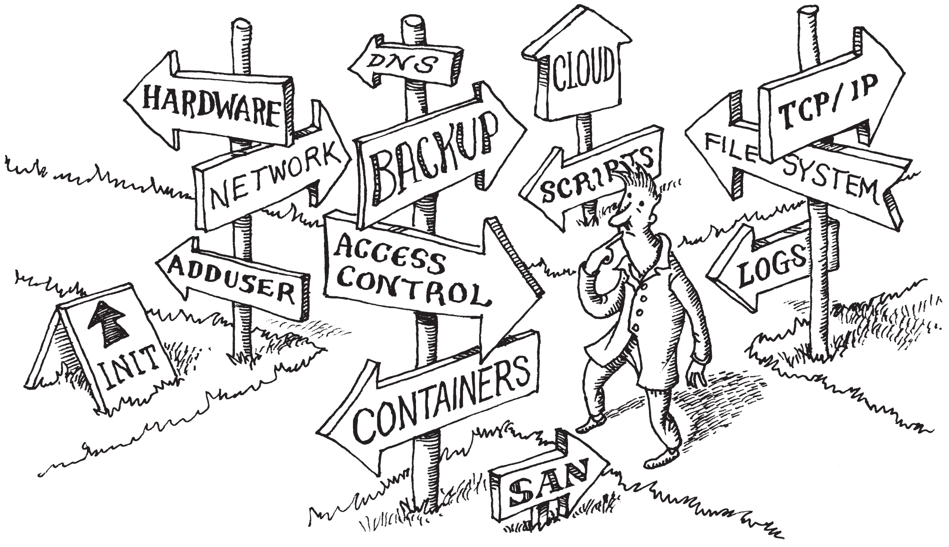
1 Where to Start



We've designed this book to occupy a specific niche in the vast ecosystem of man pages, blogs, magazines, books, and other reference materials that address the needs of UNIX and Linux 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 among various implementations of a concept, we describe the advantages and drawbacks of the most popular options.

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 the majority of a system administrator's needs; we summarize them on page 98.

Finally, this book focuses on the administration of enterprise servers and networks. That is, *serious, professional* system administration. It's easy to set up a single system; harder to keep a distributed, cloud-based platform running smoothly in the face of viral popularity, network partitions, and targeted attacks. We describe techniques and rules of thumb that help you recover systems from adversity, and we help you choose solutions that scale as your empire 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 constitutes the most appropriate solution. We offer our subjective opinions to you as raw data. Decide for yourself how much to accept and how much of our comments apply to your environment.

1.1 Essential duties of a system administrator

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 the members of a team. However, at least one person should understand all the components and ensure that every task is performed correctly.

Controlling access

See Chapters 8, 17, and 23 for information about user account provisioning.

The system administrator creates accounts for new users, removes the accounts of inactive users, and handles all the account-related issues that come up in between (e.g., forgotten passwords and lost key pairs). The process of actually adding and removing accounts is typically automated by a configuration management system or centralized directory service.

Adding hardware

Administrators who work with physical hardware (as opposed to cloud or hosted systems) must install it and configure it to be recognized by the operating system. Hardware support chores might range from the simple task of adding a network interface card to configuring a specialized external storage array.

Automating tasks

See Chapter 7, Scripting and the Shell, for information about scripting and automation.

Using tools to automate repetitive and time-consuming tasks increases your efficiency, reduces the likelihood of errors caused by humans, and improves your ability to respond rapidly to changing requirements. Administrators strive to reduce the amount of manual labor needed to keep systems functioning smoothly. Familiarity with scripting languages and automation tools is a large part of the job.

Overseeing backups

See page 788 for some tips on performing backups.

Backing up data and restoring it successfully when required are important administrative tasks. Although backups are time consuming and boring, the frequency of real-world disasters is simply too high to allow the job to be disregarded.

Operating systems and some individual software packages provide well-established tools and techniques to facilitate backups. Backups must be executed on a regular schedule and restores must be tested periodically to ensure that they are functioning correctly.

Installing and upgrading software

See Chapter 6 for information about software management.

Software must be selected, installed, and configured, often on a variety of operating systems. As patches and security updates are released, they must be tested, reviewed, and incorporated into the local environment without endangering the stability of production systems.

See Chapter 26 for information about software deployment and continuous delivery.

The term "software delivery" refers to the process of releasing updated versions of software—especially software developed in-house—to downstream users. "Continuous delivery" takes this process to the next level by automatically releasing software to users at a regular cadence as it is developed. Administrators help implement robust delivery processes that meet the requirements of the enterprise.

Monitoring

See Chapter 28 for information about monitoring.

Working around a problem is usually faster than taking the time to document and report it, and users internal to an organization often follow the path of least resistance. External users are more likely to voice their complaints publicly than to open a support inquiry. Administrators can help to prevent both of these outcomes by detecting problems and fixing them before public failures occur.

Some monitoring tasks include ensuring that web services respond quickly and correctly, collecting and analyzing log files, and keeping tabs on the availability of server resources such as disk space. All of these are excellent opportunities for automation, and a slew of open source and commercial monitoring systems can help sysadmins with these tasks.

Troubleshooting

See page 428 for an introduction to network troubleshooting.

Networked systems fail in unexpected and sometimes spectacular fashion. It's the administrator's job to play mechanic by diagnosing problems and calling in subject-matter experts as needed. Finding the source of a problem is often more challenging than resolving it.

Maintaining local documentation

See page 1115 for suggestions regarding documentation.

Administrators choose vendors, write scripts, deploy software, and make many other decisions that may not be immediately obvious or intuitive to others. Thorough and accurate documentation is a blessing for team members who would otherwise need to reverse-engineer a system to resolve problems in the middle of the night. A lovingly crafted network diagram is more useful than many paragraphs of text when describing a design.

Vigilantly monitoring security

See Chapter 27 for more information about security.

Administrators are the first line of defense for protecting network-attached systems. The administrator must implement a security policy and set up procedures to prevent systems from being breached. This responsibility might include only a few basic checks for unauthorized access, or it might involve an elaborate network of traps and auditing programs, depending on the context. System administrators are cautious by nature and are often the primary champions of security across a technical organization.

Tuning performance

See Chapter 29 for more information about performance.

UNIX and Linux are general purpose operating systems that are well suited to almost any conceivable computing task. Administrators can tailor systems for optimal performance in accord with the needs of users, the available infrastructure, and the services the systems provide. When a server is performing poorly, it is the administrator's job to investigate its operation and identify areas that need improvement.

Developing site policies

See the sections starting on page 17 for information about local policy-making.

For legal and compliance reasons, most sites need policies that govern the acceptable use of computer systems, the management and retention of data, the privacy and security of networks and systems, and other areas of regulatory interest. System administrators often help organizations develop sensible policies that meet the letter and intent of the law and yet still promote progress and productivity.

Working with vendors

Most sites rely on third parties to provide a variety of ancillary services and products related to their computing infrastructure. These providers might include software developers, cloud infrastructure providers, hosted software-as-a-service (SaaS) shops, help-desk support staff, consultants, contractors, security experts, and platform or infrastructure vendors. Administrators may be tasked with selecting vendors, assisting with contract negotiations, and implementing solutions once the paperwork has been completed.

Fire fighting

Although helping other people with their various problems is rarely included in a system administrator's job description, these tasks claim a measurable 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 more brownie points than five hours of midnight debugging. Your choice!

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 a user's perspective since we do not review that material. Several good books can get you up to speed; see *Recommended reading* on page 28.

We love well-designed graphical interfaces. Unfortunately, GUI tools for system administration on UNIX and Linux remain rudimentary in comparison with the richness of the underlying software. In the real world, administrators must be comfortable using the command line.

For text editing, we strongly recommend learning **vi** (now seen more commonly in its enhanced form, **vim**), which is standard on all systems. It is simple, powerful, and efficient. Mastering **vim** is perhaps the single best productivity enhancement available to administrators. Use the **vimtutor** command for an excellent, interactive introduction.

Alternatively, GNU's **nano** is a simple and low-impact "starter editor" that has on-screen prompts. Use it discreetly; professional administrators may be visibly distressed if they witness a peer running **nano**.

See Chapter 7 for an introduction to scripting.

Although administrators are not usually considered software developers, industry trends are blurring the lines between these functions. Capable administrators are usually polyglot programmers who don't mind picking up a new language when the need arises.

For new scripting projects, we recommend Bash (aka **bash**, aka **sh**), Ruby, or Python. Bash is the default command shell on most UNIX and Linux systems. It is primitive as a programming language, but it serves well as the duct tape in an administrative tool box. Python is a clever language with a highly readable syntax, a large developer community, and libraries that facilitate many common tasks. Ruby developers describe the language as "a joy to work with" and "beautiful to behold." Ruby and Python are similar in many ways, and we've found them to be equally functional for administration. The choice between them is mostly a matter of personal preference.

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 and is easy to learn.

Chapter 7, *Scripting and the Shell*, summarizes the most important things to know about scripting for Bash, Python, and Ruby. It also reviews regular expressions (text matching patterns) and some shell idioms that are useful for sysadmins.

1.3 Linux distributions

A Linux distribution comprises the Linux kernel, which is the core of the operating system, and packages that make up all the commands you can run on the system. All distributions share the same kernel lineage, but the format, type, and number of packages differ quite a bit. Distributions also vary in their focus, support, and popularity. There continue to be hundreds of independent Linux distributions, but our sense is that distributions derived from the Debian and Red Hat lineages will predominate in production environments in the years ahead.

By and large, the differences among Linux distributions are not cosmically significant. In fact, it is something of a mystery why so many different distributions exist, 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.

Most major distributions include a relatively painless installation procedure, a desktop environment, and some form of package management. You can try them out easily by starting up a cloud instance or a local virtual machine.

See Chapter 25, Containers, for more information about Docker and containers.

Much of the insecurity of general-purpose operating systems derives from their complexity. Virtually all leading distributions are cluttered with scores of unused software packages; security vulnerabilities and administrative anguish often come along for the ride. In response, a relatively new breed of minimalist distributions has been gaining traction. CoreOS is leading the charge against the status quo and prefers to run all software in containers. Alpine Linux is a lightweight distribution that is used as the basis of many public Docker images. Given this reductionist trend, we expect the footprint of Linux to shrink over the coming years.

By adopting 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. 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?
- Does this distribution have an active community and sufficient documentation?
- If I have problems, will the vendor talk to me, and how much will that cost?

Table 1.1 lists some of the most popular mainstream distributions.

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 enterprise-level support. Debian benefits from a committed group of contributors and from the enormous popularity of the Ubuntu distribution, which is based on it.

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

Table 1.1 Most popular general-purpose Linux distributions

Distribution	Web site	Comments
Arch	archlinux.org	For those who fear not the command line
CentOS	centos.org	Free analog of Red Hat Enterprise
CoreOS	coreos.com	Containers, containers everywhere
Debian	debian.org	Free as in freedom, most GNUish distro
Fedora	fedoraproject.org	Test bed for Red Hat Linux
Kali	kali.org	For penetration testers
Linux Mint	linuxmint.com	Ubuntu-based, desktop-friendly
openSUSE	opensuse.org	Free analog of SUSE Linux Enterprise
openWRT	openwrt.org	Linux for routers and embedded devices
Oracle Linux	oracle.com	Oracle-supported version of RHEL
RancherOS	rancher.com	20MiB, everything in containers
Red Hat Enterprise	redhat.com	Reliable, slow-changing, commercial
Slackware	slackware.com	Grizzled, long-surviving distro
SUSE Linux Enterprise	suse.com	Strong in Europe, multilingual
Ubuntu	ubuntu.com	Cleaned-up version of Debian

1.4 Example systems used in this book

We have chosen three popular Linux distributions and one UNIX variant as our primary examples for this book: Debian GNU/Linux, Ubuntu Linux, Red Hat Enterprise Linux (and its dopplegänger CentOS), and FreeBSD. These systems are representative of the overall marketplace and account collectively for a substantial portion of 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 a logo:

Debian GNU/Linux 9.0 “Stretch”

Ubuntu® 17.04 “Zesty Zapus”

RHEL Red Hat® Enterprise Linux® 7.1 and CentOS® 7.1

FreeBSD® 11.0

Most of these marks belong to the vendors that release the corresponding software and are used with the kind permission of their respective owners. However, the vendors have not reviewed or endorsed the contents of this book.

We repeatedly attempted and failed to obtain permission from Red Hat to use their famous red fedora logo, so you're stuck with yet another technical acronym. At least this one is in the margins.

The paragraphs below provide a bit more detail about each of the 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.

Debian (pronounced *deb-ian*, named after the late founder Ian Murdock and his wife Debra), is one of the oldest and most well-regarded distributions. It is a non-commercial project with more than a thousand contributors worldwide. Debian maintains 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 defines three releases that are maintained simultaneously: `stable`, targeting production servers; `unstable`, with current packages that may have bugs and security vulnerabilities; and `testing`, which is somewhere in between.

Ubuntu is based on Debian and maintains Debian's commitment to free and open source software. The business behind Ubuntu is Canonical Ltd., founded by entrepreneur Mark Shuttleworth.

Canonical offers a variety of editions of Ubuntu targeting the cloud, the desktop, and bare metal. There are even releases intended for phones and tablets. Ubuntu version numbers derive from the year and month of release, so version 16.10 is from October, 2016. Each release also has an alliterative code name such as Vivid Vervet or Wily Werewolf.

Two versions of Ubuntu are released annually: one in April and one in October. The April releases in even-numbered years are long-term support (LTS) editions that promise five years of maintenance updates. These are the releases recommended for production use.

RHEL

Red Hat has been a dominant force in the Linux world for more than two decades, and its distributions are widely used in North America and beyond. By the numbers, Red Hat, Inc., is the most successful open source software company in the world.

Red Hat Enterprise Linux, often shortened to RHEL, targets production environments at large enterprises that require support and consulting services to keep their systems running smoothly. Somewhat paradoxically, RHEL is open source but requires a license. If you're not willing to pay for the license, you're not going to be running Red Hat.

Red Hat also sponsors Fedora, a community-based distribution that serves as an incubator for bleeding-edge software not considered stable enough for RHEL.

Fedora is used as the initial test bed for software and configurations that later find their way to RHEL.

CentOS is virtually identical to Red Hat Enterprise Linux, but free of charge. The CentOS Project (centos.org) is owned by Red Hat and employs its lead developers. However, they operate separately from the Red Hat Enterprise Linux team. The CentOS distribution lacks Red Hat's branding and a few proprietary tools, but is in other respects equivalent.

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, even as nonproduction systems run CentOS. This arrangement covers the important bases in terms of risk and support while also minimizing cost and administrative complexity.

CentOS aspires to full binary and bug-for-bug compatibility with Red Hat Enterprise Linux. Rather than repeating "Red Hat and CentOS" ad nauseam, we generally mention only one or the other in this book. The text applies equally to Red Hat and CentOS unless we note otherwise.

Other popular distributions are also Red Hat descendants. Oracle sells a rebranded and customized version of CentOS to customers of its enterprise database software. Amazon Linux, available to Amazon Web Services users, was initially derived from CentOS and still shares many of its conventions.

Most administrators will encounter a Red Hat-like system at some point in their careers, and familiarity with its nuances is helpful even if it isn't the system of choice at your site.

Example UNIX distribution

The popularity of UNIX has been waning for some time, and most of the stalwart UNIX distributions (e.g., Solaris, HP-UX, and AIX) are no longer in common use. The open source descendants of BSD are exceptions to this trend and continue to enjoy a cult following, particularly among operating system experts, free software evangelists, and security-minded administrators. In other words, some of the world's foremost operating system authorities rely on the various BSD distributions. Apple's macOS has a BSD heritage.

FreeBSD, first released in late 1993, is the most widely used of the BSD derivatives. It commands a 70% market share among BSD variants according to some usage statistics. Users include major Internet companies such as WhatsApp, Google, and Netflix.

Unlike Linux, FreeBSD is a complete operating system, not just a kernel. Both the kernel and userland software are licensed under the permissive BSD License, a fact that encourages development by and additions from the business community.

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 code font. Sometimes, we annotate sessions with the **bash** comment character # and italic text. For example:

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

We use $ to denote the shell prompt for a normal, unprivileged user, and # for the root user. When a command is specific to a distribution or family of distributions, we prefix the prompt with the distribution name. For example:

```
$ sudo su - root           # Become root
# passwd                   # Change root's password
debian# dpkg -l          # List installed packages on Debian and Ubuntu
```

This convention is aligned with the one used by standard UNIX and Linux shells.

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 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.
- ~*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.

1.6 Units

Metric prefixes such as kilo-, mega-, and giga- are defined as powers of 10; one megabuck is $1,000,000. However, computer types have long poached these prefixes and used them to refer to powers of 2. For example, one "megabyte" of 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, or where the delineation is not important, we leave the original values and unit designators. We abbreviate bit as b and byte as B. Table 1.2 on the next page shows some examples.

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 upper case, the analogy doesn't scale. Later, people became confused about the distinction and started using K for factors of 1,000, too.

Most of the world doesn't consider this to be an important matter and, like the use of imperial units in the United States, metric prefixes are likely to be misused for

Table 1.2 Unit decoding examples

Example	Meaning
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 on page 13
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	1,073,741,824 bytes of memory
1 Gb/s Ethernet	A network that transmits 1,000,000,000 bits per second
6TB hard disk	A hard disk that stores about 6,000,000,000,000 bytes

a. That is, 10^8 rounded down to the nearest whole multiple of the disk's 512-byte block size

the foreseeable future. Ubuntu maintains a helpful units policy, though we suspect it has not been widely adopted even at Canonical; see wiki.ubuntu.com/UnitsPolicy for some additional details.

1.7 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 documentation is on-line in some form or another.) Program-specific man pages come along for the ride when you install new software packages. Even in the age of Google, we continue to consult man pages as an authoritative resource because they are accessible from the command line, typically include complete details on a program's options, and show helpful examples and related commands.

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?"

Organization of the man pages

FreeBSD and Linux divide the man pages into sections. Table 1.3 shows the basic schema. Other UNIX variants sometimes define the sections slightly differently.

The exact structure of the sections isn't important for most topics because **man** finds the appropriate page wherever it is stored. Just 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 5.

Table 1.3 Sections of the 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

man: read man pages

See page 193 to learn about environment variables.

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 and 8) 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.

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 useful information
tr (1p)            - translate characters
...
```

The keywords database can become outdated. If you add additional man pages to your system, you may need to rebuild this file with **makewhatis** (Red Hat and FreeBSD) or **mandb** (Ubuntu).

Storage of man pages

nroff input for man pages (i.e., the man page source code) is stored in directories under **/usr/share/man** and compressed with **gzip** to save space. The **man** command knows how to decompress them on the fly.

man maintains a cache of formatted pages in **/var/cache/man** or **/usr/share/man** if the appropriate directories are writable; however, this is a security risk. Most systems preformat the man pages once at installation time (see **catman**) or not at all.

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. 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 6, *Software Installation and Management*, for more details.

1.8 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. Many continue to produce useful book-length manuals, including administration and installation guides. These are generally available on-line and as downloadable PDF files. Table 1.4 shows where to look.

Although this documentation is helpful, it's not the sort of thing you keep next to your bed for light evening reading (though some vendors' versions would make useful sleep aids). We generally Google for answers before turning to vendor docs.

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 *Pro Git* from git-scm.com/book make the hunt worthwhile.

Table 1.4 Where to find OS vendors' proprietary documentation

OS	URL	Comments
Debian	debian.org/doc	Admin handbook lags behind the current version
Ubuntu	help.ubuntu.com	User oriented, see "server guide" for LTS releases
RHEL	redhat.com/docs	Comprehensive docs for administrators
CentOS	wiki.centos.org	Includes tips, HowTos, and FAQs
FreeBSD	freebsd.org/docs.html	See the *FreeBSD Handbook* for sysadmin info

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 a long-form article. For example, the man page for **vim** tells you about the command-line arguments that **vim** understands, but you have to turn to an in-depth treatment to learn how to actually edit a file.

Most software projects have user and developer mailing lists and IRC channels. This is the first place to visit if you have questions about a specific configuration issue or if you encounter a bug.

Books

The O'Reilly books are favorites in the technology industry. The business began with *UNIX in a Nutshell* and now includes a separate volume on just about every important UNIX and Linux subsystem and command. O'Reilly also publishes books on network protocols, programming languages, Microsoft Windows, and other non-UNIX tech topics. All the books are reasonably priced, timely, and focused.

Many readers turn to O'Reilly's Safari Books Online, a subscription service that offers unlimited electronic access to books, videos, and other learning resources. Content from many publishers is included—not just O'Reilly—and you can choose from an immense library of material.

RFC publications

Request for Comments documents describe the protocols and procedures used on the Internet. Most of these are relatively detailed and technical, but some are written as overviews. The phrase "reference implementation" applied to software usually translates to "implemented by a trusted source according to the RFC specification."

RFCs are absolutely authoritative, and many are quite useful for system administrators. See page 376 for a more complete description of these documents. We refer to various RFCs throughout this book.

1.9 Other sources of information

The sources discussed in the previous section are peer reviewed and written by authoritative sources, but they're hardly the last word in UNIX and Linux administration. 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 or an equivalent search engine 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.[1] *When stuck, search the web.*

Keeping current

Operating systems and the tools and techniques that support them change rapidly. Read the sites in Table 1.5 with your morning coffee to keep abreast of industry trends.

Table 1.5 Resources for keeping up to date

Web site	Description
darkreading.com	Security news, trends, and discussion
devopsreactions.tumblr.com	Sysadmin humor in animated GIF form
linux.com	A Linux Foundation site; forum, good for new users
linuxfoundation.org	Nonprofit fostering OSS, employer of Linus Torvalds
lwn.net	High-quality, timely articles on Linux and OSS
lxer.com	Linux news aggregator
securityfocus.com	Vulnerability reports and security-related mailing lists
@SwiftOnSecurity	Infosec opinion from Taylor Swift (parody account)
@nixcraft	Tweets about UNIX and Linux administration
everythingsysadmin.com	Blog of Thomas Limoncelli, respected sysadmin [a]
sysadvent.blogspot.com	Advent for sysadmins with articles each December
oreilly.com/topics	Learning resources from O'Reilly on many topics
schneier.com	Blog of Bruce Schneier, privacy and security expert

a. See also Tom's collection of April Fools' Day RFCs at rfc-humor.com

Social media are also useful. Twitter and reddit in particular have strong, engaged communities with a lot to offer, though the signal-to-noise ratio can sometimes be quite bad. On reddit, join the sysadmin, linux, linuxadmin, and netsec subreddits.

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

HowTos and reference sites

The sites listed in Table 1.6 contain guides, tutorials, and articles about how to accomplish specific tasks on UNIX and Linux.

Table 1.6 Task-specific forums and reference sites

Web site	Description
wiki.archlinux.org	Articles and guides for Arch Linux; many are more general
askubuntu.com	Q&A for Ubuntu users and developers
digitalocean.com	Tutorials on many OSS, development, and sysadmin topics [a]
kernel.org	Official Linux kernel site
serverfault.com	Collaboratively edited database of sysadmin questions [b]
serversforhackers.com	High-quality videos, forums, and articles on administration

a. See digitalocean.com/community/tutorials
b. Also see the sister site stackoverflow.com, which is dedicated to programming but useful for sysadmins

Stack Overflow and Server Fault, both listed in Table 1.6 (and both members of the Stack Exchange group of sites), warrant a closer look. If you're having a problem, chances are that somebody else has already seen it and asked for help on one of these sites. The reputation-based Q&A format used by the Stack Exchange sites has proved well suited to the kinds of problems that sysadmins and programmers encounter. It's worth creating an account and joining this large community.

Conferences

Industry conferences are a great way to network with other professionals, keep tabs on technology trends, take training classes, gain certifications, and learn about the latest services and products. The number of conferences pertinent to administration has exploded in recent years. Table 1.7 on the next page highlights some of the most prominent ones.

Meetups (meetup.com) are another way to network and engage with like-minded people. Most urban areas in the United States and around the world have a Linux user group or DevOps meetup that sponsors speakers, discussions, and hack days.

1.10 Ways to find and install software

Chapter 6, *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 and contract according to your needs. When adding

Table 1.7 Conferences relevant to system administrators

Conference	Location	When	Description
LISA	Varies	Q4	Large Installation System Administration
Monitorama	Portland	June	Monitoring tools and techniques
OSCON	Varies (US/EU)	Q2 or Q3	Long-running O'Reilly OSS conference
SCALE	Pasadena	Jan	Southern California Linux Expo
DefCon	Las Vegas	July	Oldest and largest hacker convention
Velocity	Global	Varies	O'Reilly conference on web operations
BSDCan	Ottawa	May/June	Everything BSD from novices to gurus
re:Invent	Las Vegas	Q4	AWS cloud computing conference
VMWorld	Varies (US/EU)	Q3 or Q4	Virtualization and cloud computing
LinuxCon	Global	Varies	The future of Linux
RSA	San Francisco	Q1 or Q2	Enterprise cryptography and infosec
DevOpsDays	Global	Varies	A range of topics on bridging the gap between development and ops teams
QCon	Global	Varies	A conference for software developers

software, don your security hat and remember that additional software creates additional attack surface. Only install what's necessary.

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.

Our example systems provide excellent package management systems that include tools for accessing and searching hosted software repositories. Distributors aggressively maintain these repositories on behalf of the community, to facilitate patching and software updates. Life is good.

When the packaged format is insufficient, administrators 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 if software is already installed

For a variety of reasons, it can be a bit tricky to determine which package contains the software 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:

```
ubuntu$ 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.

Another alternative is the incredibly useful **locate** command, which consults a precompiled index of the filesystem to locate filenames that match a particular pattern.

FreeBSD includes **locate** as part of the base system. In Linux, the current implementation of **locate** is in the **mlocate** package. On Red Hat and CentOS, install the **mlocate** package with **yum**; see page 174.

locate can find any type of file; it is not specific to commands or packages. For example, if you weren't sure where to find the **signal.h** include file, you could try

```
freebsd$ locate signal.h
/usr/include/machine/signal.h
/usr/include/signal.h
/usr/include/sys/signal.h
...
```

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

See Chapter 6 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 system, the following command checks for the presence (and installed version) of the Python interpreter:

```
redhat$ rpm -q python
python-2.7.5-18.el7_1.1.x86_64
```

You can also find out which package a particular file belongs to:

```
redhat$ rpm -qf /etc/httpd
httpd-2.4.6-31.el7.centos.x86_64

freebsd$ pkg which /usr/local/sbin/httpd
/usr/local/sbin/httpd was installed by package apache24-2.4.12

ubuntu$ dpkg-query -S /etc/apache2
apache2: /etc/apache2
```

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 **mlocate** 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.

The following examples show the installation of the **tcpdump** command on each of our example systems. **tcpdump** is a packet capture tool that lets you view the raw packets being sent to and from the system on the network.

Debian and Ubuntu use APT, the Debian Advanced Package Tool:

```
ubuntu# sudo apt-get install tcpdump
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  tcpdump
0 upgraded, 1 newly installed, 0 to remove and 81 not upgraded.
Need to get 0 B/360 kB of archives.
After this operation, 1,179 kB of additional disk space will be used.
Selecting previously unselected package tcpdump.
(Reading database ... 63846 files and directories currently installed.)
Preparing to unpack .../tcpdump_4.6.2-4ubuntu1_amd64.deb ...
Unpacking tcpdump (4.6.2-4ubuntu1) ...
Processing triggers for man-db (2.7.0.2-5) ...
Setting up tcpdump (4.6.2-4ubuntu1) ...
```

RHEL

The Red Hat and CentOS version is

```
redhat# sudo yum install tcpdump
Loaded plugins: fastestmirror
Determining fastest mirrors
 * base: mirrors.xmission.com
 * epel: linux.mirrors.es.net
 * extras: centos.arvixe.com
 * updates: repos.lax.quadranet.com
```

```
Resolving Dependencies
--> Running transaction check
---> Package tcpdump.x86_64 14:4.5.1-2.el7 will be installed
--> Finished Dependency Resolution
tcpdump-4.5.1-2.el7.x86_64.rpm                   | 387 kB  00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : 14:tcpdump-4.5.1-2.el7.x86_64   1/1
  Verifying  : 14:tcpdump-4.5.1-2.el7.x86_64   1/1
Installed:
  tcpdump.x86_64 14:4.5.1-2.el7
Complete!
```

The package manager for FreeBSD is **pkg**.

```
freebsd# sudo pkg install -y tcpdump
Updating FreeBSD repository catalogue...
Fetching meta.txz:          100%    944 B   0.9kB/s    00:01
Fetching packagesite.txz:   100%    5 MiB   5.5MB/s    00:01
Processing entries: 100%
FreeBSD repository update completed. 24632 packages processed.
All repositories are up-to-date.
The following 2 package(s) will be affected (of 0 checked):

New packages to be INSTALLED:
     tcpdump: 4.7.4
     libsmi: 0.4.8_1

The process will require 17 MiB more space.
2 MiB to be downloaded.
Fetching tcpdump-4.7.4.txz:   100%  301 KiB 307.7kB/s    00:01
Fetching libsmi-0.4.8_1.txz:  100%    2 MiB   2.0MB/s    00:01
Checking integrity... done (0 conflicting)
[1/2] Installing libsmi-0.4.8_1...
[1/2] Extracting libsmi-0.4.8_1: 100%
[2/2] Installing tcpdump-4.7.4...
[2/2] Extracting tcpdump-4.7.4: 100%
```

Building software from source code

As an illustration, here's how you build a version of **tcpdump** from the source code.

The first chore is to identify the code. Software maintainers sometimes keep an index of releases on the project's web site that are downloadable as tarballs. For open source projects, you're most likely to find the code in a Git repository.

The **tcpdump** source is kept on GitHub. Clone the repository in the **/tmp** directory, create a branch of the tagged version you want to build, then unpack, configure, build, and install it:

```
redhat$ cd /tmp
redhat$ git clone https://github.com/the-tcpdump-group/tcpdump.git
<status messages as repository is cloned>
redhat$ cd tcpdump
redhat$ git checkout tags/tcpdump-4.7.4 -b tcpdump-4.7.4
Switched to a new branch 'tcpdump-4.7.4'
redhat$ ./configure
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
...
redhat$ make
<several pages of compilation output>
redhat$ sudo make install
<files are moved in to place>
```

This **configure**/**make**/**make install** sequence is common to most software written in C and works on all UNIX and Linux systems. It's always a good idea to check the package's **INSTALL** or **README** file for specifics. You must have the development environment and any package-specific prerequisites installed. (In the case of **tcpdump**, **libpcap** and its libraries are prerequisites.)

You'll often need to tweak the build configuration, so use **./configure --help** to see the options available for each particular package. Another useful **configure** option is **--prefix=***directory*, which lets you compile the software for installation somewhere other than **/usr/local**, which is usually the default.

Installing from a web script

Cross-platform software bundles increasingly offer an expedited installation process that's driven by a shell script you download from the web with **curl**, **fetch**, or **wget**.[2] For example, to set up a machine as a Salt client, you can run the following commands:

```
$ curl -o /tmp/saltboot -sL https://bootstrap.saltstack.com
$ sudo sh /tmp/saltboot
```

The bootstrap script investigates the local environment, then downloads, installs, and configures an appropriate version of the software. This type of installation is particularly common in cases where the process itself is somewhat complex, but the vendor is highly motivated to make things easy for users. (Another good example is RVM; see page 232.)

2. These are all simple HTTP clients that download the contents of a URL to a local file or, optionally, print the contents to their standard output.

See Chapter 6 for more information about package installation.

This installation method is perfectly fine, but it raises a couple of issues that are worth mentioning. To begin with, it leaves no proper record of the installation for future reference. If your operating system offers a packagized version of the software, it's usually preferable to install the package instead of running a web installer. Packages are easy to track, upgrade, and remove. (On the other hand, most OS-level packages are out of date. You probably won't end up with the most current version of the software.)

See page 1007 for details on HTTPS's chain of trust.

Be very suspicious if the URL of the boot script is not secure (that is, it does not start with https:). Unsecured HTTP is trivial to hijack, and installation URLs are of particular interest to hackers because they know you're likely to run, as root, whatever code comes back. By contrast, HTTPS validates the identity of the server through a cryptographic chain of trust. Not foolproof, but reliable enough.

A few vendors publicize an HTTP installation URL that automatically redirects to an HTTPS version. This is dumb and is in fact no more secure than straight-up HTTP. There's nothing to prevent the initial HTTP exchange from being intercepted, so you might never reach the vendor's redirect. However, the existence of such redirects does mean it's worth trying your own substitution of https for http in insecure URLs. More often than not, it works just fine.

The shell accepts script text on its standard input, and this feature enables tidy, one-line installation procedures such as the following:

```
$ curl -L https://badvendor.com | sudo sh
```

However, there's a potential issue with this construction in that the root shell still runs even if **curl** outputs a partial script and then fails—say, because of a transient network glitch. The end result is unpredictable and potentially not good.

We are not aware of any documented cases of problems attributable to this cause. Nevertheless, it is a plausible failure mode. More to the point, piping the output of **curl** to a shell has entered the collective sysadmin unconscious as a prototypical rookie blunder, so if you must do it, at least keep it on the sly.

The fix is easy: just save the script to a temporary file, then run the script in a separate step after the download successfully completes.

1.11 Where to host

Operating systems and software can be hosted in private data centers, at co-location facilities, on a cloud platform, or on some combination of these. Most burgeoning startups choose the cloud. Established enterprises are likely to have existing data centers and may run a private cloud internally.

The most practical choice, and our recommendation for new projects, is a public cloud provider. These facilities offer numerous advantages over data centers:

- No capital expenses and low initial operating costs
- No need to install, secure, and manage hardware
- On-demand adjustment of storage, bandwidth, and compute capacity
- Ready-made solutions for common ancillary needs such as databases, load balancers, queues, monitoring, and more
- Cheaper and simpler implementation of highly available/redundant systems

Early cloud systems acquired a reputation for inferior security and performance, but these are no longer major concerns. These days, most of our administration work is in the cloud. See Chapter 9 for a general introduction to this space.

Our preferred cloud platform is the leader in the space: Amazon Web Services (AWS). Gartner, a leading technology research firm, found that AWS is ten times the size of all competitors combined. AWS innovates rapidly and offers a much broader array of services than does any other provider. It also has a reputation for excellent customer service and supports a large and engaged community. AWS offers a free service tier to cut your teeth on, including a year's use of a low powered cloud server.

Google Cloud Platform (GCP) is aggressively improving and marketing its products. Some claim that its technology is unmatched by other providers. GCP's growth has been slow, in part due to Google's reputation for dropping support for popular offerings. However, its customer-friendly pricing terms and unique features are appealing differentiators.

DigitalOcean is a simpler service with a stated goal of high performance. Its target market is developers, whom it woos with a clean API, low pricing, and extremely fast boot times. DigitalOcean is a strong proponent of open source software, and their tutorials and guides for popular Internet technologies are some of the best available.

1.12 Specialization and adjacent disciplines

System administrators do not exist in a vacuum; a team of experts is required to build and maintain a complex network. This section describes some of the roles with which system administrators overlap in skills and scope. Some administrators choose to specialize in one or more of these areas.

Your goal as a system administrator, or as a professional working in any of these related areas, is to achieve the objectives of the organization. Avoid letting politics or hierarchy interfere with progress. The best administrators solve problems and share information freely with others.

DevOps

See page 1106 for more comments on DevOps.

DevOps is not so much a specific function as a culture or operational philosophy. It aims to improve the efficiency of building and delivering software, especially at large sites that have many interrelated services and teams. Organizations with a DevOps practice promote integration among engineering teams and may draw little or no distinction between development and operations. Experts who work in this area seek out inefficient processes and replace them with small shell scripts or large and unwieldy Chef repositories.

Site reliability engineers

Site reliability engineers value uptime and correctness above all else. Monitoring networks, deploying production software, taking pager duty, planning future expansion, and debugging outages all lie within the realm of these availability crusaders. Single points of failure are site reliability engineers' nemeses.

Security operations engineers

Security operations engineers focus on the practical, day-to-day side of an information security program. These folks install and operate tools that search for vulnerabilities and monitor for attacks on the network. They also participate in attack simulations to gauge the effectiveness of their prevention and detection techniques.

Network administrators

Network administrators design, install, configure, and operate networks. Sites that operate data centers are most likely to employ network administrators; that's because these facilities have a variety of physical switches, routers, firewalls, and other devices that need management. Cloud platforms also offer a variety of networking options, but these usually don't require a dedicated administrator because most of the work is handled by the provider.

Database administrators

Database administrators (sometimes known as DBAs) are experts at installing and managing database software. They manage database schemas, perform installations and upgrades, configure clustering, tune settings for optimal performance, and help users formulate efficient queries. DBAs are usually wizards with one or more query languages and have experience with both relational and nonrelational (NoSQL) databases.

Network operations center (NOC) engineers

NOC engineers monitor the real-time health of large sites and track incidents and outages. They troubleshoot tickets from users, perform routine upgrades, and coordinate actions among other teams. They can most often be found watching a wall of monitors that show graphs and measurements.

Data center technicians

Data center technicians work with hardware. They receive new equipment, track equipment inventory and life cycles, install servers in racks, run cabling, maintain power and air conditioning, and handle the daily operations of a data center. As a system administrator, it's in your best interest to befriend data center technicians and bribe them with coffee, caffeinated soft drinks, and alcoholic beverages.

Architects

Systems architects have deep expertise in more than one area. They use their experience to design distributed systems. Their job descriptions may include defining security zones and segmentation, eliminating single points of failure, planning for future growth, ensuring connectivity among multiple networks and third parties, and other site-wide decision making. Good architects are technically proficient and generally prefer to implement and test their own designs.

1.13 Recommended reading

Abbott, Martin L., and Michael T. Fisher. *The Art of Scalability: Scalable Web Architecture, Processes, and Organizations for the Modern Enterprise (2nd Edition)*. Addison-Wesley Professional, 2015.

Gancarz, Mike. *Linux and the Unix Philosophy*. Boston: Digital Press, 2003.

Limoncelli, Thomas A., and Peter Salus. *The Complete April Fools' Day RFCs.* Peer-to-Peer Communications LLC. 2007. Engineering humor. You can read this collection on-line for free at rfc-humor.com.

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

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.

Siever, Ellen, Stephen Figgins, Robert Love, and Arnold Robbins. *Linux in a Nutshell (6th Edition)*. Sebastopol, CA: O'Reilly Media, 2009.

System administration and DevOps

Kim, Gene, Kevin Behr, and George Spafford. *The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win.* Portland, OR: IT Revolution Press, 2014. A guide to the philosophy and mindset needed to run a modern IT organization, written as a narrative. An instant classic.

Kim, Gene, Jez Humble, Patrick Debois, and John Willis. *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations.* Portland, OR: IT Revolution Press, 2016.

Limoncelli, Thomas A., Christina J. Hogan, and Strata R. Chalup. *The Practice of System and Network Administration (2nd 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.

Limoncelli, Thomas A., Christina J. Hogan, and Strata R. Chalup. *The Practice of Cloud System Administration.* Reading, MA: Addison-Wesley, 2014. From the same authors as the previous title, now with a focus on distributed systems and cloud computing.

Essential tools

Blum, Richard, and Christine Bresnahan. *Linux Command Line and Shell Scripting Bible (3rd Edition).* Wiley, 2015.

Dougherty, Dale, and Arnold Robins. *Sed & Awk (2nd Edition).* Sebastopol, CA: O'Reilly Media, 1997. Classic O'Reilly book on the powerful, indispensable text processors **sed** and **awk**.

Kim, Peter. *The Hacker Playbook 2: Practical Guide To Penetration Testing.* CreateSpace Independent Publishing Platform, 2015.

Neil, Drew. *Practical Vim: Edit Text at the Speed of Thought*. Pragmatic Bookshelf, 2012.

Shotts, William E. *The Linux Command Line: A Complete Introduction.* San Francisco, CA: No Starch Press, 2012.

Sweigart, Al. *Automate the Boring Stuff with Python: Practical Programming for Total Beginners.* San Francisco, CA: No Starch Press, 2015.