Scott Granneman

ESSENTIAL CODE AND COMMANDS

Second Edition

Linux®

PHRASEBOOK

FREE SAMPLE CHAPTER

SHARE WITH OTHERS

Linux

PHRASEBOOK

SECOND EDITION

Scott Granneman

Addison-Wesley
800 East 96th Street, Indianapolis, Indiana 46240 USA

Linux Phrasebook, Second Edition

Library of Congress Control Number: 2015955629

ISBN-13: 978-0-321-83388-4
ISBN-10: 0-321-83388-0

First printing: December 2015

Trademarks
Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

Warning and Disclaimer
The author and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

Special Sales
For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact international@pearsoned.com.

Visit us on the Web: informit.com/aw

Acquisitions Editor
Mark Taber

Managing Editor
Sandra Schroeder

Senior Project Editor
Tonya Simpson

Copy Editor
Anne Goebel

Senior Indexer
Cheryl Lenser

Proofreader
Laura Hernandez

Technical Reviewer
Brian Tiemann

Editorial Assistant
Vanessa Evans

Cover Designer
Chuti Prasertsith

Compositor
Mary Sudul

Table of Contents

About the Author

Scott Granneman is an author, educator, and small business owner. He has written seven books (about Firefox, Linux, Google Apps, and Mac OS X) and contributed to two. In addition, he was a columnist for *SecurityFocus*, one of the largest and most important security-focused sites on the Web, and *Linux Magazine* while it was in print.

As an educator, he has taught thousands of people of all ages—from preteens to senior citizens—on a wide variety of topics, including both literature and technology. He is currently an adjunct professor at Washington University in St. Louis and at Webster University, where he teaches a variety of courses about technology, social media, the Internet, and Web development. With the shift in focus over the past few decades to Linux and other open-source technologies, he has worked to bring knowledge of these powerful new directions in software to people at all technical skill levels.

As a Principal of WebSanity, a website planning, development, and hosting firm with clients in 12 states, he manages the firm's Linux-based servers and infrastructure, researches new technologies, and works closely with other partners on the underlying WebSanity Content Management System (CMS).

Dedication

This book is dedicated to Linux users,
both old and new. Welcome!

Acknowledgments for the First Edition (2005)

No one learns about the Linux shell in a vacuum, and I have hundreds of writers, working over decades, to thank for educating me about the awesome possibilities provided by the Linux command line. Books, websites, blogs, handouts at LUG meetings: All helped me learn about `bash` and the beauty of the Linux command line, and they continue to teach me today. If I can give back just a fraction of what I have absorbed, I'll be satisfied.

In addition to that general pool of knowledgeable individuals, I'd like to thank the people (and animals) who gave me help and support during the writing of *Linux Phrasebook*.

My agent, Laura Lewin, who has been helpful in too many ways for me to recount.

My editors at Pearson, who gave me the opportunity to write this book in the first place and have encouraged me whenever I needed prodding.

Robert Citek provided invaluable help with RPM and was always there if I had a question. That man knows Linux.

My business partner and lifelong buddy, Jans Carton, helped me focus when I needed it, and was a (mostly) willing guinea pig for many new commands and options. Looking back at that day in fifth grade when we met, who'da thunk it?

Jerry Bryan looked over everything I wrote and fixed all the little grammatical mistakes and typos I made. I promise, Jerry: One day I'll learn the difference between "may" and "might"!

My wife, Denise Lieberman, patiently listened to me babble excitedly whenever I figured out something cool, even though she had absolutely no idea what I was talking about. That's true love. Thanks, Denise!

Finally, I must point to my cute lil' Shih Tzu, Libby, who always knew exactly the right time to put her front paws on my leg and demand ear scratches and belly rubs.

Acknowledgments for the Second Edition (2015)

For the second edition, many things have changed (sadly, Libby has gone to that great dog house in the sky), but one thing is the same: Lots of folks helped me during the writing of this book, and they deserve recognition and my gratitude.

Mark Taber at Pearson supported me over a *very* lengthy writing period. I can't thank you enough, Mark. You are why this book exists today.

My wife, Robin Woltman, read through every chapter and made many invaluable edits and suggestions. You did a great job, Robin!

Robert Citek looked over many of the chapters and made suggestions and technical edits that were always right on the nose. He's who I turn to when I need help!

Craig Buchek looked over a chapter or two and made some good points. As the former benevolent dictator for life of the St. Louis Linux Users Group, that was expected.

Tom Kirk loaned me a few laptops from his enormous collection that enabled me to test out things from the networking sections. Tom, you are a lifesaver, and the best hardware guy I know.

In addition to my friends above, the following people emailed to let me know about problems in the first edition.

- Joe Hancharik pointed out a problem in Chapter 1's "Wildcards and What They Mean" that was a very silly mistake on my part.
- William H. Ferguson found another silly mistake I made in Chapter 3's "Copy Files."
- Brian Greer pointed out a problem I should have noticed in Chapter 15's "Troubleshooting Network Problems."

Thank you, gentlemen! It's readers like Joe, William, and Brian that make writing a book fun. Feedback is a great thing, even if it is a bug report. If you see an error, let me know, so I can fix it for the third edition!

We Want to Hear from You!

As the reader of this book, *you* are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

We welcome your comments. You can email or write directly to let us know what you did or didn't like about this book—as well as what we can do to make our books better.

Please note that we cannot help you with technical problems related to the topic of this book, and that due to the high volume of mail we receive, we might not be able to reply to every message.

When you write, please be sure to include this book's title and author, as well as your name and phone number or email address.

E-mail: feedback@developers-library.info

Mail: Reader Feedback
Addison-Wesley Developer's Library
800 East 96th Street
Indianapolis, IN 46240 USA

Reader Services

Visit our website and register this book at **www.informit.com/register** for convenient access to any updates, downloads, or errata that might be available for this book.

Introduction

Among key Linux features, the command-line shell is one of the most important. If you run a Linux server, your main interface is more than likely going to be the shell. If you're a power user running Linux on the desktop, you probably have a terminal open at all times. If you're a Linux newbie, you may think that you'll never open up the command line, but you will sometime … and the more you use Linux, the more you're going to want to use that shell.

The shell in many ways is the key to Linux's power and elegance. You can do things with the command line that you simply can't do with whatever GUI you favor. No matter how powerful KDE or GNOME may be (or IceWM or XFCE or any of the other kajillion windowing environments out there), you will always be able to do many things faster and more efficiently with a terminal. If you want to master Linux, you need to begin by mastering the Linux command line.

The traditional method has been to use the Linux `man` pages. While `man` pages are useful, they are often not enough, for one simple reason: They lack examples. Oh, a few `man` pages here and there have a few examples, but by and large, good examples are hard to come by. This presents a real problem for users at all experience levels: It's one thing to see options listed

and explained, but it's another thing entirely to see those options used in real-world situations.

This book is all about those missing examples. I've been using Linux for two decades, and I consider myself pretty knowledgeable about this amazing, powerful operating system. I'm so addicted to the command line that I always have a terminal open. And to top it off, the Linux servers my company relies on don't have GUIs on them (just the way I like it!), so I *have* to use the terminal to work with them. But I'm always lamenting—along with my Linux-using friends, acquaintances, and LUG members—the dearth of examples found in `man` pages. When I was asked to write *Linux Phrasebook*, and told that it was to consist of hundreds of examples illustrating the most important Linux commands, I replied, "I can't wait! That's a book I'd buy in a heartbeat!"

You're holding the result in your hands: a book about the Linux commands you just have to know, with examples illustrating how to use each and every one. This is a reference book that will be useful now and for years to come, but I also hope you find it enjoyable as well, and even a little fun.

NOTE: Visit our website and register this book at informit.com/register for convenient access to any updates, downloads, or errata that might be available for this book.

Audience for This Book

I've written this book to be useful both to beginners—the folks that show up to meetings of our Linux Users Group seeking guidance and a helping hand as they begin the adventure of using Linux—and to experienced users who use the shell for everything from systems administration to games to programming. If you've just started using Linux, this book will help teach you about the shell and its power; if you've been using Linux for years and years, *Linux Phrasebook* will teach you some new tricks and remind you of some features you'd long ago forgotten.

There are many shells out there—csh, tcsh, zsh, to name but a few—but I use the default shell for virtually every Linux distro: bash, the Bourne Again Shell. The bash shell is not only ubiquitous, but also powerful and flexible. After you get comfortable with bash, you may choose to explore other options, but knowledge of bash is required in the world of Linux.

I wrote this book using Debian, which is about as generic a Linux distro as you can get, as well as one of the most widely used. Even though I used Debian, the commands I discuss should work on your distro as well. The only major difference comes when you run a command as root. Instead of logging in as root, some distros (like Ubuntu, for instance) encourage the use of the `sudo` command; in other words, instead of running `lsof firefox` as root, an Ubuntu user would run `sudo lsof firefox`.

NOTE: Sharp-eyed readers may note that, in the first edition, I used Ubuntu (or, as I referred to it there, K/Ubuntu—I was trying to emphasis that I was using KDE with Ubuntu). Now I've gone more generic and I just use straight Debian, the distro upon which Ubuntu is based.

In order to appeal to the widest number of readers out there, I showed the commands as though you have to run them as root, without `sudo`. If you see a `#` in front of a command, that's the shell indicating that root is logged in, which means you need to be root to run that command, or utilize `sudo` if you're using Ubuntu or a similar distro.

One final thing: In order to keep this book from being even longer, I've often truncated the outputs of commands. For instance, on your Linux computer, you'd normally see the following output after entering `ls -l`:

```
-rwxr-xr-x 1 scott admins 1261 Jun 1 2012 script.sh
```

You will see that in a few places in this book in which it's appropriate, but often you'll instead find something like this:

```
-rw-r--r-- 1 scott admins script.sh
```

In that case, I cut out the stuff that wasn't important to my point, which helped keep the output to only one line instead of two. You'd be surprised how those lines can add up, which is why I made those changes when it seemed appropriate (or when my editor was about to freak out at the number of pages the book was going to have!!).

TIP: Much of the information I provide about Linux in this book can also be applied to other flavors of UNIX, such as BSD and OS X. Note that I did not say *all* or *most*—I said *much*. As long as you keep that in mind, you'll find *Linux Phrasebook* to be helpful with those operating systems as well.

About the Second Edition

When Mark Taber, my editor at Pearson, first approached me about writing a second edition of *Linux Phrasebook*, I jumped at the chance. I actually use my own book as a reference a few times a month, so over the years, I've noticed errors (wincing every time) and have seen many things that I'd like to change, remove, or add.

My goal has been to make the second edition a must-buy for new readers as well as for owners of the first edition of *Linux Phrasebook*. This isn't some so-called new edition with just a few small changes. Oh no. So here's what I've done that's new or different in this book:

- I split the former Chapter 2, "The Basics," into two chapters: Chapters 2, "Navigating Your File System," and 3, "Creation and Destruction." The old Chapter 2 was ridiculously long, and it crammed together different kinds of commands. The new split makes things far more manageable and sensible (although it did mean every chapter after those two had to be re-numbered).
- I removed Chapters 6, "Printing and Managing Print Jobs," and 16, "Windows Networking," because they just don't seem as vital as they did ten years ago. In addition, most people who need

to print or connect to and work on a Windows-based network will have GUI tools that more than adequately do those jobs. However, don't despair—you can still find those original chapters from the first edition on my website, www.granneman.com/linux-redaction.

- I added a new chapter: Chapter 7, "Manipulating Text Files with Filters." There is a *large* amount of great new information there, and I know you will find it very useful!
- I removed sections from the old Chapters 2 (now Chapters 2 and 3), 3 (now 4), 7 (8), 8 (9), 9 (10), 10 (11), and 14 (15). Again, you will find those original sections from the first edition on my website, www.granneman.com/linux-redaction.
- I added new sections in Chapters 1–6 and 8–13. And I almost doubled the size of Chapter 15 by keeping the deprecated commands (since they're still on most distros) while adding all the new ones.
- I moved sections into new chapters where they made sense, which you can especially see in Chapter 8.
- I revised things in every chapter of the book. I fixed mistakes, rewrote parts that were unclear, added additional notes and tips, and improved examples by adding or revising text.
- I mentioned many more commands in passing, like `ssh-agent`, `wput`, `htop`, `dnf`, `pandoc`, `rename`, `whoami`, and `iconv`.
- I provided little tidbits of information about many different things that are good to know about, like variables, `for` loops, cron jobs, arguments, and `sources`. Oh, and there's more H.P. Lovecraft!

Finally, a tip: If any links are broken or don't work any longer, try entering them at the Internet Archive's Wayback Machine, which you can find at https://archive.org, to see if you can still find the content. Then let me know so I can fix that reference in future editions.

Thank you for reading this far, and I really hope you enjoy this new edition of *Linux Phrasebook*!

Conventions Used in This Book

This book uses the following conventions.

- `Monospace` type is used to differentiate between code/programming-related terms and regular English, and to indicate text that should appear on your screen. For example:

  The `df` command shows results in kilobytes by default, but it's usually easier to comprehend if you instead use the `-h` (or `--human-readable`) option.

  ```
  It will look like this to mimic the way text
  looks on your screen.
  ```

- An arrow (➥) at the beginning of a line of code means that a single line of code is too long to fit on the printed page. It signifies to readers that the author meant for the continued code to appear on the same line.
- In addition to this, the following elements are used to introduce other pertinent information used in this book.

NOTE: A note presents interesting pieces of information related to the surrounding discussion.

TIP: A tip offers advice or teaches an easier way to do something.

CAUTION: A caution advises you about potential problems and helps you steer clear of disaster.

This page intentionally left blank

2

Navigating Your File System

This chapter introduces the basic commands you'll find yourself using several times every day. Think of these as the hammer, screwdriver, and pliers that a carpenter keeps in the top of his toolbox. After you learn these commands, you can start controlling your shell and finding out all sorts of interesting things about your files, folders, data, and environment. In particular, you'll be learning about some of the metadata—the data describing your data—that Linux has to keep track of, and it may just surprise you how much there is.

NOTE: When I updated this book for its second edition, I removed the section about `mkdir -v` (which shows you what `mkdir` is doing as it does it) and `rm -v` (which does the same thing, but for `rm`). You can find the original text on my website, www.granneman.com/linux-redactions.

Also, I took the sections on `touch`, `mkdir`, `cp`, `mv`, `rm`, and `rmdir` and used them to create a new Chapter 3 titled "Creation and Destruction" (which of course renumbered everything after it!). Finally, the section on `su` was moved to Chapter 8, "Ownership and Permissions," which makes a lot more sense.

List Files and Folders

```
ls
```

The `ls` command is probably the one that people find themselves using the most. After all, before you can manipulate and use files in a directory (remember, *file* and *directory* are interchangeable), you first have to know what files are available. That's where `ls` comes in, as it lists the files and subdirectories found in a directory.

NOTE: The `ls` command might sound simple—just show me the files!—but there are a surprising number of permutations to this amazingly pliable command, as you'll see.

Typing `ls` lists the contents of the directory in which you're currently working. When you first log in to your shell, you'll find yourself in your home directory. Enter `ls`, and you might see something like the following:

```
$ ls
alias Desktop   iso   pictures program_files todo
bin   documents music podcasts src           videos
```

List the Contents of Other Folders

```
ls [folder]
```

You don't have to be in a directory to find out what's in it. Let's say that you're in your home directory, but you want to find out what's in the `music` directory.

Simply type the `ls` command, followed by the folder whose contents you want to view, and you'll see this:

```
$ ls music
Buddy_Holly  Clash  Donald_Fagen  new
```

In the previous example, a relative path is used, but absolute paths work just as well.

```
$ ls /home/scott/music
Buddy_Holly  Clash  Donald_Fagen  new
```

The ability to specify relative or absolute paths can be incredibly handy when you don't feel like moving all over your file system every time you want to view a list of the contents of a directory. Not sure if you still have that video of a Bengal Tiger your brother took at the zoo? Try this (~ represents your home directory):

```
$ ls ~/videos
airhorn_surprise.wmv
apple_knowledge_navigator.mov
b-ball-e-mail.mov
carwreck.mpg
nerdtv_1_andy_hertzfeld
nerdtv_2_max_levchin_paypal
nerdtv_3_bill_joy
tiger.wmv
Ubuntu_Talk-Mark_Shuttleworth.mpeg
```

Yes, there it is: `tiger.wmv`.

List Folder Contents Using Wildcards

```
ls *
```

You just learned how to find a file by listing out its entire directory; but there's a faster method, especially for really long directory listings. If you know that the video of the Bengal Tiger you're looking for is in Windows Media format (Boo! Hiss!), and therefore ends with `.wmv`, you can use a wildcard to show just the files ending with that particular extension.

```
$ ls ~/videos
airhorn_surprise.wmv
apple_knowledge_navigator.mov
b-ball-e-mail.mov
carwreck.mpg
nerdtv_1_andy_hertzfeld
nerdtv_2_max_levchin_paypal
nerdtv_3_bill_joy
tiger.wmv
Ubuntu_Talk-Mark_Shuttleworth.mpeg
$ ls ~/videos/*.wmv
airhorn_surprise.wmv   tiger.wmv
```

There's another faster method, also involving wildcards: Look just for files that contain the word *tiger*.

```
$ ls ~/videos/*tiger*
tiger.wmv
```

NOTE: If your wildcard results in a match with a directory, the contents of that directory will be listed as well. If you want to omit that match and avoid showing the contents of subdirectories, add the `-d` option.

View a List of Files in Subfolders

```
ls -R
```

You can also view the contents of several subdirectories with one command. Say you're at a Linux Users Group (LUG) meeting and installations are occurring around you fast and furious. "Hey," someone hollers out, "does anyone have an ISO image of the new Kubuntu that I can use?" You think you downloaded that recently, so to be sure, you run the following command (instead of `ls -R`, you could have also used `ls --recursive`):

```
$ ls -R ~/iso
/home/scott/iso:
debian-6.0.4-i386-CD-1.iso  knoppix  ubuntu

/home/scott/iso/knoppix:
KNOPPIX_V7.2.0CD-2013-06-16-EN.iso
➥ KNOPPIX_V7.4.2DVD-2014-09-28-EN.iso

/home/scott/iso/ubuntu:
kubuntu-15.04-desktop-amd64.iso
➥ ubuntu-15.04-desktop-amd64.iso
ubuntu-14.04.3-server-amd64.iso
```

There it is, in `~/iso/ubuntu: kubuntu-15.04-desktop-amd64.iso`. The `-R` option traverses the `iso` directory recursively, showing you the contents of the main `iso` directory and every subdirectory as well. Each folder is introduced with its path—relative to the directory in which you started—followed by a colon, and then the items in that folder are listed. Keep in mind that the recursive option becomes less useful when you have many items in many subdirectories, as the listing goes

on for screen after screen, making it hard to find the particular item for which you're looking. Of course, if all you want to do is verify that there are many files and folders in a directory, it's useful just to see everything stream by, but that won't happen very often.

View a List of Contents in a Single Column

```
ls -1
```

So far, you've just been working with the default outputs of `ls`. Notice that `ls` prints the contents of the directory in alphabetical columns, with a minimum of two spaces between each column for readability. But what if you want to see the contents in a different manner?

If multiple columns aren't your thing, you can instead view the results of the `ls` command as a single column using, logically enough, `ls -1` (or `ls --format=single-column`).

```
$ ls -1 ~/bin
Desktop
documents
iso
music
pictures
podcaststodo
videos
```

This listing can get out of hand if you have an enormous number of items in a directory, and more so if you use the recursive option as well, as in `ls -1R ~/`. Be prepared to press Ctrl+C to cancel the command

if a list is streaming down your terminal with no end in sight.

View Contents As a Comma-Separated List

```
ls -m
```

Another option for those who can't stand columns of any form, whether it's one or many, is the `-m` option (or `--format=commas`).

```
$ ls -m ~/
alias, bin, Desktop, documents, iso, music, pictures,
➥ podcasts, program_files, src, todo, videos
```

Think of the *m* in `-m` as a mnemonic for *comma*, and it is easier to remember the option. Of course, this option is also useful if you're writing a script and need the contents of a directory in a comma-separated list, but that's a more advanced use of a valuable option.

View Hidden Files and Folders

```
ls -a
```

Up to this point, you've been viewing the visible files in directories, but don't forget that many directories contain hidden files in them as well. Your home directory, for instance, is just bursting with hidden files and folders, all made invisible by the placement of a `.` at the beginning of their names. If you want to view these hidden elements, just use the `-a` option (or `--all`).

```
$ ls -a ~/
.              .gimp-2.2        .openoffice.org1.9.95
..             .gksu.lock       .openoffice.org1.9
.3ddesktop     .glade2          .openoffice.org2
.abbrev_defs   .gnome           .opera
.acrorc        .gnome2          .padminrc
.adobe         .gnome2_private  pictures
alias          .gnome_private   podcasts
[List condensed due to length]
```

You should know several things about this listing. First, `ls -a` displays both hidden and unhidden items, so you see both `.gnome` and `pictures`. Second, you'll always see the `.` and `..` because `.` refers to the current directory, while `..` points to the directory above this one, the parent directory. These two hidden files exist in every single folder on your system, and you can't get rid of them. Expect to see them every time you use the `-a` option. Finally, depending on the directory, the `-a` option can reveal a great number of hidden items of which you weren't aware.

TIP: I just said that `ls -a` shows you everything beginning with a dot, including the folder references `.` and `..`; but if you don't want to see those two, use `ls -A` instead.

Visually Display a File's Type

```
ls -F
```

The `ls` command doesn't tell you much about an item in a directory besides its name. By itself, it's hard to tell if an item is a file, a directory, or something else.

An easy way to solve this problem and make `ls` really informative is to use the `-F` option (or `--classify`).

```
$ ls -F ~/bin
adblock_filters.txt   fixm3u*        pix2tn.pl*
addext*               flash.xml*     pop_login*
address_book.csv      getip*         procmail/
address_book.sxc      homesize*      programs_usual*
address_book.xls      html2text.py*  quickrename*
backup_to_chaucer*    list-urls.py*
```

This tells you quite a bit. An `*` or asterisk after a file means that it is executable, while a `/` or forward slash indicates a directory. If the filename lacks any sort of appendage at all, it's just a regular ol' file. Other possible endings are shown in Table 2.1.

Table 2.1 **Symbols and File Types**

Character	Meaning
*	Executable
/	Directory
@	Symbolic link
|	FIFO (AKA a *named pipe*)
=	Socket

Display Contents in Color

```
ls --color
```

In addition to the symbols that are appended to files and folders when you use the `-F` option, you can also ask your shell to display things in color, which gives an additional way to classify items and tell them

apart. Many Linux installs come with colors already enabled for shells, but if yours does not, just use the `--color` option (I know you can't see the colors, so just pretend).

```
$ ls --color
adblock_filters.txt   fixm3u       pix2tn.pl
addext                flash.xml    pop_login
address_book.csv      getip        procmail
address_book.sxc      homesize     programs_kill
address_book.xls      html2text.py programs_usual
backup_ssh_to_chaucer list-urls.py quickrename
```

In this setup, executable files are green, folders are blue, and normal files are black (which is the default color for text in my shell). Table 2.2 gives you the full list of common color associations (but keep in mind that these colors may vary on your particular distro).

Table 2.2 **Colors and File Types**

Color	Meaning
Default shell text color	Regular file
Green	Executable
Blue	Directory
Magenta	Symbolic link
Yellow	FIFO
Magenta	Socket
Red	Archive (`.tar`, `.zip`, `.deb`, `.rpm`)
Magenta	Images (`.jpg`, `.gif`, `.png`, `.tiff`)
Magenta	Audio (`.mp3`, `.ogg`, `.wav`)

TIP: Want to see what colors are mapped to the various kinds of files on your system? Enter `dircolors --print-database`, and then read the results carefully. You can also use the `dircolors` command to change those colors as well.

With the combination of `--color` and `-F`, you can see at a glance what kinds of files you're working with in a directory. Now we're cookin' with gas!

```
$ ls -F --color
adblock_filters.txt    fixm3u*       pix2tn.pl*
addext*                flash.xml*    pop_login*
address_book.csv       getip*        procmail/
address_book.sxc       homesize*     programs_kill*
address_book.xls       html2text.py* programs_usual*
backup_ssh_to_chaucer* list-urls.py* quickrename*
```

List Permissions, Ownership, and More

```
ls -l
```

You've now learned how to format the results of `ls` to tell you more about the contents of directories, but what about the actual contents themselves? How can you learn more about the files and folders, such as their size, their owners, and who can do what with them? For that information, you need to use the `-l` option (or `--format=long`).

```
$ ls -l ~/bin
total 2951
-rw-r--r--  1 scott scott  15058 2015-10-03 18:49
➥adblock_filters.txt
-rwxr-xr--  1 scott root      33 2015-04-19 09:45
➥addext
-rw-r--r--  1 scott scott  84480 2015-04-19 09:45
➥addressbook.xls
-rwxr--r--  1 scott scott     55 2015-04-19 09:45
➥batchprint_home
drwxr-xr-x  9 scott scott   1080 2015-09-22 14:42
➥bin_on_bacon
-rwxr-xr--  1 scott scott    173 2015-04-19 09:45
➥changeext
-rwxr-xr--  1 scott root     190 2015-04-19 09:45
➥convertsize
drwxr-xr-x  2 scott scott     48 2015-04-19 09:45
➥credentials
[List condensed and edited due to length]
```

The `-l` option stands for *long*, and as you can see, it provides a wealth of data about the files found in a directory. Let's move from right to left and discuss what you see.

On the farthest right is the easiest item: the name of the listed item. Want `ls` to display more about it? Then add the `-F` option to `-l`, like this: `ls -lF`. Color is easily available as well, with `ls -lF --color`.

Moving left, you next see a date and time. This is when the file was last modified, including the date (in year-month-day format) and then the time (in 24-hour military time).

Further left is a number that indicates the size of the item, in bytes. This is a bit tricky with folders—for instance, the previous readout says that `bin_on_bacon` is 1080 bytes, or just a little more than one kilobyte,

yet it contains 887KB of content inside it. The `credentials` directory, according to `ls -l`, is 48 bytes, but contains nothing inside it whatsoever! What is happening?

Remember in Chapter 1, "Things to Know About Your Command Line," when you learned that directories are just special files that contain a list of their contents? In this case, the contents of `credentials` consists of nothing more than the `..` that all directories have in order to refer to their parent, so it's a paltry 48 bytes; while `bin_on_bacon` contains information about more than 30 items, bringing its size up to 1080 bytes.

The next two columns to the left indicate, respectively, the file's owner and its group. As you can see in the previous listing, almost every file is owned by the user `scott` and the group `scott`, except for `addext` and `convertsize`, which are owned by the user `scott` and the group `root`.

NOTE: Those permissions need to be changed, which you'll learn how to do in Chapter 8 (hint: the commands are `chown` and `chgrp`).

The next to last column as you move left contains a number. If you're examining a file, this number tells you how many hard links exist for that file (for more, see Chapter 3's "Create a Link Pointing to Another File or Directory" section); if it's a directory, it refers to the number of subdirectories it contains, including the two hidden pointers `.` (the current directory) and `..` (the parent directory), which means that even if there are no subdirectories, you will still see a `2` there.

And now you reach the final item on the left: the actual permissions for each file and directory. This might seem like some arcane code, but it's actually very understandable with just a little knowledge. There are ten items, divided (although it doesn't look that way) into four groups. The first group consists of the first character; the second group contains characters 2 through 4; the third consists of characters 5 through 7; and the fourth and final group is made up of characters 8 through 10. For instance, here's how the permissions for the credentials directory would be split up: `d|rwx|r-x|r-x`.

That first group tells you what kind of item it is. You've already seen that `-F` and `--color` do this in different ways, but so does `-l`. A `d` indicates that `credentials` is a directory, while a `-` in that first position indicates a file. (Even if the file is executable, `ls -l` still uses just a `-`, which means that `-F` and `--color` here give you more information.) There are, of course, other options that you might see in that first position, as detailed in Table 2.3.

Table 2.3 **Permission Characters and File Types**

Character	Meaning
`-`	Regular file
`-`	Executable
`d`	Directory
`l`	Symbolic link
`s`	Socket
`b`	Block device
`c`	Character device
`p`	Named pipe (AKA FIFO)

TIP: To view a list of files that shows at least one of almost everything listed in Table 2.3, try `ls -l /dev`.

The next nine characters—making up groups two, three, and four—stand for, respectively, the permissions given to the file's owner, the file's group, and all the other users on the system. In the case of `addext`, shown previously, its permissions are `rwxr-xr--`, which means that the owner `scott` has `rwx`, the group (in this case, also `scott`) has `r-x`, and the other users on the box have `r--`. What's that mean?

In each case, `r` means "yes, read is allowed"; `w` means "yes, write is allowed" (with "write" meaning both changing and deleting); and `x` means "yes, execute is allowed." A `-` means "no, do not allow this action." If the `-` is located where an `r` would otherwise show itself, that means "no, read is not allowed." The same holds true for both `w` and `x`.

Looking at `addext` and its permissions of `rwxr-xr--`, it's suddenly clear that the owner (`scott`) can read, write, and execute the file; the members of the group (`root`) can read and execute the file, but not write to it; and everyone else on the machine (often called the "world") can read the file but cannot write to it or run it as a program.

Now that you understand what permissions mean, you'll start to notice that certain combinations seem to appear frequently. For instance, it's common to see `rw-r--r--` for many files, which means that the owner can both read and write to the file, but both the group and world can only read the file. For programs, you'll often see `rwxr-xr-x`, which allows everyone on the computer to read and run the program, but restricts changing the file to its owner.

Directories, however, are a bit different. The permissions of `r`, `w`, and `x` are pretty clear for a file: You can read the file, write (or change) it, or execute it. But how do you execute a directory?

Let's start with the easy one: `r`. In the case of a directory, `r` means that the user can list the contents of the directory with the `ls` command. A `w` indicates that users can add more files into the directory, rename files that are already there, or delete files that are no longer needed. That brings us to `x`, which corresponds to the capability to access a directory in order to run commands that access and use files in that directory, or to access subdirectories inside that directory.

As you can see, `-l` is incredibly powerful all by itself, but it becomes even more useful when combined with other options. You've already learned about `-a`, which shows all files in a directory, so now it should be obvious what `-la` would do (or `--format=long --all`).

```
$ la -la ~/
drwxr-xr-x    2 scott scott   200 2015-07-28 01:31
➥alias
drwx------    2 root  root     72 2015-09-16 19:14
➥.aptitude
-rw-------    1 scott scott  8800 2015-10-18 19:55
➥.bash_history
-rw-r--r--    1 scott scott    69 2015-04-20 11:00
➥.bash_logout
-rw-r--r--    1 scott scott   428 2015-04-20 11:00
➥.bash_profile
-rw-r--r--    1 scott scott  4954 2015-09-13 19:46
➥.bashrc
[List condensed and edited due to length]
```

Reverse the Order Contents Are Listed

```
ls -r
```

If you don't like the default alphabetical order that `-l` uses, you can reverse it by adding `-r` (or `--reverse`).

```
$ ls -lar ~/
-rw-r--r--    1 scott scott  4954 2015-09-13 19:46
➥.bashrc
-rw-r--r--    1 scott scott   428 2015-04-20 11:00
➥.bash_profile
-rw-r--r--    1 scott scott    69 2015-04-20 11:00
➥.bash_logout
-rw-------    1 scott scott  8800 2015-10-18 19:55
➥.bash_history
drwx------    2 root  root     72 2015-09-16 19:14
➥.aptitude
drwxr-xr-x    2 scott scott   200 2015-07-28 01:31
➥alias
[List condensed and edited due to length]
```

NOTE: Keep in mind that this is `-r`, not `-R`. `-r` means reverse, but `-R` means recursive. Yes, that is confusing.

When you use `-l`, the output is sorted alphabetically based on the name of the files and folders; the addition of `-r` reverses the output, but is still based on the filename. Keep in mind that you can add `-r` virtually any time you use `ls` if you want to reverse the default output of the command and options you're inputting.

Sort Contents by Date and Time

```
ls -t
```

Letters are great, but sometimes you need to sort a directory's contents by date and time. To do so, use `-t` (or `--sort=time`) along with `-l`; to reverse the sort, use `-tr` (or `--sort=time --reverse`) along with `-l`. Using the reverse sort can be pretty handy—the newer stuff ends up at the bottom, which is easier to see, because that's where your prompt will be when the command finishes!

```
$ ls -latr ~/
-rw-------    1 scott scott  8800 2015-10-18 19:55
➥.bash_history
drwx------   15 scott scott  1280 2015-10-18 20:07
➥.opera
drwx------    2 scott scott    80 2015-10-18 20:07
➥.gconfd
drwxr-xr-x    2 scott scott   432 2015-10-18 23:11
➥.qt
drwxr-xr-x  116 scott scott  5680 2015-10-18 23:11 .
drwx------    3 scott scott   368 2015-10-18 23:12
➥.gnupg
drwxr-xr-x   12 scott scott  2760 2015-10-18 23:14
➥bin
drwx------    4 scott scott   168 2015-10-19 00:13
➥.Skype
[list condensed and edited due to length]
```

All of these items except the last one were modified on the same day; the last one would have been first if you weren't using the `-r` option and thereby reversing the results.

NOTE: Notice that you're using four options at one time in the previous command: `-latr`. You could have instead used `-l -a -t -r`, but who wants to type all of those hyphens and spaces? It's quicker and easier to just combine them all into one giant option. The long version of the options (those that start with two hyphens and consist of a word or two), however, cannot be combined and have to be entered separately, as in `-la --sort=time --reverse`. Note that this technique works with many Linux commands and programs, but not all of them.

Sort Contents by Size

```
ls -S
```

You can also sort by size instead of alphabetically by filename or extension, or by date and time. To sort by size, use `-S` (or `--sort=size`).

```
$ ls -laS ~/
-rw-r--r--    1 scott scott 109587 2015-10-19 11:53
➥.xsession-errors
-rw-------    1 scott scott  40122 2015-04-20 11:00
➥.nessusrc
-rw-r--r--    1 scott scott  24988 2015-04-20 11:00
➥.abbrev_defs
-rwxr--r--    1 scott scott  15465 2015-10-12 15:45
➥.vimrc
-rw-------    1 scott scott  11794 2015-10-19 10:59
➥.viminfo
-rw-------    1 scott scott   8757 2015-10-19 08:43
➥.bash_history
[List condensed and edited due to length]
```

When you sort by size, the largest items come first. To sort in reverse, with the smallest at the top, just use `-r`.

Express File Sizes in Terms of K, M, and G

```
ls -h
```

In the previous section, the 15465 on `.vimrc`'s line means that the file is about 15KB, but it's not always convenient to mentally translate bytes into the equivalent kilobytes, megabytes, or gigabytes. Most of the time, it's more convenient to use the `-h` option (or `--human-readable`), which makes things easier to understand.

```
$ ls -laSh ~/
-rw-r--r--    1 scott scott 100K 2015-10-19 11:44
➥.xsession-errors
-rw-------    1 scott scott  40K 2015-04-20 11:00
➥.nessusrc
-rw-r--r--    1 scott scott  25K 2015-04-20 11:00
➥.abbrev_defs
-rwxr--r--    1 scott scott  16K 2015-10-12 15:45
➥.vimrc
-rw-------    1 scott scott  12K 2015-10-19 10:59
➥.viminfo
-rw-------    1 scott scott 8.6K 2015-10-19 08:43
➥.bash_history
[List condensed and edited due to length]
```

In this example, you see `K` for kilobytes; if the files were big enough, you'd see `M` for megabytes or even `G` for gigabytes. Some of you might be wondering how 40122 bytes for `.nessusrc` became 40K when you used

`-h`. Remember that 1024 bytes make up a kilobyte, so when you divide 40122 by 1024, you get 39.1816406 kilobytes, which `ls -h` rounds up to 40K. A megabyte is actually 1,048,576 bytes, and a gigabyte is 1,073,741,824 bytes, so a similar sort of rounding takes place with those as well. If you want exact powers of ten in your divisions (that is, what are referred to as "kibibytes," "mebibytes," and so on—see Chapter 6, "Viewing (Mostly Text) Files," for more), try the `--si` option.

NOTE: In my `~/.bashrc` file, I have the following aliases defined, which have served me well for years. Use what you've learned in this section to extend these examples and create aliases that exactly meet your needs (for more on aliases, see Chapter 12, "Your Shell").

```
alias l='ls -F'
alias l1='ls -1F'
alias la='ls -aF'
alias ll='ls -laFh'
alias ls='ls -F'
```

Display the Path of Your Current Directory

```
pwd
```

Of course, while you're listing the contents of directories hither and yon, you might find yourself confused about just where you are in the file system. How can you tell in which directory you're currently working? The answer is the `pwd` command, which stands for *print working directory*.

NOTE: The word *print* in *print working directory* means "print to the screen," not "send to printer."

The `pwd` command displays the full, absolute path of the current, or working, directory. It's not something you'll use all the time, but it can be incredibly handy when you get a bit discombobulated.

```
$ pwd
/home/scott/music/new
```

There is one thing you should be aware of, however, and this can really confuse people. In Chapter 3 you're going to find out about the `ln` command ("Create a Link Pointing to Another File or Directory"), so you might want to skip ahead and read that to fully understand what I'm about to show you. Assuming you have, check out the following:

```
# ls -l
lrwxrwxrwx  scott scott  websites -> /var/www/
$ cd websites
$ pwd
/websites
$ pwd -P
/var/www
```

So there's a soft link with a source `websites` that points at a target `/var/www`. I `cd` using the soft link and enter `pwd`. Notice what comes back: the *`logical`* directory of `/websites`, the *`source`* of the soft link, which isn't the actual working target directory of `/var/www`. This is the default behavior of `pwd`, equivalent to entering `pwd -L` (or `--logical`).

On the other hand, if you enter `pwd -P` (or `--physical`), you instead get back the target of the soft link, which is `/var/www`.

I don't know about you, but when I use `pwd`, I usually want to get back the actual physical location (the target), not the logical location (the source). Since I actually prefer the `-P` option as the default, I like to use an alias in my `.bash_aliases` file (discussed in Chapter 12's "Create a New Permanent Alias" section) that looks like this:

```
alias pwd="pwd -P"
```

Now you understand not only how to use `pwd`, but also the gotchas you might run into when you use it.

Change to a Different Directory

`cd`

It's possible to list the contents of any directory simply by specifying its path, but often you actually want to move into a new directory. That's where the `cd` command comes in, another one that is almost constantly used by shell aficionados.

The `cd` command is simple to use: Just enter `cd`, followed by the directory into which you want to move. You can use a relative path, based on where you are currently—`cd src` or `cd ../../`—or you can use an absolute path, such as `cd /tmp` or `cd /home/scott/bin`.

Change to Your Home Directory

```
cd ~
```

You should know about a few nice shortcuts for `cd`. No matter where you are, just enter a simple `cd`, and you'll immediately be whisked back to your home directory. This is a fantastic timesaver, one that you'll use all the time. Or, if you'd like, you can use `cd ~` because the `~` is like a shortcut meaning "my home directory."

```
$ pwd
/home/scott/music
$ cd ~
$ pwd
/home/scott
```

Change to Your Previous Directory

```
cd -
```

Another interesting possibility is `cd -`, which takes you back to your previous directory and then runs the `pwd` command for you, printing your new (or is it old?) location. It's the shell equivalent of the Back button in a GUI file navigator window. You can see it in action in this example:

```
$ pwd
/home/scott
$ cd music/new
$ pwd
/home/scott/music/new
$ cd -
/home/scott
```

Using `cd -` can be useful when you want to jump into a directory, perform some action there, and then jump back to your original directory. The additional printing to the screen of the information provided by `pwd` is just icing on the cake to make sure you're where you want to be.

Conclusion

If this was law school, you would have learned in this chapter what misdemeanors, torts, and felonies are. If this was a hardware repair class, it would have been RAM, hard drives, and motherboards. Since this is a book about the Linux shell, you've examined the most basic commands that a Linux user needs to know to effectively use the command line: `ls`, `pwd`, and `cd`. Now that you know how to get around the file system, it's time to learn about two things that govern the universe: creation and destruction. Now not only will you be observing your environment, you'll be modifying it to your whims. Read on!

This page intentionally left blank

Index

Symbols

A

B

C

D

E

F

G

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z