sections below briefly summarize some tasks that system administrators are typically expected to perform. These duties need not necessarily be performed by one person, and at many sites the work is distributed among several people. However, at least one person must understand all the chores and make sure that someone is doing them.

Adding, removing, and managing user accounts

See Chapter 6 for more information about adding new users.

The system administrator adds accounts for new users and removes the accounts of users that are no longer active. The process of adding and removing users can be automated, but certain administrative decisions (where to put the user's home directory, on which machines to create the account, etc.) must still be made before a new user can be added.

When a user should no longer have access to the system, the user's account must be disabled. All the files owned by the account should be backed up to tape and disposed of so that the system does not accumulate unwanted baggage over time.

Adding and removing hardware

See Chapters 7, 28, and 23 for more information about these topics.

When new hardware is purchased or when hardware is moved from one machine to another, the system must be configured to recognize and use that hardware. Hardware-support chores can range from the simple task of adding a printer to the more complex job of adding a disk array.

Performing backups

See Chapter 9 for more information about backups.

Performing backups is perhaps the most important job of the system administrator, and it is also the job that is most often ignored or sloppily done. Backups are time consuming and boring, but they are absolutely necessary. Backups can be automated and delegated to an underling, but it is still the system administrator's job to make sure that backups are executed correctly and on schedule (and that the resulting media can actually be used to restore files).

Installing and upgrading software

See Chapter 11 for more information about software management.

When new software is acquired, it must be installed and tested, often under several operating systems and on several types of hardware. Once the software is working correctly, users must be informed of its availability and location. As patches and security updates are released, they must be incorporated smoothly into the local environment.

Local software should be installed in a place that makes it easy to differentiate local from system software. This organization simplifies the task of upgrading the operating system since the local software won't be overwritten by the upgrade procedure.

Monitoring the system

Large installations require vigilant supervision. Daily activities include making sure that email and web service are working correctly, watching log files for early signs of trouble, ensuring that local networks are all properly connected, and keeping an eye on the availability of system resources such as disk space.

Troubleshooting

Linux systems and the hardware they run on occasionally break down. It is the administrator's job to play mechanic by diagnosing problems and calling in experts if needed. Finding the problem is often harder than fixing it.

Maintaining local documentation

See page 930 for suggestions regarding documentation.

As the system is changed to suit an organization's needs, it begins to differ from the plain-vanilla system described by the documentation. It is the system administrator's duty to document aspects of the system that are specific to the local environment. This chore includes documenting any software that is installed but did not come with the operating system, documenting where cables are run and how they are constructed, keeping maintenance records for all hardware, recording the status of backups, and documenting local procedures and policies.

Vigilantly monitoring security

See Chapter 20 for more information about security.

The system administrator must implement a security policy and periodically check to be sure that the security of the system has not been violated. On low-security systems, this chore might involve only a few cursory checks for unauthorized access. On a high-security system, it can include an elaborate network of traps and auditing programs.

Helping users

Although helping users with their various problems is rarely included in a system administrator's job description, it claims a significant portion of most administrators' workdays. System administrators are bombarded with problems ranging from "My program worked yesterday and now it doesn't! What did you change?" to "I spilled coffee on my keyboard! Should I pour water on it to wash it out?"

1.9 SYSTEM ADMINISTRATION UNDER DURESS

System administrators wear many hats. In the real world, they are often people with other jobs who have been asked to look after a few computers on the side. If you are in this situation, you may want to think a bit about where it might eventually lead.

The more you learn about your system, the more the user community will come to depend on you. Networks invariably grow, and you may be pressured to spend an increasing portion of your time on administration. You will soon find that you are the only person in your organization who knows how to perform a variety of important tasks.

Once coworkers come to think of you as the local system administrator, it is difficult to extricate yourself from this role. We know several people who have changed jobs to escape it. Since many administrative tasks are intangible, you may also find that you're expected to be both a full-time administrator and a full-time engineer, writer, or secretary.

Some unwilling administrators try to fend off requests by adopting an ornery attitude and providing poor service. We do not recommend this approach; it makes you look bad and creates additional problems.

Instead, we suggest that you document the time you spend on system administration. Your goal should be to keep the work at a manageable level and to assemble evidence that you can use when you ask to be relieved of administrative duties. In most organizations, you will need to lobby the management from six months to a year to get yourself replaced, so plan ahead.

On the other hand, you may find that you enjoy system administration and that you yearn to be a full-time administrator. Your prospects for employment are good. Unfortunately, your political problems will probably intensify. Refer to Chapter 30, *Management, Policy, and Politics*, for a preview of the political aspects of system administration.

System Administration Personality Syndrome

One unfortunate but common clinical condition resulting from working as a system administrator is System Administration Personality Syndrome. The onset of this condition usually begins early in the third year of a system administrator's career and the

syndrome can last well into retirement. Characteristic symptoms include but are not limited to

- Acute phantom pagerphobia: the disturbing feeling that your pager has gone off (when it really hasn't) and that your peaceful evening with your significant other is about to abruptly end, resulting in a 72-hour work marathon without food
- User voodoographia: the compulsive creation of voodoo-doll representations of the subset of your user population that doesn't seem to understand that their persistent lack of planning doesn't constitute an emergency in your world
- Idiopathic anal tapereadaplexia: the sudden, late-night urge to mount backup tapes to see if they're actually readable and labeled correctly
- Scientifica inapplicia: the strong desire to violently shake fellow system administrators who seem never to have encountered the scientific method

Many curative therapies can be used to treat this unfortunate condition. The most effective are a well-developed sense of humor and the construction of a small but well-endowed office wine cellar. You might also consider the more meditative approach of silently staring off into space and clicking your heels together whenever the words "Is the server down again?" are spoken in your vicinity. If all else fails, take a vacation.

1.10 RECOMMENDED READING

The best resources for system administrators in the printed realm (aside from this book :-) are the O'Reilly series of books. The series began with *UNIX in a Nutshell* over 20 years ago and now includes a separate volume on just about every important UNIX and Linux subsystem and command. The series also includes books on the Internet, Windows, and other non-UNIX topics. All the books are reasonably priced, timely, and focused. Tim O'Reilly has become quite interested in the open source movement and runs a conference, OSCON, on this topic as well as conferences on other trendy techie topics. OSCON occurs twice yearly, once in the United States and once in Europe. See www.oreilly.com for more information.

Although a variety of introductory Linux books are on the market, we have not yet found one that we could recommend without reservation. In general, you're better off looking for the UNIX "classics." Almost everything you read will apply equally well to Linux.

SIEVER, ELLEN, AARON WEBER, AND STEPHEN FIGGINS. *Linux in a Nutshell (5th Edition)*. Sebastopol, CA: O'Reilly Media, 2006.

LAMB, LINDA, AND ARNOLD ROBBINS. *Learning the vi Editor (6th Edition)*. Sebastopol, CA: O'Reilly & Associates, 1998.

POWERS, SHELLY, JERRY PEEK, TIM O'REILLY, AND MIKE LOUKIDES. *UNIX Power Tools (3rd Edition)*. Sebastopol, CA: O'Reilly Media, 2003.

WALL, LARRY, TOM CHRISTIANSEN, AND JON ORWANT. *Programming Perl (3rd Edition)*. Cambridge, MA: O'Reilly Media, 2000.

CHRISTIANSEN, TOM, AND NATHAN TORKINGTON. *Perl Cookbook (2nd Edition)*. Sebastopol, CA: O'Reilly Media, 2003.

GANCARZ, MIKE. *Linux and the Unix Philosophy*. Boston: Digital Press, 2003.

SALUS, PETER. *The Daemon, the GNU & the Penguin*. Groklaw. 2006.

This fascinating history of the open source movement by UNIX's best-known historian is being serialized at groklaw.com under the Creative Commons license. It's currently about 75% complete. The URL for the book itself is quite long; look for a currently link at groklaw.com or try this compressed equivalent: tinyurl.com/d6u7j.

1.11 EXERCISES

E1.1 What command would you use to read about the **sync** system call (*not* the **sync** command)? How would you read **sync**'s local man page that was kept in **/usr/local/share/man**?

E1.2 Does a system-wide config file control the behavior of the **man** command at your site? What lines would you add to this file if you wanted to store local material in **/doc/man**? What directory structure would you have to use in **/doc/man** to make it a full citizen of the man page hierarchy?

E1.3 What are the main differences between **man** and **info**? What are some advantages of each?

☆ **E1.4** What is the current status of Linux kernel development? What are the hot issues? Who are some of the key players? How is the project managed?

☆ **E1.5** Research several Linux distributions (see page 7 for a starter list) and recommend a distribution for each of the following applications. Explain your choices.

a) A single user working in a home office
b) A university computer science lab
c) A corporate web server

☆ **E1.6** Suppose you discover that a certain feature of Apache **httpd** does not appear to work as documented on Fedora Core 5.

a) What should you do before reporting the bug?
b) If you decide that the bug is real, whom should you notify and how?
c) What information must be included to make the bug report useful?