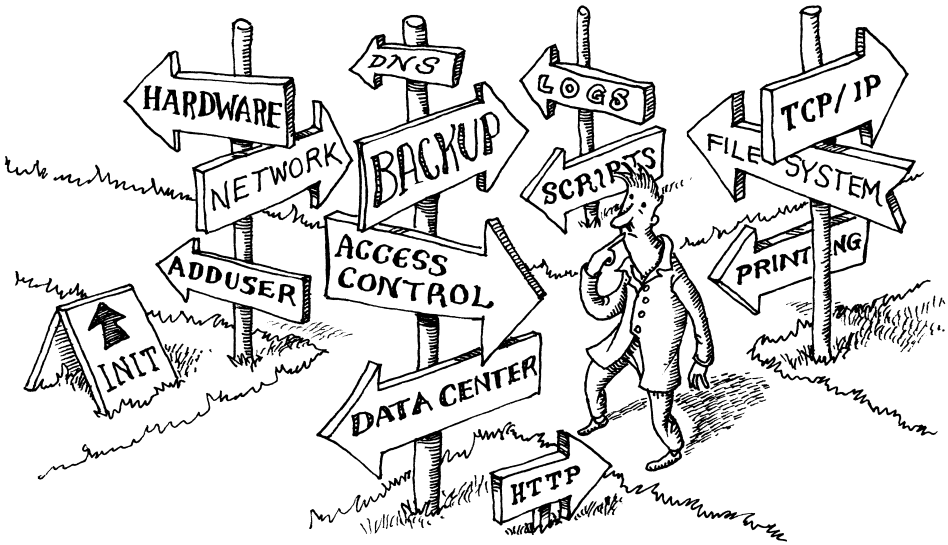
1 Where to Start



An awful lot of UNIX and Linux information is available these days, so we've designed this book to occupy a specific niche in the ecosystem of man pages, blogs, magazines, books, and other reference materials that address the needs of system administrators.

First, it's an orientation guide. It reviews the major administrative systems, identifies the different pieces of each, and explains how they work together. In the many cases where you must choose between various implementations of a concept, we describe the advantages and drawbacks of the major players.

Second, it's a quick-reference handbook that summarizes what you need to know to perform common tasks on a variety of common UNIX and Linux systems. For example, the **ps** command, which shows the status of running processes, supports more than 80 command-line options on Linux systems. But a few combinations of options satisfy 99% of a system administrator's needs; see them on page 130.

Finally, this book focuses on the administration of enterprise servers and networks. That is, *serious* system administration. It's easy to set up a single desktop system; harder to keep a virtualized network running smoothly in the face of load spikes, disk failures, and intentional attacks. We describe techniques and rules of

thumb that help networks recover from adversity, and we help you choose solutions that scale as your site grows in size, complexity, and heterogeneity.

We don't claim to do all of this 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 ESSENTIAL DUTIES OF THE SYSTEM ADMINISTRATOR

See Chapter 2 for more information about scripting.

The Wikipedia page for "system administrator" includes a nice discussion of the tasks that system administration is generally thought to include. This page currently draws a rather sharp distinction between administration and software development, but in our experience, professional administrators spend much of their time writing scripts. That doesn't make system administrators developers per se, but it does mean that they need many of the same analytical and architectural skills.

The sections below summarize some of the main tasks that administrators are expected to perform. These duties need not necessarily be carried out by a single person, and at many sites the work is distributed among a team. However, at least one person must understand all the components and make sure that every task is being done correctly.

Account provisioning

See Chapter 7 for more information about adding new users.

The system administrator adds accounts for new users, removes the accounts of users that are no longer active, and handles all the account-related issues that come up in between (e.g., forgotten passwords). The process of adding and removing users can be automated, but certain administrative decisions (where to put a user's home directory, which machines to create the account on, 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 and then disposed of so that the system does not accumulate unwanted baggage over time.

Adding and removing hardware

See Chapters 8, 13, and 26 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.

Now that virtualization has arrived in the enterprise computing sphere, hardware configuration can be more complicated than ever. Devices may need installation

at several layers of the virtualization stack, and the system administrator may need to formulate policies that allow the hardware to be shared securely and fairly.

Performing backups

See Chapter 10 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 12 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 and administrative scripts should be properly packaged and managed in a fashion that's compatible with the native upgrade procedures used on systems at your site. As this software evolves, new releases should be staged for testing before being deployed to the entire site.

Monitoring the system

Large installations require vigilant supervision. Don't expect users to report problems to you unless the issues are severe. Working around a problem is usually faster than taking the time to document and report it, so users often follow the path of least resistance.

Regularly ensure that email and web services are working correctly, watch log files for early signs of trouble, make sure that local networks are properly connected, and keep an eye on the availability of system resources such as disk space. All of these chores are excellent opportunities for automation, and a variety of off-the-shelf monitoring systems can help sysadmins with this task.

Troubleshooting

System failures are inevitable. 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 1200 for suggestions regarding documentation.

As a system is changed to suit an organization's needs, it begins to differ from the plain-vanilla system described by the documentation. Since the system administrator is responsible for making these customizations, it's also the sysadmin's duty to document the changes. This chore includes 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 22 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 basic checks for unauthorized access. On a high-security system, it can include an elaborate network of traps and auditing programs.

Fire fighting

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 "It 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?"

In most cases, your response to these issues affects your perceived value as an administrator far more than does any actual technical skill you might possess. You can either howl at the injustice of it all, or you can delight in the fact that a single well-handled trouble ticket scores as many brownie points as five hours of midnight debugging. You pick!

1.2 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 the system looks and feels from the user's perspective since we don't review this material. Several good books can get you up to speed; see the reading list on page 27.

Even in these days of Compiz-powered 3D desktops, the GUI tools for system administration on UNIX and Linux systems remain fairly simplified in comparison with the richness of the underlying software. In the real world, we still administer by editing configuration files and writing scripts, so you'll need to be comfortable with both a command-line shell and a text editor.

Your editor can be a GUI tool like **gedit** or a command-line tool such as **vi** or **emacs**. Word processors such as Microsoft Word and OpenOffice Writer are quite different from text editors and are nearly useless for administrative tasks. Command-line tools have an edge because they can run over simple SSH connections and on ailing systems that won't boot; there's no need for a window system. They are also much faster for the quick little edits that administrators often make.

We recommend learning **vi** (now seen most commonly in its rewritten form, **vim**), which is standard on all UNIX and Linux systems. Although it may appear a bit pallid when compared with glitzier offerings such as **emacs**, it is powerful and complete. GNU's **nano** is a simple and low-impact "starter editor" that has

on-screen prompts. Be wary of nonstandard editors, though; 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 **bash**/**sh** scripts.

See Chapter 2 for more information about scripting.

For new scripting projects, we recommend Perl or Python. As a programming language, Perl is admittedly a bit strange. 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 27.

Many administrators prefer Python to Perl, and we know of sites that are making a concerted effort to convert. Python is a more elegant language, and Python scripts are generally more readable and easier to maintain. (As Amazon's Steve Yegge said, "The Python community has long been the refuge for folks who finally took the red pill and woke up from the Perl Matrix.") A useful set of links that compare Python to other scripting languages (including Perl) can be found at python.org/doc/Comparisons.html.

Ruby is an up-and-coming language that maintains many of the strengths of Perl while avoiding some of Perl's syntactic pitfalls and adding modern object-oriented features. It doesn't yet have a strong tradition as a scripting language for system administrators, but that will likely change over the next few years.

We also suggest that you learn **expect**, which is not a programming language so much as a front end for driving interactive programs. It's an efficient glue technology that can replace some complex scripting. **expect** is easy to learn.

Chapter 2, *Scripting and the Shell*, summarizes the most important things to know about scripting for **bash**, Perl, and Python. It also reviews regular expressions (text matching patterns) and some shell idioms that are useful for sysadmins.

1.3 Friction between UNIX and Linux

See the section starting on page 1264 for more of the history of UNIX and Linux.

Because they are similar, this book covers both UNIX and Linux systems. Unfortunately, mentioning UNIX and Linux together in the same sentence can sometimes be like stepping into a political minefield, or perhaps blundering into a large patch of quicksand. But since the relationship between UNIX and Linux seems to engender some confusion as well as animosity, it's hard to avoid staking out a position. Here is our perspective and our short version of the facts.

Linux is a reimplementation and elaboration of the UNIX kernel. It conforms to the POSIX standard, runs on several hardware platforms, and is compatible with most existing UNIX software. It differs from many—but not all—variants of UNIX in that it is free, open source, and cooperatively developed. Linux includes

technical advances that did not exist in UNIX, so it is more than just a UNIX clone. At the same time, traditional UNIX vendors have continued to refine their systems, so there are certainly areas in which commercial UNIX systems are superior to Linux.

Whatever the relative merits of the systems, Linux is a legally, developmentally, and historically distinct entity that cannot properly be referred to as "UNIX" or as a "version of UNIX." To do so is to slight the work and innovation of the Linux community. At the same time, it's somewhat misleading to insist that Linux is "not UNIX." If your creation walks like a duck and quacks like a duck, you may have invented a duck.

Schisms exist even within the Linux camp. It has been argued, with some justification, that referring to Linux distributions simply as "Linux" fails to acknowledge the work that went into the software that runs outside the kernel (which in fact constitutes the vast majority of software on an average system). 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.[1] Interestingly, the use of open source software is now predominant even on most UNIX systems, but no one seems to be pushing for a GNU/UNIX designation just yet.[2]

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.[3] The same code runs on Linux and non-Linux systems. The Apache web server, for example, doesn't much care whether it's running on Linux or Solaris. From the standpoint of applications and most administrative software, Linux is simply one of the best-supported and most widely available varieties of UNIX.

It's also worth noting that Linux is not the only free UNIX-like operating system in the world. OpenSolaris is free and open source, although its exact licensing terms have earned suspicious looks from some open source purists. 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.

1. Since Wikipedia contains Linux information and must therefore refer to Linux frequently, the debate has particular relevance to Wikipedia itself. The discussion page for the Wikipedia article is also well worth reading.
2. After all, "GNU's not UNIX!"
3. Several of our technical reviewers protested that we seem to be crediting GNU with the creation of most of the world's free software. We are not! However, GNU has certainly done more than any other group to promote the *idea* of free software as a social enterprise and to structure ongoing debate about licensing terms and interactions between free and nonfree software.

UNIX and Linux systems have both been used in production environments for many years, and they both work well.[4] At this point, the choice between them has more to do with packaging, support, and institutional inertia than any real difference in quality or modernity.

In this book, comments about "Linux" generally apply to Linux distributions but not to traditional UNIX variants. The meaning of "UNIX" is a bit more fluid, as we occasionally apply it to attributes shared by all UNIX derivatives, including Linux (e.g., "UNIX file permissions"). To avoid ambiguity, we usually say "UNIX and Linux" when we mean both.

1.4 LINUX DISTRIBUTIONS

All Linux distributions share the same kernel lineage, but the ancillary materials that go along with that kernel can vary quite a bit. Distributions vary in their focus, support, and popularity. There continue to be hundreds of independent Linux distributions, but our sense is that distributions based on the Debian, Red Hat, and SUSE lineages will continue to predominate in production environments over the next five years.

The differences among Linux distributions 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 features. It's hard to avoid the conclusion that people just like to make new Linux distributions.

Many smaller distributions are surprisingly competitive in terms of fit and finish. All major distributions, including the second tier, include a relatively painless installation procedure, a well-tuned desktop environment, and some form of package management. Most distributions also allow you to boot from the distribution DVD, which can be handy for debugging and is also a nice way to take a quick look at a new distribution you are considering.

Since our focus in this book is the management of large-scale 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.

4. We consider a "production" environment to be one that an organization relies on to accomplish real work (as opposed to testing, research, or development).

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 distributions don't sound quite so appealing. But don't count them out: E*Trade, for example, runs on Gentoo Linux.

The most viable distributions are not necessarily the most corporate. For example, we expect Debian Linux (OK, OK, Debian GNU/Linux!) to remain viable for a long time despite the fact that Debian is not a company, doesn't sell anything, and offers no formal, on-demand support. Debian itself isn't one of the most widely used distributions, but it benefits from a committed group of contributors and from the enormous popularity of the Ubuntu distribution, which is based on it.

Table 1.1 lists some of the most popular mainstream distributions.

Table 1.1 Most popular general-purpose Linux distributions

Distribution	Web site	Comments
CentOS	centos.org	Free analog of Red Hat Enterprise
Debian	debian.org	Closest to GNU
Fedora	fedoraproject.org	De-corporatized Red Hat Linux
Gentoo	gentoo.org	Compile-it-yourself, optimized
Linux Mint	linuxmint.com	Ubuntu-based, elegant apps
Mandriva	mandriva.com	Long history, "easy to try"
openSUSE	opensuse.org	Free analog of SUSE Linux Enterprise
Oracle Enterprise Linux	oracle.com	Oracle-supported version of RHEL
PCLinuxOS	pclinuxos.com	Fork of Mandriva, KDE-oriented
Red Flag	redflag-linux.com	Chinese distro, similar to Red Hat
Red Hat Enterprise	redhat.com	Reliable, slow-changing, commercial
Slackware	slackware.com	Grizzled, long-surviving distro
SUSE Linux Enterprise	novell.com/linux	Strong in Europe, multilingual
Ubuntu	ubuntu.com	Cleaned-up version of Debian

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

1.5 EXAMPLE SYSTEMS USED IN THIS BOOK

We have chosen three popular Linux distributions and three UNIX variants as our examples to discuss throughout this book: Ubuntu Linux, openSUSE, Red Hat Enterprise Linux, Solaris, HP-UX, and AIX. These systems are representative of

the overall marketplace and account collectively for an overwhelming majority of the installations in use at large sites today.

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


Ubuntu® 9.10 "Karmic Koala"


openSUSE® 11.2


Red Hat® Enterprise Linux® 5.5


Solaris™ 11 and OpenSolaris™ 2009.06


HP-UX® 11i v3


AIX® 6.1

These logos are used with the kind permission of their respective owners. However, the vendors have not reviewed or endorsed the contents of this book. The paragraphs below provide a bit more detail about each of these example systems.

Example Linux distributions


Information that's specific to Linux but not to any particular distribution is marked with the Tux penguin logo shown at left.


The 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. Ubuntu currently enjoys philanthropic funding from South African entrepreneur Mark Shuttleworth.

Ubuntu is based on the Debian distribution and uses Debian's packaging system. It comes in two main forms, a Desktop Edition and a Server Edition. They are essentially similar, but the Server Edition kernel comes pretuned for server use and does not install a GUI or GUI applications such as OpenOffice.


SUSE, now part of Novell, has 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. Nothing in this book is specific to one SUSE distribution or the other, so we simply refer to them collectively as "SUSE."


Red Hat has been a dominant force in the Linux world for most of the last decade, and its distributions are widely used 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 refer to as RHEL or Red Hat 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.

The distributions were initially similar, but Fedora has made some significant changes over the last five years and the two systems aren't currently synchronized in any meaningful way. RHEL offers great support and stability but is effectively impossible to use without paying licensing fees to Red Hat.

The CentOS Project (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.

Example UNIX distributions

Solaris is a System V derivative with many extensions from the company formerly known as Sun Microsystems, now part of Oracle.[5] Sun UNIX (as it was called in the mid-80s) was originally the progeny of Berkeley UNIX, but a (now historic) corporate partnership between Sun and AT&T forced a change of code base. Solaris runs on a variety of hardware platforms, most notably Intel x86 and SPARC.

In Sun's hands, Solaris was free to download and use. However, Oracle has changed this policy, and current downloads are labeled as 90-day free trial editions. The existence of OpenSolaris, an explicitly free and open source version of Solaris, complicates the picture as well. At this point (mid-2010), Oracle's exact plans for Solaris and OpenSolaris remain unclear.

The release of Solaris 11 is expected some time this year, and every indication so far is that it will hew closely to OpenSolaris. In this book, the composite system we refer to as "Solaris" is based on production Solaris 10 and OpenSolaris releases, adjusted with guidance from our network of deep-cover spies within Oracle. In a few cases, we note specifics for Solaris 10 or OpenSolaris.

HP-UX is based on System V and is tied to Hewlett-Packard's hardware platforms. It's closer to the ancestral source tree than either Solaris or AIX, but HP has kept pace with developments in the OS world and has added a variety of its own enhancements. Now that HP has begun supporting Linux as well, the future of HP-UX is somewhat less clear.

5. See page 1264 for some background on BSD, System V, and the general history of UNIX.

IBM's AIX started as a variant of Berkeley's 4.2BSD, but as of version 4 in 1994, most parts of the system migrated to System V. At this point, AIX has drifted rather far from both origins.

In general, we have the impression that AIX has enjoyed less cross-pollination from other systems than most UNIX variants. It also seems to have fallen under the Svengali-like influence of some of IBM's mainframe and AS/400 operating systems, from which it inherits conventions such as the Object Data Manager (see page 432), the use of configuration commands rather than configuration files, and the SMIT administrative interface. Over time, one might charitably say, it has grown to be more and more like itself.

IBM has been pursuing an interestingly OS-agnostic approach to marketing its hardware for most of the last decade. IBM continues to develop and promote AIX, but it's also engaged in partnerships with Red Hat and Novell to ensure that their respective Linux distributions run smoothly on IBM hardware. It will be interesting to see how this approach plays out in the years ahead.

1.6 System-specific administration tools

Modern systems include a variety of visually oriented tools and control panels (such as SUSE's YaST2 and IBM's SMIT) that help you configure or administer selected aspects of the system. These tools are useful, especially for novice administrators, but they also tend to be relatively incomplete reflections of the underlying software. They make many administrative tasks easier, but most fall short of being authoritative.

In this book, we cover the underlying mechanisms that the visual tools manipulate rather than the tools themselves, for several reasons. For one, the visual tools tend to be proprietary (or at least, system-specific). They introduce variation into processes that may actually be quite consistent among systems 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 often not helpful in tracking down and fixing problems. Finally, manual configuration is often faster, more flexible, more reliable, and easier to script.

1.7 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.[6] Sometimes, we annotate sessions with italic text. For example:

```
$ grep Bob /pub/phonelist          # Look up Bob's phone number
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 daemon group or the printer anchor-lw with no special formatting at all.

We use the same conventions as the manual pages for command syntax:

- 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/smartd.conf
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.[7]
- ~*user* means the home directory of *user*.

For example, we might refer to the 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 U.S. English and put punctuation outside the quotes so that there can be no confusion about what's included and what's not.

1.8 Units

Metric prefixes such as kilo-, mega-, and giga- are defined as powers of 10: one megabuck is 1,000,000 dollars. However, computer types have long poached these prefixes and used them to refer to powers of 2. For example, one "megabyte" of

6. 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.
7. Solaris 10's default shell for root is the original Bourne shell, which (rather surprisingly) does not understand ~ or ~*user* notation.

memory is really 2^{20} or 1,048,576 bytes. The stolen units have even made their way into formal standards such as the JEDEC Solid State Technology Association's Standard 100B.01, which recognizes the prefixes as denoting powers of 2 (albeit with some misgivings).

In an attempt to restore clarity, the International Electrotechnical Commission has defined a set of numeric prefixes (kibi-, mebi-, gibi-, and so on, abbreviated Ki, Mi, and Gi) based explicitly on powers of 2. Those units are always unambiguous, but they are just starting to be widely used. The original kilo-series prefixes are still used in both senses.

Context helps with decoding. RAM is always denominated in powers of 2, but network bandwidth is always a power of 10. Storage space is usually quoted in power-of-10 units, but block and page sizes are in fact powers of 2.

In this book, we use IEC units for powers of 2, metric units for powers of 10, and metric units for rough values and cases in which the exact basis is unclear, undocumented, or impossible to determine. In command output and in excerpts from configuration files, we leave the original values and unit designators. We abbreviate bit as b and byte as B. Table 1.2 shows some examples.

Table 1.2 Unit decoding examples

Example	Meaning
56 kb/s serial line	A serial line that transmits 56,000 bits per second
1kB file	A file that contains 1,000 bytes
4KiB SSD pages	SSD pages that contain 4,096 bytes
8KB of memory	Not used in this book; see note below
100MB file size limit	Nominally 10^8 bytes; in context, ambiguous
100MB disk partition	Nominally 10^8 bytes; in context, probably 99,999,744 bytes[a]
1GiB of RAM	Exactly 1,073,741,824 bytes of memory[b]
1 Gb/s Ethernet	A network that transmits 1,000,000,000 bits per second
1TB hard disk	A hard disk that stores 1,000,000,000,000 bytes

a. That is, 10^8 rounded down to the nearest whole multiple of the disk's 512-byte block size
b. But according to Microsoft, still not enough memory to run the 64-bit version of Windows 7

The abbreviation K, as in "8KB of RAM!", is not part of any standard. It's a computerese adaptation of the metric abbreviation k, for kilo-, and originally meant 1,024 as opposed to 1,000. But since the abbreviations for the larger metric prefixes are already uppercase, the analogy doesn't scale. Later, people became confused about the distinction and started using K for factors of 1,000, too.

The Ubuntu Linux distribution is making a valiant attempt to bring rationality and consistency to this issue; see wiki.ubuntu.com/UnitsPolicy for some additional details.

1.9 MAN PAGES AND OTHER ON-LINE DOCUMENTATION

The manual pages, usually called "man pages" because they are read with the **man** command, constitute the traditional "on-line" documentation. (Of course, these days all the documentation is on-line in some form or another.) Man pages are typically installed with the system. Program-specific man pages come along for the ride when you install new software packages.

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 this system so damn slow?" For those questions, consult your vendor's administration guides (see page 18) or, for Linux systems, the documents available from the Linux Documentation Project.

Organization of the man pages

All systems divide the man pages into sections, but there are minor variations in the way some sections are defined. The basic schema used by our example systems is shown in Table 1.3.

Table 1.3 Sections of the man pages

Linux	Solaris	HP-UX	AIX	Contents
1	1	1	1	User-level commands and applications
2	2	2	2	System calls and kernel error codes
3	3	3	3	Library calls
4	7	7	4	Device drivers and network protocols
5	4	4	5	Standard file formats
6	6	–	6	Games and demonstrations
7	5	5	7	Miscellaneous files and documents
8	1m	1m	8	System administration commands
9	9	–	–	Obscure kernel specs and interfaces
–	–	9	–	HP-UX general information

Some sections may be further subdivided. For example, Solaris's section 3c contains man pages about the system's standard C library. There is also considerable variation in the exact distribution of pages; some systems leave section 8 empty and lump the system administration commands into section 1. A lot of systems have discontinued games and demos, leaving nothing in section 6. Many systems have a section of the manuals called "l" (lowercase L) for local man pages.

The exact structure of the sections isn't important for most topics because **man** finds the appropriate page wherever it is stored. You only need to be aware of the section definitions when a topic with the same name appears in multiple sections. For example, **passwd** is both a command and a configuration file, so it has entries in both section 1 and section 4 or 5.

man: read man pages

man *title* formats a specific manual page and sends it to your terminal through **more**, **less**, or whatever program is specified in your PAGER environment variable. *title* is usually a command, device, filename, or name of a library routine. 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, on most systems, **man sync** gets you the man page for the **sync** command, and **man 2 sync** gets you the man page for the **sync** system call.

Under Solaris, you must preface the section number with the **-s** flag, for example, **man -s 2 sync**.

man -k *keyword* or **apropos** *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
…
```

The keywords database can become out of date. If you add additional man pages to your system, you may need to rebuild this file with **mandb** (Ubuntu, SUSE), **makewhatis** (Red Hat), or **catman -w** (Solaris, HP-UX, AIX).

Storage of man pages

nroff input for man pages is usually kept in directories under **/usr/share/man**. Linux systems compress them with **gzip** to save space. (The **man** command knows how to uncompress them on the fly.) The **man** command maintains a cache of formatted pages in **/var/cache/man** or **/usr/share/man** if the appropriate directories are writable, but this is a security risk. Most systems preformat the man pages once at installation time (see **catman**) or not at all.

Solaris understands man pages formatted with SGML in addition to the traditional **nroff**. The SGML pages have their own section directories underneath **/usr/share/man**.

The **man** command can search several man page repositories to find the manual pages you request. On Linux systems, you can find out the current default search path with the **manpath** command. This path (from Ubuntu) is typical:

```
ubuntu$ manpath
/usr/local/man:/usr/local/share/man:/usr/share/man
```

If necessary, you can set your MANPATH environment variable to override the default path:

```
export MANPATH=/home/share/localman:/usr/share/man
```

Some systems let you set a custom system-wide default search path for man pages, which can be useful if you need to maintain a parallel tree of man pages such as those generated by OpenPKG. If you want to distribute local documentation in the form of man pages, however, it is simpler to use your system's standard packaging mechanism and to put man pages in the standard man directories. See Chapter 12, *Software Installation and Management*, for more details.

GNU Texinfo

Linux systems include a sort of supplemental on-line man page system called Texinfo. It was 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, but Texinfo still lumbers along like a zombie in search of human brains.

Although the use of Texinfo seems to be gradually fading, a few GNU packages persist in documenting themselves with Texinfo files rather than man pages. You can pipe the output of the Texinfo reader, **info**, through **less** to evade **info**'s built-in navigation system.

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.

1.10 Other authoritative documentation

Man pages are just a small part of the official documentation. Most of the rest, unfortunately, is scattered about on the web.

System-specific guides

Major vendors have their own dedicated documentation projects, and many continue to produce useful book-length manuals. These days the manuals are usually found on-line rather than in the form of printed books. The extent and quality of the documentation vary widely, but most vendors produce at least an administration guide and an installation guide. Table 1.4 shows where to look for each of our example systems.

The standout in this crowd is IBM, which produces a raft of full-length books on a variety of administration topics. You can buy them as books, but they're also available for free as downloads. The downside to IBM's completeness is that many of the documents seem to lag a version or two behind the current release of AIX.

Table 1.4 Where to find OS vendors' proprietary documentation

System	URL	Comments
Ubuntu	help.ubuntu.com	Mostly user-oriented; see "server guide"
SUSE	novell.com/documentation	Admin stuff is in "reference guide"
RHEL	redhat.com/docs	Mostly documents Red Hat extensions
Solaris	docs.sun.com	Extensive catalog of materials
HP-UX	docs.hp.com	Books, white papers, and tech guides
AIX	www.redbooks.ibm.com ibm.com/support	Numerous real books in PDF format Support gateway to notes, FAQs, etc.

Red Hat is the unfortunate laggard in the documentation race. Most of its documents relate to its proprietary value-added systems rather than to Linux administration generally.

Package-specific documentation

Most of the important software packages in the UNIX and Linux world are maintained by individuals or by third parties such as the Internet Systems Consortium and the Apache Software Foundation. These groups write their own documentation. The quality runs the gamut from embarrassing to spectacular, but jewels such as *Version Control with Subversion* from svnbook.red-bean.com make the hunt worthwhile.

UNIX vendors and Linux distributors always include the appropriate man pages in their packages. Unfortunately, they tend to skimp on other documentation, mostly because there really isn't a standard place to put it (check **/usr/share/doc**). It's often useful to check the original source of the software to see if additional materials are available.

Supplemental documents include white papers (technical reports), design rationales, and book- or pamphlet-length treatments of particular topics. These supplemental materials are not limited to describing just one command, so they can adopt a tutorial or procedural approach. Many pieces of software have both a man page and an article. For example, the man page for **vi** tells you about the command-line arguments that **vi** understands, but you have to go to the in-depth treatment to learn how to actually edit a file.

Books

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 oreilly.com for more information.

RFCs and other Internet documents

The Request for Comments document series describes the protocols and procedures used on the Internet. Most of these documents are relatively detailed and technical, but some are written as overviews. They are absolutely authoritative, and many are quite useful for system administrators. See page 449 for a more complete description of these documents.

The Linux Documentation Project

Linux systems have another major source of reference information: the Linux Documentation Project at tldp.org. This site hosts a huge array of user-contributed documentation ranging from FAQs to full-length guides. The LDP also centralizes efforts to translate Linux-related documents into additional languages.

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

1.11 Other sources of information

The sources discussed in the previous section are generally the most reliable, but they're hardly the last word in UNIX and Linux documentation. Countless blogs, discussion forums, and news feeds are available on the Internet.

It should go without saying, but Google is a system administrator's best friend. Unless you're looking up the details of a specific command or file format, Google should be the first resource you consult for any sysadmin question. Make it a habit; if nothing else, you'll avoid the delay and humiliation of having your questions in an on-line forum answered with a link to Google.[8] *When stuck, Google.*

We can't enumerate every useful collection of UNIX and Linux information on the Internet, but a few of the most significant ones are shown in Table 1.5.

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.

If you're a Linux site, don't be shy about accessing general UNIX resources. Most information is directly applicable to Linux.

8. Or worse yet, a link to Google through lmgtfy.com

Table 1.5 Sysadmin resources on the web

Web site	Description
blogs.sun.com	Great collection of technical articles, many Solaris-related
cpan.org	Authoritative collection of Perl modules
freshmeat.net	Large index of Linux and UNIX software
kernel.org	Official Linux kernel site
linux.com	Linux forum, good for new users[a]
linux.org	General Linux information clearing house
linux.slashdot.org	Linux-specific arm of tech news giant Slashdot
linuxhq.com	Compilation of kernel-related info and patches
lwn.net	Linux and open source news service
lxer.com	Linux news aggregator
rootvg.net	AIX-oriented site with lots of links and good forums
securityfocus.com	General computer security info
serverfault.com	Collaboratively edited database of sysadmin questions
ServerFiles.com	Directory of network admin software and hardware
slashdot.org	Tech news in a variety of categories
solariscentral.org	Open blog with Solaris-related news and articles
sun.com/bigadmin	Sun-specific aggregation site for admin info
sunhelp.org	Very nice collection of Sun-related material
ugu.com	UNIX Guru Universe – all things sysadmin

a. This site is now run by the Linux Foundation.

1.12 WAYS TO FIND AND INSTALL SOFTWARE

Chapter 12, *Software Installation and Management*, addresses software provisioning in detail. But for the impatient, here's a quick primer on how to find out what's installed on your system and how to obtain and install new software.

Modern operating systems divide their contents into packages that can be installed independently of one another. The default installation includes a range of starter packages that you can expand according to your needs.

Add-on software is often provided in the form of precompiled packages as well, although the degree to which this is a mainstream approach varies widely among systems. Most software is developed by independent groups that release the software in the form of source code. Package repositories then pick up the source code, compile it appropriately for the conventions in use on the systems they serve, and package the resulting binaries. It's usually easier to install a system-specific binary package than to fetch and compile the original source code. However, packagers are sometimes a release or two behind the current version.

The fact that two systems use the same package format 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 best to use packages designed for your particular system if they are available.

Major Linux distributions provide excellent package management systems that include tools for accessing and searching Internet software repositories. Distributors aggressively maintain these repositories on behalf of the community, so there is rarely a need for Linux administrators to step outside the bounds of their systems' default package manager. Life is good.

UNIX systems show more ambivalence about package management. Solaris, HP-UX, and AIX all provide packaging software that works at the level of individual machines. However, the vendors of these systems don't maintain repositories of open source software, so the user communities are mostly left to fend for themselves.[9] Unfortunately, one of the main pieces of glue that holds a packaging universe together is a way for packages to reliably refer to other packages in order to express dependency or compatibility information. Without some central coordination, the whole ecosystem can quickly fall apart.

In the real world, results have varied. Solaris has an add-on system (**pkgutil** from blastwave.org) that provides for easy software installation from an Internet repository, much like the native systems found on Linux distributions. HP-UX has a nice Internet repository in the form of the HP-UX Porting and Archiving Centre at hpux.connect.org.uk, but packages must be manually and individually downloaded. At the more dismal end of the spectrum, the availability of prepackaged software for AIX is somewhat scattershot.

Administrators without access to prepackaged binaries must install software the old-fashioned way: by downloading a **tar** archive of the source code and manually configuring, compiling, and installing it. Depending on the software and the operating system, this process can range from trivial to nightmarish.

In this book, we generally assume that optional software is already installed rather than torturing you with boilerplate instructions for installing every package. If there's a potential for confusion, we sometimes mention the exact names of the packages needed to complete a particular project. For the most part, however, we don't repeat installation instructions since they tend to be similar from one package to the next.

Determining whether software has already been installed

For a variety of reasons, it can be a bit tricky to determine which software package contains the component you actually need. Rather than starting at the package level, it's easier to use the shell's **which** command to find out if a relevant binary is already in your search path. For example, the following command reveals that the GNU C compiler has already been installed on this machine:

```
aix$ which gcc
/opt/pware/bin/gcc
```

9. OpenSolaris does offer a Linux-quality package management system and Internet repository. This feature does not exist in Solaris 10, but it's likely to be featured in Solaris 11.

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.

Another alternative is the incredibly useful **locate** command, which consults a precompiled index of the filesystem to locate filenames that match a particular pattern. **locate** is part of the GNU **findutils** package, which is included by default on most Linux systems but must be installed by hand on UNIX.

locate 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, you could try

```
ubuntu$ locate signal.h
/usr/include/signal.h
/usr/include/asm/signal.h
/usr/include/asm-generic/signal.h
/usr/include/linux/signal.h
…
```

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

See Chapter 12 for more information about package management.

If you know the name of the 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 or SUSE system, the following command checks for the presence (and installed version) of the Python scripting language:

```
redhat$ rpm -q python
python-2.4.3-21.el5
```

Adding new software

If you do need to install additional software, you first need to determine the canonical name of the relevant software package. For example, you'd need to translate "I want to install **locate**" to "I need to install the **findutils** package," or translate "I need **named**" to "I have to install BIND." A variety of system-specific indexes on the web can help with this, but Google is usually just as effective. For example, a search for "locate command" takes you directly to several relevant discussions. If you're on a UNIX system, throw in the name of the operating system as well.

Once you know the name of the relevant software, you can download and install it. The complete installation is usually a single command on Linux systems and on Solaris systems that have **pkgutil** installed. For HP-UX and AIX you'll have to download either a prebuilt binary package or the project's original source code. If the latter, try to locate the project's official web page through Google and download the source code from one of the project's mirrors.

The following examples show the installation of the **wget** command on each of our example systems. It's a nifty GNU utility that turns HTTP and FTP downloads into atomic commands—very useful for scripting. **wget** is installed by

default on each of our example Linux systems, but the commands shown below can be used for both initial installation and later updates.

Ubuntu uses APT, the Debian Advanced Package Tool:

```
ubuntu# apt-get install wget
Reading package lists... Done
Building dependency tree
Reading state information... Done
wget is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

The SUSE version is

```
suse# yast --install wget
<runs in a terminal-based UI>
```

The Red Hat version is

```
redhat# yum install wget
Loaded plugins: fastestmirror
...
Parsing package install arguments
Package wget-1.10.2-7.el5.i386 is already installed and latest version
Nothing to do
```

On a Solaris system with **pkutil** already installed (see blastwave.org for instructions on setting this up)

```
solaris# /opt/csw/bin/pkgutil --install wget
<multiple pages of output as seven packages are installed>
```

For HP-UX, we found an appropriate binary package on hpux.connect.org.uk and downloaded it to the **/tmp** directory. The commands to unpack and install it were

```
hpux# gunzip /tmp/wget-1.11.4-hppa-11.31.depot.gz
hpux# swinstall -s /tmp/wget-1.11.4-hppa-11.31.depot wget
=======  05/27/09 13:01:31 EDT  BEGIN swinstall SESSION
     (non-interactive) (jobid=hpux11-0030)

     * Session started for user "root@hpux11".

     * Beginning Selection
     * Target connection succeeded for "hpux11:/".
     * Source:           /tmp/wget-1.11.4-hppa-11.31.depot
     * Targets:          hpux11:/
     * Software selections:
         wget.wget-RUN,r=1.11.4,a=HP-UX_B./800
     * Selection succeeded.
     * Beginning Analysis and Execution
     ...
     * Analysis and Execution succeeded.
     ...
```

The package depot on the **swinstall** command line must be specified as a full path starting with **/**; otherwise, **swinstall** tries to find the file on the network. The **wget** at the end tells **swinstall** which package to install from within the depot file.

Unfortunately, the installation is not really as easy as it first appears. The installed version of **wget** won't actually run because several of the libraries on which it depends have not been installed:

```
hpux$ wget http://samba.org/samba/docs/Samba3-HOWTO.pdf
/usr/lib/dld.sl: Can't open shared library: /usr/local/lib/libcrypto.sl
/usr/lib/dld.sl: No such file or directory
[HP ARIES32]: Core file for 32 bit PA-RISC application
[HP ARIES32]: /usr/local/bin/wget saved to /tmp/core.wget.
```

swinstall does have some dependency management built in, but its abilities unfortunately do not extend to Internet repositories. You'll have to read the fine print and install all the appropriate prerequisite packages (in this case, six more) to make things right.

Building software from source code

There is in fact at least one binary **wget** package available for AIX in RPM format. A Google search for "aix wget rpm" should turn up some good leads. After downloading, the installation command would be a simple

```
aix# rpm --install wget-1.11.4-1.aix5.1.ppc.rpm
```

But just for illustration, let's build the AIX version of **wget** from the original source code.

Our first chore is to find the code, but that's easy: the first Google result for "wget" takes us right to the project page at GNU, and the source tarball is just a link away. After downloading the current version into the **/tmp** directory, we unpack, configure, build, and install it:

```
aix# cd /tmp; gunzip wget-1.11.4.tar.gz
aix# tar xfp wget-1.11.4.tar
aix# cd wget-1.11.4
aix# ./configure --disable-ssl --disable-nls      # See comments below
configure: configuring for GNU Wget 1.11.4
checking build system type... rs6000-ibm-aix
...
config.status: creating src/config.h
config.status: executing default commands
generating po/POTFILES from ./po/POTFILES.in
creating po/Makefile
aix# make
<several pages of compilation output>
aix# make install
<about a page of output>
```

This **configure**/**make**/**make install** sequence is common to the majority of UNIX and Linux software and works on all systems as long as you have the development environment and any package-specific prerequisites installed. However, it's always a good idea to check the package's **INSTALL** or **README** file for specifics.

In this case, the **--disable-ssl** and **--disable-nls** options to **configure** omit some **wget** features that depend on other libraries that haven't been installed. In real life, you'd probably want to install the prerequisites. Use **configure --help** to see all the configuration options. Another useful **configure** option is **--prefix=***directory*, which lets you put the software somewhere other than **/usr/local**.

1.13 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 this is your situation, tread carefully and be aware of how this scenario tends to play out over the long term.

The more experienced you become at system management, the more the user community comes to depend on you. Networks invariably grow, and administrative work tends to accumulate over time as your administration system becomes more sophisticated and you add additional layers. 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. That is not necessarily a bad thing, but 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 analyst.

There is a common tendency for unwilling administrators to fend off requests by adopting a surly attitude and providing poor service.[10] This approach usually backfires; it makes you look bad and creates additional problems.

Instead, consider keeping detailed records of 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 prefer it to real work. Employment prospects remain good. Unfortunately, your political problems will probably intensify. See Chapter 32, *Management, Policy, and Politics*, for a preview of the delights in store.

10. A tendency lovingly and sadistically documented in Simon Travaglia's Bastard Operator from Hell stories; see bofh.ntk.net for the archive. (Look under BOFH.)

1.14 RECOMMENDED READING

ROBBINS, ARNOLD. *UNIX in a Nutshell (4th Edition)*. Sebastopol, CA: O'Reilly Media, 2008.

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

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

SALUS, PETER H. *The Daemon, the GNU & the Penguin: How Free and Open Software is Changing the World*. Reed Media Services, 2008.

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

RAYMOND, ERIC S. *The Cathedral & The Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. Sebastopol, CA: O'Reilly Media, 2001.

System administration

LIMONCELLI, THOMAS A., CHRISTINA J. HOGAN, AND STRATA R. CHALUP. *The Practice of System and Network Administration (Second Edition)*. Reading, MA: Addison-Wesley, 2008.

This is a good book with particularly strong coverage of the policy and procedural aspects of system administration. The authors maintain a system administration blog at everythingsysadmin.com.

FRISCH, ÆLEEN. *Essential System Administration (3rd Edition)*. Sebastopol, CA: O'Reilly Media, 2002.

This is a classic all-around guide to UNIX system administration that is sadly somewhat out of date. We hope a new version is in the works!

Essential tools

ROBBINS, ARNOLD, ELBERT HANNAH, AND LINDA LAMB. *Learning the vi and Vim Editors*. Sebastopol, CA: O'Reilly Media, 2008.

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

MICHAEL, RANDAL K. *Mastering UNIX Shell Scripting: BASH, Bourne, and Korn Shell Scripting for Programmers, System Administrators, and UNIX Gurus*. Indianapolis, IN: Wiley Publishing, Inc., 2008.

ROBBINS, ARNOLD AND NELSON H. F. BEEBE. *Classic Shell Scripting*. Sebastopol, CA: O'Reilly Media, 2005.

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.

BLANK-EDELMAN, DAVID N. *Automating System Administration with Perl (2nd Edition)*. Sebastopol, CA: O'Reilly Media, 2009.

PILGRIM, MARK. *Dive Into Python.* Berkeley, CA: Apress, 2004.

This book is also available for free on the web at diveintopython.org.

FLANAGAN, DAVID, AND YUKIHIRO MATSUMOTO. *The Ruby Programming Language*. Sebastopol, CA: O'Reilly Media, 2008.

This book, optimistically subtitled *Everything You Need to Know,* is unfortunately a bit on the dry side. However, it covers the Ruby 1.9 release and includes a wealth of detail that only the language designer is really in a position to know. Best for those who already have a working knowledge of Perl or Python.

1.15 EXERCISES

E1.1 What command would you use to read about the terminal driver, **tty** (*not* the **tty** command)? How would you read a local **tty** 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 is the current status of Linux kernel development? What are the hot issues? Who are the key players? How is the project managed?

☆ E1.4 Research several UNIX and Linux systems and recommend an operating system 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
d) A server cluster that runs the database for a shipping company

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

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?

☆☆ E1.6 Linux has made dramatic inroads into production environments. Is UNIX doomed? Why or why not?