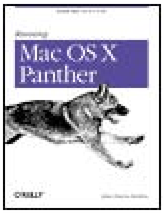


- Table of Contents
- Reviews
- Reader Reviews
- Errata
- Academic
- Ten Things I Dig About Panther
- Scheduling Tasks in Panther

Running Mac OS X Panther

ByJames Duncan Davidson

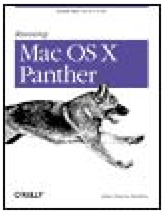
Publisher: O'Reilly

Pub Date: December 2003

ISBN: 0-596-00500-8

Pages: 326

Running Mac OS X Panther is the ultimate Swiss Army Knife(TM); for power users who want to customize, tweak, and generally rev up their Mac. This book takes readers deep inside Mac OS X's core, revealing the inner workings of Panther for those who want to get the most out of their system. You may not be a Mac guru when you start this book, but once you've read it, you'll be well on your way.



- Table of Contents
- Reviews
- Reader Reviews
- Errata
- Academic
- Ten Things I Dig About Panther
- Scheduling Tasks in Panther

Running Mac OS X Panther

By James Duncan Davidson

Publisher: O'Reilly

Pub Date: December 2003

ISBN: 0-596-00500-8

Pages: 326

Running Mac OS X Panther
by James Duncan Davidson

Published by O'Reilly & Associates, Inc., 1005 Gravenstein Highway North,
Sebastopol, CA 95472.

Editor: Chuck Toporek

Copyeditor: Jill Steinberg

Production Manager: Claire Cloutier

Cover Designer: Emma Colby

Interior Designer: David Futato

Interior Layout: James Duncan Davidson

Printing History:
December 2003: First Edition.

While every precaution has been taken in the preparation of this book, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

ISBN: 0-596-00500-8
[M]

Preface

Mac OS X is the first real operating system for the 21st century. A stunning achievement not to be confused with Mac OS 9 and its predecessors even as it builds on their legacy, Mac OS X redefines our expectations of what a computer should be. On the surface, Mac OS X has a graphical user interface (GUI) with usability that can't be touched by any other OS on the planet. Under the hood it has a powerful Unix engine known as Darwin, which was developed through Apple's open source initiative and based on FreeBSD 5.0 Unix distribution and the fully buzzword-compliant Mach 3.0 kernel. This combination of features gives the system its smooth multitasking behavior and virtual memory management.

This strong foundation provides the system the stability it needs to amaze, intrigue, and serve you. It is also this foundation that lets you run the Apache web server, the Postfix mail server, and X11 applications next to Microsoft Word, Adobe Photoshop, and Macromedia Dreamweaver—a feat that nobody thought possible a few years ago. For many, Mac OS X has replaced the dual boot systems that they used to have: one partition for Windows so you could run Office and another for Linux for doing Unixrelated work. Now, it's all under one roof.

Even better, Mac OS X continues to improve. Each release brings new features, speed improvements, and more polish. This book—*Running Mac OS X Panther*—will help you master the latest version of Mac OS X. You'll learn how to get the most out of the pretty GUI as well as how to dive into the Unix layer of the system to take fine-grained control when you need to.

Audience

This book is written for users and power users of Mac OS X and assumes that you already have some experience with Mac OS X and with computers in general. For example, this book assumes that you have found and used the System Preferences application, have discovered how to change the way that your windows minimize and how the Dock works, and that you've figured out how to change your Desktop background. I'm going to speak to you as somebody who wants to truly master what is going on with your system. Maybe you want to turn your Mac into a web server. Or maybe you want to know the pros and cons of the various filesystem choices that you have. Possibly you want to understand how Mac OS X's networking subsystem makes it possible to switch effortlessly between wired and unwired networks as well as letting you share your connection with others.

The command line will be covered extensively in this book and while I expect that you are at least comfortable with the idea of the command line, one thing I won't assume is that you are a Unix guru who is knowledgeable about all the shell's dark corners. I'll give you enough help and explanation of how the command line works so you can accomplish everything you need to. If you decide you like the Unix side of Mac OS X, you can learn more about it by reading *Learning Unix for Mac OS X Panther*, by Dave Taylor and Brian Jepson.

On the other hand, if you are a Unix guru, you may be interested in *Mac OS X Panther for Unix Geeks*, by Brian Jepson and Ernest E. Rothman, as a companion book that serves as a magic decoder ring describing the differences between Mac OS X's internals and those of Linux, Solaris, and other variants of Unix.

And, if you really do want to learn how to set your Desktop pictures and how to use Stickies, iChat, and iTunes, I suggest you put this book down and pick up the latest edition of *Mac OS X: The Missing Manual, Panther Edition*, by David Pogue; surely it's located somewhere nearby in the bookstore. Better yet, buy it at the same time you pick up this one. The two books go well with each other.

There are two other books that you should consider companion books to this one:

- *Mac OS X Panther in a Nutshell,* by Chuck Toporek, Chris Stone, and Jason McIntosh, serves as a quick reference to many of Panther's features.
- *Mac OS X Panther Pocket Guide,* by Chuck Toporek, squeezes the most used features into a pocket-sized quick reference guide.

How This Book Is Organized

This book consists of 14 chapters along with three appendixes.

Part I, Getting Started

This section of the book introduces where Mac OS X came from, how it is put together, and how it works. It covers how the system looks, how multiple users coexist on the system, and how to use the Terminal to get at the Unix foundation of the operating system.

Chapter 1, *Where It All Came From*

> Mac OS X is the successful hybrid of technologies from Unix and the original Mac OS. On its own, the history would be interesting enough. But the impact of Apple's decisions while fusing these two systems together affects how the system operates to this day.

Chapter 2, *Lay of the Land*

> The layout of the Mac OS X filesystem is different from that of most other operating systems. It shares history with traditional Unix filesystem layouts, but introduces several concepts that make it easier for the system to keep its data separate from yours. This chapter looks beneath the Finder's presentation of the filesystem and explains how the filesystem is laid out, with details on what is located where.

Chapter 3, *The Terminal and Shell*

> The Terminal gives you access to the Unix core of Mac OS X. While it is possible to become a decent user without ever using the Terminal, you'll need a good understanding of it to become a guru and unleash the full potential of the system. This chapter shows how to use the Terminal and how to configure the various shells on the system.

Part II, Essentials

The next section of the book gives you the tools you need to examine how your system is running and how to adjust some of the knobs behind its operation. It covers topics ranging from process control to managing software installations and backup strategies.

Chapter 4, *System Startup and Login*

> Quite a bit happens behind the simple gray Apple logo and spinner that appear when Mac OS X starts up. This chapter looks at what gets executed when and what the various boot options are. In addition, it takes a look at the process by which events happen when you

log in and out of the system.

Chapter 5, *Users and Groups*

Thanks to its Unix heritage, Mac OS X is a multiuser operating system through and through. This chapter dives into the question of why there are multiple users on the system and why they are gathered into groups. It also explains how to add users from both the Accounts preference panel and the command line, how user security works, and how to access the functionality of the root user.

Chapter 6, *Files and Permissions*

Most administration tasks that you perform will involve working with files in one capacity or another. This chapter explains the various attributes and permissions that a file can have and how to work with them. It also takes a look at how to effectively find files on the system.

Chapter 7, *Monitoring the System*

The first step to keeping your system in tip-top shape is to know where to go to get information about how the system is running. This chapter covers the Console, System Profiler, Activity Monitor, and many other tools as well as gives you the skinny on what to look for in your system's log files.

Chapter 8, *Scheduling Tasks*

Behind the scenes, Mac OS X runs many tasks on a periodic basis. These tasks perform essential housekeeping services for the system. This chapter dives into how these tasks are run and how to set up your own tasks to run at periodic intervals.

Chapter 9, *Preferences and Defaults*

All user configuration data on Mac OS X is stored in the defaults system. This system is structured, in a way similar to the filesystem, to keep your data separate from the system's data, allowing you to migrate it between machines easily. In addition to discussing how the defaults system works, this chapter shows you how to modify preferences stored in the system in a variety of ways.

Chapter 10, *Disks and Filesystems*

Mac OS X supports many different kinds of disks including: hard drives, removable USB and FireWire drives, floppy drives, CDs, DVDs, and virtual disk images. This chapter gives you the lowdown on the different kinds of filesystems you can use and the pros and cons of each. In addition, it covers various backup strategies to help keep your data safe.

Part III, Advanced Topics

This part of the book covers some advanced topics such as how to work with Directory Services, the printing system of Mac OS X, networking, and how the various network services work.

Chapter 11, *Open Directory*

> Open Directory stores the most important data about your system—the list of users, groups, printers, and so forth. Based on LDAP, it works with NetInfo, flat files, other LDAP servers, and Microsoft's Active Directory. This chapter shows you how this system works and how to modify data contained within it.

Chapter 12, *Printing*

> The printing architecture that Mac OS X uses is very flexible and supports both local and network printers. This chapter gives you a detailed look at how printing works, how to use print servers, how to connect with Windows-hosted printers, and even how to share your ink jet printer to Windows users. It also offers a look at the Common Unix Print System (CUPS), which serves as the foundation for the rest of the printing system.

Chapter 13, *Networking*

> Mac OS X lives to be connected to other machines. This chapter shows you how the networking system works including wireless, spontaneous networking (Rendezvous), and Internet Connection Sharing. It also covers how to connect to corporate VPN networks.

Chapter 14 , *Network Services*

> In addition to being able to connect to other machines, Mac OS X can serve as a server for many different protocols. It can share both files and printers to other Macs as well as Windows machines. It can serve web pages to other machines from anywhere on the network. And it can even serve as a full fledged mail server. This chapter shows you how to configure these services and where to go to look for more information.

Appendixes

The appendixes include quick reference material.

Appendix A, *Installing from Scratch*

> Every new Mac comes with Mac OS X installed. For many reasons, however, you may want (or need) to reinstall your system. This can be an easy process, but there are a few things you should think about as you do it. This appendix provides a set of recipes to use when rebuilding your system.

Appendix B, *Boot Command Keys*

While your Mac boots up you can give it all sorts of commands, and they all involve somewhat cryptic key combinations. This appendix gives you the full reference to all the commands.

Appendix C, *Other Sources of Information*

There are a wide variety of good resources containing more information about Mac OS X. This appendix serves as a list of resources that will help you develop knowledge about Mac OS X.

How to Use This Book

You should be able to jump into this book wherever you wish. Reading straight through from front to back, however, will ensure that you won't encounter any surprises as every effort has been made to avoid forward references. If at any time you need to dig deep and find out why something isn't working as you expect, read Chapter 7,*Monitoring the System*.

There are a zillion things I'd love to show you about how your Mac operates under the hood. Due to the desire to both get this book into your hands before the end of this century as well as to allow you to carry it next to your PowerBook in your backpack, I'm going to cover the most salient and interesting aspects of each subject and, where appropriate, point you to additional sources of information.

Don't be shy about dog-earing the pages as well as highlighting parts of the book that you find useful. Computer books should not be treated as pristine coffee table books. They should be used. And, it won't be long before Apple releases a new version of Mac OS X, which will prompt a revision of this book that you'll surely want to buy. So go ahead, scribble notes in the margins!

Compatibility

Mac OS X is evolving rapidly—there have been three major releases since it was first released as version 10.0 in March 2001—and there is no sign that Apple is slowing down. To keep things simple, this book is targeted directly at Mac OS X Panther (version 10.3.1 was the most current version when this book went to press). While most of this information may work with previous or later releases of Mac OS X, there is no way to ensure this—and I'm certainly not going to pretend that I'm psychic and that the advice in this book will apply to the next release of Mac OS X (whatever big cat they name it after). Use this book with Panther and you should be safe.

Conventions Used in This Book

The following is a list of the typographical conventions used in this book:

Italic

> Used to indicate new terms, URLs, filenames, file extensions, directories, commands and options, and program names, as well as to highlight comments in examples. For example, a path in the filesystem will appear as */Developer/Applications*.

`Constant Width`

> Used to show code examples, the contents of a file, or the output from commands.

`Constant Width Italic`

> Used in examples and tables to show text that should be replaced with user-supplied values.

`Constant Width Bold`

> Used in examples and tables to show text that is input by the user.

↵

> A carriage return (↵) at the end of a line of code is used to denote an unnatural line break; that is, you should not enter these as two lines of code but as one continuous line. Multiple lines are used in these cases due to printing constraints.

`$, %, #`

> The dollar (`$`) and percent (`%`) signs are used in some examples to show the prompt of the *bash* or *tcsh* shell; the hash (`#`) mark is the prompt for the root user.

Menus/Navigation

> Menus and their options are referred to in the text as File→Open, Edit→Copy, and so on. Arrows will also be used to signify a navigation path when using window options; for example, System Preferences→Accounts→Startup Items means that you would launch

System Preferences, click the icon for the Accounts preference panel, and select the Startup Items pane within that panel.

Menu Symbols

When looking at the menus for any application, you will see some symbols associated with keyboard shortcuts for a particular command. For example, to create a new project in Project Builder, you would go to the File menu and select New Project (File→New Project), or you could issue the keyboard shortcut, Shift-⌘-N. The symbol is used to refer to the Apple menu in the upper-left corner of the screen.

Directories and Folders

In the text you will see the same thing referred to sometimes as "directories" and sometimes as "folders". The term "directory" is used when referring to directories in the command line or in the Unix part of the OS. When using the Finder and other GUI applications, however, the traditional Mac OS term "folder" is used.

You should pay special attention to notes set apart from the text with the following icons:

> This is a tip, suggestion, or general note. It contains useful supplemental information about the topic at hand.

> This indicates a warning or a caution. It will help you solve and avoid annoying problems.

How to Contact Us

All the people who have worked on this book, including (but not limited to) the author, editor, and copyeditor, have tested and verified the information in this book to the best of their ability, but you may find that features have changed—or even that we have made mistakes! As a reader of this book, you can help us to improve future editions by sending along your feedback. Please let us know about any errors, inaccuracies, bugs, misleading or confusing statements, and typos that you find anywhere in this book.

Please also let us know what can be done to make this book more useful to you. Your comments will be taken seriously, and we will try to incorporate reasonable suggestions into future editions. You can write to us at:

O'Reilly & Associates, Inc.

1005 Gravenstein Highway North

Sebastopol, CA 95472

(800) 998-9938 (in the U.S. or Canada)

(707) 829-0515 (international or local)

(707) 829-0104 (fax)

You can also send us messages electronically. To be put on the mailing list or to request a catalog, send email to:

info@oreilly.com

To ask technical questions or to comment on the book, send email to:

bookquestions@oreilly.com

The web site for *Running Mac OS X Panther* lists examples, errata, and plans for future editions. You can find this site at:

www.runningosx.com/

There is also a page on O'Reilly's web site dedicated to this book. You can find this page at:

www.oreilly.com/catalog/runmacxpanther

For more information about this book and others, see the O'Reilly web site:

www.oreilly.com/

Acknowledgments

First and foremost, I'd like to thank all the people who reviewed the material in this book as it was written: Gabe Benveniste, Mike Clark, Damon Clinkscales, Stuart Halloway, Bethany Jane Hanson, Joseph Heck, Manton Reece, Daniel Steinberg, Dave Thomas, and Glenn Vanderburg as well as a whole lot of people at Apple (you guys know who you are). These people had front row seats to the creative process, watched a book grow, made tons of helpful comments, and asked all the right questions along the way. They even came through when I asked them to review chapters over the Thanksgiving holiday. The book you hold in your hands is, in large part, a result of their input. I owe them all my gratitude and thanks.

Special thanks go to Dave and Daniel who both went way beyond the call of duty. Any time I can repay the favor, guys, and review material for you, I'll be there.

Of course, there's the editorial team without whom an author is just somebody who scribbles random gibberish on paper. Chuck Toporek, who has worked with me on three books and become a friend along the way, brought this idea to me and helped me bring it to fruition. In addition to editing the book, Chuck also picked up the pen and authored Appendix A. Jill Steinberg, also a friend, was the copyeditor for the book. She fretted over my use of words and helped turn them into something that I'm proud to put on the bookshelf. In addition to wielding the red pencil, she also offered valuable advice and critique during the early work on this book. Both Chuck and Jill are top-notch. Without them, this would not be a coherent book.

Many thanks to the women and men of the Apple Developer Connection who were of great assistance throughout the project. Without their support, this book couldn't have been written and printed in 2003. Writing about an OS is not an easy task. Writing about one that is evolving as fast as Mac OS X is and still being timely is even harder. Without the support of the ADC and others at Apple, by the time we got a decent book together, news of the next version of Mac OS X would have made it obsolete.

Great thanks, as always, to Tim O'Reilly for so many things, but most of all for always listening and being willing to try to do what's right, even if it's risky and hasn't been done before. The production of this book was anything but typical. The last chapter was written during the 2003 Thanksgiving holiday and the book should be in stores by the end of the year or just after—in other words, it came together very rapidly. We've bent a few rules along the way and without Tim's support, it couldn't have happened.

David Futato watched as I took a design tuned from years of FrameMaker use and implemented it afresh using Adobe InDesign. Along the way he offered many helpful suggestions and tips. I may have executed the layout and tweaked it to work well in a new environment, but the design is his and I hope it was implemented to his satisfaction.

During the writing of this book, many things happened in my personal life for which I am thankful. I moved to Portland, Oregon, a fantastic place to live even with the rain. I watched two of my best friends get married: Jason Hunter to Kathlyn in Yosemite; and Jim Driscoll to Sent-si in Hawaii. And, with the help of Dr. Atkins, I weigh less than I have in 10 years. I'm not sure how I managed to do all that while writing a book—a process that found me up until the wee hours of the night and, on more than one occasion, watching the Sun rise before going to bed.

The music. Always the music. iTunes shows the following artists as the most frequently played during the creation of this book: Amon Tobin, BT, The Chemical Brothers, Coldplay, The Crystal Method, Electric Skychurch, Everclear, Fatboy Slim (Norman Cook), Filter, Fischerspooner, Fluke, Foo Fighters, Groove Armada, Josh Wink, Juno Reactor, No Doubt, The Orb, Paul Oakenfold, Paul van Dyk, Pink Floyd, Tabla Beat Science, Timo Maas, Tori Amos, Underworld, and Venus Hum. You better believe that I can't write without music.

And finally, many thanks to my family and friends who lent their support to the book writing process and who always encouraged me to chase my dreams.

Part I: Getting Started

Any master at their subject knows that to truly understand a subject, you have to first understand the basics—what the subject is and where it all came from. The first part of this book sets the stage for the rest of the book by showing you where today's Mac OS X came from and how it is put together. This section also covers two essential elements of Mac OS X, the Terminal and the shell, which you'll need to be proficient at to be a true power user.

Chapters in this part of the book include:

Chapter 1,*Where it All Came From*

Chapter 2,*Lay of the Land*

Chapter 3,*The Terminal and Shell*

Chapter 1. Where It All Came From

The original Mac OS defined the basic metaphors that influenced personal computing and for a decade was the standard by which all other personal computer operating systems were measured. It was not until Microsoft Windows 95 was introduced that anything else came close. Unfortunately, due to many mistakes made by Apple while Microsoft continued to improve Windows, close was good enough and Microsoft ended up owning the user desktop experience for the last half of the 1990s. Now, with the advent of a technological tour de force known as Mac OS X, the Mac is once again setting the standard for what a personal operating system should be.

The Mac OS as we now know it is far different from the one released in 1984. It is a successful hybrid of ideas that were first expressed in the original Macintosh and of technologies that come from a computing philosophy long considered to be the antithesis of the Mac experience: Unix. The fusion of these two very different camps gives us the platform we know and love today.

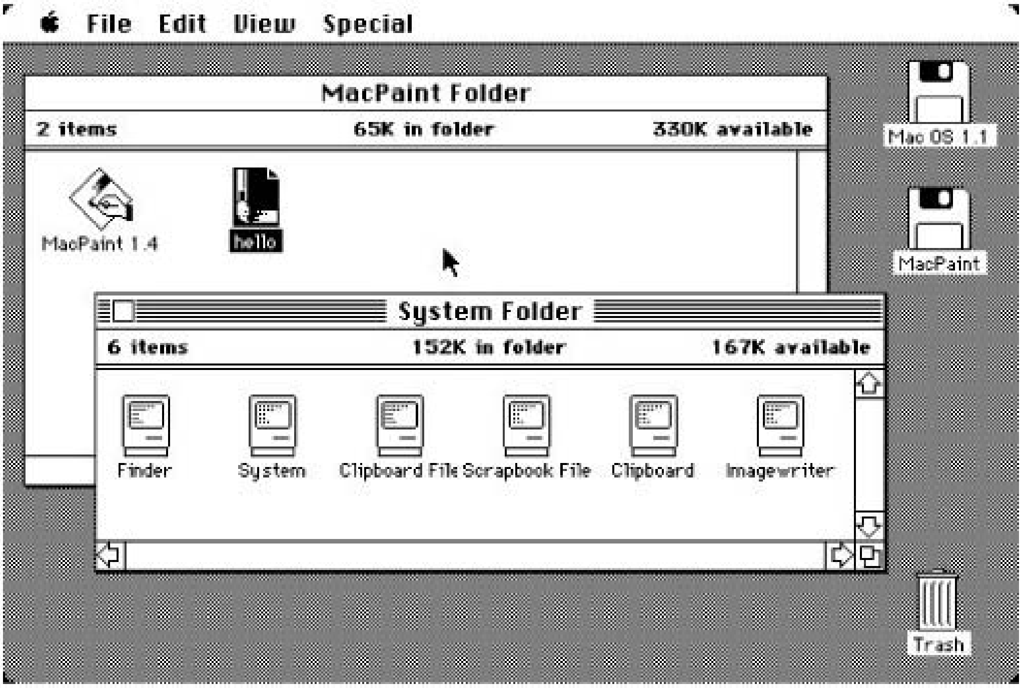
Instead of giving the same laundry list of features in this chapter that you can get from Apple's web site or from most other books covering Mac OS X, I'm going to take a look at the predecessors to Mac OS X and explore the various origins of the modern system. The intent isn't to tell the story of Apple Computer or of Steve Jobs, though those histories are interesting. The focus here is on the technologies at hand, where they came from, and how they influence the development of the Mac operating system. Each major technology of the current Mac OS X will be introduced and its source identified.

1.1 The Classic Mac OS

In 1983, after the incredible success of the original Apple II, which was the first computer that was a true personal computer for most people, and after the later relative disappointments of the Apple III and the Lisa, Steve Jobs organized a group of engineers under a pirate flag to make a computer that could be used by anybody.

The result of their efforts, known as the Macintosh, was introduced to the world a year later accompanied by the fanfare of the famous "1984" commercial (directed by Ridley Scott of *Alien* and*Blade Runner* fame). With a message of empowerment, the Mac redefined what a personal computer could be. What made the Mac different from every other personal computer at the time was its Graphical User Interface (GUI), inspired by research that originated at Xerox PARC: a desktop where files lived in folders, and you used a mouse to move things where you wanted them. The Mac's original desktop is shown in Figure 1-1.

Figure 1-1. The original Mac OS desktop



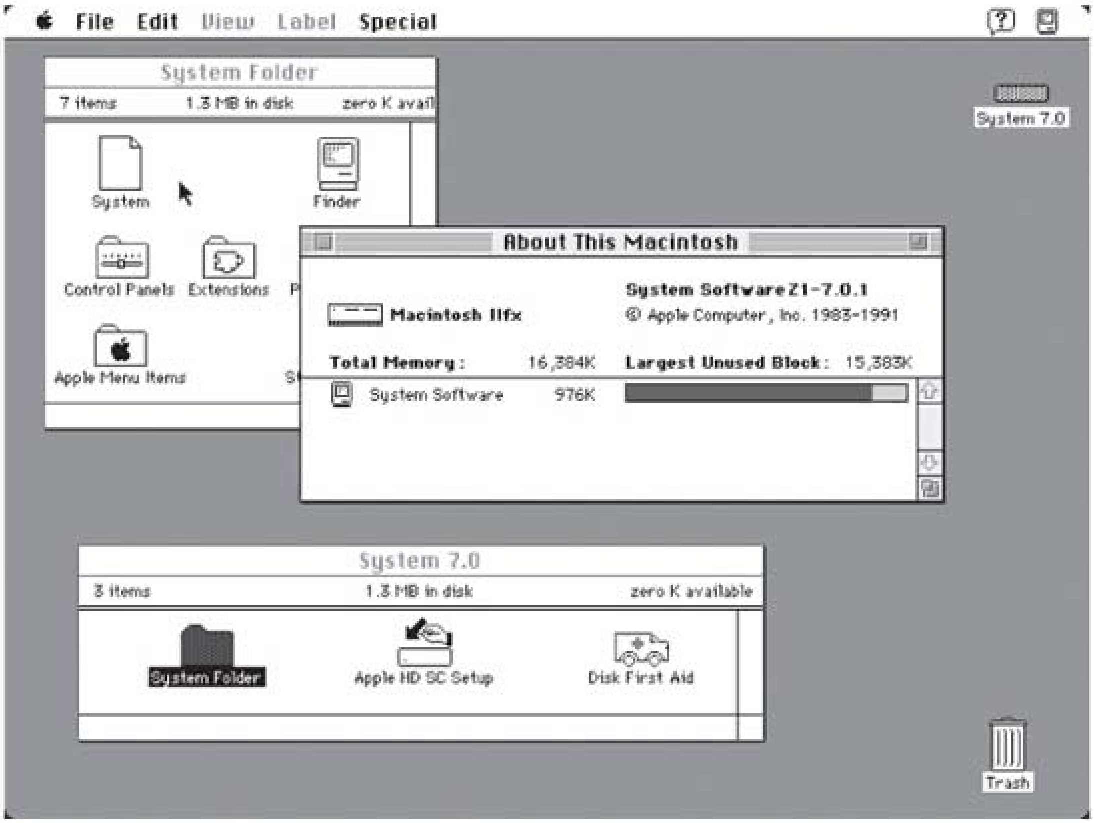
The desktop metaphor introduced with the original Mac in 1984 lives on today in the current Finder. Sure, today's desktop is rendered in full color and the original was comparatively crudely drawn in black and white, but the basic idea remains much the same: opening disks and folders yields windows that contain icons that represent files.

At the heart of the system was a set of programmatic routines called the Macintosh Toolbox that allowed programmers to implement the various essential interface components, such as windows, menus, alert boxes, scrollbars, and other controls, in their applications. In 1984, the Toolbox was an amazing feat of engineering and contributed to a consistent user experience that was the envy of other operating system vendors. But, over the years after its release, more and more routines were added to the Toolbox as the system developed. This resulted in more features, but those features came with a price—the system became fragile.

1.2 System 7

In 1991 Apple introduced System 7, an upgrade that greatly modernized the Mac. It incorporated many features that would guide future development: seamless multitasking (albeit cooperative), color icons, personal file sharing, virtual memory, and a hierarchical System folder to help organize Control Panels and Extensions. The Mac interface also evolved for System 7, growing a bit more sophisticated as shown in Figure 1-2.

Figure 1-2. The System 7 desktop



Macintosh IIfx

System Software Z1-7.0.1
© Apple Computer, Inc. 1983-1991

Total Memory : 16,384K
Largest Unused Block : 15,383K
System Software 976K

System 7 also lay the foundation for three very important technologies:

QuickTime

A powerful multimedia technology for manipulating, enhancing, storing, and streaming video, graphics, sound, and animation.

ColorSync

A set of system routines for color management allowing for graphics to be color managed from editing to print.

AppleScript

An easy-to-use scripting language that allows you to automate tasks on your system.

Although System 7 was years ahead of every other operating system from the ease-of-use perspective, not all was rosy with this release. With all the new functionality, the Toolbox had become even more fragile than its predecessors. It turned out that cooperative multitasking, where applications are expected to play nice with each other, meant that any application could hog all the processor time and, in effect, lock up a system. And, most glaringly, the Mac OS did not provide memory protection between programs. This not only let applications run faster as well as enable all sorts of nifty customizations (also known as hacks), but it also let applications scribble into each other's memory space, inevitably corrupting the system. As Apple released new hardware, developers dreamed up new features that could only be implemented using system extensions. While extensions had good intentions, they frequently conflicted with each other, which forced users to reboot their Macs several times a day.

At this point, even though development continued on the Mac OS, eventually leading to OS 8, OS 9, and the Classic environment under Mac OS X, it was obvious to Apple that a radical overhaul was needed to provide a more solid foundation for the future.

1.3 Copland

To address the need for a more stable operating system, Apple embarked on the development of a new operating system, dubbed Copland, in 1993. With Copland, the focus was on increasing stability, portability, ease-of-use, and performance. Copland was intended to be Apple's stepping stone to a future OS that would include preemptive multitasking and protected memory—keeping applications separate and preventing them from crashing each other or taking the entire system down. Much to the chagrin of Mac users as well as Apple, Copland descended into a death spiral of budget and schedule overruns.

Three years later, in 1996, Copland was cancelled and Apple desperately looked elsewhere for a new foundation for the Mac OS. For a time, it looked like Apple would purchase Be, started by Apple alum Jean-Louis Gassée, and use the technically sophisticated BeOS as the foundation for its next generation operating system. The BeOS was developed on the PowerPC chip fitting in with Apple's Mac hardware strategy, had impressive multitasking abilities, an advanced filesystem, and provided the memory protection that the Mac OS so desperately needed. The only catch: BeOS was an unfinished work-in-progress that was unproven in the marketplace.

After several months into negotiations to acquire Be, the two companies were not able to agree on a price. Be wanted $400 million dollars for its unfinished and unproven system, and Apple did not want to take that much risk for the price. After negotiations fell through late in 1996, Be struggled for several more years, eventually being acquired by Palm, Inc.

Once again, Apple was left with no direction for its operating system, and many thought that the demise of Apple was near. Then, discussions started with Steve Jobs that led to the purchase in December 1996 of the company that he founded after leaving Apple in 1985: NeXT Computer. With this purchase came the portable and feature-rich NEXTSTEP operating system.

1.4 NEXTSTEP

Originally introduced to the world in 1987, along with the elegant NeXT cube, NEXTSTEP was intended to, "Create the next insanely great thing," as only Steve Jobs could say. The driving mantra was to do everything right, and not to repeat the mistakes that Apple had made. Built on top of BSD Unix and the Mach microkernel, NEXTSTEP had the preemptive multitasking and memory-protected core that Mac OS needed. NEXTSTEP also utilized Display PostScript from Adobe allowing developers to use the same code to display documents on screen as well as to print to paper—enabling truly WYSIWYG applications. NEXTSTEP also featured a rich GUI desktop, as shown in Figure 1-3.

For developers, NEXTSTEP came with a set of libraries, called "frameworks," and tools to enable programmers to build applications using the Objective-C language. Developers loved this mix of technologies. For example, Tim Berners-Lee used NEXTSTEP to write the first few versions of both the client browser and server software that would start the World Wide Web.

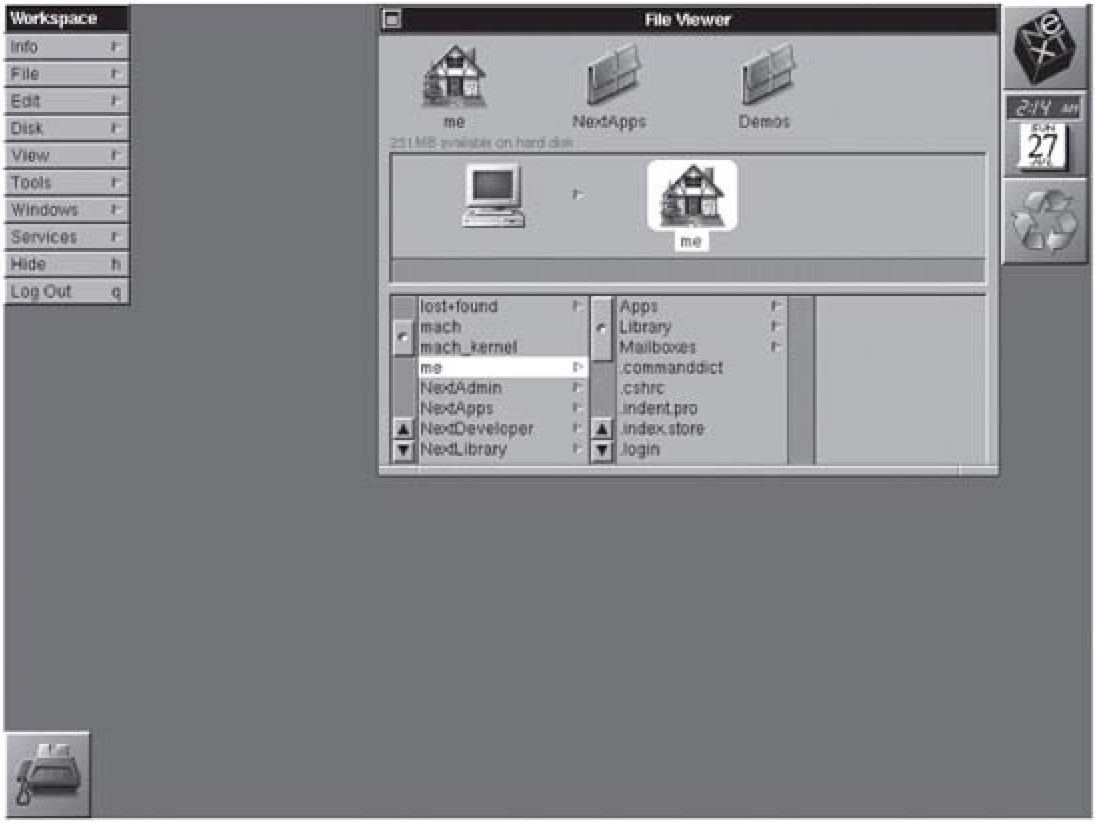
NEXTSTEP evolved through many releases and was adopted by many governments and companies as their platform of choice. It made inroads into the military, banking, healthcare, and telecommunications industries and received glowing reviews from the press. Because it was created as a fresh start, avoiding the mistakes of the design of the original Macintosh and yet building on its ideas, it was able to jump far ahead of anything else.

On Spelling and Capitalization

Over the years, there's been quite a bit of confusion as to how Copland was spelled, with many using the spelling "Copeland". However, a quick Google search with the term**`copland +site:apple.com`**turns up many old documents on Apple's web site that set the record straight. It's Copland without an "e".

There's also been quite a bit of confusion about how the word NEXTSTEP is capitalized, with camps of rabid curmudgeons gathered around the variations of NeXTSTEP, NeXTStep, and NEXTSTEP—each claiming their capitalization is correct. Well, the truth of the matter is they are all right. NeXT varied the capitalization over the various releases of the system. This book uses the all-caps form of the word as that's how Apple referred to the operating system in the press release dated December 26, 1996, announcing the merger of Apple and NeXT.

Figure 1-3. The NEXTSTEP desktop



In 1993, NEXTSTEP was ported to the Intel *x*86 architecture. Subsequently, other ports were performed for the Sun SPARC, Digital Alpha, and Hewlett-Packard PARISC architectures. Later, the development frameworks and tools were revised to run on other operating systems, such as Windows and Solaris. These revised frameworks became known as OpenStep. The NeXT team gained quite a bit of experience in implementing its technology on multiple platforms.

From NEXTSTEP, Mac OS X directly inherits the following:

The Mach kernel

> Descended from a research project at Carnegie Mellon University and first designed by Avi Tevanian (Apple's current Chief Software Technology Officer) in the mid-1980s, the Mach kernel performs the critical functions of abstracting the hardware of the computer from the software that runs on it. It provides symmetric multiprocessing, preemptive multitasking, protected and virtual memory, and support for real-time applications. Because the kernel is the gatekeeper to hardware and controls each application's view of memory, individual applications can't crash or corrupt the system.

The BSD layer

> Derived from FreeBSD, the BSD layer provides the "user visible" part of the Unix layer: the process model, the concept of users, basic security policies, networking, and support for filesystems. This layer allows many Unix applications, such as the Apache web server, to be easily ported and run on Mac OS X.

The NEXTSTEP programming frameworks

> Now known as Cocoa, NEXTSTEP defined a rich set of object-oriented APIs, a set of libraries, a runtime, and a development environment for building applications. It provides most of the infrastructure needed to build graphical user applications and to insulate those applications from the internal workings of the core system.

You don't necessarily *have* to ever see this layer if you don't open up the Terminal, but it's there nonetheless, and knowledge of it is *de rigeur* to becoming a power user. Through the course of this book, you're going to get quite familiar with this layer; mastery of the BSD Unix core of Mac OS X is essential to gaining complete mastery of your system.

1.5 Rhapsody

At the 1997 World Wide Developer Conference (WWDC), Apple demonstrated an early build of Rhapsody, a version of NEXTSTEP that ran on Macintosh hardware. With NeXT's experience porting the system to multiple hardware architectures, it was easy enough for Apple to get a build up and running on a single model Mac model in time for the conference. The development community was invigorated as an ambitious project plan was announced that would supposedly lead to a release in 1998 of a new Mac OS for the PowerPC and Intel platforms.

As part of the project plan, three major components were announced: The Yellow Box, which would run OpenStep-based applications on top of Rhapsody; the Blue Box, which would run a future version of the classic Mac OS as a Rhapsody process; and the Red Box, which would allow OpenStep applications to run on top of Windows with a simple recompile.

Rhapsody brought the integration of the following technologies into the current Mac OS X:

Classic

> With the outgrowth of the Blue Box, Classic allows older Mac applications to run unmodified under the new operating system. Essentially an operating system running within a process, it doesn't confer all the advantages of a protected-memory preemptive system to the applications running within it. If an application crashes inside Classic, it can corrupt any other applications also running in Classic. However, Classic allows the old and new operating systems to run at once with the same screen, mouse, and a modicum of cross-platform application utility. It's not a perfect solution, but it allows users to continue using their older Mac OS applications as they migrate to the new system.

Java

> From the beginning, the Rhapsody team made it a priority to implement good Java support on the platform. This support would allow Java applications and applets to run and operate without modification. As well, a layer of functionality—known as the *Java Bridge*—was implemented, allowing Java and Objective-C to work together. With this functionality, Cocoa applications can be built using Java.

Several problems arose that would keep Rhapsody from ever seeing the light of day as a consumer system. The first problem was getting Rhapsody to run on Apple's entire product line. Each model of Macintosh was very different from other models, and clever engineering hacks had been put into place over time in earlier versions of the Mac OS so it would run on all Macs at the time (including PowerPC and 680x0-based Macs). This variation of hardware made it harder to port the Mac OS as each new Mac model required its own port of the OS. This led to the reduction in the number of older Macs that the new OS would be supported on as well as to the introduction of many changes to the hardware platform to ease the marriage of software and hardware.

The second major problem to face Rhapsody was the lack of commitment by the major Mac software vendors, such as Adobe and Microsoft, to rewrite their products using the Objective-C language and the frameworks derived from OpenStep. Without a complete rewrite, their applications were doomed to live as second-class citizens in the new OS by running in the Blue Box (that is, Classic mode). It became clear to Apple that without the support of the major software vendors shipping first-class applications, the new OS would not succeed. Apple rethought its strategy and decided to postpone the new OS as a consumer system. This delay gave Apple more time to make its next-generation OS run on more hardware as well as to develop a compatibility layer that would make it easier for older Mac software to be ported to run on the new system. This layer was called Carbon, which was introduced at WWDC 1998 along with the rest of the plan for Mac OS X development after Rhapsody.

Even though Rhapsody wasn't widely released, it wasn't a failure. It represented an important phase of the transition of the old NeXT-based operating system into something that could be the next generation of the Mac OS. Without Rhapsody, all that followed could not have happened. In fact, the major distinction between Rhapsody and Mac OS X is the name change and the integration of Carbon.

1.6 Continued Development of the Classic Mac OS

After Apple bought NeXT, not all its development efforts were spent creating Rhapsody and eventually Mac OS X. While development proceeded with the new operating system, several releases of the classic Mac OS bridged the gap between System 7 and Mac OS X.

Rhapsody, Mac OS X, and Intel

Over the years it has been rumored that there would be an official Mac OS X release for Intel x86 hardware. These rumors started from the day Apple purchased NeXT and were well founded as NEXTSTEP had long run on *x*86 hardware. They were further fueled by Apple's early confusion about what it was going to do with the raft of technologies acquired from NeXT. And all the talk about boxes—red, yellow, and blue—didn't help matters.

While Apple maintains Darwin (the underlying Unix layer of Mac OS X) to run on *x*86 processors, there are continued rumors that Apple is secretly working on a version of Mac OS X to run on Intel-based hardware. This project, reportedly codenamed *Marklar* (after the aliens in the *South Park* cartoons) is supposedly being developed somewhere deep inside Apple's Cupertino labs. However, nothing has ever seen the light of day and Apple neither confirms nor denies that such a project exists. In fact, Apple is committed to the RISC-based PowerPC platform (at least for the next few years) with the adoption of the new PowerPC 750 (aka the G5), designed and manufactured by IBM.

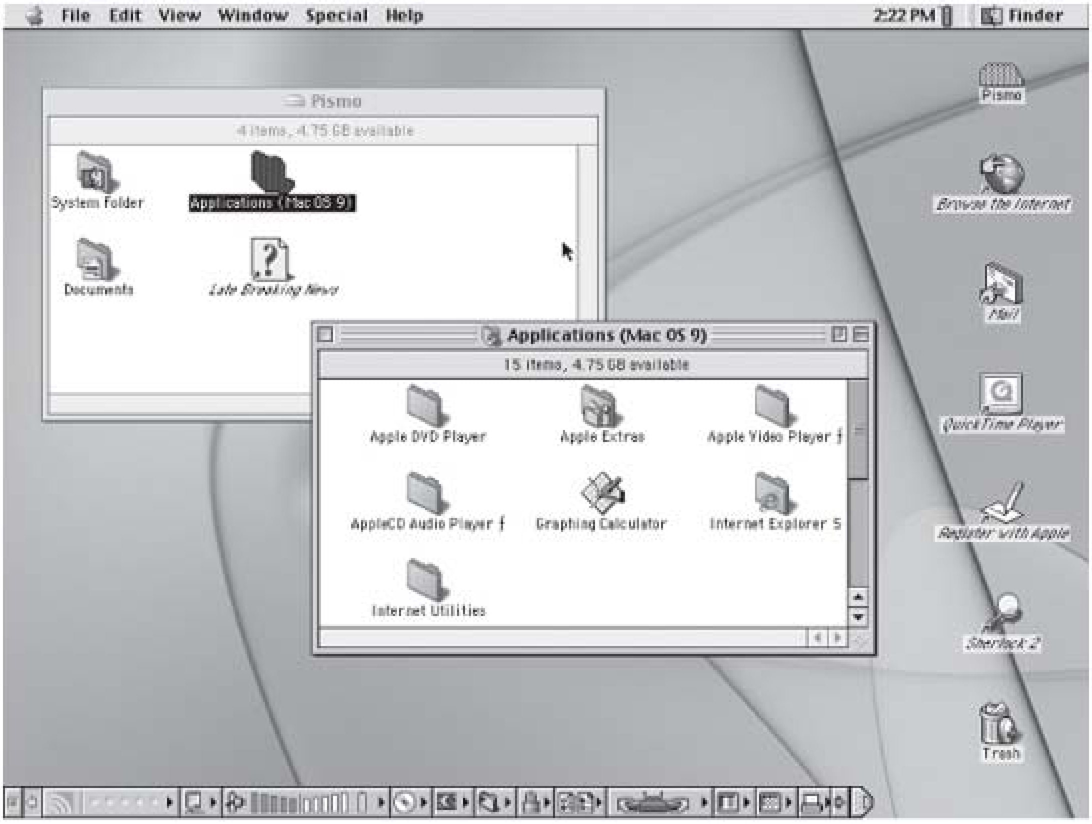
It is always possible, however, that Apple may one day change to a different chip architecture. And it's not outside the realm of possibility that there might one day be an Intel chip at the heart of a Macintosh. For now though, it seems Apple is putting effort into maintaining the *x*86 port of Darwin in order to preserve the cross-platform abilities of the Unix core of Mac OS X.

Mac OS 8, which made its appearance on July 22, 1997, was the first of these releases. Hailed at the time as the "Most significant Macintosh operating system since 1984," it brought a new platinum-based look and feel to the desktop (based on the look and feel from Copland), complete with spring-loaded folders that popped up when a file was dragged onto them, and contextual menus activated by holding down the Control key while clicking an icon. Mac OS 8 also made it easier for users to connect to the Internet with an Internet Set Up Assistant and bundled Netscape Navigator and Microsoft Internet Explorer. Finally, Mac OS 8 marked the introduction of the Macintosh Runtime for Java, allowing applications written in the Java cross-platform language to run on the Mac.

More important to the migration to Mac OS X, Apple was busy making the modifi- cations required so it would run well both as its own operating system and in Mac OS X's Classic mode. In addition, a lot of work went into the Carbon library so that developers could start writing applications that would run on both the old and new systems. This turned out to be a protracted exercise in identifying the system calls that developers were using and resulted in several versions of the Carbon libraries being released for the classic Mac OS.

After several releases of Mac OS 8 (8.5, 8.5.1, and 8.6), the development of the original Mac OS moved into its final phase with the release of Mac OS 9 on October 23, 1999. This release introduced: an updated desktop look and feel, as shown in Figure 1-4; the Keychain application, allowing all of your passwords to be held in one place; automatic software updating over the Internet; and the Network Browser, allowing easy browsing of the servers and computers connected to the local area network. This release came with an updated Carbon library for developers to write against and many more changes to the core of the OS to enable it to run under Mac OS X as Classic. Mac OS 9 also sported many performance enhancements; after all, for a considerable amount of time it was still the OS that shipped by default on every new Mac.

Figure 1-4. The Mac OS 9 Desktop



Development of Mac OS 9 wrapped up with the release of version 9.2.2 on December 5, 2001. This version ships as Classic in every new Macintosh. At WWDC 2002, Steve Jobs read a eulogy to Mac OS 9, dramatizing the message with a fog-shrouded coffin.

1.6.1 The Introduction of the iApps

October 1999 didn't just bring Mac OS 9, it also brought to light iMovie, the first of the so-called iApps. Using the Mac's built-in FireWire port, iMovie was the first product of its kind to allow home users to connect a handheld camcorder to a Mac and edit and produce their own movies. Following the success of iMovie, Apple moved forward in January 2001 with its iApps strategy.

iMovie was soon followed by iTunes, for managing digital music, and iDVD for taking the movies created in iMovie and creating DVDs that could then be used on any home DVD player.

The release of these three applications marks a turning point for Apple. Not only was the Mac OS being conceived of as an operating system that would help sell Macs, it was also being conceived of as the center of a larger strategy whereby the Mac would enable the so-called Digital Hub. Instead of just being a provider of hardware and the operating system that ran on it, Apple positioned itself as a provider of solutions that pulled together the various digital devices that you might have and make them more useful.

1.7 Mac OS X 10.0

Apple shipped Mac OS X Public Beta in September 2000. More than 100,000 people bought the beta, and Apple reported that there were more than 75,000 feedback submissions. Not only did the Public Beta serve to indicate that Apple was going to ship Mac OS X, it helped Apple identify the issues that users needed to have addressed in order to replace the original Mac OS.

After four years in development, the first full release of Mac OS X, known to the engineers who worked on it as *Cheetah* and to the public as version 10.0, finally shipped on March 24, 2001—17 years and 3 months after the introduction of the original Macintosh. People gathered at stores everywhere to welcome its introduction. The fusion of Mac OS and Unix was complete, and Apple finally had the next-generation operating system that it wanted.

This release brought the following features:

Carbon

> A subset of the original Mac OS Toolbox APIs that could be safely transitioned from the old Mac OS to the new one, Carbon was the lifeline for older Mac OS applications, allowing them to be ported to the new system. This meant that applications essential for attracting users to the new system, such as Microsoft Office and Adobe Photoshop, could be ported to Mac OS X and take advantage of preemptive multitasking and protected memory.

Quartz 2D

> The powerful imaging layer of Mac OS X, Quartz 2D handles all the drawing on the system and delivers a rich imaging model based on PDF (replacing the Display PostScript used by NEXTSTEP), on-the-fly rendering, and antialiasing—all managed by ColorSync to ensure graphics look their best.

Quartz Compositor

> The windowing system for the OS and provider of low-level services such as event handling and cursor management, the Quartz Compositor is based on a "videomixer" model where every pixel on the screen can be shared among windows in real time. This model allows for smooth transitions between the states of a GUI; one of the traits of the Aqua experience.

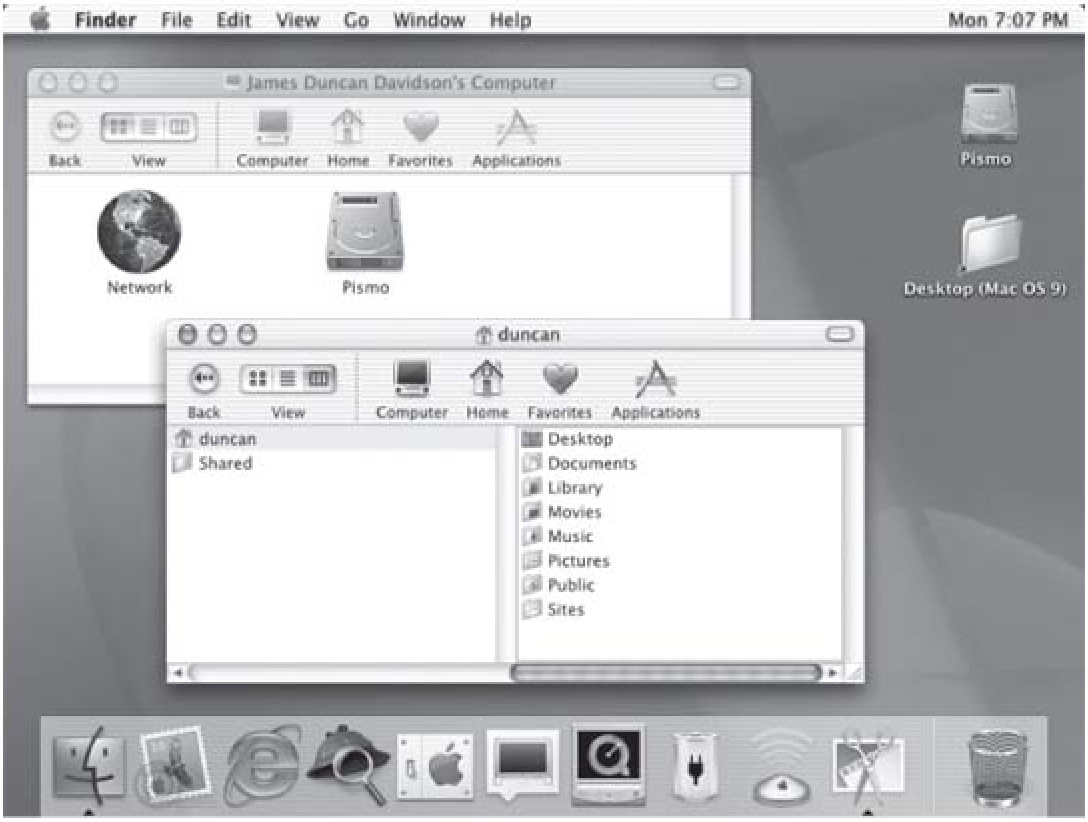
Aqua

> More than a color, and more than something that brings to mind the properties of water, Aqua is an attitude. When Apple set out to design the interface for Mac OS X, it wanted to take all the good parts of the previous Macintosh and mix them with new features made possible by modern computer hardware.
>
> Aqua brings the user interface to life with depth, transparency, translucence, and motion.

Aqua also recognizes that it is managing a thoroughly multitasking machine. Instead of popping up dialog boxes in the middle of the screen, Aqua provides sheets that are attached to the window that they pertain to, allowing other tasks to be performed without hindrance and clearly communicating function to the user.

The Aqua interface is pictured in Figure 1-5.

Figure 1-5. The Mac OS X 10.0 Desktop and the Aqua user interface



Mail

An easy-to-use mail client compatible with Internet-standard mail servers that use the SMTP, IMAP, and POP protocols, Mail shipped with a built-in configuration to seamlessly access iTools-based Mac.com mail.

iTunes and iMovie

Critical to Apple's Digital Hub strategy, the first release of Mac OS X also included ports of the iTunes and iMovie applications, which first appeared on Mac OS 9.

Impressive as it was, by all measures the first release of Mac OS X was not quite ready for most of the Mac faithful. It lacked DVD playback, and the major applications like Adobe Photoshop and

Microsoft Office weren't released yet. Still, it was more than usable for the Unix geeks, who were among the early adopters. Many improvements were to come, but Mac OS X's first consumer release signaled that it would not be another failure like Copland.

1.7.1 Darwin and Open Source

As Mac OS X was developed, Apple made sure the Open Source foundations of Mac OS X remained open by starting up the Darwin project. Darwin is essentially the non- GUI part of Mac OS X and is a complete operating system in itself. If you are interested in running Darwin separately, you can download it for free from Apple's Darwin Site (*developer.apple.com/darwin/*) and have a full working operating system. It won't have the Mac OS X Aqua desktop, but you can use Darwin just like you would Linux, from the command line and using the X11 window manager.

The main tree of Darwin is kept under pretty careful control, after all, Apple builds the base to an operating system that it distributes to millions of people from this tree. In order to provide a place for experimentation, Apple and the Internet Software Consortium run the OpenDarwin project. You can find the OpenDarwin site at *www. opendarwin.org/*.

1.7.2 Developer Tools

When Apple released Mac OS X, they made a great decision by also deciding to provide development tools to every Mac user for free including the Project Builder IDE and the Interface Builder GUI layout application. These aren't just simplified tools to learn development with; they are the same tools that Apple uses to develop the operating system itself as well as its various applications. These tools allow development of Carbon- and Cocoa-based applications, system libraries, BSD command-line utilities, hardware device drivers, and even kernel extensions (known as KEXTs).

The Developer Tools aren't installed by default because Apple doesn't think most users will want them and would probably want the almost 500 MB of disk space for something else. But developers, as well as power users who want to compile programs available in source code form on the Internet, can easily find them and install them from a variety of sources. And, since they are free, any user who wants to try developing software can do so purely by investing the time it takes to learn.

1.7.3 Command-Line Access

It is difficult to say what kind of success Mac OS X would enjoy without having access to the command line. According to legend, as Rhapsody developed into Mac OS X, there were many within Apple who didn't want to ship the Terminal command-line tool—or at the very least only wanted it installed as part of the developer tools. After all, the command line was seen as the antithesis of the Mac experience. However, those who wanted the command line to be present prevailed and the Terminal shipped in the */Applications/Utilities* directory.

This was a fortunate move indeed as a large percentage of the early adopters of Mac OS X were not traditional Mac OS users, but were switchers from Linux and other flavors of Unix including Solaris and FreeBSD. Without easy access to the command line, these users—a market many at Apple in 2000 probably didn't expect to sell to—would have never moved to the platform.

1.8 Mac OS X 10.1

The next release of Mac OS X, version 10.1 (known to Apple engineers as *Puma*), was released in September 2001, just six months after the initial release of the system. With Puma, Apple focused on performance and ensuring that the critical Carbon layer was robust enough for Microsoft Office and Adobe Photoshop to be released. Puma, along with the large influx of applications that came with it, made it clear that Mac OS X was going to be a success and that the strategy for migrating from the old Classic Mac OS to the new OS was going to work.

The quick release cycle of Puma also indicated that Apple was moving rapidly to improve the OS in ways that were meaningful for both Classic Mac OS users as well as those switching from other platforms like Windows. As a maintenance release, it was released as a no-charge upgrade—the only version of Mac OS X to date that didn't have a price tag associated with it.

With Puma came the release of the last of the original iApps to make the jump from OS 9: *iDVD*. Apple's Digital Hub strategy was now fully on Mac OS X, signalling that Apple's application development efforts were now entirely focused on Mac OS X. Apple further strengthened the iApps strategy in January 2002 with the release of *iPhoto* for managing digital photos, and continued in July 2002 with the release of *iTunes 3* as an update to the popular music jukebox application.

Also revealed in July was .Mac (pronounced "dot-Mac"), a suite of Internet services, including those that used to be part of the older iTools service, and software. The new pay service included the McAfee Virex virus scanner and the new Backup software for making safe backups of critical data to .Mac.

1.9 Mac OS X 10.2 Jaguar

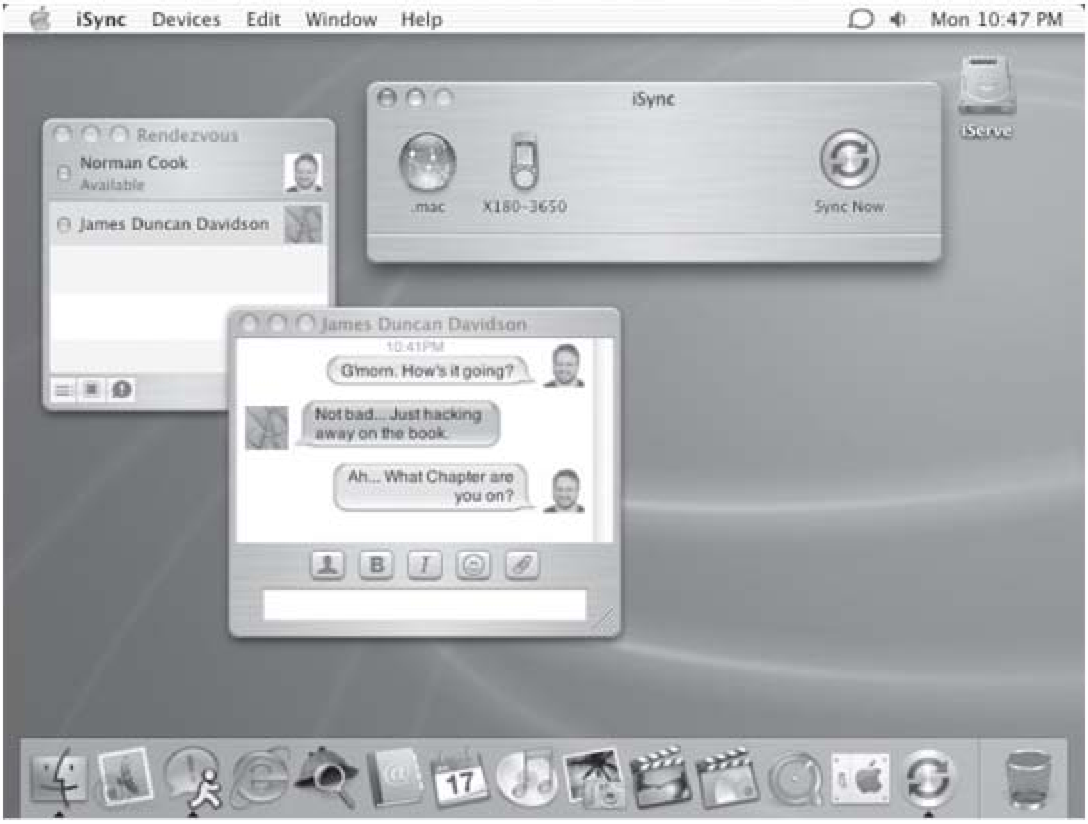
The next release took Mac OS X to the next level. Unlike previous versions of Mac OS, Apple decided to put the codename, *Jaguar*, into the name of the product itself. On the box and at its web site, it is known as Mac OS X Jaguar. It is also the first release of Mac OS X to come as the default boot OS on every system Apple sold, marking an important step in the transition between the classic Mac OS and Mac OS X. The Jaguar desktop is shown in Figure 1-6.

Jaguar introduced the following features:

Quartz Extreme

> Building on the already powerful Quartz Compositor, Quartz Extreme performs most of its work in OpenGL, allowing it to take advantage of the enormous potential of modern graphics processors—and offloading the tasks from the computer's CPU. With Quartz Extreme, all applications on the system have easy access to 3D capabilities, and this paves the way for later user interface improvements.

Figure 1-6. The Jaguar desktop with iChat and iSync



G'morn. How's it going?

Not bad... Just hacking away on the book.

Ah... What Chapter are you on?

Address Book

> Previous versions of Mac OS X had a rudimentary contact database. Jaguar introduced a new system-wide contact database, along with a management application, called simply the Address Book. With its simple and elegant API, the Address Book can be incorporated into any application, and all the built-in applications on Jaguar, such as Mail and iChat, support it.

Rendezvous

> Built on the work of the IETF Zero Configuration (Zeroconf) effort, Rendezvous enables machines to discover each other without user intervention. This lets you share services with ease between machines on the same network and allows you to seamlessly connect to machines by setting up a local network and just hooking up cables or setting up an ad hoc wireless network. You can find out more about ZeroConf at *www.zeroconf.org*.

iChat

> Another in Apple's series of iApps, iChat is an elegant AIM- (AOL Instant Messenger) compatible application that also leverages the strength of Rendezvous to allow chats between users on the local network. This is handy for ad hoc networks such as those at conferences and other events.

Windows-Compatible Networking

> Leveraging the capabilities of Samba, an open source software package for working with the Windows SMB file sharing protocol, Jaguar can share its filesystems with Windows clients as well as mount Windows-based filesystems. Jaguar also introduced the capability to integrate with Windows Active Directory to ease the migration of Macs into Windows-dominated networks.

Even though Jaguar shipped with enough features to make most people happy, Apple didn't stop there. After Jaguar was released, Apple released the following products for it:

iCal

> This personal calendaring application handles multiple calendars and enables sharing of those calendars via the Internet. Calendars can be published either through .Mac or to any server that supports the WebDAV protocol.

iSync

> A tool for synchronizing the address book and calendars on a Mac to mobile phones, PDAs, and the iPod. iSync allows information to be kept on multiple machines by synchronizing data against the .Mac service.

iLife

iLife is a bundle of new versions of iPhoto 2, iMovie 3, and iDVD 2 along with already available iTunes 3 as a shrink-wrapped, boxed product available for sale in stores. Even though iLife has a cost associated with it, iPhoto, iMovie, and iTunes remained free downloads. Only iDVD is available exclusively through iLife.

Safari

A brand new browser with a highly-tuned rendering engine, Safari replaces Microsoft's Internet Explorer as the browser of choice on Mac OS X. This browser ful- filled a real need as evidenced by the fact that there were 500,000 downloads of the public beta between January 7 and 9, 2003.

1.10 Mac OS X Panther

Panther, the name given to version 10.3 of Mac OS X and the system that this book covers, brings another host of improvements to the system and continues in the Mac OS X tradition of "It's getting better all the time." A few of the improvements made to the system are:

A new Finder

The Finder that shipped with Mac OS X 10.0 through 10.2 was a simplified interpretation of the classic Finder from the original Mac OS that was designed to work on a single-user system. On the multiuser Mac OS X, this meant the files that the typical user cared about—those in their Home directory—were located three levels away from the filesystem root.

The new Finder in Panther, shown in Figure 1-7, introduces a new Sidebar on the left-hand side of every Finder window that gives you quick access to the various disks attached to your computer, your directory, and any other locations that you put there. When you select a disk or a folder from the Sidebar, it becomes the start of what you see in the window view. And, changing a paradigm in place since 1984, removable disks can be ejected or disconnected by clicking on an eject icon in the Finder's Sidebar, rather than having to drag disks to the Trash.

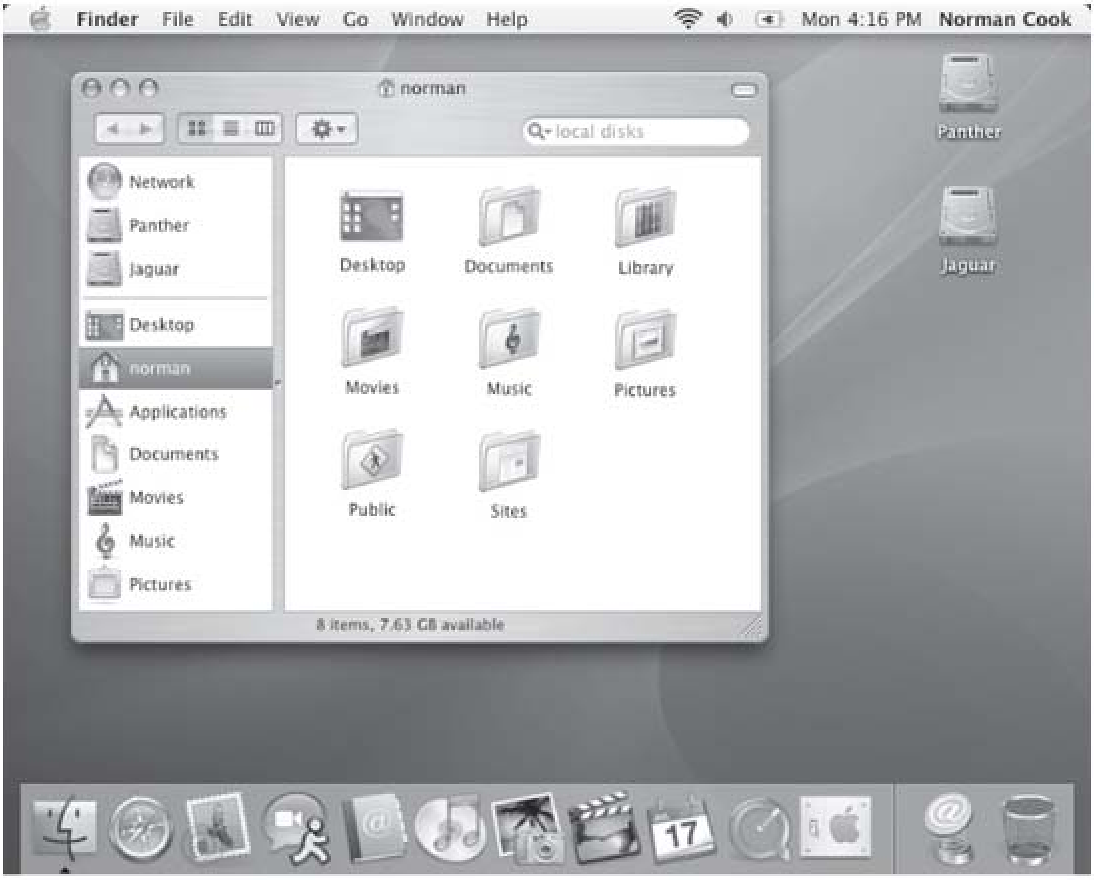
Exposé

Building on the power of Quartz Extreme, Exposé introduces a new way to work with the dozens of windows that are open on the typical Mac OS X user's desktop. Instead of having to dig through the open windows one by one, you can use either a keystroke or a mouse gesture and perform one of three actions:

- Show all the windows open in miniature so you can see all of them at once.
- Show all the windows belonging to a particular application.
- Clear all the windows off the screen so you can see the desktop.

These actions can be set to a variety of keystrokes and mouse gestures by using the Exposé preference panel in System Preferences.

Figure 1-7. Panther's Desktop and the new Finder



Fast User Switching

Built on Unix, Mac OS X has always allowed multiple users and allows there to be multiple users logged in at once via the command line. Panther takes a cue from Windows XP and introduces the capability for multiple users to be logged in to the GUI at the same time. When you change users, the new desktop rotates into place giving a very clear indication that you are entering into another user's desktop; an example of the many ways in which the graphics abilities given to the system by Quartz Extreme have been put to use.

Fast Application Switching

You can quickly switch between applications using the ⌘-Tab keystroke. This feature was present in previous versions of Mac OS X, but in Panther it works the same way it does in Windows, featuring an elegant transparent window that appears on screen to allow you to see what application you are switching to.

Security

Panther introduces a host of security features designed to make data safe. The new FileVault feature can encrypt the contents of your home directory so even if somebody gets physical access to your disk, the contents of your files aren't exposed. When it's time to empty your Trash, you can choose to overwrite the deleted files with random data so the old files cannot be recovered.

In addition, all the GUI applications in Panther, and most of the command-line ones, are aware of the Kerberos security services that Panther makes available. Kerberos is a cryptographic secure network authentication protocol that enables single sign-on. After entering your password when you log in, all of your network applications, such as Mail, can automatically log in to Kerberos-enabled servers.

Font Book

Panther's Font Book application builds advanced font management into Mac OS X. Instead of managing fonts via the filesystem, Font Book lets you activate and deactivate fonts on the fly, organize them into collections, and preview fonts. It also allows you to take advantage of ligatures, kerning, and many other font features.

iDisk Synchronization

Building on the .Mac online service, iDisk is now a permanent fixture in the Finder. As a new feature in Panther, you can opt to have your iDisk synchronized with your computer. This allows you to work transparently with the same files on multiple machines whether or not you are online at the time you want to work with them. The only catch is that you have to dedicate as much space on your local hard drive as you have on your iDisk, which with a basic .Mac account is 100 MB.

Xcode Tools

As part of Panther, Apple revamped the developer tools, replacing Project Builder with a new IDE called Xcode and dubbing the entire toolset Xcode Tools. Including the Xcode IDE, Interface Builder, *gcc* 3.3, updated documentation, and performance tools. Xcode provides everything you need to develop applications on, and for, Mac OS X.

Installing the Xcode Tools

You can quickly check to see if you have Xcode Tools installed by seeing if you have a */Developer* folder on your hard drive. If so, you are ready to go. If not, you'll need to install the tools either from the Xcode Tools CD that came with your copy of Mac OS X or from a disk image that you can download from the Apple Developer Connection (ADC) web site. If you bought your Mac with Mac OS X installed on it, you'll find the Xcode Tools disk image on your hard drive.

A special note: Apple provides regular updates (two to three times a year) to the Xcode Tools through the ADC web site. These releases introduce new features, fix bugs, and improve the available documentation. If you don't want to download the rather large files, Apple will send you a copy on CD for a nominal charge. Log in to the ADC Member web site and go to the Purchase section.

1.11 What Does the Future Hold?

Without a doubt, Apple has many more features in store for future versions of Mac OS X. There are many at Apple who say building on top of Unix has allowed the company to innovate faster than expected—and with features like Exposé building on top of Quartz Extreme I'd have to agree. So, even though nobody outside Apple's headquarters in Cupertino has a clue as to what may be coming up next, it is certain that Mac OS X will continue to break new ground. Apple's choice to use Unix—a well understood, stable, and time-tested design—as the foundation of the system means Apple can focus on innovating above Unix and continue to bring improvements in the way that we use the system.

Whatever plans Apple has, it has surely noticed that Microsoft adopted several of Jaguar's features into Windows XP and that the next version of Windows—codenamed *Longhorn*—has a feature list that borrows many more features from Panther. Whatever Apple has up its sleeve, it's safe to say that the company is aware of this and fully intends to leapfrog Longhorn even before it finally shows up.

Enough about the past and the future. Let's look at what you can do with Panther today!

Chapter 2. Lay of the Land

When you first log in to Mac OS X you see a user interface that is the end result of 20 years of development; quite a bit of work has gone into making it an elegant and usable interface for your computer. Panther's new Finder goes even further, bending the computer's view of the world to your own, allowing you to view the filesystem any way you want to. Underneath it all, however, Mac OS X is a structured environment based on Unix and to truly master it you'll need to know how it all fits together.

To navigate the system, you use two programs: The GUI-based Finder, and the command- line-based Terminal. Each of these programs gives access to different layers of the system. For some tasks, the Finder is the best tool for the job. For others, using the Terminal is a way of life.

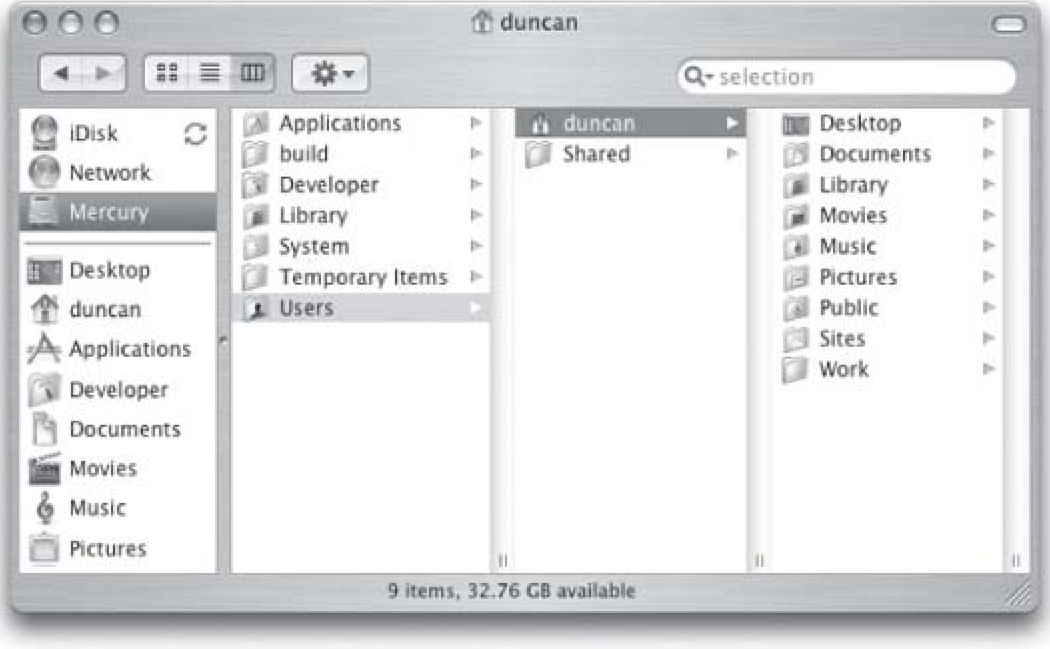
This chapter explains how the filesystem in Mac OS X is put together and shows you how to navigate your way through it using the Finder and the Terminal.

2.1 Filesystem Hierarchy

Like most other filesystems, the Mac OS X filesystem is conceptually a hierarchical tree-based structure that branches from a root. In the case of Mac OS X, the filesystem is rooted on the drive partition that the system booted from. In the typical case where you boot off the single partition of the hard drive that is in your machine, your filesystem is rooted there. If you have more than one disk drive, then the filesystem is rooted from whichever partition you booted from. Likewise, if you boot from an external disk drive or even a disk image on a network server, the filesystem will be rooted from there. Figure 2-1 shows a Finder window displaying the boot disk's filesystem.

The structure of the Mac OS X filesystem at the root level is very strict, and almost every file that Mac OS X needs to run has a specific place within it. Some of the folders at the root of the filesystem are visible in the Finder when you click on your boot drive, others are not.

Figure 2-1. The Finder showing the boot disk



2.1.1 The Filesystem through the Finder

The folders at the root of the filesystem that are visible in the Finder are:

Applications

This folder contains applications that are available to all users on the system. Most applications that Apple ships for Mac OS X (such as iCal, iPhoto, and Safari) are located here. In addition, a large number of useful utility programs, such as the Terminal and Activity Monitor, are installed in the Utilities subfolder.

Developer

If you have installed the developer tools on your system, most of the applications, documentation, examples, and other files needed for building applications for Mac OS X are located here.

Library

This folder contains application- and system-specific resources, such as fonts and application preferences, that are needed by all the users on the system. In some respects, this folder is like its namesake in that it holds a lot of useful information for the various parts of the system.

System

This folder contains the resources used by the operating system itself. Mac OS X is very particular about the files in this directory and will prevent you from messing with them. It's best to consider this directory strictly Apple's domain.

File Extensions and the Finder

By default, file extensions, the part of a filename that comes after the dot, are hidden in the Finder. This means a file named *DialUpNumbers.txt* will appear in the Finder, by default, as *DialUpNumbers*. Some folks, especially old-time Mac OS 9 users, prefer this approach as they wish to look at the icon to determine what kind of file it is. Others, however, including most of us who cut our teeth on Unix, prefer to see the file extensions intact.

To make file extensions viewable all the time, use the Finder→Preferences menu (⌘-,), click on the Advanced button in the toolbar, and then click on the checkbox next to "Show all file extensions".

Users

This folder contains the Home folders for the users on the system. Within a Home folder, a user is the master of his domain. Outside a user's Home folder, the ability to make changes is greatly limited and depends on the type of user they are.

If the Classic Mac OS 9 environment is installed on your machine, you'll also see the following folders:

Applications (Mac OS 9)

This folder contains applications that aren't designed to run on Mac OS X.

Desktop Folder

This folder contains all the files and folders that are part of the Mac OS 9 desktop. This folder is useful when you boot back and forth between Mac OS 9 and Mac OS X and need to get to the files that you left on the Mac OS 9 desktop.

Documents

This is the traditional folder in Mac OS 9 for storing your documents and is placed here for convenience. In Mac OS X, you should always use the Documents folder in your Home folder instead of this folder.

System Folder

This folder contains the Mac OS 9 system and supporting files.

These folders aren't part of Mac OS X but are located at this level of the filesystem for ease of use when you boot into Mac OS 9 from the drive.

2.1.2 The Filesystem Under the Hood

The Finder's view of the filesystem doesn't reveal everything that's there. The Finder is just one view, and for most users it is sufficient because it keeps hidden system files, otherwise known as *dot files*, from view. The Terminal (*/Applications/Utilities*), however, lets you dive beneath the pretty GUI view and see everything in the filesystem. Example 2-1 shows the output of the *ls* command.

> I am going to give you a little bit more grounding in the Terminal in the next chapter. For now, just play along.

Example 2-1. The command-line view of the filesystem root

```
$ ls /
```

```
total 8457
Applications    Network           Volumes      etc          private
Desktop DB      SalesBackups      automount    mach         sbin
Desktop DF      System            bin          mach.sym     tmp
Developer       Temporaryn Items  cores        mach_kernel  usr
Library         Users             dev          mypants      var
```

As you can see, this is a much different view of the filesystem than you'll see in the Finder. Some of the folders—*Applications*,*Library*,*Network*,*System*, and *Users*—are the same as those in the Finder, but many aren't exposed. You'll see this kind of difference time and again in Mac OS X. The GUI gives you a filtered view of the system, and most of the time this filtered view is exactly what you want. The Terminal, on the other hand, gives the raw view of the system giving you unfettered access to the depths of the system.

The folders and files visible using the command line at this level but hidden in the Finder are:

Desktop DB

Desktop DF

> These are the hidden files behind Mac OS 9's desktop. The Desktop folder in the Finder view uses these files to present your Mac OS 9 desktop to you while you are booted into OS X.

Network

> This virtual filesystem allows you to browse the filesystems shared from servers on your network.

Volumes

> This directory contains all the disks, other than the boot disk, that are attached to your computer. For example, external FireWire disks (including an iPod) as well as CDs, will show up here.

automount

This directory contains links to any filesystems mounted from network file servers using network protocols such as AFP, NFS, and SMB.

bin

This directory contains many of the executable Unix programs that your system needs.

cores

This directory contains crash files for the various Unix programs. This is the traditional location in which Unix programs write out their crash data.

dev

This directory contains the various device nodes for the system. One of the core ideas of Unix is that the various devices can be treated as a files—including some devices that you normally wouldn't think of as being a file. The *dev* tree contains the "files" that represent everything attached to the operating system.

etc

This directory contains configuration files used by the various Unix components of the system.

mach, mach.sym, and *mach_kernel*

These files make up the kernel, the core part of the operating system that manages everything else.

private

This is the directory that actually contains the *etc*,*var*, and *tmp* directories. These directories actually don't live at this level of the filesystem but are linked into the root (/) directory.

sbin

This directory contains system executables needed at startup as well as some configuration utilities for the system.

tmp

This is a place for the temporary files that some Unix programs create.

usr

This directory contains the Unix tools and libraries that are intended for all users on the system.

var

This directory contains data that changes frequently such as mail spools and log files. The disk files that implement the backing store for virtual memory are also held here.

2.1.2.1 More hidden files

There's actually another layer of hidden files beyond what you see with the ls command. If you change the command and add an *-a* option (to show all files), you'll see the extra files shown in Example 2-2.

Example 2-2. Hidden files in the root directory

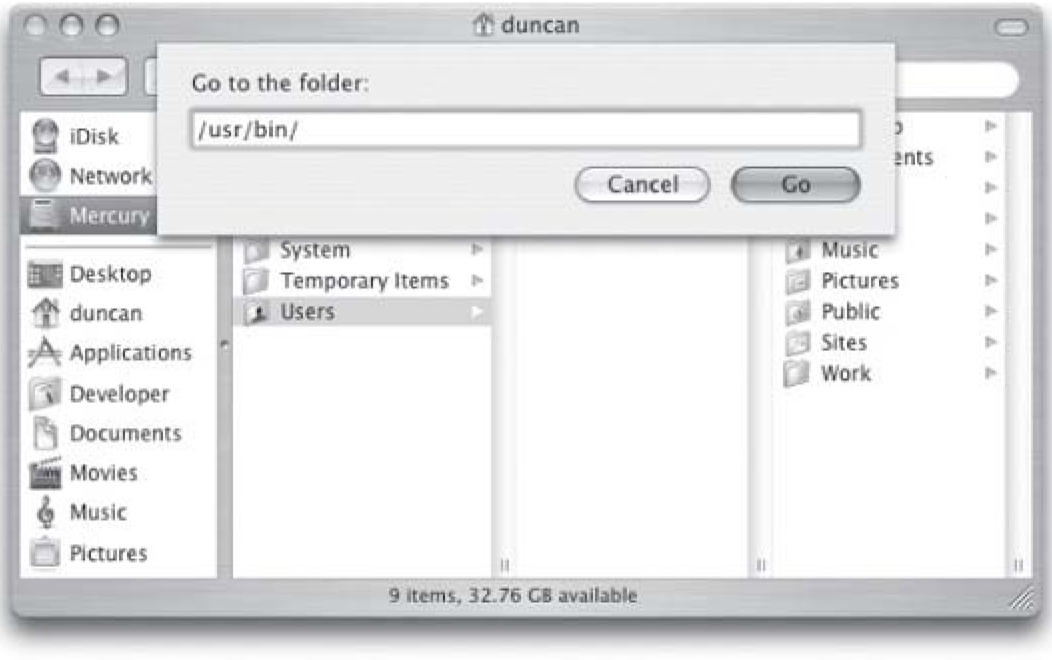
```
$ ls -a /
.                   .vol            Network           bin         mach_kernel
..                  Applications    System            cores       private
.DS_Store           Desktop DB      Temporary Items   dev         sbin
.Trashes            Desktop DF      Users             etc         tmp
.hidden             Developer       Volumes           mach        usr
.hotfiles.btree     Library         automount         mach.sym    var
```

The*.DS_Store* file contains the Finder settings and the comment text for the various files within the folder. The rest of these files are, well, files that you really don't need to see. They are hidden from the command-line view which indicates that you should only mess with them if you really need to and you know what you are doing.

2.1.2.2 Opening hidden directories in the Finder

Even though the Finder hides these directories from you, there is an easy way to get to them. Simply use the Go To Folder command (Go→Go to Folder or Shift-⌘-G) and enter the path to the folder you want to see. An example of this is shown in Figure 2-2.

Figure 2-2. The Go To Folder dialog



If you want to see every file on your system all the time, you can do so by typing the following command:

```
$ defaults write com.apple.Finder AppleShowAllFiles YES
```

Then, either log out and log back in, or just give the Finder a few minutes to pick up the change. This seems cute at first (and some people may find it useful), but it can quickly become irritating, so you might find yourself preferring the Finder's regular view instead. To reverse this setting, use the following command:

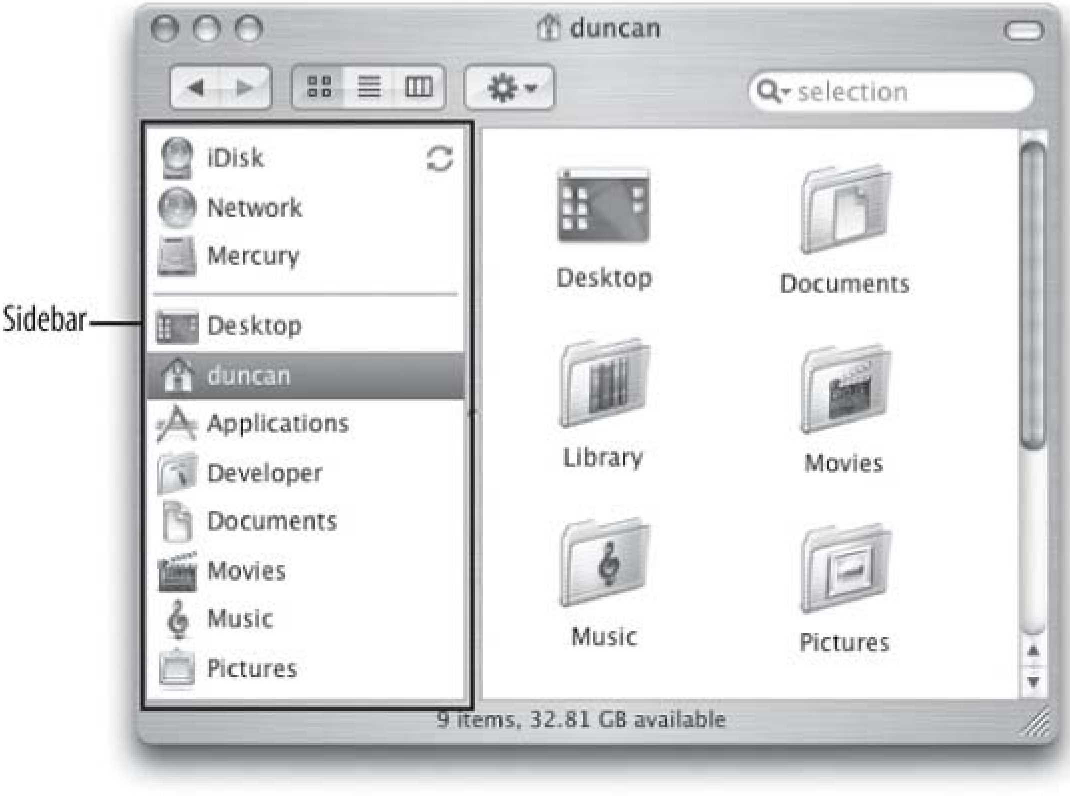
```
$ defaults write com.apple.Finder AppleShowAllFiles NO
```

This command uses the defaults system which I'll talk about in Chapter 9,*Preferences*and *Defaults*.

2.2 The Many Roots of the Finder

Panther's new Finder allows you to ignore the way the filesystem is structured at the command-line level and look at it in a variety of ways that are more relevant to everyday tasks. By selecting one of the items in the Sidebar (shown in Figure 2-3), the view to the right of the Sidebar changes to reveal that folder's contents, or specific details about a selected file.

Figure 2-3. The Finder's sidebar



The most useful feature about this new tool in the Finder is that it gives you quick access to your Home folder (which is usually the center of activity on your computer), or to any other folder that you place in the Sidebar.

2.2.1 The Home Folder

All the files, applications, preferences, and resources that are yours and yours alone are located within your Home folder. This is where you should make all your modifications and additions. If you are an old-school Mac OS 9 user, this is where you should feel like customizing your system. And, even better, if you play by the rules, you'll be able to move to a new machine simply by copying your Home folder.

The folders you'll find inside your Home folder are:

Desktop

Contains all the files and folders that appear on the Mac OS X desktop for the user.

Documents

Intended to contain your documents. Of course, you can save your documents anywhere in your Home folder, but this is the recommended location. It is also the default location that will be proposed for you when you save a document from an application.

Library

Contains application- and user-specific resources that belong to a single user only. This allows you to have fonts on the system that nobody else can use. It also allows your applications to save your preferences separate from those of other users.

Movies

Intended to contain your movies. This is where iMovie creates its project files.

Music

Intended to contain your music. This folder is where iTunes keeps its data files as well as your MP3 and AAC files.

Pictures

Intended to contain your digital photographs. iPhoto keeps its data here—including photos you upload from a camera.

Public

Intended to contain the files that you are willing to share with other users, either on the same machine or across the network.

Sites

Intended to hold your personal web site, which can be served by Mac OS X's built-in web server. When personal web sharing is turned on, these documents can be accessed by passing a path of the pattern *~username*. For example, a user with the username *norman* could access these documents from his machine with the URL *http://localhost/~norman/*. Even if you don't run a web server on the public Internet, this is a handy way to publish files to other people on the network.

2.2.2 The Command-Line View of a Home Folder

Unlike the base of the filesystem, the command-line view of the Home folder looks pretty much the same as what you see in the Finder. The only difference is, using the command line you can see the hidden "dot" files that you can't (by default) see in the Finder, as shown in Example 2-3.

Example 2-3. The command-line view of the Home folder

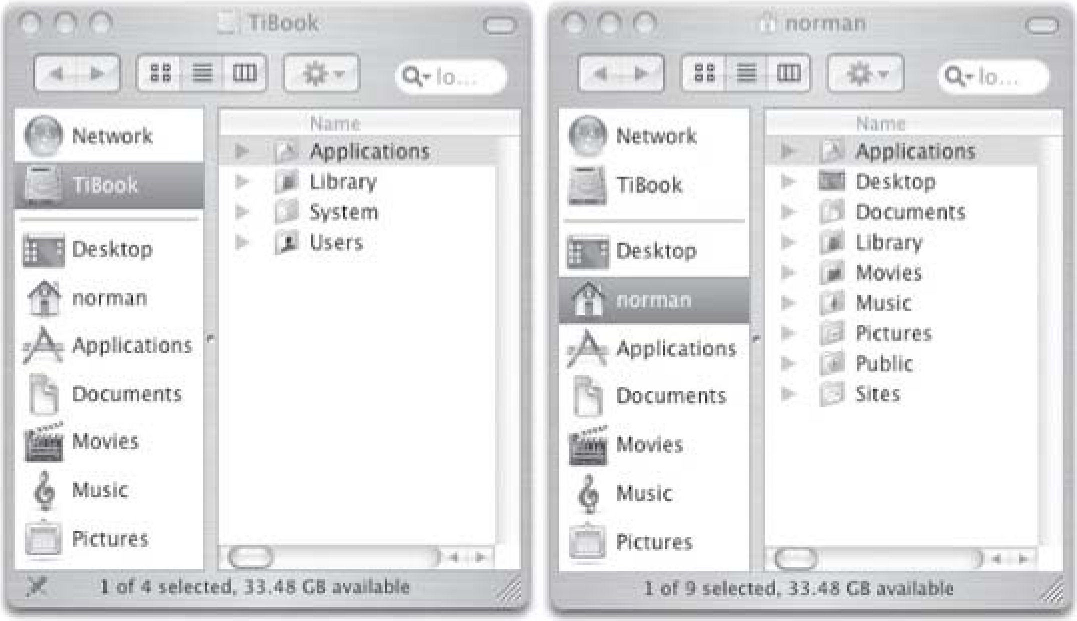
```
$ ls -a
.                        Desktop        Music
..                       Documents      Pictures
.CFUserTextEncoding      Library        Public
.Trash                   Movies         Sites
```

These dot files are either self-explanatory or are files that you usually don't need to worry about. The one thing you should notice is the ~ symbol when you first open a Terminal. This symbol is a shorthand for your Home directory. From anywhere on the system you can construct a path using the ~ symbol, and the operating system will automatically use the full path to your Home directory.

2.3 Filesystem Domains

In our discussion about the filesystem, you've no doubt identified a certain amount of redundancy. There's a *Library* folder at the root of the filesystem, one in your Home folder, and one in the System folder. And if you create an *Applications* folder in your Home folder to store applications that aren't for use by others, you'll note that it automatically gets the same folder icon as the *Applications* folder at the root of the filesystem, as shown in Figure 2-4.

Figure 2-4. The Local and User domains in the Finder



Where to Put Your Applications

The various filesystem domains give you a choice as to where to put your applications. If you are the only user on a system that is going to be using an application, you should consider putting it into an *Applications* folder in your Home folder. This will keep them seperate from all of the applications that come on the system in the */Applications* folder. However, there are many applications—typically older applications that have been migrated from Mac OS 9, that have to be installed in the */Applications* folder. There's no winning sometimes, but at least for many applications, what domain you place them into is your choice.

This is the result of a concept known as Filesystem Domains and is structured in such a way as to allow multiple users to share the same system or to be hosted on a server so they can use multiple systems and yet provide a consistent experience.

There are four domains in Mac OS X:

User

> Contains the resources for a user logged in to the system. As you would expect from the similarity of descriptions, this domain is contained within the user's Home folder.

Local

> Contains the resources that users of a particular system share with each other. The local domain consists of the Applications and Library folders at the root of the filesystem. These resources are available to users of the system but are not available to users on networked computers.

Network

> Contains the resources available to all users of a local area network. Applications, documents, and other resources located in this domain are available on any machine that is part of the network. The folders that hold this domain vary according to network setup but typically appear as a Network folder in the Finder.

System

> Contains the resources required for the system to run. These resources are part of the operating system installation and can only be modified by administrative users.

When a resource is requested by an application, Mac OS X searches these domains—in the order above—to satisfy the request. For example, when an application requests a particular font, the system will search the Fonts directory in the User domain first. If it doesn't find the font there, it will look in the Local domain. If it doesn't find it there, it will look in the Network domain, and finally it will look in the System domain.

This hierarchical search allows a user's configuration file to override a system-wide preference. Furthermore, the consistency in naming of directories between the domains, such as the *Library* directory that appears in each domain, allows for easy management of resources.

2.4 The Library

I've made lots of references to the *Library* directory and have indicated that it is for the storage of resources. Any kind of resource that an application needs can be located in the Library. Here's a list of some of the most common directories you'll find in the Library and the kinds of resources they contain (and remember, you can create one of these directories in a domain in which it doesn't exist):

Application Support

> Contains third-party plug-ins, helper applications, templates, and even data for the applications on your system. You'll usually find the resources for an application in a subdirectory named after the application.

Audio

> Contains drivers, plug-ins, and sounds for Mac OS X's audio subsystems.

Caches

> Contains temporary data used by the applications on your system. For example, Safari will keep web page data here so when you revisit a page, you don't have to download all the content on it again.

Calendars

> Contains the ICS files used to store the calendar data for a user. This directory is used by iCal and only appears in the User domain.

ColorSync

> Contains profiles and scripts used by Mac OS X's ColorSync color management subsystem.

Documentation

> Contains documentation for various parts of the system. Also, the Help application uses this folder to hold files that it displays.

Favorites

Contains aliases to frequently accessed folders, files, or web sites. This directory only appears in the User domain.

Fonts

Contains fonts for applications to use. This fonts in this directory are easily managed using Font Book (*/Applications*).

Frameworks

Contains the frameworks and shared libraries that applications need to operate.

Internet Plug-Ins

Contains the various helper applications, such as the Flash Player, that extend the functionality of your web browsers.

Mail

Only appearing in the User domain, this directory contains the mail for a user.

Preference Panes

Contains applets that will appear in the System Preferences application.

Preferences

Contains the preferences for an application.

Printers

Contains printer drivers and printer definition (PPD) files. The drivers are organized by vendor name.

Scripting Additions

Contains scripts and scripting resources to extend AppleScript's capabilities.

StartupItems

Contains scripts and programs that are run at boot time. This only appears at the local and system level.

WebServer

Appearing in the Local domain, this directory contains the content and CGI scripts for the system's web server. When you turn on web sharing on your machine, you can access the documents from your local machine using the URL *http://localhost/*.

Many other directories may appear in your Library, but as you can see, the files that you find here affect, control, and configure your experience on Mac OS X.

2.5 Further Explorations

A great resource for expanding your understanding of the way in which the system is put together is installed on your hard drive when you install Xcode Tools. It's a 191-page PDF book titled*System Overview* and is located at */Developer/Documentation/MacOSX/Conceptual/SystemOverview/SystemOverview.pdf*.

Chapter 3. The Terminal and Shell

The Terminal application (*/Applications/Utilities*) is the portal to the internals of Mac OS X. You can use, and become proficient with, the system without ever touching the Terminal. But if you truly want to dig deep and learn how to unleash the full potential of the underlying Unix capabilities of the system, the command line is essential. And once you know how to use it, the Terminal becomes a tool so valuable that many power users keep it in their Dock or in the Finder's sidebar for quick access.

Tempting as it may be to think of the Terminal as the Unix part of Mac OS X, it's simply an interface to the underlying Unix operating system, and specifically to those programs that give the system its Unix character.

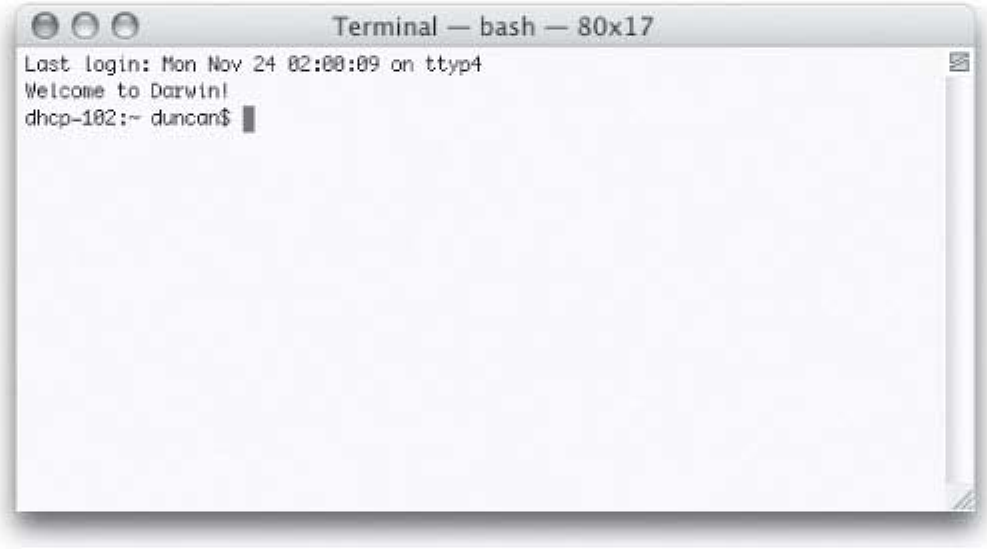
I'm going to make the assumption that you have at least a passing familiarity with the idea of the command line. Maybe you remember using a shared system at a school somewhere. Or maybe you had a DOS-based machine that required you to go sleuthing into the depths of the `C:\` world. In any case, the aim of this chapter is to familiarize you with the Terminal, the shell, and some of the other tools you'll need through the rest of the book.

> If this chapter goes over your head and you need more of a grounding on the subject, you should pick up *Learning Unix for Mac OS X Panther*, by David Taylor, et al. (O'Reilly & Associates, Inc., 2004).

3.1 Terminal Overview

When you launch the Terminal application, shown in Figure 3-1, you are greeted with a single, rather plain-looking window. What you see here is essentially the same command- line interface that Unix users have been seeing since the days when 8-track tapes were en vogue. The system greets you with a friendly "Welcome to Darwin!" message and then gives you a prompt for the *bash* shell, indicated by the dollar sign ($). This is your cue that you are interacting with the command line. At this point, you have direct access to the internals of Panther, and if you have administrator privileges and a little bit of know-how, you can do everything you ever wanted to, and more.

Figure 3-1. The Terminal window



```
Last login: Mon Nov 24 02:00:09 on ttyp4
Welcome to Darwin!
dhcp-102:~ duncan$
```

The prompt that is being displayed in the Terminal is the output from a program known as the *shell*. The shell is the mediator between you and the internals of the Unix system. The shell's job is to interpret the commands you type and invoke the various programs on your system to satisfy those instructions. Each command you enter typically consists of a program name and some parameters to pass to that program. When the shell executes the program, its result are displayed in the Terminal window. Example 3-1 shows the use of the *date* command, which returns the current date and time.

Example 3-1. Using the date command

```
$date

Mon Dec 8 18:25:05 PST 2003
```

After the *date* command exits, the shell resumes control of the display and outputs another prompt enabling you to enter your next command.

3.1.1 Essential Filesystem Commands

In the Finder, it's easy to find your way around; after all, it was designed to be easy. The command line isn't quite such a walk in the park. It's just you and the prompt. So, how do you find your way around? Quite simply by using a set of commands at the prompt. Table 3-1 lists the commands that you will most often use to navigate your way around.

Table 3-1. Common shell navigation commands

Command	**Description**
pwd	Displays the current working directory
ls [options] *filename*	Lists the files in the given directory
cd*directory*	Changes directory
mkdir*dirname*	Makes a directory
rmdir*dirname*	Removes a directory
cp*from* to	Copies a file
mv*from* to	Moves a file
rm*file*	Removes a file
sudo [*command*]	Invokes the *command* as the superuser

3.1.2 Wildcards

Quite often when you use the command line, you will want to perform operations on groups of files at one time. For instance, you may want to copy all the files in a directory that start with a particular prefix. Or, you may just want to see all the files in a directory that end with the *.jpg* extension. Fortunately, the shell provides a painless way to accomplish this with the help of some wildcards.Table 3-2 lists the most common wildcards.

Table 3-2. Common wildcards

Wildcard	**Matches**
`?`	Any single character
`*`	Any string of characters
`[`*set*`]`	Any character in the *set*
`[!`*set*`]`	Any character not in the *set*

By far the most widely used wildcard is the asterisk (*). As an example, to find all the files ending with *.plist* in the *~/Library/Preferences* directory, use the following:

```
$ ls ~/Library/Preferences/*.plist
```

You can also use the asterisk by itself as an argument. This selects every file in a directory. For example, the following command can be dangerous indeed:

```
$ rm *
```

This command is dangerous as it will remove every file in a directory. Even more dangerous is combining wildcards with the rm command's *-r* option, which puts rm into recursive mode and removes every directory as well as every file in a directory. Execute this command at the root of the filesystem and you could lose everything. Wildcards are powerful, so use them with care.

3.1.3 Searching for Files

While Panther's new Finder has made great progress in making it easy to find files using the GUI, there will be times when it is easier or more convenient to use the command- line tools *find, grep*, and *locate*.

3.1.3.1 find

The aptly named *find* is one of the original Unix file-searching tools and allows you to search for files based on their filenames and other attributes such as when they were last modified. The basic syntax for using *find* is:

```
find pathname options
```

where the pathname argument indicates where in the file system hierarchy the search should occur, and the conditions argument indicates the attributes a file should have to match the search. There are more than 40 types of conditions you can pass, some of the most common of which are listed in Table 3-3, but the option you'll probably find yourself using most is *-name*.

Table 3-3. Commonly used find conditions

Condition	**Description**
`-group` *groupname*	Find files belonging to the specified group
`-mtime+`*n* `| -`*n* `|` *n*	Find files that were last modified more than *n (+n)*, less than *n (-n)*, or exactly *n* days ago
`-name`*pattern*	Find files with names matching the given pattern
`-newer`*file*	Find files that have been modified more recently than the given file
`-user`*username*	Find files that belong to the specfiied user

Example 3-2 shows how to find all the files that have a *.txt* extension in the *~/Documents* directory.

Example 3-2. Using the find command with the -name argument

```
$find ~/Documents -name *.txt

/Users/duncan/Documents/Financial/Planning.txt

/Users/duncan/Documents/ToDo.txt

/Users/duncan/Documents/Weather.txt
```

When run, the *find* command looks at every file in the *~/Documents* directory, including recursing through the subdirectories, and then lists each file with a name matching the pattern `*.txt`.

Another useful option is *-mtime*, which lets you search for files that were modified in a particular time frame. Example 3-3 shows how to find all the files that were modified in the last day in the *~/Documents* directory.

Example 3-3. Finding documents based on modification time

```
$find ~/Documents -mtime -1

/Users/duncan/Documents/ToDo.txt
```

3.1.3.2 grep

While*find* searches for files based on their filename, *grep* looks within a file, which allows you to search for something based on the contents of the file instead of just by its filename. You can either search for a string within a single file or a whole group of files. Its basic syntax is:

```
  grep pattern file
```

where pattern is what grep will look for and file is the file, or list of files, to look in. By itself, this command will only search the files you give it. Quite often, however, you'll want to search an entire directory. To do so you can use the *-r* option, which will put grep into recursive mode. Example 3-4 shows how to find a document in the Documents directory that has the word "thunder" in it.

Example 3-4. Finding documents based on content with grep

```
$grep -r thunder ~/Documents

/Users/duncan/Documents/Weather.txt:The thunder struck loudly and shook the house
```

In addition to telling you the filename that the term is found in, the command shown in Example 3-4 also gives you the line of text that the term is in, which is a way to make sure that it has found the document you really want.

3.1.3.3 locate

One of the fastest ways to find files—and quite a bit easier to use than *find*—is to use the *locate* command. Unfortunately, on most Macs this tool won't return any results because it depends on a database (located in the file */var/db/locate.database*) that only gets created if your machine is running at 4:00 a.m. on a Sunday morning. Chapter 8 will show you how to reschedule these tasks to a better time, but it's possible to create this database manually with the following command:

```
$ sudo /usr/libexec/locate.updatedb &
```

When you issue this command, you will be prompted for your password. Because you're using *sudo*, you must be an administrative user to execute the command. If you've noticed, there's an ampersand (&) at the end of that string. When you add the ampersand to the end of a command string, it tells the shell to run the command in the background, which means you get your command prompt back right away while the shell runs the command under the hood. Since it takes a while to create the locate database, it's always best to run this command in the background.

Once the database has been created, you can use the *locate* command with the following syntax:

```
locate pattern
```

Example 3-5 shows how to find a file named *icon.psd* that you can't remember the location of.

Example 3-5. Using the locate command

```
$ locate icon.psd
/Users/duncan/Work/Code/Backdrop/icon.psd
```

3.1.4 Terminal User Interface

In many ways, the command-line environment and the GUI are as diametrically opposed as possible. One stresses exactness while the other emphasizes ease-of-use. However, Apple has provided several features in the Terminal application to help it work better with the Aqua side of Mac OS X.

One of these features is the ability to drag and drop files from the Finder to the Terminal. When you do so, the full path name of the file is inserted on the command line. This means instead of trying to remember a long path to a file, you can quickly find it in the Finder, compose a command in the Terminal, and then drag and drop the file or folder icon into the Terminal window.

Also, while using the Terminal, there's a plethora of keystrokes you can use to whiz through tasks with ease. Table 3-4 lists the most commonly used ones.

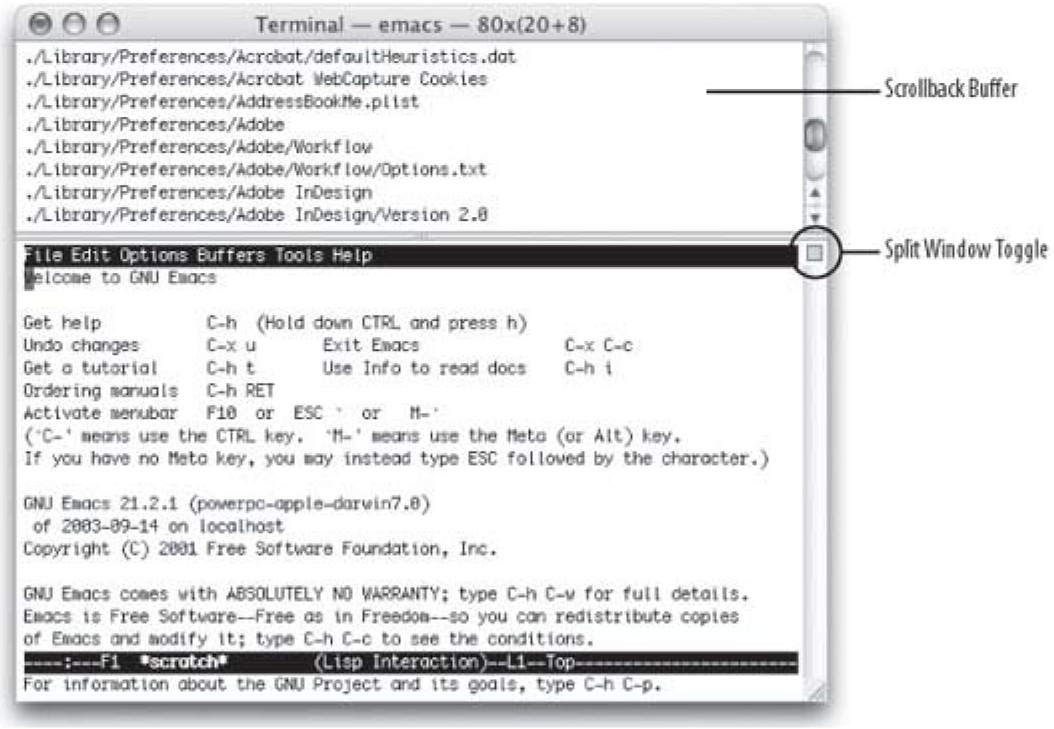
Table 3-4. Commonly used Terminal keyboard shortcuts.

Key	**Description**
⌘-N	Create a new Terminal window
Shift-⌘-N	Connect to server via SSH, SFTP, FTP, or Telnet
⌘-.	Send a break character (equivalent to Control-C)
Shift-⌘-V	Paste selected text (like a middle button mouse in X Windows)
⌘-Home	Jump to the top of the scrollback buffer
⌘-End	Jump to the bottom of the scrollback buffer
⌘-Page Up	Scroll one page up in the scrollback buffer
⌘-Page Down	Scroll one page down in the scrollback buffer
⌘-Up Arrow	Scroll one line up in the scrollback buffer
⌘-Down Arrow	Scroll one line down in the scrollback buffer
⌘-Left Arrow	Switch to the next Terminal window when you have multiple windows open
⌘-Right Arrow	Switch to the previous Terminal window when you have multiple windows open

One keystroke that should be mentioned in particular—once you start using it, you'll never stop—is Shift-⌘-V, which causes whatever text you have highlighted in the buffer to be instantly copied and then pasted at the prompt. If you've ever used an X Windows application, you'll recognize this as the equivalent of selecting text with the mouse and using the middle mouse button to copy and paste the text.

Another feature of the Terminal that is useful is the ability to split the window, as shown in Figure 3-2, and use the top part to scroll back through the buffer while working in the bottom part. You activate and deactivate this feature by using the button directly above the scrollbar.

Figure 3-2. The Terminal's split window



```
./Library/Preferences/Acrobat/defaultHeuristics.dat
./Library/Preferences/Acrobat WebCapture Cookies
./Library/Preferences/AddressBookMe.plist
./Library/Preferences/Adobe
./Library/Preferences/Adobe/Workflow
./Library/Preferences/Adobe/Workflow/Options.txt
./Library/Preferences/Adobe InDesign
./Library/Preferences/Adobe InDesign/Version 2.0
File Edit Options Buffers Tools Help
Welcome to GNU Emacs

Get help           C-h  (Hold down CTRL and press h)
Undo changes       C-x u       Exit Emacs               C-x C-c
Get a tutorial     C-h t       Use Info to read docs    C-h i
Ordering manuals   C-h RET
Activate menubar   F10  or  ESC `  or   M-`
(`C-' means use the CTRL key.  `M-' means use the Meta (or Alt) key.
If you have no Meta key, you may instead type ESC followed by the character.)

GNU Emacs 21.2.1 (powerpc-apple-darwin7.0)
 of 2003-09-14 on localhost
Copyright (C) 2001 Free Software Foundation, Inc.

GNU Emacs comes with ABSOLUTELY NO WARRANTY; type C-h C-w for full details.
Emacs is Free Software--Free as in Freedom--so you can redistribute copies
of Emacs and modify it; type C-h C-c to see the conditions.
----:---F1  *scratch*         (Lisp Interaction)--L1--Top----------------------
For information about the GNU Project and its goals, type C-h C-p.
```

You can change the colors used by the Terminal, including the background and foreground colors, as well as the transparency of the Terminal window. When you are staring at nothing but text, it's nice to be able to set it up with colors that are most comfortable for you to use. Just use the Terminal→Window Settings menu item and tweak away.

3.2 Mac-specific Shell Commands

Although Mac OS X is a Unix system, its filesystem contains some features over and beyond what t Unix*cp* and *mv* commands can handle. Apple has made it a priority to integrate the GUI and command-line worlds by including several tools.

3.2.1 open

The*open* command is the Terminal version of double-clicking a file or an application. Its syntax is:

```
open -a application filename
```

For example, to open the *~/Sites/index.html* file in the default browser on the system (typically Safari), use the following:

```
$ open ~/Sites/index.html
```

When you execute this command, the system will find the appropriate application to open *index.ht* with. However, if want to specify another application to open the file with, such as BBEdit (www.barebones.com) so that you can edit the HTML, you can use the -a option as shown:

```
$ open -a Safari ~/Sites/index.html
```

One thing to keep in mind about the *open* command is that the application is free to interpret the files passed to it however it wants to. For example, if you use the following command:

```
$ open -a Safari ~/Sites
```

Safari will launch, see that *~/Sites* is a directory, and then use its default behavior for local directories, which is to cause it to be opened up in the Finder.

If you want to open an application from the command line, you don't have to specify a filename. Ju execute the command like this:

```
$ open -a Safari
```

If you have Safari set to open a web site at startup, that page will load if your Mac is connected to the Internet after Safari launches.

3.2.2 ditto

The*cp* and *mv* commands know how to deal with plain files, but because they were designed for Unix, they don't know about some of the special attributes that Mac applications can associate with files. Most of the time, these attributes, known as *resources*, are stored in a folder with the application that created the file, icons, and other pieces of supplemental information. When you co a file with *cp*, all the special attributes associated with that file are lost.

The*ditto* command copies either files or, recursively, directories and will preserve the special file attributes that can be associated with files on the Mac. For example, to copy the contents of the *~/Documents* directory to another directory you would use the following command:

```
$ ditto ~/Documents ~/DocumentsCopy
```

3.2.3 CpMac and MvMac

In addition to ditto, Apple provides the *CpMac* and *MvMac* tools, which also understand how to preserve Mac-specific file attributes, as part of the Xcode Tools. Unfortunately, since these commands aren't installed in the locations the shell usually uses to find tools, you have to use the full path of the program. You use *CpMac* the same way you use *cp.* For example, to copy the file *foo.txt* to *bar.txt*, you would use the following:

```
$ /Developer/Tools/CpMac foo.txt bar.txt
```

To move the file instead, you would use the following:

```
$ /Developer/Tools/MvMac foo.txt ../foo.txt
```

3.2.4 osascript

The*osascript* command allows you to execute AppleScript from the command line just as easily as you can execute shell scripts. To execute the AppleScript contained in the file */Library/Scripts/URLs/Download Weather Map.scpt*, you would use the following command:

```
  $ osascript MyScript.applescript "/Library/Scripts/URLs/Download Weather Map.scp
```

When executed, the AppleScript fetches the current weather map for your country and opens it in Preview. You can also execute one liners of AppleScript with the -*e* option. For example, if you wanted to run a script in the background and have it announce when it was done, you could insert the following line into the script:

```
      osascript -e 'say "hello"'
```

> Mac OS X provides several commands to interact with Open Directory, the underlying database that contains system configuration information. These tools are *dscl, nicl, nidump, nifind, niload*, and *niutil* and are covered in Chapter 11,*Open Directory*.

3.3 Configuring and Using bash

As mentioned previously, with the Panther release of Mac OS X, *bash* is the default shell. Apple ma switch from *tcsh* to *bash* because of its support for Unicode text, something that's very important i international market. Another logical reason for switching to *bash* is that it is the default shell for m distributions and is easier to script with than *tcsh*. Because *bash* is now the default shell, the book on its use with an occasional nod to the other shells where appropriate.

The default configuration of *bash* is perfectly adequate for casual usage, but you'll inevitably want it to your own liking. This section takes a look at the various configuration files for *bash*, its enviro variables, how to set up command aliases, and how to use *bash*'s history to your advantage.

3.3.1 Environment Variables

Every program on the system runs in an *environment*. The environment consists of a set of name-v known as *environment variables*, which communicate a variety of configuration settings to a progra example, the shell uses the PATH environment variable to find a program to execute in response to command. To get an idea of what kinds of data are stored in environment variables, execute the *se* command, as shown in Example 3-6.

Example 3-6. Examining environment variables with set

```
$set
BASH=/bin/bash
BASH_VERSINFO=([0]="2" [1]="05b" [2]="0" [3]="1" [4]="release" [5]="powerpc-appled
BASH_VERSION='2.05b.0(1)-release'
COLUMNS=80
DIRSTACK=()
EDITOR=/Users/duncan/bin/usebbedit
EUID=501
GROUPS=()
HISTFILE=/Users/duncan/.bash_history
HISTFILESIZE=500
HISTSIZE=500
HOME=/Users/duncan
HOSTNAME=Incognita.local
```

```
HOSTTYPE=powerpc

...
```

When you execute the *set* command, you'll see quite a bit of output—probably around 40 lines. So environment variables may make sense when you first look at them, and some won't. Table 3-5 lis the more commonly-used environment variables you are likely to use on occasion.

To see the value of a single environment variable such as PATH, you can use the *echo* command as Example 3-7.

Example 3-7. Using the echo command to examine an environment varia

```
$echo $PATH

/bin:/sbin:/usr/bin:/usr/sbin
```

Table 3-5. Commonly used bash environment variables.

Variable	Description
BASH	Location of the *bash* shell program
BASH_VERSION	Version of *bash* that is being run
COLUMNS	Number of columns that are being used in the terminal view
DIRSTACK	List of directories being used by the *pushd* and *popd* commands
GROUPS	Various groups that the user is associated with
HISTFILE	File containing the shell history
HOME	Home directory for the user
HOSTNAME	Name of the system that the shell is running on
LINES	Number of lines currently being used by the shell
PATH	List of directories used by the shell to resolve commands
PS1	String used as the primary prompt
PS2	String used as the secondary prompt
SHELL	Shell program being used
SHELLOPTS	Options that are in effect for the shell

TERM	Type of terminal that the shell is displaying its content to
UID	User id of the currently logged-in user
USER	Username of the currently logged-in user
_ (underscore)	Previously executed command

The dollar sign in front of PATH means you are referring to an environment variable. Otherwise, if y have just entered *echo PATH*, the shell would return PATH, as shown in Example 3-8.

Example 3-8. Result of using the echo command without a dollar sign.

```
$echo PATH

PATH
```

To set or change an environment variable for the lifetime of the shell, use the *export* command. Fo to change the PATH variable so that you can run your own commands in *~/bin*, you could use the f command:

```
  $ export PATH=$PATH:~/bin
```

This sets the PATH variable to the currently existing PATH with the *~/bin* directory appended to it. I wanted to include the */Developer/Tools* directory on your path so that you have access to *CpMac* a you would instead use the following:

```
  $ export PATH=$PATH:/Developer/Tools:~/bin
```

The important thing to remember here is that you need to use a colon (:) as a delimiter between c paths. These commands will last for the lifetime of the current shell, which is as long as that Termi is open. In order to permanently add these paths to the shell, you'll have to edit one of *bash*'s conf files, described next.

3.3.2 Configuration Files

Three files in the Home directory, if they exist, are used to configure *bash*:

.bash_profile

> Contains environment variables and commands that are read and executed every time you c Terminal window and a shell is created for it, or when you SSH into your machine and are pr with a prompt. This allows you to customize the shell to your liking. If *bash* doesn't find this for*.bash_login* and *.profile* respectively to fill in for it.

.bashrc

> Contains environment variables and commands that are only read and executed when you cr subshell by typing bash in an already running shell.

.bash_logout

> Contains commands that are read and executed when you log out of a shell. You could use th up files before you log out.

By default, these files don't exist as part of a user's Home directory until you create them. The mos these three files is *.bash_profile*, which is used to customize the shell. For example, if you wanted t permanently modify the PATH that the shell uses to resolve commands, you could create a *.bash_p* your Home directory and add the following line:

```
PATH=$PATH:~/bin:/Developer/Tools
```

This causes the PATH environment variable to be set to the given string each time you open a new window. Because *.bash_profile* is only read when the shell is created, any changes you make to it v effect until you start the next shell. If you don't want to close your shell and start a new one, you c *source* command to load the contents of the *.bash_profile* file:

```
$ source ~/.bash_profile
```

3.3.3 Aliases

In addition to searching the PATH for commands, the *bash* shell also lets you define a set of aliases commonly used to create a shorter command name for long command strings so that they're a bit manageable or to rename commonly used commands.

Shell History

The first Unix shell, known simply as *sh*, was written by Steven Bourne in the 1970s and spawned the development of dozens of shells over the last 30 years. The first of the alternative shells to achieve prominence was the C shell (*csh*), written by Bill Joy at Berkeley as part of the Berkeley Standard Distribution. Then came the Korn Shell (*ksh*), written by David Korn at AT&T, Bell Laboratories, that combines the best features of the Bourne and C shells. Later, as part of the GNU project by the Free Software Foundation, *bash*—a freely available shell licensed according to the GNU Public License—was created by Brian Fox. As part of the GNU project, *bash* became the default shell on Linux and became possibly the most prevalent of the shells. Just as *bash* reinterpreted *sh, tcsh* was written as a derivative of the C shell . And most recently, there's the Z Shell (*zsh*), written by Paul Falstad while at Princeton, which combines many of the interesting features of *bash, ksh*, and *tcsh*.

Even though there are so many shells to choose from, several of the core Unix utilities still depend on the functionality of the original *sh*. So much so that despite the fact that the original *sh* shell doesn't ship with Mac OS X, there is a version of bash installed as */bin/sh* so these programs can execute. When run as */bin/sh, bash* tries to mimic the behavior of the original shell as much as possible to maintain compatibility.

Mac OS X ships with *sh, bash, tcsh*, and *zsh*. The Korn shell has also been ported to Mac OS X and can be obtained from the official website at www.kornshell.com.

To define an *alias* for a command, use the following syntax:

```
alias name=command
```

Where*name* is the name of the *command* alias you are defining, and command is the command tha executed by the shell when you invoke the alias. One common use of aliases is to accommodate fa of commands. For example, if you are always typing *sl* instead of *ls*, you could define the following that you don't get scolded by the shell again:

```
$ alias sl=ls
```

Another use for aliases is to create a simple command for a longer one. For example, if you are alw to change directories to somewhere deep in the hierarchy, you can set up an alias that will allow yo there quickly:

```
$ alias fdocs='cd ~/Documents/Corporate/Master/Forecasts'
```

Notice the use of quotes around the command. This is required when a command consists of more word.

To get a list of all the aliases currently defined, use the *alias* command by itself as shown in Examp

Example 3-9. Examining the currently defined aliases

```
    $ alias

alias sl=ls

alias fdocs='cd ~/Documents/Corporate/Master/Forecasts'
```

You can even make aliases to GUI applications. For example, if you wanted to create a quick shortc the Safari browser while on the command line, you could define the following alias:

```
  $ alias safari='open -a Safari'
```

With this alias in place, to launch Safari you simply need to type safari into the command line. And to make an alias permanent you'll need to create a *.bash_profile* file and place the alias into it or e existing*.bash_profile* file and then source it so the change takes effect.

3.3.4 History

As*bash* runs, it keeps a history of the commands that you've executed. This feature is quite handy you look at and reuse commands that you've previously entered. Where the shell's history is partic is when you need to invoke a command that has a lengthy set of parameters that you can't remem

The simplest way to use the history is to use the Up and Down arrows on your keyboard. This will s and forth through the commands that you've executed and display each in turn at the prompt. To g out of the history, you can use the *history* command, which displays a list of previously executed co Example 3-10 shows some output from history.

Example 3-10. Using the history command

```
$history
```

```
    1 cd Documents

    2 ls

    3 open -a Safari

    4 history
```

The list of commands is in the order in which they were executed. To reuse a particular command, (exclamation point—also called "bang" by Unix geeks), followed by the number of the command yc reuse.Example 3-11 shows how to use this command.

Example 3-11. Using a command from the history

```
$ !2

ls

Adobe SVG 3.0 Installer Log    Music

Desktop                        Pictures

Documents                Public

Library                 Sites

Movies                        Work
```

Note that the shell tells you which command is running as it runs the command. This is useful beca tell what arguments are being used. Another way to navigate the history list is to use the first few the command instead of the command number. Example 3-12 shows how to quickly execute the la that started with an 'o' character.

Example 3-12. Executing a command from the history based on characte

```
$!o

open -a Safari
```

Whenever you exit *bash*, it writes its history to the *~/.bash_history* file. Likewise whenever you sta populates its history with the contents of the *~/.bash_history* file. This allows you to quit and resta and still have your history available to you. By default, the history file is set to retain up to the last commands. To change this value, set the HISTFILESIZE environment variable to the number of lin want to keep. For example, to change it to remember 966 commands instead of 500, you would us following:

```
$ export HISTFILESIZE=966
```

3.4 Using Other Shells

Essentially, all the shells on the system (*sh, bash, tcsh*, and *zsh*) do the same thing: they take input from you and translate it into commands that are run on the system. However, each shell differs in the specifics of how you interact with it and the special features that it offers. If you've never used the Unix shell, you'll do just fine sticking with the default *bash* shell. But if you've become accustomed to the *tcsh* shell that was the default on previous versions of Mac OS X, or if you are an old-hand Unix user who simply prefers another shell, you may want to change the default shell to suit your preferences.

Because a shell is a program like any other, all you have to do to use a different shell temporarily is to type its name at the command line. For example, if you want to use the *tcsh* shell, execute the following:

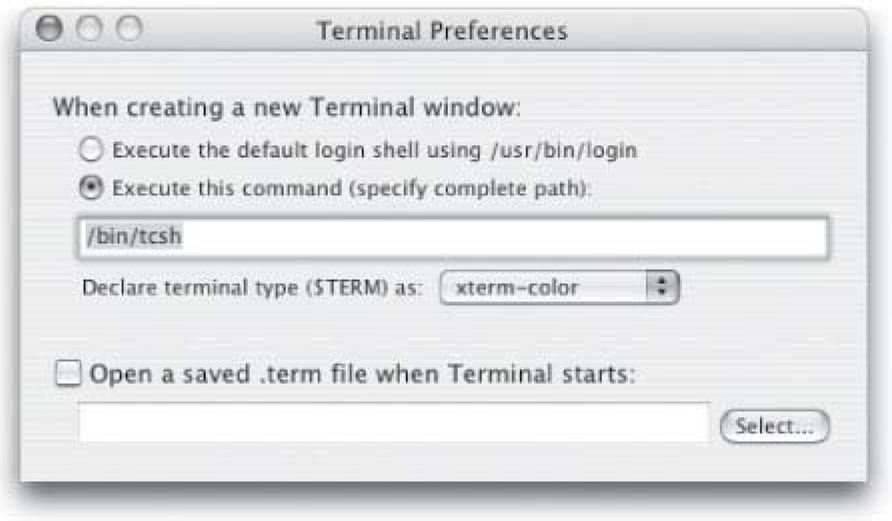
```
$ tcsh
```

The prompt will change to a percent sign (%), which is the default prompt for the *tcsh* shell, and will be in effect until you exit out of it.

3.4.1 Changing the Shell

To change the default shell used by the Terminal when it launches, simply open the Terminal preferences (Terminal→Preferences), then specify the shell you want to execute in the "Execute this command" text field, as shown in Figure 3-3. Now, whenever you open a Terminal window, you'll get the shell that you want.

Figure 3-3. Changing the shell using the Terminal Preferences pane



When creating a new Terminal window:

○ Execute the default login shell using /usr/bin/login

● Execute this command (specify complete path):

/bin/tcsh

Declare terminal type ($TERM) as: xterm-color

□ Open a saved .term file when Terminal starts:

Note that this setting doesn't change your default shell when you remotely log in. To change your default shell at the system level rather than only in the Terminal application, you'll need to modify your user account using NetInfo as shown in Chapter 5.

3.5 Shell Scripts

One of the true powers of the shell is that you don't have to always type in your commands by hand tediously one after the other. When you find a set of tasks that you do all of the time, you can consolidate them into a *shell script*. A shell script is nothing more than a collection of shell commands that is saved in a file on the filesystem. The shell script is made into an executable by changing its file permissions using *chmod +x*. A simple shell script is shown in Example 3-13.

Example 3-13. A simple shell script

```
#!/bin/sh

echo Hello World!
```

Colorizing File Listings

One nifty feature of the *ls* command that you can discover by reading through its manpage its ability to color code its output. For example, directories will be printed in blue and executable files will be printed in red. If you'd like to enable this, simply set the CLICOLOR environment variable to YES:

```
  $ export CLICOLOR=YES
```

You can customize the colors used by setting the LSCOLORS environment variable. See the *ls* manpage for more information.

What Can I Execute?

To quickly list all the command line utilities available, in other words those utilities that are in directories listed by your PATH environment variable, hit Shift-Esc twice in a row. A message will appear:

```
Display all 1024 possibilities? (y or n)
```

Answer yes and you can page through all of the commands.

The first line of a shell script is special and indicates to the system that it is to be executed with the*/bin/sh* program. The first line always starts with the #! characters (known to Unix hands as the*shebang* line) followed by the name of the tool to execute the script with. For example, if you wanted to write a shell script using *tcsh*, the first line would read #!/bin/tsch. For most purposes it's best to write shell scripts using either the *sh* or *bash* shells. Writing to the *sh* shell will result in scripts that are portable to the greatest number of systems. Writing to *bash* is almost as good due to the popularity of the *bash* shell.

When you create shell scripts, you shouldn't scatter them all over the filesystem. Just as all of your applications are gathered together in the Applications directory, you should place all your shell scripts together in one place. The common Unix convention for user-created shell scripts is to place them into a *~/bin* directory.

Writing shell scripts that are more than just a collection of simple statements is incredibly useful. It's also something that you can write a book on—and many people have. See "Further Explorations" at the end of the chapter for book recommendations that will help you explore the world of shell programming.

3.6 Getting Help

The*bash* shell provides a *help* command that will give information about all its built-in commands.Example 3-14 shows how to get information about the *alias* command.

Example 3-14. Getting help about alias

```
$help alias

alias: alias [-p] [name[=value] ... ]

  'alias' with no arguments or with the -p option prints the list

  of aliases in the form alias NAME=VALUE on standard output.

  Otherwise, an alias is defined for each NAME whose VALUE is given.

  A trailing space in VALUE causes the next word to be checked for

  alias substitution when the alias is expanded. Alias returns

  true unless a NAME is given for which no alias has been defined.
```

If you want to get a list of commands that *help* can help you with, use the command by itself.

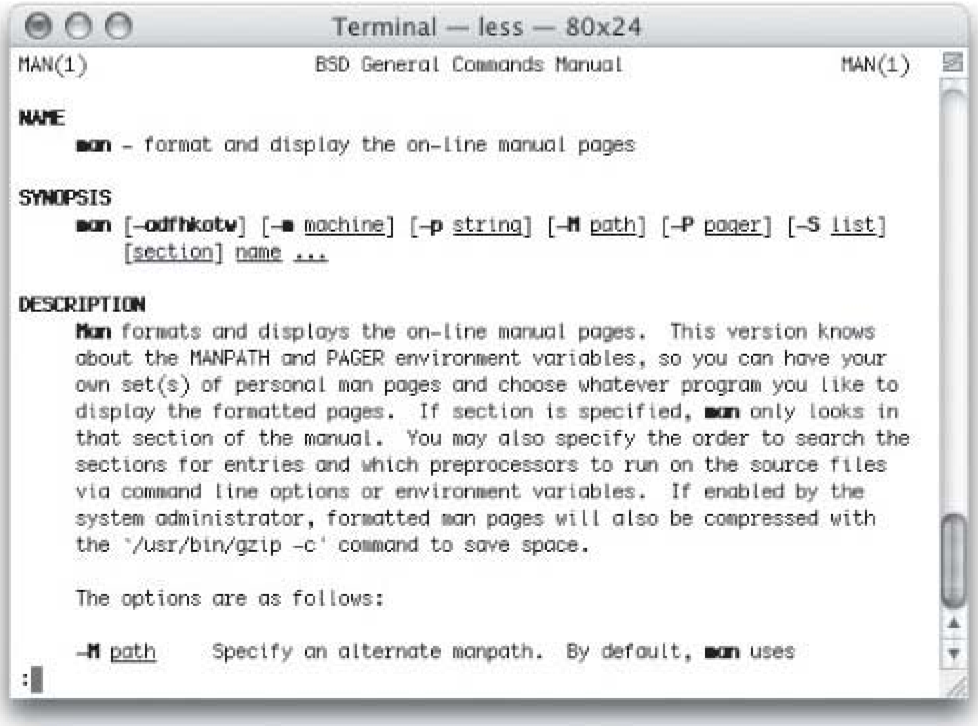
To get help about any other command in the Unix shell, use *man*, the all-purpose, Swiss Army Knife for getting information for most commands on the system in a form known as a *manpage*. For example, to see the manpage for *man*, type the following:

```
  $ man man
```

When you execute this command you'll see the output shown in Figure 3-4. Notice that there's a colon (:) at the bottom of the window with the cursor next to it. This indicates that there's more content to be seen in the manpage. At the prompt you can:

- Hit the Spacebar or the F key to page down
- Hit the B key to go back a page
- Type a / (slash) followed by a word to search forward for that word
- Hit the Q key to quit

Figure 3-4. The man command



```
MAN(1)                    BSD General Commands Manual                   MAN(1)

NAME
     man - format and display the on-line manual pages

SYNOPSIS
     man [-adfhkotw] [-m machine] [-p string] [-M path] [-P pager] [-S list]
         [section] name ...

DESCRIPTION
     Man formats and displays the on-line manual pages.  This version knows
     about the MANPATH and PAGER environment variables, so you can have your
     own set(s) of personal man pages and choose whatever program you like to
     display the formatted pages.  If section is specified, man only looks in
     that section of the manual.  You may also specify the order to search the
     sections for entries and which preprocessors to run on the source files
     via command line options or environment variables.  If enabled by the
     system administrator, formatted man pages will also be compressed with
     the `/usr/bin/gzip -c' command to save space.

     The options are as follows:

     -M path     Specify an alternate manpath.  By default, man uses
:
```

When you quit out of viewing a manpage, the information goes away and you are left with the same contents in your window as you had before you executed the *man* command. This is because*man* actually uses the program configured in your environment to be the pager—a program that can take output and step through it a page at a time. By default the pager is the *less* tool, which is the application that allows you to use the keystrokes listed above. If you want, you can use the *more* pager by setting your PAGER environment variable as follows:

```
$ export PAGER=more
```

The*more* pager isn't as full-featured as *less*, but it does leave the contents of the manpage onscreen when you exit. All things being equal though, I suggest you open a second Terminal window rather than change your pager to browse the manpage documentation.

The real trick to using *man* is knowing which command might be the one that you want information on. To help you find the right manpage, use the *apropos* command. For example, if you were wanting to find the manpages containing information about power management, you could use the following command:

```
$ apropos power
```

This returns six manpages, with one for *pmset* that has a summary indicating that it's for modifying power management settings.

3.7 Editing Text Files

While editing text files in the GUI is straightforward using the built-in TextEdit (*/Applications*), editing files using the command-line tools hasn't always been a walk in the park unless you are comfortable with old-school text editors like *vi* or *Emacs*. But thanks to the *open* command, several of the command-line editing tasks that you need to perform can be done in the GUI. For example, to open up your *bash* history file in TextEdit, use the following command:

```
$ open ~/.bash_history
```

For most text file editing, however, TextEdit is a little underpowered. Also, it can't be used as the command-line editor program (via the `EDITOR` environment variable), and it can't be used to edit files while you are logged in remotely.

3.7.1 pico

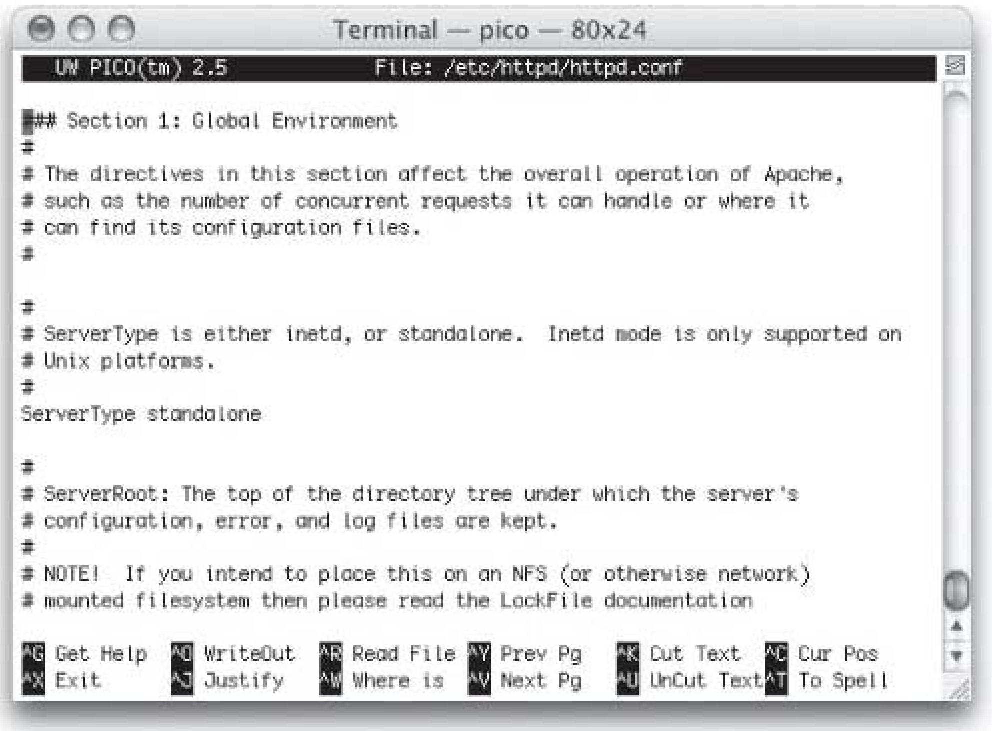
If you do need to edit text files on the command line—say you are logged in remotely or need to edit a file owned by root—the easiest text editor to use on the system is *pico*. To use pico to edit a file, use the following command:

```
$ picofilename
```

When you use *pico*, you'll see the user interface shown in Figure 3-5. In place of a toolbar, there is a series of commands at the bottom of the screen. The caret (^) character prefixing all those lines stands for the Control key on your keyboard. Simply press the Control key plus the letter of the command to perform some action. For example, to save a file, which pico refers to as "WriteOut," hit Control-O. To open, or "Read" a file, hit Control-R. And to exit, hit Control-X.

Aside from the Control key combinations, *pico* is fairly straightforward to use. Arrow keys move the cursor up and down, and whatever you type is inserted wherever the cursor is located at.

Figure 3-5. The pico text editor



```
  UW PICO(tm) 2.5              File: /etc/httpd/httpd.conf

### Section 1: Global Environment
#
# The directives in this section affect the overall operation of Apache,
# such as the number of concurrent requests it can handle or where it
# can find its configuration files.
#

#
# ServerType is either inetd, or standalone.  Inetd mode is only supported on
# Unix platforms.
#
ServerType standalone

#
# ServerRoot: The top of the directory tree under which the server's
# configuration, error, and log files are kept.
#
# NOTE!  If you intend to place this on an NFS (or otherwise network)
# mounted filesystem then please read the LockFile documentation

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Pg   ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where is  ^V Next Pg   ^U UnCut Text^T To Spell
```

By default, *pico* is not set as the default command-line `editor` on the system. This means when a command-line tool, such as *cvs* or *crontab* (which I'll talk about in Chapter 8), wants you to edit a file, it will pop up the system default editor of *vi*. To set *pico* as your default editor, simply set the `EDITOR` environment variable:

```
$ export EDITOR=pico
```

And, as always, to make this change permanent you can put it into your *.bash_profile*.

3.7.2 vi and Emacs

Mac OS X also ships with the two venerable Unix powerhouses of text editing: *vi* and Emacs. Each of these tools has a long history and feature sets that are even longer. These tools are powerful indeed and up to the task of any amount of text editing that you want to do, including hardcore development, but in many ways they are more complicated than what you need for everyday tasks. However, if you are already a user of either of these two editors, they are available for use on the system.

> In Mac OS X Panther, the default *vi* is now *vim*, a *vi* replacement that duplicates all the functionality of the original *vi* and extends it in many useful ways, adding unlimited undo and syntax highlighting.

This isn't the place for a full tutorial on how to use *vi* or Emacs, but you should be able to exit out of either of these shells if you find yourself in one. To exit out of *vi*, type the following:

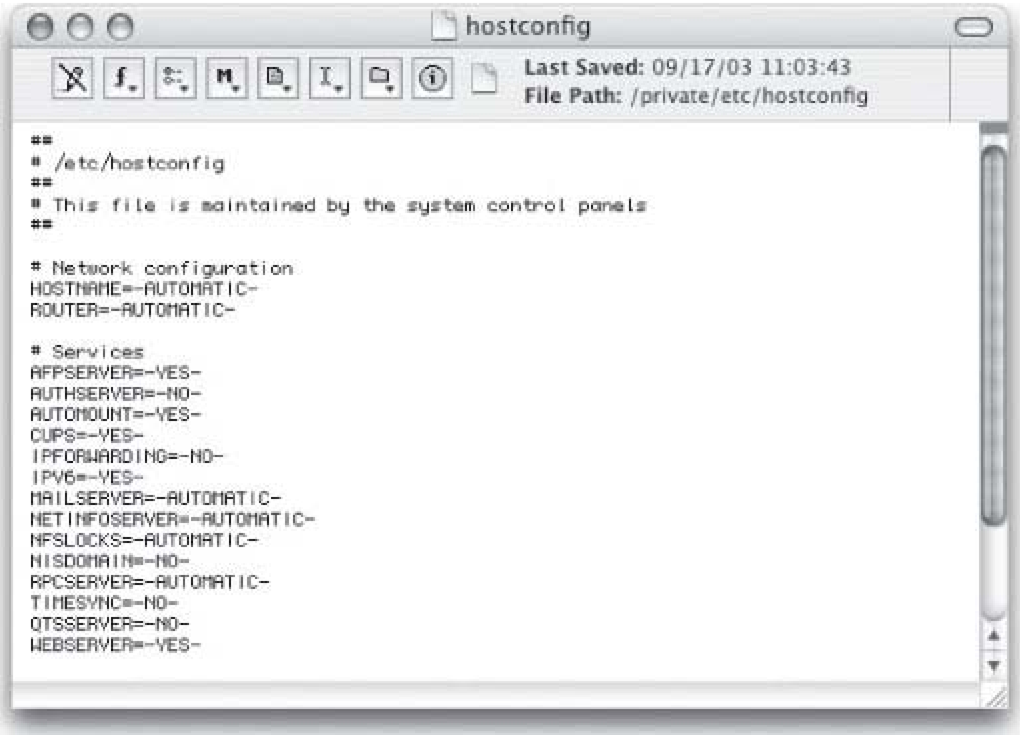
```
:q!
```

To exit out of Emacs, type Control-X, then Control-C.

3.7.3 BBEdit

If you are going to be doing a lot of text editing and want seamless integration between the command line and a GUI-based editor, I suggest that you take a look at either BBEdit, shown in Figure 3-6, or its little brother, TextWrangler, which shares many of BBedit's features. Both of these are from Bare Bones Software (www.barebones.com). Not only is BBEdit a great text editor, but it integrates well with the Unix layer of Mac OS X and the command line.

Figure 3-6. The /etc/hostconfig file loaded in BBEdit



Last Saved: 09/17/03 11:03:43
File Path: /private/etc/hostconfig

```
##
# /etc/hostconfig
##
# This file is maintained by the system control panels
##

# Network configuration
HOSTNAME=-AUTOMATIC-
ROUTER=-AUTOMATIC-

# Services
AFPSERVER=-YES-
AUTHSERVER=-NO-
AUTOMOUNT=-YES-
CUPS=-YES-
IPFORWARDING=-NO-
IPV6=-YES-
MAILSERVER=-AUTOMATIC-
NETINFOSERVER=-AUTOMATIC-
NFSLOCKS=-AUTOMATIC-
NISDOMAIN=-NO-
RPCSERVER=-AUTOMATIC-
TIMESYNC=-NO-
QTSSERVER=-NO-
WEBSERVER=-YES-
```

The only downside to BBEdit is that it doesn't come with the system. Instead, BBEdit will set you back about $150 for a single license. TextWrangler, which also integrates well with the command line (even though it sports fewer features than BBEdit), costs about $50. You can download demos of either editor through Bare Bones Software's web site.

3.7.3.1 Command-line integration

When you run BBEdit for the first time, it prompts you to install its command-line tool, *bbedit*. After you do so, you'll be able to open files with the following command:

```
bbedit filename
```

> If you are using TextWrangler, the command-line tool is named *edit* instead of*bbedit*.

Unlike the command-line editors that allow you to specify a file that doesn't already exist, you have to tell *bbedit* to explicitly create a new file when needed. To do so, use the *-c* argument as shown:

```
$ bbedit -c newfile.txt
```

BBEdit can even serve as your editor, totally replacing *pico, vi*, or *Emacs* in this role. Example 3-15 shows a script that can be used as your editor. Simply create the script and save it as *~/bin/editwithbbedit*.

Example 3-15. A script to use BBEdit as an EDITOR on the command line

```
#!/bin/sh

/usr/bin/bbedit --wait --resume "$@"
```

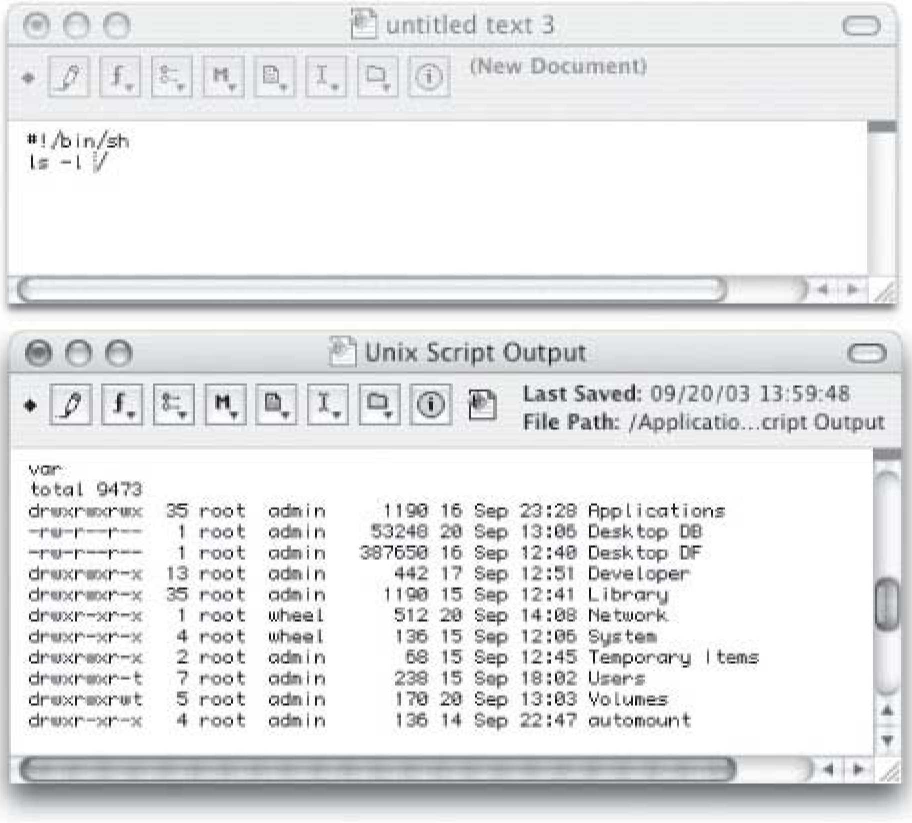
This command launches BBEdit in such a way that it keeps the process that requested the editor waiting until you close the window, then it pops the Terminal window from which the editor was called back to the foreground. After you save the script, make it executable with *chmod +x editwithbbedit* and then set your `EDITOR` environment variable to point at this shell script. To make the change permanent, add the following line to your *.bash_profile*:

```
export EDITOR=~/bin/editwithbbedit
```

3.7.3.2 Writing and executing shell scripts

BBEdit allows you to write and execute shell scripts without ever going to the command line. Simply write your script in a BBEdit window, then use the Run→Run menu. BBEdit runs the script and then put its output into a new window as shown in Figure 3-7.

Figure 3-7. Running a script in BBEdit



```
#!/bin/sh
ls -l /
```

Last Saved: 09/20/03 13:59:48
File Path: /Applicatio...cript Output

```
var
total 9473
drwxrwxrwx  35 root  admin     1190 16 Sep 23:28 Applications
-rw-r--r--   1 root  admin    53248 20 Sep 13:06 Desktop DB
-rw-r--r--   1 root  admin   387650 16 Sep 12:40 Desktop DF
drwxrwxr-x  13 root  admin      442 17 Sep 12:51 Developer
drwxrwxr-x  35 root  admin     1190 15 Sep 12:41 Library
drwxr-xr-x   1 root  wheel      512 20 Sep 14:08 Network
drwxr-xr-x   4 root  wheel      136 15 Sep 12:06 System
drwxrwxr-x   2 root  admin       68 15 Sep 12:45 Temporary Items
drwxrwxr-t   7 root  admin      238 15 Sep 18:02 Users
drwxrwxrwt   5 root  admin      170 20 Sep 13:03 Volumes
drwxr-xr-x   4 root  admin      136 14 Sep 22:47 automount
```

3.8 Further Explorations

This chapter's aim was to give you enough knowledge of how the shell works to successfully execute the commands that I'll be covering through the rest of the book. However, this chapter has just scratched the surface of what is possible with the shell.

The following books are recommended to learn more about the subjects in this chapter:

- *Learning the bash Shell, 2nd Edition*, by Cameron Newham, et al. (O'Reilly & Associates, Inc., 1998)
- *Learning the vi Editor, 6th Edition*, by Linda Lamb, et al. (O'Reilly & Associates, 1998)
- *Learning GNU Emacs, 2nd Edition*, by Debra Cameron, et al. (O'Reilly & Associates, 1996)

Part PART II: Essentials

This part of the book gives you the tools you need to examine how your system is running and how to adjust most of the knobs behind its operation. I'll cover topics including process control, managing software installations, and formulating a backup strategy for your data.

Chapters in this part of the book include:

Chapter 4. System Startup and Login

Usually when you sit down at your computer, the first thing you do is either hit the power-on button on the system, or—for those who leave their machines on all the time—log in. Most of the time you probably don't think much about what happens behind the Apple logo and spinner as the system boots up or about the events that happen between the time you enter a username and password and when the desktop appears. In this chapter, however, I'm going to remove the veil of mystery and show how it all happens.

Booting a Mac is not a single-step process; over 100 programs take part in the process of transforming your machine from an inert collection of plastic and metal into a running system. In general, though, the process can be broken down into two major steps. The first is the hardware powering up and organizing itself. The second is the hardware launching the operating system and the operating system starting itself up. First, let's take a look at what happens at the hardware layer.

4.1 The Hardware Boot Process

When you press the power-on button on your Mac, a small hardware program embedded in the ma your machine, known as the POST (Power-On Self Test), is activated. The first thing the POST does then initialize the CPU. Next the POST performs some tests on the core system components, such a make sure the system can boot. If everything checks out OK, the POST starts a program called Ope on a reprogrammable chip on the motherboard. It's at this point in the boot process that you hear chime.

Open Firmware's first job is to find the various hardware devices attached to the motherboard of th includes internal hardware such as video cards and hard drives as well as external hardware conne FireWire, USB, and SCSI buses. As Open Firmware discovers the various hardware devices, it orga device tree, which will make it easier for the higher levels of the system to access those devices.

So What's Open Firmware?

You're no doubt familiar with the terms *software* and *hardware*. But *firmware*? Well, when chip designers make a mistake in a hardware program, they pretty much have to throw away the chip start over unlike with software where all you need to do is replace the instructions without throwi away a physical device. The solution that hardware designers came up with is to create a chip tha be reprogrammed. The instructions on the chip are permanently stored on it, so it's not quite soft But they can be changed. Being neither hardware nor software, they settled on the name firmwar

Open Firmware is the name given to the IEEE standard for a machine-independent BIOS that gre of the boot code used in workstations and servers from Sun Microsystems. What separates it from BIOS implementations is that it uses a bytecode representation of the Forth language. This allows same code to run on different processors without change, which lets hardware makers provide on drivers that will work on PowerPC, Sparc, and Intel *x*86 architectures. In addition, Open Firmware provides a mechanism for each piece of hardware to describe itself so that the host system can be it.

You can find out more information about Open Firmware at www.openfirmware.org/.

After Open Firmware finds the hardware, it looks for an operating system to boot. First it looks at i configuration for the boot device. Assuming the configured device is a disk and is attached, it will t disk for the blessed operating system booter and load and start it. If a bootable operating system i configured device, Open Firmware will conduct a search of any other drives attached to your syster boot device can be found, you'll be presented with a folder icon with a blinking question mark indic system could be found.

4.1.1 Taking a Peek at Open Firmware

Even though you will rarely (if ever) need to deal directly with Open Firmware, you can take a quic configuration using the *nvram* command-line tool. Example 4-1 shows the use of this command an (slightly abbreviated).

Example 4-1. Looking at Open Firmware's configuration

```
$nvram -p
oem-banner?       false
boot-script
virt-size         -1
output-device     screen
output-device-1   scca
input-device      keyboard
input-device-1    scca
mouse-device      mouse
selftest-#megs    0
boot-volume       3
boot-device       mac-io/ata-4@1f000/@0:10,\\:tbxi
boot-screen
boot-args
default-server-ip
default-gateway-ip
default-router-ip
default-client-ip
default-subnet-mask
default-mac-address?    false
screen-#columns   100
screen-#rows      40
scroll-lock       true
skip-netboot?     false
diag-switch?      false
boot-command      mac-boot
```

```
ram-size          0x30000000
```

Notice the boot-device line in the output. This tells Open Firmware what device it should boot from Firmware is set to boot from the ATA-based disk drive. This string will be different depending on w (ATA or SCSI) your hard drive is attached to and what partition on that drive you are using for you

The*nvram* command can also be used to set these values if you have an administrator account on t a particular value in Open Firmware, you would use the following syntax:

```
sudo nvram name=value
```

And if you want to set a large number of values at once, you can load them from a text file. The file list of name=value statements. The syntax for loading the contents of the file into Open Firmware i

```
sudo nvram -f filename
```

4.1.1.1 Resetting Open Firmware

It should go without saying but mucking about with your Open Firmware settings can put your syst odd state. Luckily, if you go overboard and things do get mucked up, there's an easy way to reset t referred to as "Zapping the PRAM". To do this, hold down Option-⌘-P-R while booting your machin

4.1.2 Updating Firmware

Every so often an update to Mac OS X is released that requires a firmware upgrade for certain mac work correctly. Since Open Firmware is primarily used at boot time, not having an up-to-date firmw machine that won't boot—it'll get stuck at a gray screen instead. You should be sure to read the rel major update to Mac OS X to see if you need a firmware update before installing it.

Another place to look for documentation about firmware updates is on Apple's support web site at www.info.apple.com/. To quickly see the pages dealing with firmware updates required for Mac OS `Firmware Update` in the "Search Apple Support" box. These pages will tell you how to determine w need a firmware update for your Mac before you install a new operating system.

There are two ways to determine the firmware version you have. The first is to use the System Pro (seeChapter 6 for details on System Profiler). The second is to access Open Firmware directly.

4.1.3 Accessing Open Firmware

If the *nvram* command doesn't give you enough access to the firmware of the system, you can dire holding down the Option-⌘-O-F keys as you boot your machine. This will drop you directly into the command line. You should be greeted by something that looks like Example 4-2.

Example 4-2. The Open Firmware command-line interface

```
Apple PowerBook3,1 4.1.8f5 BootROM built on 03/21/01 at 11:49:53

Copyright 1994-2001 Apple Computer, Inc.

All Rights Reserved


Welcome to Open Firmware, the system time and date is: 22:51:00 09/21/2003


To continue booting, type 'mac-boot' and press return.

To shut down, type 'shut-down' and press return.


 ok

0 > _
```

The first line of output tells you the model of your computer followed by the version of your firmwa computer is an Apple PowerBook3,1 and the firmware version is 4.1.8f5.

After the banner is printed, you are at the Open Firmware prompt (>), a pretty limited place unles what you are doing. Table 4-1 lists a few of the commands that you can use. If you get yourself in Example 4-3 shows how to reset your firmware and restart the system.

Table 4-1. Open Firmware commands.

Key	Description
`printenv`	Prints all the variables held in firmware
`setenv`*`name=value`*	Sets a variable to the given value
`password`	Sets an Open Firmware password
`reset-nvram`	Clears the variables in firmware and replaces them with defa
`reset-all`	Causes the machine to reboot
`mac-boot`	Causes the machine to continue booting into the configured s
`shut-down`	Shuts down the machine

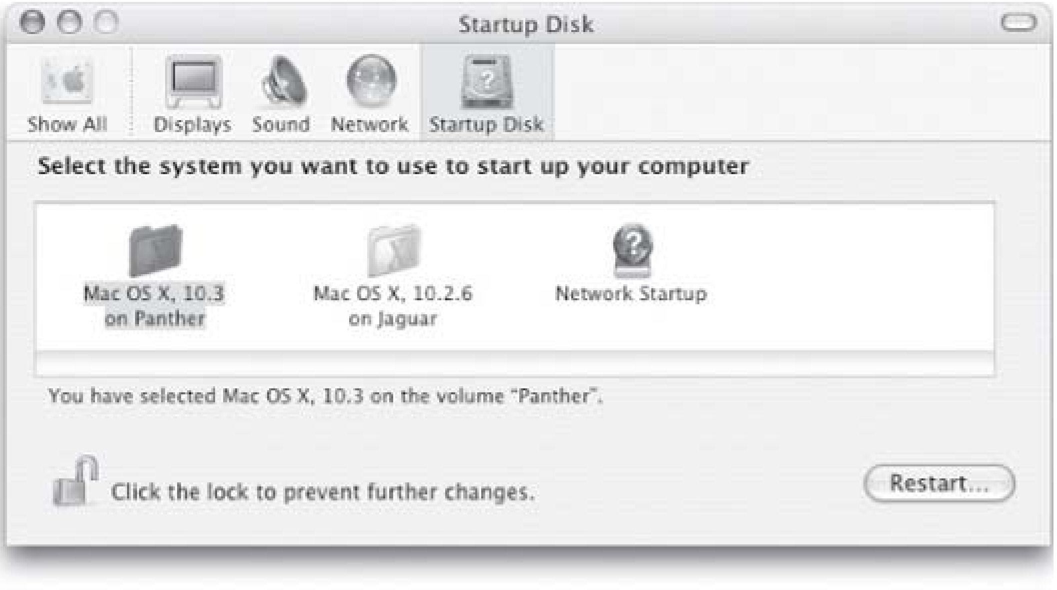
Example 4-3. Resetting Open Firmware and restarting the system

```
0 > reset-nvram

0 > reset-all
```

4.1.4 Setting the Boot Disk

To set the disk that you want to boot from, use the Startup Disk preference panel in System Prefer launch this tool, as shown in Figure 4-1, it will find all the system folders on the hard drives attach and offer them as options to choose from. As well, there's an item labeled "Network Startup". This boot from a NetBoot server on your network, if available.

Figure 4-1. The Startup Disk preference panel



Select the system you want to use to start up your computer

You have selected Mac OS X, 10.3 on the volume "Panther".

Click the lock to prevent further changes.

Macintosh Model Names

When you see your Mac's model name in Open Firmware or in System Profiler as something like " you might wonder how that maps to its more wellknown name of "PowerBook G3 (FireWire)". It r machine using the latter name while Open Firmware and the system internals know it as the shor name—and there's no direct mapping between the two names to be found on the Apple web site. however, on your hard drive at */System/Library/SystemProfiler/SPPlatformReporter.spreporter/Contents/Resources/English.lproj/* that gives the mappings between these names. It's a Unicode file, so you'll need to open it in Text *open -e filename* command, which forces a file to be opened in TextEdit) or some other editor tha Unicode text encoding.

When you select an operating system to boot from in the Startup Disk preference panel, two things Open Firmware is programmed with the location of the boot device. Second, a small bit of informa the boot blocks of the hard disk that contain the system indicating where on the drive the boot pro operating system is located. This second step, known as *blessing the system*, allows more than one on a hard drive or partition. This which means you can boot into either Mac OS X or Mac OS 9 ever installed on the same physical drive.

4.1.4.1 Blessed system disk

To look at the startup settings for your machine from the command line, use the *bless* command. F wanted to take a look at the various settings of the drive you are currently running from, you woul shown in Example 4-4.

4-4. Executing bless to examine boot settings

```
$sudo bless -info -bootBlocks

finderinfo[0]:   2483 => Blessed System Folder is /System/Library/CoreServices

finderinfo[1]:      0 => No Startup App folder (ignored anyway)

finderinfo[2]:      0 => Open-folder linked list empty

finderinfo[3]:      0 => No OS 9 + X blessed 9 folder

finderinfo[4]:      0 => Unused field unset

finderinfo[5]:   2483 => OS X blessed folder is /System/Library/CoreServices

64-bit VSDB volume id: 0x1BD7BA028DDA6A89
```

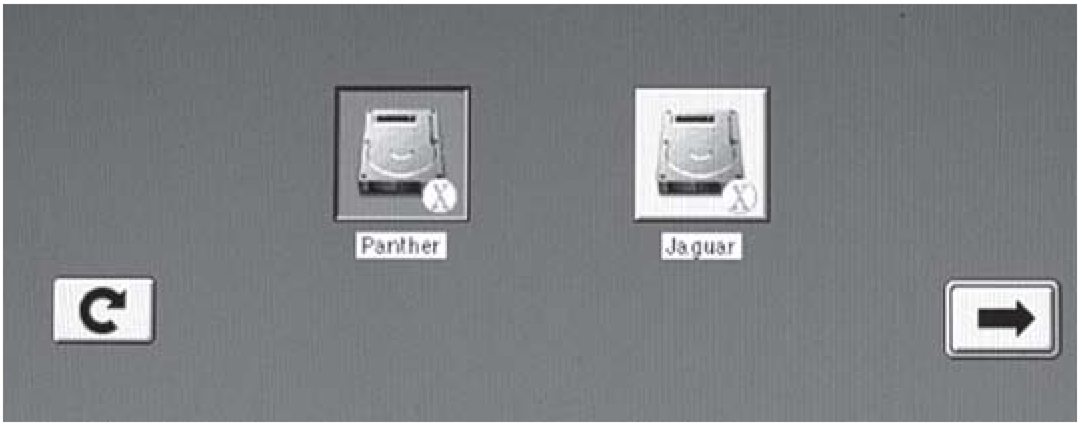
Try changing the settings around in your Startup Disk preference panel and executing this comma changes. You may have to switch out of the Startup Disk preference panel for it to save your chang bit more you can do with the *bless* command such as blessing a particular folder or partition on yo *bless* manpage for details.

4.1.5 Changing the Boot Disk at Boot Time

Sometimes you will want to boot from some other device than the default. Usually this happens wh boot from a CD or a FireWire drive to replace or repair your primary OS installation. You can use th to choose the boot disk you want to use:

- To boot from a CD, hold down the C key as you boot.
- To boot from a NetBoot server, hold down the N key as you boot.
- To see the local drives that contain a bootable partition, hold down the Option key as you boo Open Firmware program called the *OS Picker*, as shown in Figure 4-2.
- If you have one disk that contains Mac OS 9 and one disk that contains Mac OS X, you can bo holding down the X key while the system boots. If you have more then one Mac OS X system this trick won't work.

Figure 4-2. The OS Picker



OS Picker duplicates some, but not all, of the functionality of the Startup Disk preference panel. It can only see the blessed operating system on a partition. If you have both Mac OS X and Mac OS 9 same partition, OS Picker will show only the one that was configured for boot in the Startup Disk p you want to boot into the non-blessed operating system on a partition, you'll have to make the cha and not with the OS Picker.

4.1.6 Locking Down Open Firmware

Since Open Firmware controls the boot process and will accept input from the keyboard during sys time—either in the form of boot-key combinations or by accessing the Open Firmware prompt—it is to make your Mac boot from any other drive. For most personal users, this isn't a problem. In fact, flexibility in starting up your system with multiple versions of Mac OS. In corporate or academic lab you may not want people to be able to muck about with the settings of the computer, or the data i easily.

There are two ways to enable an Open Firmware password. The first is to download the Open Firmw utility from Apple through its support web site at [docs.info.apple.com/article.html?artnum=120095 is at the Open Firmware command line using the following process:

1. Boot into Open Firmware (Option-⌘-O-F).
2. At the Open Firmware prompt, type your `password`. You will be prompted to enter your passw
3. Set a security mode for Open Firmware by typing `setenv security-mode full` or `setenv s command`. The first option will totally lock down your machine and require a password to be er system. The second option will let the system boot as configured but will require a password t Open Firmware or use any boot key options.
4. Restart the machine using the `reset-all` command.

To turn off password protection, boot into Open Firmware and type `setenv securitymode none`. Y for your password, if one is set, to make this change.

If you have a password-protected machine that you've forgotten the password for, there is a way t password: open the machine and change the total amount of RAM in your system. Then reset the P Option-⌘-P-R at boot time.

It should be noted that being able to lock down the system with an Open Firmware password is not feature. If somebody can open up the machine and reset the memory in it, then the password prot nothing. Sure, you can padlock desktop machines, but there is no way to keep people out of the m slots on a laptop. This matches a truism in computer security—if somebody has access to a machin can gain access to the data on it. The only real way to protect the data on a machine is to use the F discussed in Chapter 10,*Disks and Filesystems*.

4.2 The Operating System Boot Process

When Open Firmware boots Mac OS X, it does so through *BootX* (located in */System/Library/CoreS* It's important to note that Open Firmware doesn't know that it's booting Mac OS X, nor does it hav concept of the filesystem that contains BootX. All it knows is that there is a program at a particular on a drive that it can load and run. It is the job of BootX to take care of the rest.

The first thing BootX does is draw the "booting" image on the screen. This is the monochrome gray screen with the spinner on it that you see after your computer chimes. It then loads the kernel fror and the essential kernel-level device drivers (such as those for accessing disk drives) necessary to things running. Once loaded, the kernel's initialization procedure is called.

The kernel loads device drivers for all the devices connected to the computer and then finds the bo It may seem a bit odd that the kernel has to find the drive that it was loaded from, but remember was loaded from the drive and then started. It has to find the drive it was loaded from by consultin Firmware. Once it finds the boot drive, it mounts it at the Unix filesystem root (/), starts its interna server, which allows processes to communicate with each other, and then starts the BSD */sbin/init*

4.2.1 BSD Initialization

The BSD *init* program is the first Unix process to start up in the system, and it owns every other pr the system. It carries out four jobs:

1. Determines what mode to boot into; either the normal mode, which is called "multiuser," or t "single-user" maintenance mode.
2. Runs the *rc* system initialization script, located in the /etc directory, which starts many other that are vital to the system.
3. Launches the *loginwindow* application so users can log in to the system.
4. Cleans up after all processes as they terminate so system resources are not wasted.

4.2.1.1 The rc script

The*rc* script launches a variety of programs that perform the following system initialization tasks:

- Mount the local filesystem.
- Start the kernel extension loader, allowing the kernel to load device drivers dynamically.
- Load the Mach bootstrap-based kernel services listed in the */etc/mach_init.d* directory. These provide some of the essential functions of Mac OS X such as the Apple Type Server (ATS), Op Directory, Kerberos, and the window server, allowing programs to draw to screen.
- Start the local NetInfo server.
- Start the *update* background process, which makes sure that the filesystem cache in memory regularly synchronized with the disk.

- Start the virtual memory system and ensure that the swap file is properly created.

The last thing that the rc scripts do is launch the *SystemStarter* program, which takes care of the f stages of system startup.

4.2.1.2 Single-user mode

Single-user mode is a special Unix system state in which system startup is stopped and a login pro offered, allowing for administrative and maintenance activities to be performed. Historically this m been used when a system becomes unstable and needs critical areas to be fixed. In the last few ve Mac OS X, the primary, albeit rare, use for this mode was to repair a disk drive with corrupted files that Panther has built-in filesystem journaling, which guarantees the integrity of files on the hard c need for a single-user mode is all but gone. Still, if you'd like to use it, hold down ⌘-S as you boot the system has booted, you'll be presented with a simple command-line prompt. To exit single-use type*exit* or *reboot*;*exit* will cause the system to go multiuser and complete the boot into the GUI environment.

4.2.1.3 Verbose mode

If you are really interested in all the things that execute during this phase of the boot process, you down ⌘-V as you boot. Instead of the gray-on-gray Apple logo and spinner, you'll see a plain-text of the various things going on as you boot your system. If you want to see this every time you boo system, execute the following command:

```
$ sudo nvram boot-args="-v"
```

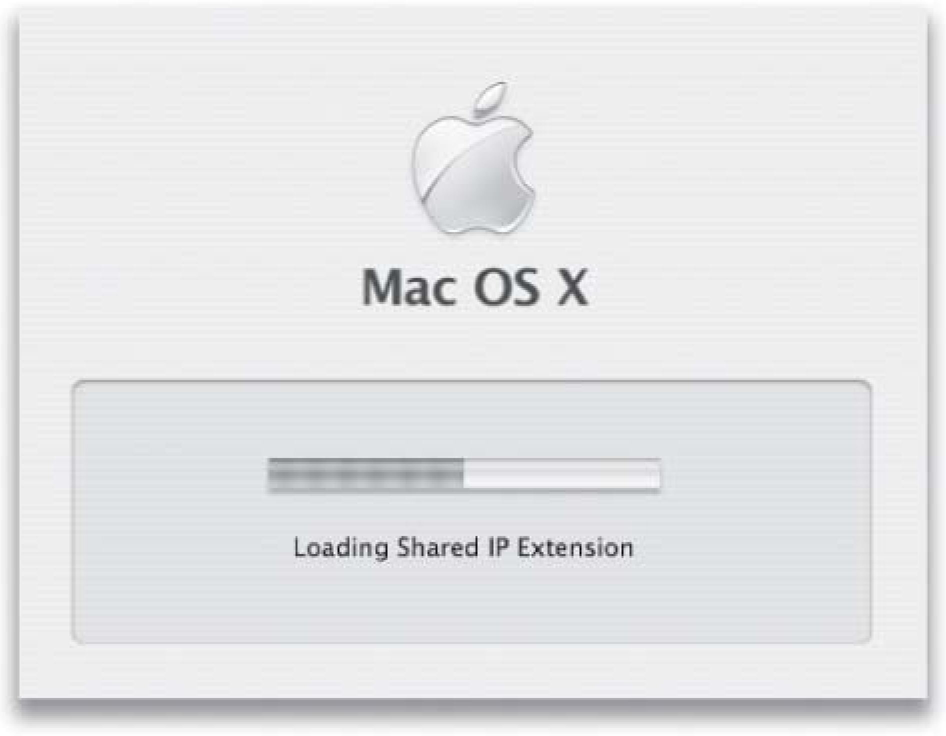
To turn off verbose booting, use the following:

```
$ sudo nvram boot-args=""
```

4.2.2 SystemStarter

The SystemStarter program takes care of the final stages of the boot process which turn the syster just another Unix variant into Mac OS X. It does so by starting up system daemons like Rendezvou Apache web server, and the AppleShare server. When SystemStarter runs, the boot screen change the gray-on-gray apple to a blue background with the Boot Panel image shown in Figure 4-3.

Figure 4-3. The Boot Panel



Mac OS X

Loading Shared IP Extension

SystemStarter determines the services it should start by scanning the */System/Library/StartupItem* the*/Library/StartupItems* directories for items to be started when the system starts up, and then la them. To help reduce boot time, SystemStarter attempts to run as many startup items as possible parallel. This means startup tasks that take a little while to complete, such as setting up the netwo stop all the others from starting. This parallel process lets the boot process take advantage of dual machines. Some of the services started are listed in Table 4-2.

Changing the Boot Panel

If you are feeling like spicing up your system a bit, you can change the image that is used for the Boot Panel shown in Figure 4-3. The image is a simple PDF file located at */System/Library/CoreServices/SystemStarter/QuartzDisplay.bundle/Resources/Boot- Panel.pdf*. You can replace it with an image of your own or one from ResExcellence's boot panel library at www.resexcellence.com/user_X_boot.shtml. Before replacing the image, consider renaming the original as *BootPanel.pdf.bak* so you can restore the original if you need to. If you happen to misname the *BootPanel.pdf* file, the system will still boot, and the progress bar will be drawn on a blank blue screen.

If you opt to use or create your own image for the boot panel, just make sure you size the image to be 472 x 360 pixels.

Table 4-2. Some core system startup items.

Startup Item	Description
SecurityServer	Manages the core system keychain, authentication, and authorization
Network	Activates and configures the network interfaces of the system sets the machine's hostname
IPServices	Starts xinetd, the Internet super-server that can serve as a fr end for other services
mDNSResponder	Starts the multicast DNS responder—a core component of Rendezvous
CrashReporter	Starts the crash reporter service which catches application cra and kernel panics to provide information about them
LDAP	Starts the LDAP service which OpenDirectory relies on
NetworkTime	Synchronizes the system clock to Network Time Protocol (NTP servers
SSH	Starts the *sshd* daemon to provide secure remote login servic
Apache	Starts the Apache web server
PrintingServices	Starts the CUPS printing service daemon, allowing local printe be shared
AppleShare	Starts the AppleShare service
Postfix	Starts the Postfix *smtp* server
Cron	Starts the *cron* daemon

As SystemStarter runs, it launches as many of the system startup items in parallel as possible whi to speed up the boot process. Once SystemStarter is finished running, it hands control over to the *loginwindow* application, which allows you to log in via the GUI interface.

4.2.2.1 The hostconfig file

Many startup items need some amount of input as to whether they should be launched or not. For if Personal Web Sharing is turned off on your machine, then the Apache startup item should not ac order to control these startup items without moving them out of their *StartupItems* folder, Mac OS provides the */etc/hostconfig*file—a simple file consisting of a set of name-value pairs indicating whi services should be run and which ones should not. This file is shown in Example 4-5.

Example 4-5. The /etc/hostconfig file

```
##
```

```
# /etc/hostconfig
##
# This file is maintained by the system control panels
##

# Network configuration
HOSTNAME=-AUTOMATIC-
ROUTER=-AUTOMATIC-

# Services
AFPSERVER=-YES-
AUTHSERVER=-NO-
AUTOMOUNT=-YES-
CUPS=-YES-
IPFORWARDING=-NO-
IPV6=-YES-
MAILSERVER=-NO-
NETINFOSERVER=-AUTOMATIC-
NFSLOCKS=-AUTOMATIC-
NISDOMAIN=-NO-
RPCSERVER=-AUTOMATIC-
TIMESYNC=-YES-
QTSSERVER=-NO-
WEBSERVER=-NO-
SMBSERVER=-NO-
DNSSERVER=-NO-
COREDUMPS=-NO-
VPNSERVER=-NO-
```

```
CRASHREPORTER=-YES-
```

The Future of SystemStarter

Past versions of Mac OS X used SystemStarter to launch most system daemons. Starting with Pan however, Apple is moving away from SystemStarter to using socalled *mach bootstrap servers*, a r way of starting up daemons that is much more resource efficient and faster as well. The only prob very little documentation exists at this point for how to properly create and set up Mach boostrap servers. The little that does exist is located on Apple's web site at developer.apple.com/documentation/MacOSX/Conceptual/BPSystemStartup/Concepts/BootProces

For now, if you need to set up a daemon you should either use Startup Items or you should use *x*

By looking at this file you can see that many of the various servers available on the system aren't t started at boot time. If you use the System Preferences application to enable a service and come b this file, you'll see that the `-NO-` next to a service name will change to a `-YES-`.

> The SystemStarter doesn't use */etc/hostconfig* to make its decisions about which services to start; individual services use this information to decide that for themselves. For example, SystemStarter always runs the Apache startup item, but when the Apache startup item is invoked, Apache checks the */etc/hostconfig* file to see whether it should run or not.

4.2.3 Anatomy of a Startup Item

So what does a startup item look like? At a basic level, a startup item is a folder with the name of t startup item that contains the following two files:

- A program; typically a shell script, whose name matches the name of the startup item.
- A configuration property list (*plist*) file named *StartupParameters.plist* declaring the services p by the item and its dependencies. Table 4-3 lists the keys that must appear in the configurati property list.

Table 4-3. Startup Parameters.plist key descriptions

Key	Description
Description	A short description of the startup item. This string will appear Boot Panel at startup.
Provides	A list of services provided by the startup item. A service typica only provides one kind of service. SystemStarter will run only first startup item it finds for any service provided.
Requires	A list of the services that this service depends on and that mu started before this item can be started. Note each item on this must contain the service name, not the startup item name, of service.
Uses	A list of services that this item uses but that aren't required fc item to be started.
OrderPreference	A string that lets SystemStarter determine in which order to s items that are equal in the dependency tree. This string can b of the following: "First," "Early," "None," "Late," and "Last." Th preference is advisory and might be ignored by SystemStarter

Creating your own startup item is easy; much easier than actually writing a program that provides service, although simple shell scripts that delete old files are useful services. Here's the step-by-ste process to create your own startup item.

1. Create a directory in */Library/StartupItems* for your startup item. The name of the directory is name of the item and should give some clue as to its function. Never create startup items in */System/Library/StartupItems*—that's the domain of Mac OS X.

2. Create an executable in the directory and name it the same thing as your startup item. The e can be a shell script or a compiled program.

3. Create the *StartupParameters.plist* file in the directory. This file must be a valid property list t contain values for the Description, Provides, Requires, Uses, and OrderPreference keys. You c this file either by using a text editor and following the format shown in Example 4-6, or by us *PropertyListEditor* application (*/Developer/Applications/Utilities*).

Example 4-6. The StartupItems.plist file for the Apache web server

```
{
 Description      = "Apache web server";
 Provides         = ("Web Server");
 Requires         = ("DirectoryServices");
 Uses             = ("Disks", "NFS");
 OrderPreference = "None";

}
```

SystemStarter is not just responsible for starting up startup items; it can also be used to restart or them. Because of this, SystemStarter passes in an argument to a startup item's executable, indicat desired action. The valid strings for this argument are: "start," "restart," or "stop." A startup item i responsible for checking this argument and acting accordingly. Example 4-7 shows a sample startu that can handle this. You can use this sample script as a template for your own startup items.

Example 4-7. A simple startup item executable

```
#!/bin/sh


# This script includes the /etc/rc.common file which provides useful

# functions that make the job of creating a startup item easier.


. /etc/rc.common


StartService()

{

   "Starting my service"

   # insert code to start service here.


}


RestartService()

{

   ConsoleMessage "Restarting my service"

   # insert code to stop service here.
}


StopService()

{

   ConsoleMessage "Stopping my service"

   # insert code to stop service here.

}


# This should be the last line of your startup item. It calls the

# RunService function defined in rc.common which will in turn call

# the appropriate function in this script.


RunService "$1"
```

4.2.3.1 Starting and stopping startup items

You can start and stop Startup Items at any time using the *SystemStarter* commandline tool. To us tool, use the following syntax:

```
  SystemStarter action service
```

where*action* is either `start`,`stop`, or `restart`, and *service* is the kind of service the Startup Item p Example 4-8 shows how to manually start the Apache web server.

Example 4-8. Manually starting the Apache web server

```
$sudo SystemStarter start "Web Server"

Welcome to Macintosh.

Initializing network

Starting Apache web server
```

```
Processing config directory: /private/etc/httpd/users/*.conf
 Processing config file: /private/etc/httpd/users/duncan.conf
 Processing config file: /private/etc/httpd/users/norman.conf
[Mon Sep 22 00:49:34 2003] [alert] httpd: Could not determine the server's fully q
domain name, using 127.0.0.1 for ServerName
/usr/sbin/apachectl start: httpd started
Startup complete.
Hangup
```

The output from *SystemStarter* lets you know what's going on. In the case of the example here, th
indicating that the server's fully qualified domain name couldn't be found is normal. To stop the Ap
server, use the command shown in Example 4-9.

Example 4-9. Manually stopping the Apache web server

```
$sudo SystemStarter stop "Web Server"
Password:
Welcome to Macintosh.
Stopping Apache web server
/usr/sbin/apachectl stop: httpd stopped
Startup complete.
Hangup
```

4.2.4 Booting into Safe Mode

At some point you may experience a problem with a kernel extension crashing during system startu
may cause a kernel panic before startup is complete. To start your machine and remove or repair t
offending item, Mac OS X can be booted into *Safe Mode*, by holding down the Shift key as you boo
Mac, which will load only the absolute minimum number of kernel extensions and will run only star
installed in */System/Library/StartupItems*.

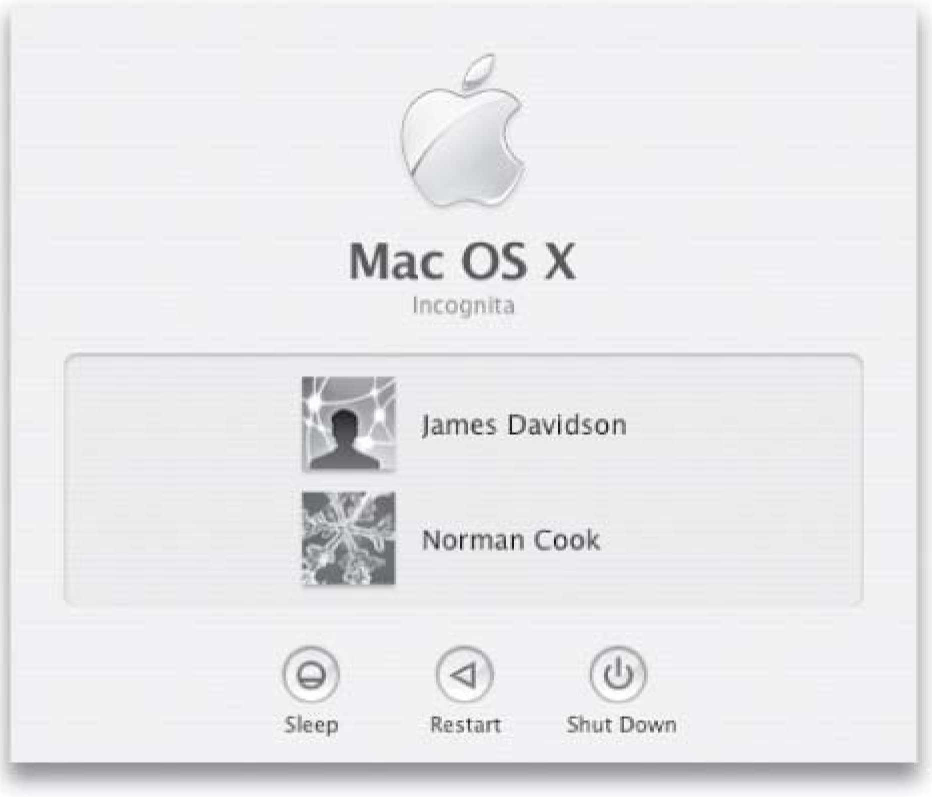
Because of Safe Mode's restrictions on kernel extensions and startup items, you are quite limited ir
you can do. You can't use a DVD player, capture video, use an AirPort wireless network, and many

things. Therefore you should really use Safe Mode only when you need to troubleshoot a startup iss

4.3 Logging In

When the system has booted and starts the *loginwindow* program, the system will either display the user login screen, as shown in Figure 4-4, or, if the auto-login preference has been set in the Accounts preference panel, the system will log in to the user specified there.

Figure 4-4. The Login Panel



When you log in, the following things happen:

- Your environment, including preferences, environment variables, and keychains, are loaded.
- The SystemUIServer is launched and handles the menu bar and menu extras (those little applets up in the upper-left area of the screen).
- The pasteboard server (*pbs*) is launched.
- The mouse, keyboard, sound, and display are configured according to the your preferences.
- Any user-defined login items are processed.
- The Dock and Finder are started.

Bypassing the GUI Login Window

If for some reason you are at the login window and want to bypass it and get straight to the command line (I can't really think of a good reason to do this, but it's possible it would be useful for something you might think of) there's a quick and easy way to do so. Simply enter >console (include the greater-than symbol) as your username in the login dialog box. The GUI will exit and dump you off at the black and white text -based console. Don't worry, the GUI will come back after you log out of your console session.

The only catch to this trick is that you must have your login screen set to display only a username and password box instead of a list of users.

Once the Finder is started and your login items have completed, login is finished and you can use your Mac. After a user has logged in, the *loginwindow* process has the following responsibilities:

- Monitors the Finder and Dock applications and restarts them if they unexpectedly exit for some reason.
- Displays alert dialogs from hidden applications.
- Manages the Force Quit window (→ Force Quit or Option-⌘-Esc).
- Manages the logout process.

> If the Finder or Dock processes die for some reason, the *loginwindow* process will automatically relaunch them. And if, for some reason, the *loginwindow* process itself dies, the init process will restart it so that the login window is displayed and the system isn't left in an unusable state.

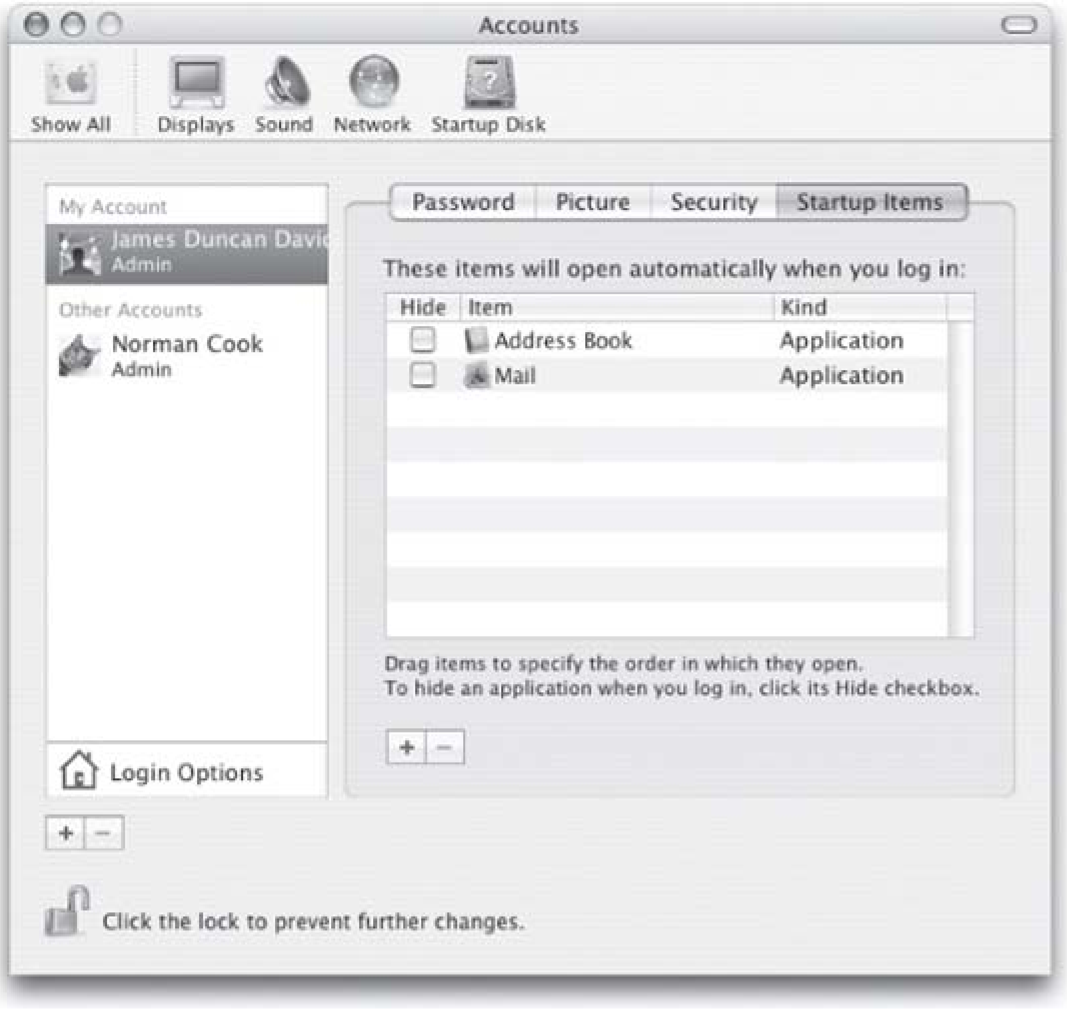
4.3.1 User Authentication

While the *loginwindow* process manages the login window and the process of logging a user in to the machine, it doesn't actually perform the authentication of the user's credentials (usually a password). Instead it passes off authentication to Directory Services. Only if a user's authentication credentials are accepted by Directory Services will the login process continue. Otherwise, the login window will shake—resembling somebody shaking her head and saying no—and will not log the user in to the system.

4.3.2 Login Items

After they log in, users can specify applications to launch automatically by using the Accounts preference panel. A list of these applications can be found and configured through the Startup Items tab of the Accounts preference panel as shown in Figure 4-5. For example, in the figure you can see that both Address Book and Mail are set to start up automatically on login. You can put pretty much any application you want into this list.

Figure 4-5. The Accounts preference panel



These items will open automatically when you log in:

Hide	Item	Kind
□	Address Book	Application
□	Mail	Application

Drag items to specify the order in which they open.
To hide an application when you log in, click its Hide checkbox.

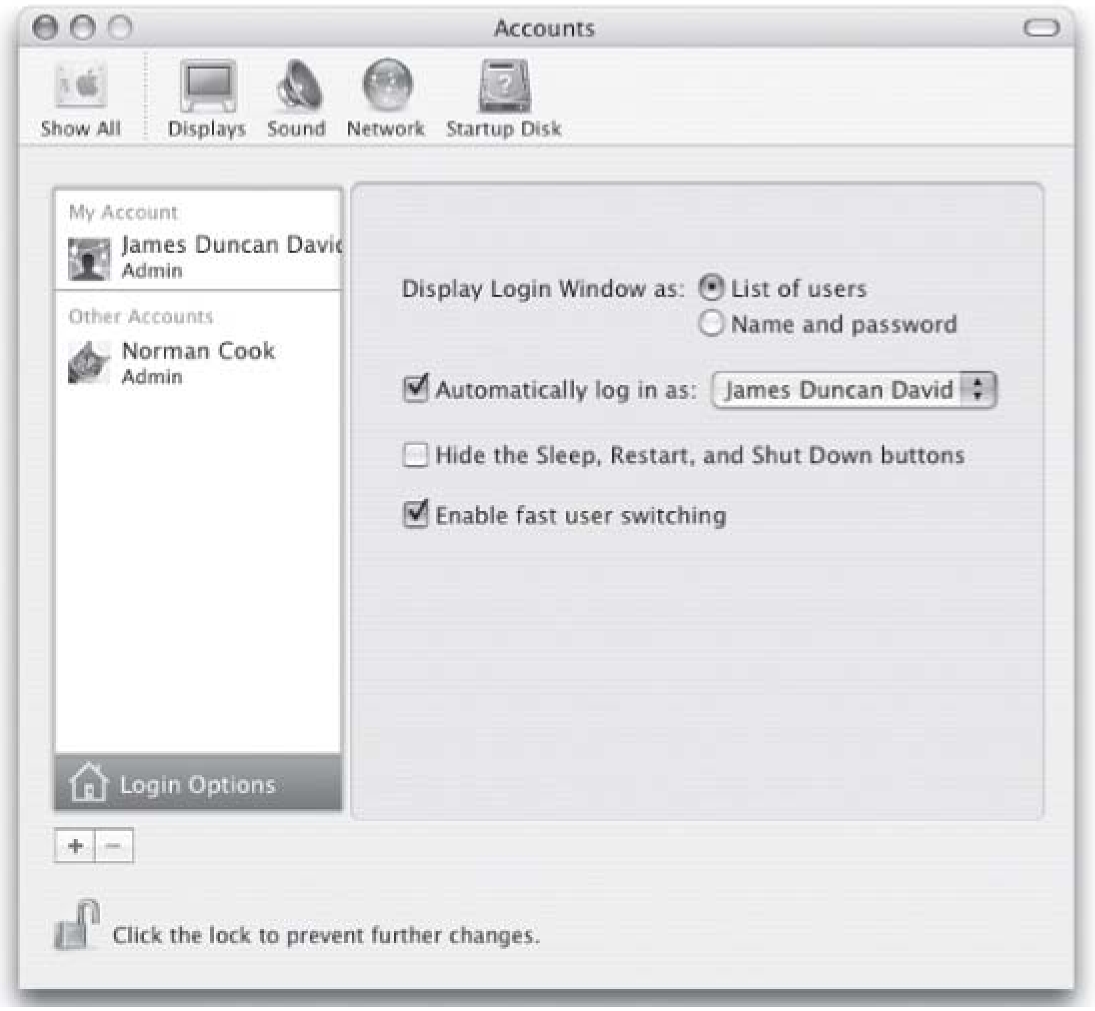
Click the lock to prevent further changes.

If you want an application to run but not show up in your Dock, click the Hide button next to the item. Many support applications, such as the Palm Desktop HotSync manager, will run hidden using the functionality provided by this panel.

4.3.3 Customizing the Login Screen

You can customize the login window through the Login Options tab of the Users System preference panel, as shown in Figure 4-6. You can have the list of users displayed as pictures or require a username and password to be entered. You can also hide the restart and shutdown buttons so random users won't be able to turn off your computer.

Figure 4-6. Login options



Display Login Window as: ● List of users
○ Name and password

☑ Automatically log in as: James Duncan David

□ Hide the Sleep, Restart, and Shut Down buttons

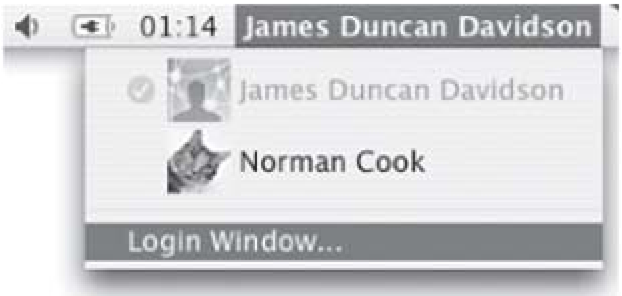
☑ Enable fast user switching

Click the lock to prevent further changes.

This panel also lets you enable Fast-User Switching, a new feature in Panther that lets you have multiple users logged in at the same time. When you activate this, your username will show up in the upper right-hand corner of the screen. You can click your name to log in as another user or to see just the login panel, as shown in Figure 4-7.

When another user logs in, either through the login window or Fast-User Switching, the same login process described in this section occurs.

Figure 4-7. Fast-User Switching menu



01:14
James Duncan Davidson
James Duncan Davidson
Norman Cook
Login Window...

4.4 Monitoring Users

You can see who's logged in to the machine right now using the *who* command, as shown in Example 4-10.

Example 4-10. Using the who command

```
$who
duncan console Dec 8 15:39
duncan ttyp1 Dec 8 23:31
norman ttyp2 Dec 8 23:37 (localhost)
duncan ttyp3 Dec 8 23:38
```

The console entry is the GUI shell that you are logged in to. The *ttyp* entries are created by active Terminal windows.

The*w* command gives a different output of this information, as shown in Example 4-11.

Example 4-11. Using the w command

```
$w
11:38PM up 1 day, 8 hrs, 4 users, load averages: 0.28, 0.50, 0.49
USER   TTY FROM           LOGIN@  IDLE WHAT
duncan  co -             Wed03PM 31:59 -
duncan  p1 -             11:31PM     0 -
norman  p2 localhost     11:37PM     0 -
duncan  p3 -             11:38PM     0 -
```

As well as listing the users logged into the system, the *w* command also gives the system uptime and load averages on the CPU.

You can see who has been logged in to the system (as well as see when the system has been rebooted) by using the command-line *last* command, as shown in Example 4-12.

Example 4-12. Using the last command

```
$last

duncan    ttyp3                     Thu Dec 8 23:38   still logged in

duncan    ttyp3                     Thu Dec 8 23:38 - 23:38 (00:00)

norman    ttyp2  localhost          Thu Dec 8 23:37   still logged in

duncan    ttyp1                     Dec May 8 15:39   still logged in

duncan    ttyp1                     Dec May 8 15:39 - 15:39 (00:00)

duncan    console Archetype.local.  Dec May 8 15:39   still logged in

reboot    ~                         Dec May 8 15:39

shutdown  ~                         Dec May 8 15:37
```

4.5 Logging Out

You can log out of your session using any of the following methods:

- Use the → Log Out menu
- Use the Shift-⌘-Q keystroke combination while using any application

When you use either of these methods, the system will display a dialog box making sure that you really want to quit. If you don't want to see this dialog box, hold down the Option key while using the menu item or use the Option-Shift-⌘-Q key combination.

When you log out, all programs that you started while logged in will be quit automatically (if you haven't already quit out of them). However, any application can block logout by asking to save a document.

4.6 Shutting Down the System

When you shut down the system or reboot, a controlled process occurs so that the system is left in a good state. This process is:

- All running startup items are called with the stop parameter giving them a chance to shut down gracefully.
- All running processes are stopped.
- The system syncs the filesystems with their in-memory caches.
- The system hands off control back to Open Firmware.
- Open Firmware either shuts down the system or reboots depending on the instructions that the system handed to it as it shut down.

You can either shut down the system by selecting → Shut Down menu from the menu bar or from the command line using either the *reboot* or *shutdown* commands. To reboot the system, simply enter the following:

```
$ sudo reboot
```

To shutdown the system:

```
$ sudo shutdown now
```

> When you reboot or shut down from the command line, all running applications will be terminated without being able to raise any dialog boxes. This means any unsaved work will be lost.

4.7 Further Explorations

For more information about the topics in this chapter, see the *Mac OS X System Overview* book. It's installed on your system as part of the Xcode Tools package and is located at */Developer/Documentation/MacOSX/Conceptual/SystemOverview/System- Overview.pdf*.

You may also want to look at the following manpages:

- *nvram*
- *bless*
- *rc*
- *SystemStarter*

Chapter 5. Users and Groups

Thanks to its Unix heritage, Mac OS X is a multiuser operating system through and through. This simple fact means there can be more than one user of your system. You can have accounts for every member of a household on one machine, and everyone's stuff will remain independent and safe. Even better, with Panther multiple users can be logged in to the same machine at the same time. While only one user at a time can use the screen, with a quick click of a menu you can switch effortlessly between user sessions.

What isn't obvious at first glance is that the concept of users runs quite deep in Mac OS X. Not only are human users treated as separate entities by the system, but many nonhuman users exist on the system as well. This means different tasks can be performed safely and in isolation from other tasks. Users can also be associated with groups, allowing the system to treat many users the same way.

5.1 What is a User Anyway?

From the operating system's point of view, a user isn't necessarily a real person who taps away at the keyboard. A user is simply an entity that can own files and execute programs. A user is defined in terms of an account that has a set of properties including a numeric user ID, such as `501`, and a username. Internally the system uses the user ID to keep track of the files and processes that belong to a user. The username is a more human-readable form that is used heavily throughout the system so that you don't have to think in terms of numbers.

Each user is also part of one or more groups. A group is a collection of users that the system can treat as a unit. Like a user, the system defines a group in terms of a numeric group ID and a more readable group name. Associating users with groups gives you the ability to control various resources of the system, and not just on a case by case basis. For example, the system uses the admin group to indicate users that can administer the computer. If your user ID is associated with the admin group, you have the ability to perform administrative tasks on the system.

To take a look at your user and group IDs, log in to the command line and execute the *id* command as shown in Example 5-1

Example 5-1. Using the id command

```
$id

uid=501(duncan) gid=501(duncan) groups=501(duncan), 80(admin)
```

The results of this example tell us that the currently logged-in user is named `duncan`, has a user ID of `501`, and is a member of the `duncan` and `admin` groups. If you take a look at a non-administrative user, you will see something like the output shown in Example 5-2.

Example 5-2. Using the id command for a specific user

```
$id norman

uid=502(norman) gid=502(norman) groups=502(norman)
```

This user isn't part of the `admin` group and therefore won't be allowed to administer the computer. Typical users have access to their Home folders and to the System Preferences panels that customize their user experience, but the rest of the system will be off limits (at least it will be if they don't know an administrator's username and password).

You'll notice in the examples above that the user IDs are 501 and 502. And if you are the only user on your machine you'll notice that your user ID is also 501. This pattern is not coincidental; it follows the numbering scheme that Mac OS X uses to separate out users that should appear in the login window from those that should not. The rule is that users with an ID less than 500 won't appear. Those with an ID greater than 500 will appear.

5.1.1 Administrative Users

Administrative users represent a special class of users on the system. Here are just a few things that they are allowed to do that users without administrative privileges aren't:

- Install new programs in the */Applications* folder.
- Add items to the */Library* folder such as startup items that take effect when the system is booted.
- Change network settings using the Network preference panel.
- Change the system time using the Date & Time preference panel.
- Add users to or remove them from the system.
- Set up or remove printers.

Without administrative privileges, users are pretty much restricted to their Home folder and are only able to change the settings in System Preferences that relate to their desktop such as screen backgrounds and Finder preferences. The various System Preferences panels that require administrator access have a closed padlock on them indicating that the user is not able to change the settings.

When you first install Mac OS X, the user that you create during the installation process automatically has administrative privileges. However, users created after that time will not have administrative privileges unless the person who creates their account (an administrator) grants the user admin privileges in the Accounts preference panel. You should consider your security needs and determine whether the user ID that you use for your normal work on the machine should have administrative privileges.

5.2 Managing Users

Unix typically stores its user and group information in the */etc/passwd* and */etc/group* files—and if you go looking you can find these files on your machine. However, after Mac OS X boots it does not use these files. Instead it uses Open Directory to store its user and group information. This allows the system to work equally well in home setups where there is only one machine and in enterprise environments where there might be hundreds of machines that use a central server for authentication.

While there are many ways to manage users on Mac OS X, the simplest and most direct by far is to use the Accounts preference panel.

5.2.1 Managing Users with the Accounts Panel

When you open the Accounts preference panel, you are presented with a list of users on the system and a set of tabbed panes to modify users, as shown in Figure 5-1.

5.2.1.1 Creating a user

To create a user, click the plus (+) button. This creates an unnamed user and takes you to the Password tab. The various fields are:

Name

> This is the full name for the user. This name shows up in most places where Mac OS X displays user information such as the login panel and any of the alert screens that prompt you for an administrator password.

Short Name

> This is the Unix-style name for the user and is what you'll typically see on the command line. The default short name runs your full name together with no spaces in it and is represented in lowercase letters. Unix usernames of yore used to be limited to 8 characters or less while Mac OS X allows short user names up to 255 characters.

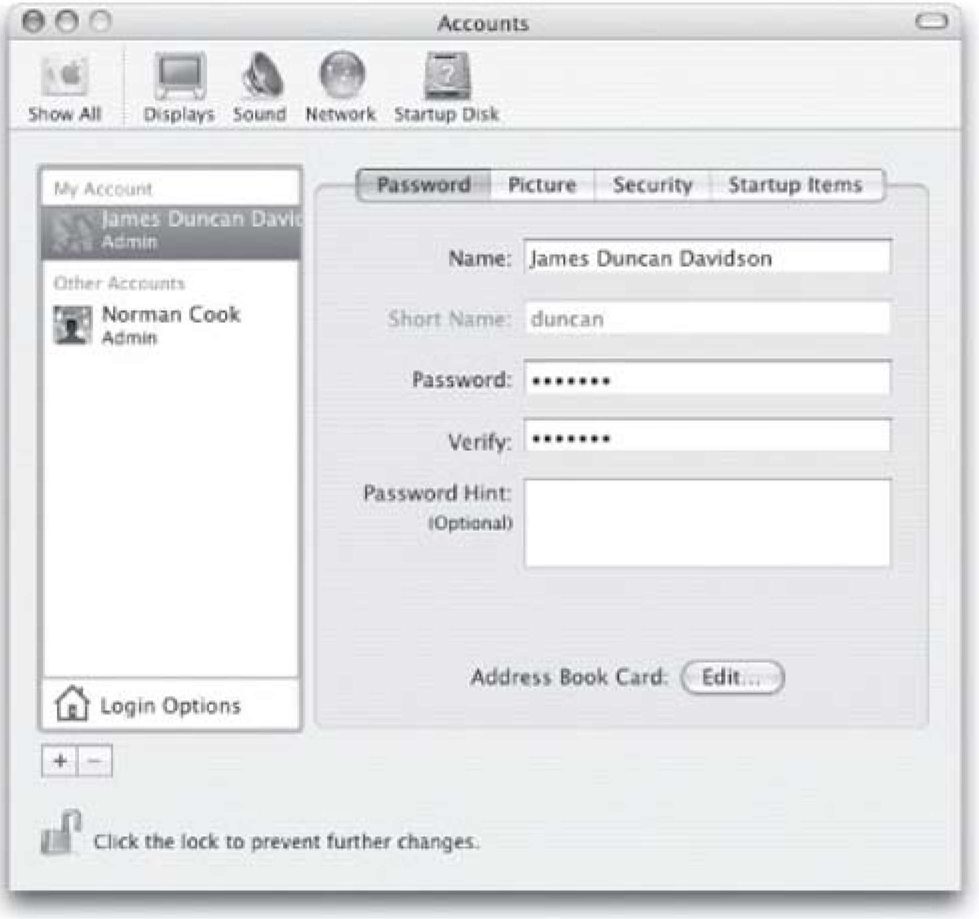
Password & Verify

> This is where you set the password for the user.

Password Hint

> This is where you define a hint that will be displayed to the user if an incorrect password is entered more than three times.

Figure 5-1. The Accounts preference panel



Name: James Duncan Davidson

Short Name: duncan

Password: •••••••

Verify: •••••••

Password Hint: (Optional)

Disabling a User

Some operating systems allow you to disable a user's account so it can't be used but the user's Home directory remains intact. Mac OS X doesn't let you disable a user per se. However, you can always accomplish the same thing by changing the user's password to something that only you, the administrator, will know. The user will be locked out, but the Home directory will still be intact.

The other three tabs of the Accounts preference panel allow you to fine tune the settings for a user. They are:

Picture

> This allows you to associate a picture with a user which is handy for the various user lists. You can either use one of the Apple provided pictures or choose one of your own. Also, if you have an iSight camera connected to your computer, the Add Picture dialog box will let you take a snapshot, which you can use for this picture.

Security

> This lets you enable FileVault for a user as well as allow the user to administer the machine. Remember, when you allow somebody to become an administrator, that user becomes a member of the *admin* group and can modify the system however they see fit.

Limitations

> This allows you to place limitations on the users' ability to use the system over and above just choosing whether or not they can administer the machine. For example, you can restrict the applications that they use or you can set them up with the Simple Finder, which is perfect for kids.

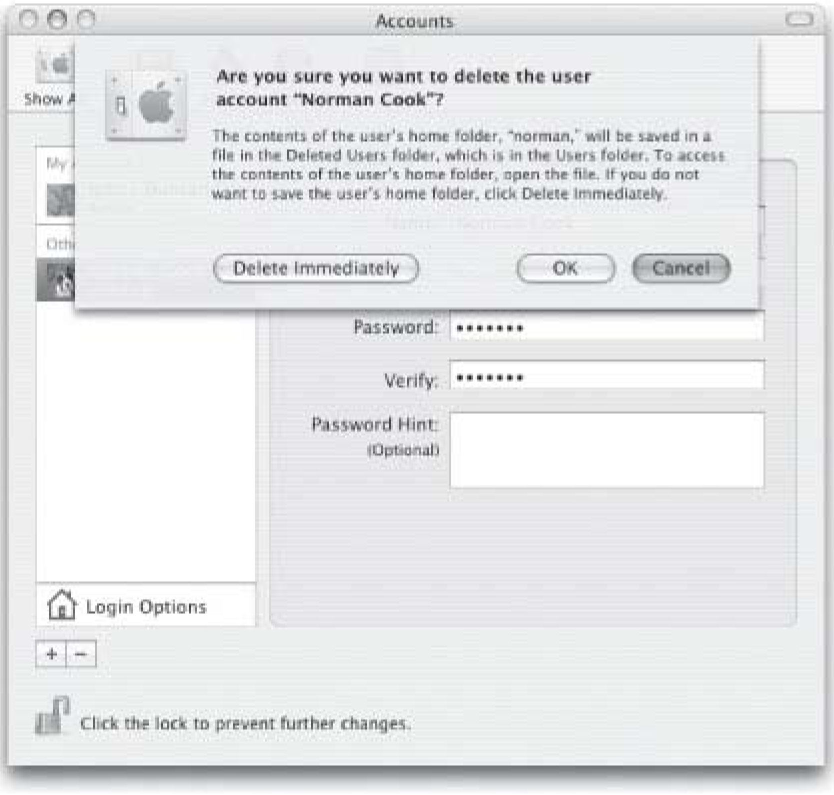
When you've finished setting up the user, his or her Home folder is created in the */Users* folder, and they will be able to log in to the system.

5.2.1.2 Deleting a user

To delete a user, select the name of the user from the list and click the minus button (-). You are presented with a dialog box asking whether or not you really want to delete the user and what you want to do with the contents of the user's Home folder. You can either archive the user's folder to a disk image (*.dmg*) file in the */Users/DeletedUsers* directory or quickly and permanently erase it, as shown in Figure 5-2.

If you choose to save the contents, you can browse through them at any time by double-clicking the *.dmg* file. This mounts a temporary drive from which you can restore a user's data. When you have decided that you no longer need the files for the user, you can delete the disk image from the */Users/Deleted Users* directory as long as you have admin privileges.

Figure 5-2. Deleting a user with the Accounts preference panel



Are you sure you want to delete the user account "Norman Cook"?

The contents of the user's home folder, "norman," will be saved in a file in the Deleted Users folder, which is in the Users folder. To access the contents of the user's home folder, open the file. If you do not want to save the user's home folder, click Delete Immediately.

Password: •••••••

Verify: •••••••

Password Hint: (Optional)

Click the lock to prevent further changes.

> As noted, after you have deleted a user, you can opt to delete their information from the system immediately or save it to a disk image. One option for system administrators is to save the disk image and then burn it to CD or DVD for historical purposes. Once the disc has successfully burned, you can then delete the disk image to free up drive space.

5.2.1.3 Managing users with NetInfo Manager

When you create an account using the Accounts preference panel, all properties about that user are stored in the local NetInfo database managed by Open Directory. To see the contents of this database, use NetInfo Manager (*/Applications/Utilities*), which provides a bare-bones view of the NetInfo database and will allow you to make substantial changes. You'll see more about NetInfo Manager and how user records are stored in Open Directory in Chapter 9,*Preferences and Defaults*.

Managing Users from the Command Line

Many hardcore Unix users are used to being able to manage users from the command line using tools like *useradd, usermod, userdel, groupadd, groupmod, groupdel, gpasswd, grpconv,* and*grpunconv*. Unfortunately, these command-based utilities don't exist on Mac OS X. The only tool of this kind on Mac OS X is the *passwd* command, which is used to change a user's password.

While it is possible to perform some user management from the command line through direct manipulation of the NetInfo database using the *niutil* command (covered in Chapter 10), several pieces of data in a user's NetInfo record, such as the generateduid property, don't have command-line tools to manage them. For almost all purposes, and unless you really need to manage users from the command line, you should use the Accounts preference panel to manage users.

5.3 Nonhuman Users

Even if you are the only human user of the system, there are well over a dozen accounts on the sys Most of these user accounts are not intended for use by you or any user on the system but are set use by various services and programs on the system. These nonhuman accounts let applications, s the Apache web server and the Postfix mail server, run in a controlled environment so if they are b by a hacker, the potential damage is limited. Table 5-1 lists the nonhuman users that are defined c system.

Many of these nonhuman users listed in the table, such as *cyrus* and *qtss*, aren't used on the avera person's system but instead are defined for use on Mac OS X Server. Others, such as *postfix* and *w* only used when you run the Postfix mail server or the Apache web server. Under most conditions, y only notice processes owned by either your own ID, or by the root user, when view processes in the Activity Monitor or with the *ps* command.

5.3.1 The Root User

As in all Unix systems, Mac OS X has a special user, named *root*, which is not subject to the contro permissions structure. The root user can modify any part of the filesystem as well as execute any p It can also stop the execution of any running program on the system.

The root user is a dangerous one. One command executed as root, such as *rm -rf /*, can immediate disable a system; you really have to think about what you are doing with every command you issu root. Therefore, Mac OS X is configured by default to allow access to the root user only through the program. To use this program you must be an administrative user, and simply preface the commar want to enter with *sudo*.Example 5-3 shows how to use *sudo* to print out the contents of *secure.log* that is only visible to the root user and that contains the records of the various actions of Panther's systems.

Table 5-1. Mac OS X's nonhuman users

Username	UserID	Description
root	0	The administrative user
daemon	1	Core system daemons
smmsp	25	Sendmail user
lp	26	Printing service
postfix	27	The Postfix SMTP server
www	70	The Apache web server
eppc	71	Apple Events
mysql	74	The MySQL database (Mac OS X Server)
sshd	75	The SSH server daemon
qtss	76	The QuickTime Streaming Server (Mac OS X Server)
cyrus	77	The Cyrus POP and IMAP servers (Mac OS X Server)
mailman	78	The Mailman mailing list manager (Mac OS X Server)
appserver	79	The JBoss application server (Mac OS X Server)
nobody	-2	A user with greatly restricted access

Example 5-3. Using sudo

```
$sudo cat /var/log/secure.log

Password: ********

Nov 12 04:16:11 localhost com.apple.SecurityServer: Entering service

Nov 12 04:36:55 localhost com.apple.SecurityServer: Succeeded authorizing right sy

preferences by process /System/Library/CoreServices/Setup Assistant.app for author

created by /System/Library/CoreServices/Setup Assistant.app.
```

Because*sudo* keeps an internal timer you can execute multiple commands without typing your pass each time. It also logs each use so you can go back and see a list of commands that were executed handy when you have multiple users with administrative privileges on a system.

If you prefer to live a bit more dangerously, you can get a shell as the root user by issuing the follo command:

```
$sudo -s
Password: ********
#
```

The prompt changes to # indicating that every command typed will be run as the root user.

> Even though it takes more work, you can and should avoid opening root shells usin the*sudo* command. By doing this, you are less likely to make a horrific mistake and hose your system. If you do make a mistake each command will be logged to */var/log/system.log* allowing you to figure out what went wrong.

5.3.1.1 Enabling the root user

Some people really want to be able to log in to their system as *root*. For some reason having administrative privileges and the ability to execute any command using *sudo* isn't enough. If you a of these people, you can enable the *root* user so that you can log in either to the GUI or the comma and have unfettered and unmonitored access to your system. I don't recommend that you do this, you insist here's how:

1. Launch NetInfo Manager (*/Applications/Utilities*).
2. Authenticate yourself using the Security→Authenticate... menu.
3. Enable the root user using the Security→Enable Root user menu.
4. Give the root user a password, one that is as secure as any password you would give an adm of the system.

If you follow this procedure, you will have a fully functional *root* user. You can log out of your syste log in as the root user. Remember: you should stick to using *sudo* instead of using the root user.

If you want to enable the root user from the command line, you can do so by executing the followi command:

```
$sudo password root
```

After you set the password, the root user will be active.

5.3.2 Creating a Nonhuman User

If you need to create a nonhuman user for some reason (for example, to run some server program securely), you shouldn't create that user account with the Accounts preference panel. The reason fo because nonhuman users don't need a Home folder and the other folders a normal user gets when account is first created. Instead, you should create the account by directly editing the NetInfo data For details on how to create a nonhuman user account, see Chapter 11,*Open Directory*.

Chapter 6. Files and Permissions

Files are central to Unix-based systems. Commands are executable files. Devices and disks are identified as files. Even most interprocess and network communication occurs through what appear to be files. This Unix view of the world permeates the lowest levels of Mac OS X, even to the point that many system privileges and permissions are controlled, in part, through access to files. Access to files is organized around the concepts of ownership and permissions.

The chapter starts out by looking at how to find files, an area in which Panther's Finder shows great improvement over the previous version of Mac OS X (appropriately enough for an application named "Finder"). You'll also learn more about how to work with files, and how to view and modify a file's permissions and attributes.

6.1 Finding Files

When you are working with your system, most of the time it's not a matter of having the correct permissions to access a file that gets in your way, it's being able to find the file in the first place. The first and easiest way to find files is by filename using the search text box in every Finder window, as shown in Figure 6-1.

Note that you can have your search occur in several places. These are:

Local Disks

> The Finder will search every disk attached to your machine to find the filename you are looking for.

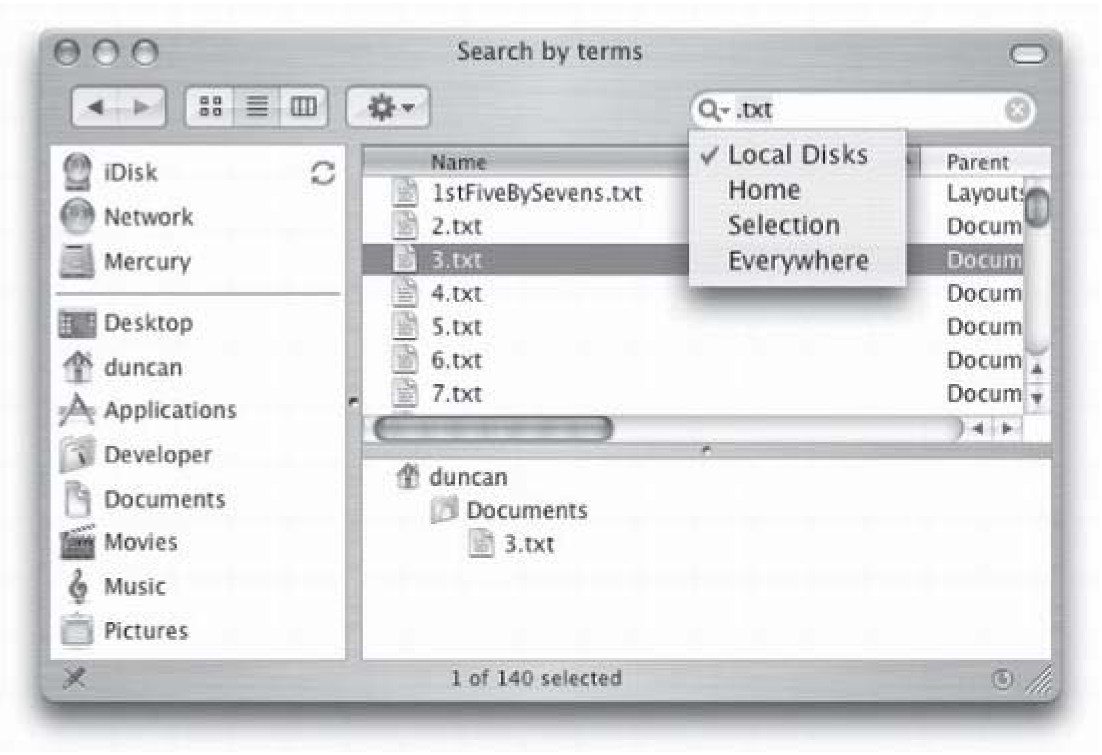
Home

> The Finder will search all the files in your Home folder.

Selection

> The Finder will search the item currently selected. This allows you to constrain a search to a particular folder.

Figure 6-1. Using the Finder to search for files



Everywhere

> The Finder searches every file it can access, including those on remote network drives, in order to find the most matches possible. This kind of search may take a long time to complete, especially if you have large network drives mounted to your Mac.

If you need to conduct more constrained searches, use the File→Find (⌘-F) in Finder to bring up the Find window, shown in Figure 6-2. This will let you build sophisticated search queries. In the Find window, you have the ability to search based on the following types of information:

Name

> Lets you search on the filename of the file. Your search can be based on either a partial or an exact match.

Content

> Lets you search the contents of a file.

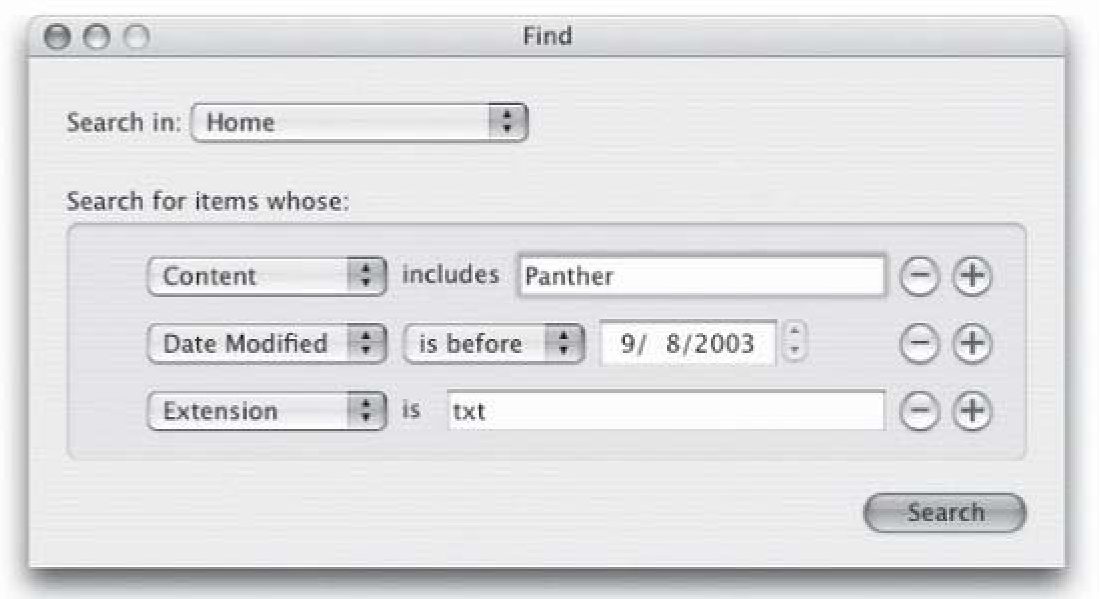
Label

> Lets you search by the label that you have applied in the Finder.

Kind

Lets you search for aliases, applications, folders, documents, audio files, image files, and movies.

Figure 6-2. Finder's Find window



Find

Search in: Home

Search for items whose:

Content includes Panther

Date Modified is before 9/ 8/2003

Extension is txt

Creator

Lets you search by creator code, a four-letter abbreviation used by some Mac applications to associate files with them.

Type

Lets you search by the type code, another four-letter abbreviation used by some Mac applications.

Extension

Lets you search the part of the filename after the dot, such as *txt*,*html*, or *jpg*.

Size

Lets you focus your search to files that are either smaller or larger than a given size.

Date Created

Lets you search for files created before, on, or after a particular date.

Date Modified

Lets you search for files last modified before, on, or after a particular date.

Each search criterion you add will introduce its own set of pop-up menus or text areas for defining your search.

6.1.1 Finding by content

When you search by content, the Finder goes an extra step beyond looking at the metadata of the file. It searches inside the files by consulting an invisible index (a hidden *.FBCIndex* file) in the folders it is traversing. If an index isn't present in the folder, the Finder creates one automatically, which can take a fair amount of time. The Finder can build up indexes in any of the languages that Mac OS X is localized for. However, you might consider speeding up the index creation by only enabling creation of indexes for the languages you care about. You can make this setting in the Finder's preferences.

6.1.2 Finding files with the command-line

When you need to find files using the command line, don't forget about the *find*,*grep*, and *locate* commands that were covered in Chapter 3.

6.2 File Ownership

File ownership in Mac OS X is based directly on the underlying BSD Unix layer and inherits its strengths (as well as a few quirks) from that legacy. On Unix systems, a file has two owners: a use and a group. Each of these owners is separate from the other; there's no requirement that the own of a file is in the group associated with that file. This split in ownership is intended to let you be as flexible as possible in the way that you structure access to files. By allowing groups as well as individual users to be associated with a file, you can give users access to an entire set of files simp by adding them to a group, and you can take away access just as easily.

You can see the owner and group for a file in the Finder using the File→Get Info (⌘-I) menu and unfolding the Ownership & Permissions section and the Details subsection, as shown in Figure 6-3. The Inspector tells you what the owner, the users in the group that owns it, and everybody else ca do with the file. In the case of Norman's Home folder, Norman can both read and write to his Home folder (which makes sense), and members of the group *norman* can only read the files, while everyone else can only read the files.

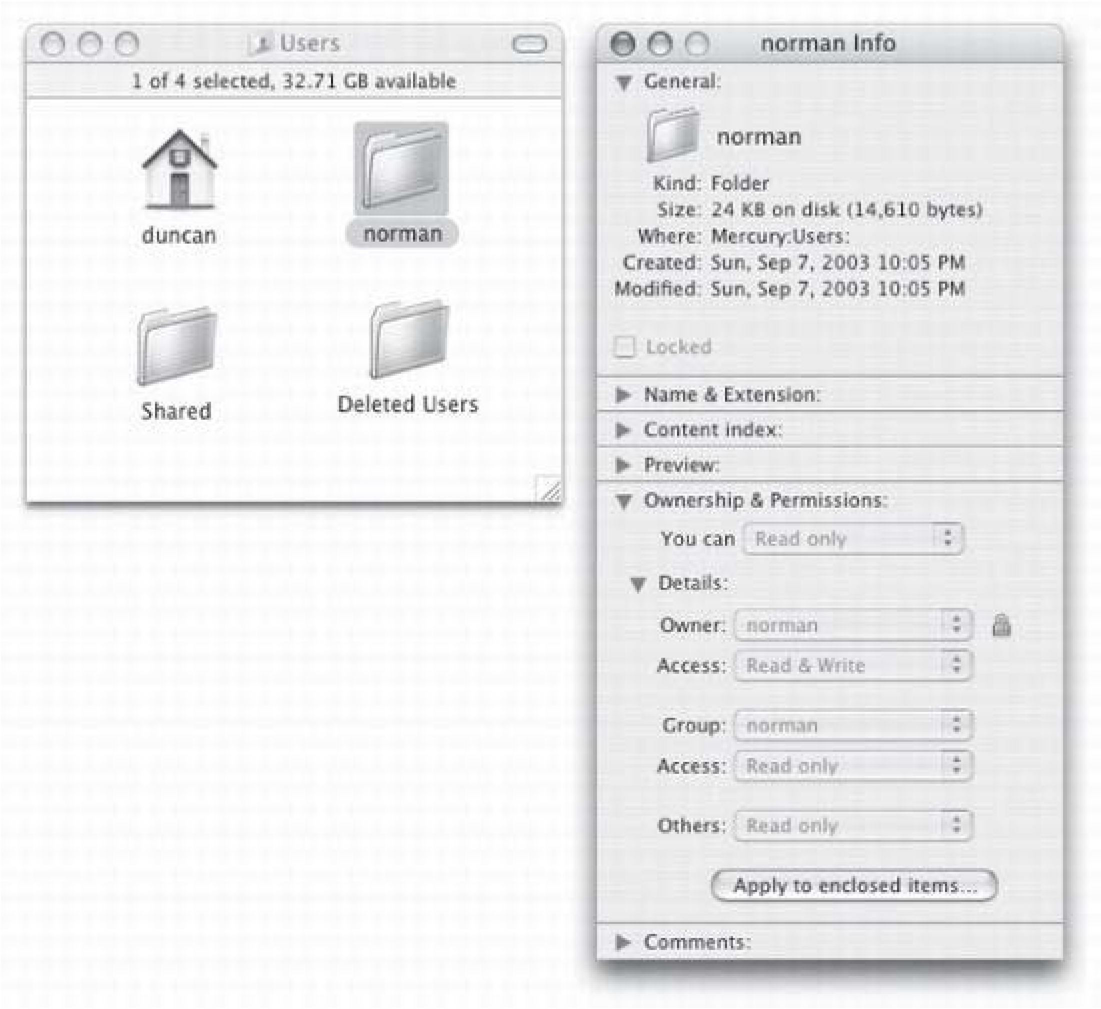
Since the file ownership model in Mac OS X comes from Unix, it follows that there is a way to view these permissions from the command line. Example 6-1 shows the use of the *ls* command to view the ownership details for the contents of the */Users* directory.

Example 6-1. A file listing of the Users directory

```
$ls -l /Users

total 0

drwxrwx---   4 root    admin  136 8 Dec 20:22 Deleted Users

drwxrwxrwt   5 root    wheel  170 8 Dec 17:13 Shared

drwxr-xr-x 23 duncan  duncan 782 8 Dec 21:59 duncan

drwxr-xr-x 11 norman  norman 374 8 Dec 22:05 norman
```

Believe it or not, the line describing Norman's Home folder gives you the same information as did t file Inspector (albeit in a very concise form that is easy enough to read once you know how). Figur 6-4 shows the kinds of data in each column. It's all quite self-explanatory except for the first set of characters, which can be broken down as follows:

Figure 6-3. Using the file inspector to look at permissions



General:

norman

Kind: Folder
Size: 24 KB on disk (14,610 bytes)
Where: Mercury:Users:
Created: Sun, Sep 7, 2003 10:05 PM
Modified: Sun, Sep 7, 2003 10:05 PM

□ Locked

Ownership & Permissions:

You can: Read only

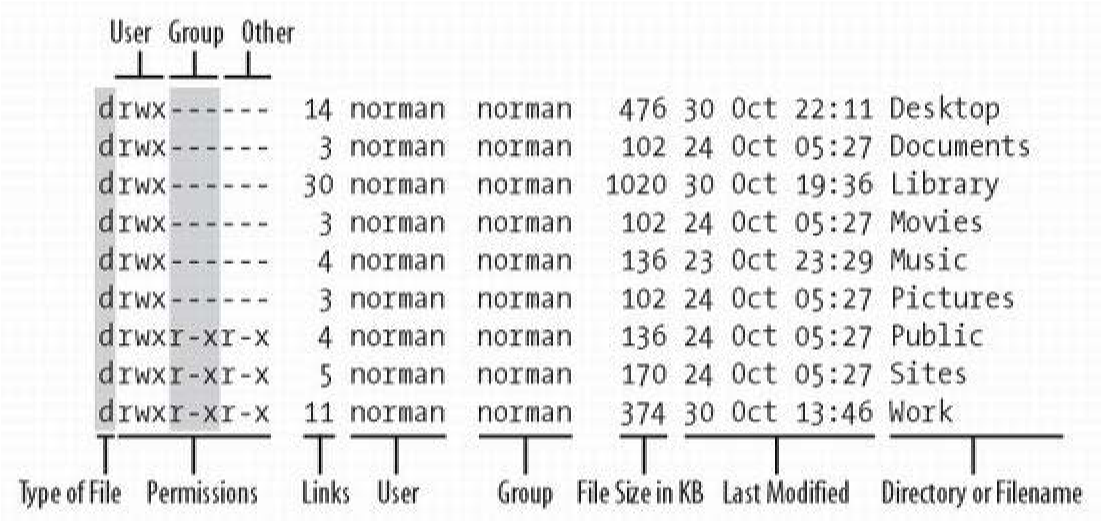
Details:

Owner: norman
Access: Read & Write

Group: norman
Access: Read only

Others: Read only

Figure 6-4. Output of ls -l



```
drwx------  14 norman  norman   476 30 Oct 22:11 Desktop
drwx------   3 norman  norman   102 24 Oct 05:27 Documents
drwx------  30 norman  norman  1020 30 Oct 19:36 Library
drwx------   3 norman  norman   102 24 Oct 05:27 Movies
drwx------   4 norman  norman   136 23 Oct 23:29 Music
drwx------   3 norman  norman   102 24 Oct 05:27 Pictures
drwxr-xr-x   4 norman  norman   136 24 Oct 05:27 Public
drwxr-xr-x   5 norman  norman   170 24 Oct 05:27 Sites
drwxr-xr-x  11 norman  norman   374 30 Oct 13:46 Work
```

- The first letter denotes the file type. In most cases this will be either a dash (-) for a file or a for a directory.
- The next three letters (characters 2-4) indicate the permissions associated with the user owne of the file.
- The next three letters (characters 5-7) indicate the permissions associated with the group owner of the file.
- The last three letters (characters 8-10) indicate the permissions associated with everyone wh is neither the file's user nor a member of the group owner.

Each of the three letter groupings consists of either the letters *r*,*w*, and *x*, or the dash (-) character A letter means a permission is given, and a dash means the permission is withheld. Table 6-1 show the meanings of these letters.

Table 6-1. File permission characters

Letter	Meaning for a directory	Meaning for a file
`r`	Can read the contents of the directory	The file can be read
`w`	Can alter the contents of the directory	The file can be written
`x`	Can make the directory the current directory (that is, use *cd*)	The file can be executed
`t, T`	The directory is "sticky" (see discussion on the sticky bit)	

Once you know the secret decoder to the string, it is pretty easy to read the permissions for a file. Using this information, you can read the line describing Norman's Home directory in the file listing Example 6-1 as follows:

- The file is a directory.
- Norman has the ability view the contents of, and to read and write files to, the directory.
- Members of the *norman* group can read files and *cd* to them.
- Everyone else can read files and *cd* to them.

The Execute Bit and the Finder

Observant Unix veterans will notice that the Finder doesn't offer a way to modify a file's execute permissions. Furthermore, directories created in the Finder will always have execute permissions set. Usually, this isn't a problem but if you do want to modify the execute permissions of a file or a directory, you will have to do so at the command line using*chmod*.

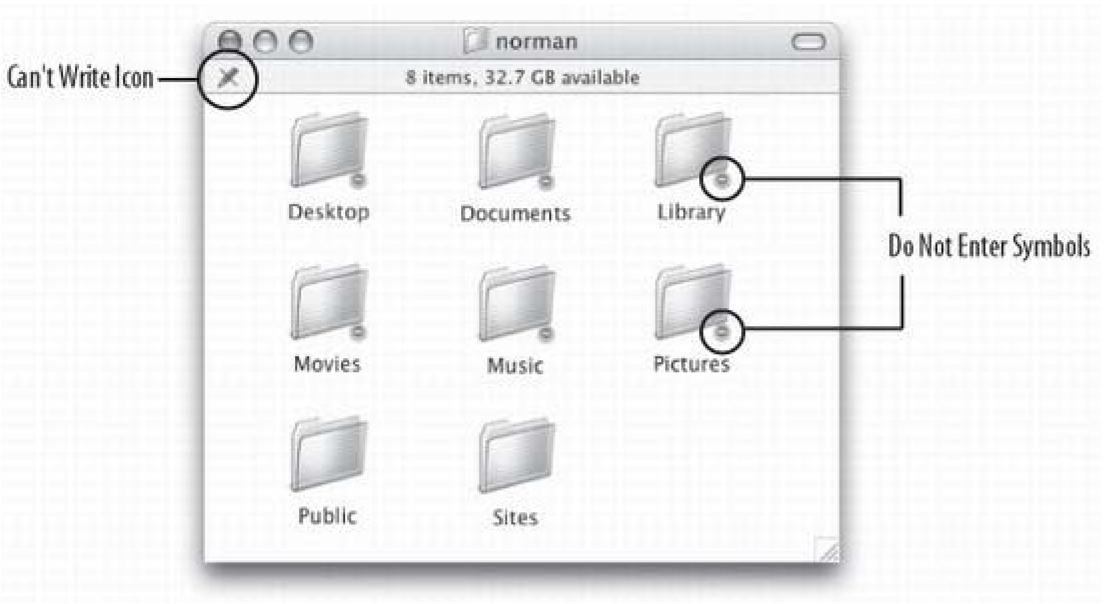
Furthermore, you can see that Norman is the only user who can access the *Desktop*,*Documents*, *Library*, and most of the other directories in this Home folder. In fact, the only directories that anybody else but Norman can look into are his *Public* and *Sites* directories. Table 6-2 gives an overview of what the various letters can mean in combination.

Table 6-2. File permission summary

Pattern	Meaning	Result
`---`	No access	No activity allowed
`r--`	Read access only	Lets users read the file
`--x`	Execute access only	Lets users execute a program
`r-x`	Read and execute access	Lets users read and execute the file
`-wx`	Write and execute access	Lets users write the file, but not read it; useful for Drop Boxes
`rwx`	Full access	Lets users read, write, and execute the file

These permissions are reflected in the Finder view of Norman's directory that another user would have shown in Figure 6-5. You can see the "Can't Write" icon in the upperleft part of the window, and you can see that the *Desktop*,*Documents*,*Library*,*Movies*,*Music*, and *Pictures* folder have a "Do Not Enter" symbol on them indicating that you can't view their contents.

Figure 6-5. Finder view of Norman's Home folder



6.2.1 Changing File Ownership

If you need to change the ownership of a file, directory, or even a directory tree, you can do so easily from the Finder by using the Inspector. Simply set the various pulldown menus to change th settings for Owner, Group, and Others. To accomplish some of the operations, you'll need to click t padlock icon and authenticate yourself. An administrative user can change ownership for any other user's files. On the command line you'll need to use the *chown* and *chmod* commands.

Special Directories (dot and dot-dot)

In the output from *ls -la*, there are two special directories, one named dot (.) and the other dot-dot (..). These are special directories that serve as useful shortcuts. The dot directory always refers to the current directory. For example, the path `./foo.txt` refers to the *foo.txt* file in the current directory. The dot-dot directory always refers to the parent of the current directory. The path `../bar.txt` refers to the *bar.txt* file in the parent of the current directory.

6.2.1.1 chown

The*chown* tool, short for "change ownership", changes the owner of a file. Its basic syntax is:

```
chown owner file
```

where*owner* is the username or user ID of the new owner for the specified *file*. For example, to change the owner of the file *ImportantDocument.doc* to *norman*, you would use the following command:

```
$ sudo chown norman ImportantDocument.doc
```

You have to issue this command with *sudo* because you need to authenticate yourself as an administrator to the system to make this change. After all, you may not want a user to put files under the ownership of somebody else without oversight.

You can also change both the user and group settings for a file by using the following syntax:

```
chown owner:group file
```

where the *owner* and *group* arguments are separated by a colon. For example, to give everyone in the group *auditor* access to the document, you would use the following command:

```
Archetype:/Users/shared/Documents $ sudo chown norman:auditor ImportantDocument.d
```

6.2.1.2 chmod

The*chmod* tool, short for "change file modes," changes the permissions on a file. Its basic syntax is

```
chmod access-string file
```

where the *access-string* states the permissions you want to set for the given *file*. The access string has three parts to it: a letter (*u*,*g*,*w*,*a*), an operation code (+, -, =), and a permission (*r*,*w*,*x*). Table 6-3 gives a summary of what these codes stand for.

Table 6-3. Commonly-used chmod code summary

String Part	Code	Description
Who	u g o a	User Group Other All (default)
Operation	+ - =	Add permission Remove permission Assign permission (and remove permission of other fields)
Permission	r w x t	Read Write Execute (file) or search (directory) Sticky bit

For example, to give everybody write access for a file that you own:

```
$ chmod a+w ImportantDocument.doc
```

To enable execution of a file for everyone in the group that owns the file:

```
$ chmod g+x generatereport
```

To remove the ability to execute a file from all users:

```
$ chmod -x generatereport
```

To remove the ability to read a file from everyone but yourself:

```
$ chmod go-r ImportantDocument.doc
```

You can recursively change the permissions of a group of files by using the *-R* option. For example, you wanted to allow everybody to see all the documents in your Documents folder, you would use the following command:

```
$ chmod -R +r Documents
```

You can also use commas to set more than one permission at a time. For example:

```
$ chmod a-wx,a+r ImportantDocument.doc
```

Alternatively, you can specify the mode of a file in another; however, this is a more arcane syntax that you will see quite often. To understand this syntax, you have to think in terms of bits and octa notation. A typical mode in this syntax contains three digits—each corresponding to the three level of permission (user, group, other). Each digit is calculated by adding the following octal values:

4	Read
2	Write
1	Execute
0	No permissions

Some common number combinations for setting permissions are shown in Table 6-4. For example, you wanted to grant read and write permissions for a user and only read permissions for group anc others:

```
$ chmod 644 ImportantDocument.doc
```

Here, 644 indicates that the user who created the file has read-write privileges (add 4+2=6), and the group and others only get read-only access, as indicated by the 4s.

To grant all permissions to the user and deny all permissions to the group and others:

```
$ chmod 700 ImportantDocument.doc
```

Table 6-4. Commonly used chmod numeric sequences

Number	Result	Description
777	rwxrwxrwx	File can be read, written, or executed by anybody
775	rwxrwxr-x	File can be read and executed by anybody, but only written to by owner or group
755	rwxr-xr-x	File can be read and executed by anybody, but only written to by owner
666	rw-rw-rw	File can be read and written by anybody, but not execute
664	rw-rw-r--	File can be read by anybody, written to by owner or group and can't be executed
644	rw-r--r--	File can be read by anybody, written to by owner, and can't be executed
444	r--r--r--	File is read only

6.2.2 The Sticky Bit

The sticky bit has an odd history. A long time ago it meant an executable file should be kept in memory even after the process using it had exited. Most Unix implementations, including Mac OS X don't use it for this purpose anymore, but somewhere along the way it got coerced into another job When the sticky bit is set on a directory, users can only delete files in the directory that belongs to them, even if they have access rights to the directory because of a group membership.

The sticky bit is set using the user part of the permission string and appears at the end of the list o permissions in a file listing. For example, to turn on the sticky bit on a directory, you would execut

```
 $ chmod u+t /Financials
```

and it shows up in a directory listing like this:

```
 drwxr-xr-t  2 duncan  admin    68   8 Sep 01:10 Finance
```

A directory on the system that has the sticky bit set is */private/tmp*, as shown in Example 6-2.

Example 6-2. The /private directory showing the sticky bit on /private /tmp

```
$ls -la /private

total 0
```

```
drwxr-xr-x   5 root wheel  170 30 Nov 17:39 .
drwxrwxr-t  34 root admin 1156 30 Nov 17:40 ..
drwxr-xr-x 102 root wheel 3468 28 Nov 23:11 etc
drwxrwxrwt   7 root wheel  238  4 Dec 21:06 tmp
drwxr-xr-x  21 root wheel  714 30 Nov 17:41 var
```

Another directory that has the sticky bit set is the /Users/Shared directory, as shown in Example 6-3.

Example 6-3. The /Users directory showing the sticky bit on /Users/Shared

```
$ls -la /Users
total 0
drwxrwxr-t  5 root    admin    170 12 Nov 04:38 .
drwxrwxr-t 34 root    admin   1156 30 Nov 17:40 ..
-rw-r--r--  1 root    wheel      0 12 Sep 20:20 .localized
drwxrwxrwt  5 root    wheel    170 12 Nov 21:39 Shared
drwxr-xr-x 29 duncan  duncan   986 30 Nov 18:05 duncan
```

6.3 Type and Creator Codes

In addition to having an owner and a set of permissions, every file also has a set of HFS+ attributes assocated with it. The most interesting of these attributes are the *Type* and *Creator* codes.

A Type code is a four character string that can be assigned to a file when it is created and is used by the filesystem to denote what type of file it is. Type codes are exactly four characters in length and are case sensitive. Table 6-5 lists some common type codes. In this table, an open square (□) means that a space is used where the character should be (remember, Type codes have to be exactly four characters).

Table 6-5. Common Type codes

File Extension	Type Code	Type of File
.pdf	PDF□	Portable Document File
.doc	W8BN	Microsoft Word document
.xls	XLS8	Microsoft Excel document
.psd	8BPS	Adobe Photoshop document
.dmg	devi	Disk image
.mov	MooV	QuickTime movie
.jpeg, .jpg	JPEG	JPEG image file

A Creator code is similar to a Type code, except that it denotes the applications that was used to create the file. Like Type codes, they are also exactly four characters in length and are case sensitive.Table 6-6 lists some common creator codes.

Table 6-6. Common Creator codes

Creator Code	Application
prvw	Preview
MSWD	Microsoft Word
XCEL	Microsoft Excel
8BIM	Adobe Photoshop

To look at a file's Type and Creator codes, you can use the *GetFileInfo* command-line tool that is installed into */Developer/Tools* as part of the Xcode Tools package. Example 6-4 shows the use of*GetFileInfo* and the resulting output.

Example 6-4. Using GetFileInfo to look at the attributes of a file.

```
$GetFileInfo Artwork.psd

file: "Artwork.psd"

type: "8BPS"

creator: "8BIM"

attributes: avbstClinmed

created: 12/06/2003 20:51:13

modified: 12/06/2003 20:51:13
```

In looking at the results for the command, you can see that the file Type is `8BPS`, which means that its a Photoshop document, and the creator code is `8BIM`, which means that it was created by Adobe Photoshop.

Type and Creator codes are used by the Finder to determine the type of a document and the application that should be used to open it. You can see this information in the Finder by using the File→Get Info menu (⌘-I).

6.4 Further Explorations

To get deeper into the subjects in this chapter, you should refer to the following books:

- *Mac OS X Panther in a Nutshell*, by Chuck Toporek, et al. (O'Reilly & Associates, Inc., 2004)
- *Learning Unix for Mac OS X Panther*, by Dave Taylor, et al. (O'Reilly & Associates, Inc., 2004)

You may also want to check out the following manpages:

- *ls*
- *chown*
- *chmod*
- *sticky*
- *GetFileInfo*

Chapter 7. Monitoring the System

The Mac, as well crafted as it is, hides a lot of complexity under the hood. This is unavoidable given what it does. Much to Apple's credit, the Mac works a lot smoother than most computers, but it's still a complex beast underneath the Aqua interface. In a lot of ways, the Mac is similar to a smooth running car. There's a set of well laid out controls, including a steering wheel, accelerator, and brake, but underneath the hood, it's a bit more complicated than that. There's system after system that work together to make it all happen.

This chapter shows you how to get under the hood and find out what kind of hardware, such as how much RAM and the kind of hard drives, is in your Mac. It also shows you how to view the software that is installed and how to watch the software running.

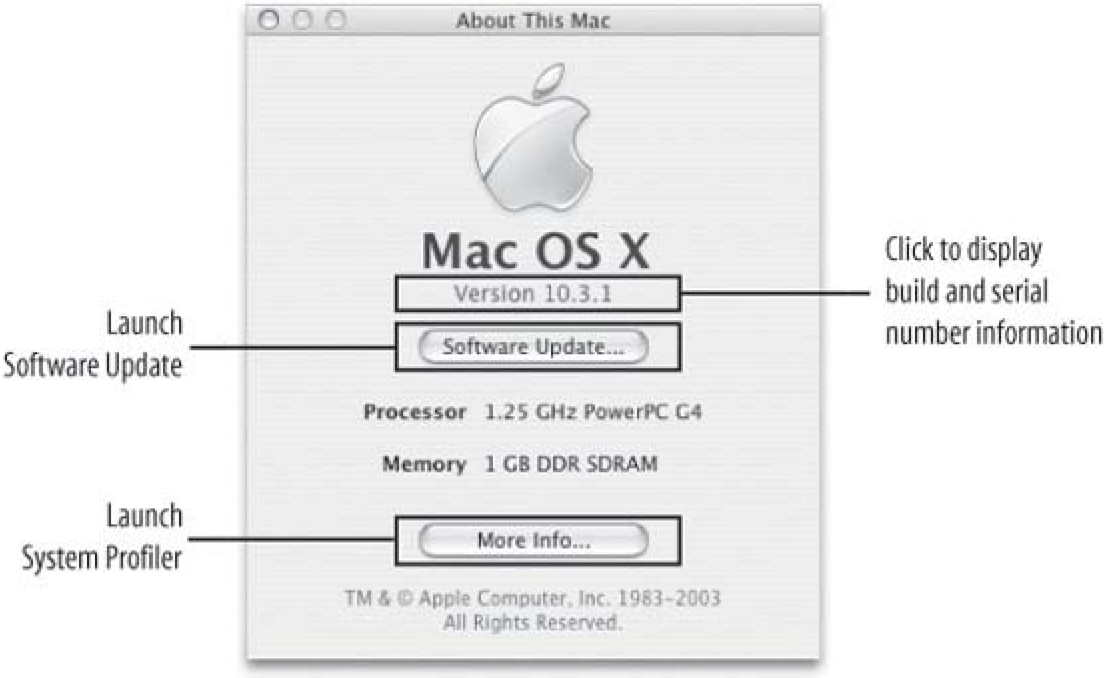
7.1 About This Mac

The first place to go to get information about your system is the → About This Mac menu, which brings up the panel shown in Figure 7-1. This simple panel shows you the kind of processor (or processors) and the amount of memory you have installed in your Mac. You can even find out the exact build of Panther you are running. Click the Version string in the panel, and it will change to the build number, such as 7C107. Click again, and you'll get the serial number of your system—a valuable piece of information when you are on hold with AppleCare. But the information here is only the surface details. Click the More Info button and System Profiler will launch.

7.2 System Profiler

The System Profiler (*/Applications/Utilities*), formerly known as the Apple System Profiler, gives you a great deal of insight into the kind of hardware that is installed and connected to your system and the software it's running. It gives fast answers to the questions: "How much memory do I have?" or "What version of Safari do I have installed?". In fact, it does such a thorough job that a System Profiler report is often the first thing Apple's support folks will ask you for when you have a problem or report a bug. Figure 7-2 shows what System Profiler looks like when you first launch it.

Figure 7-1. The About This Mac panel



Processor 1.25 GHz PowerPC G4

Memory 1 GB DDR SDRAM

TM & © Apple Computer, Inc. 1983-2003
All Rights Reserved.

Click to display build and serial number information

Launch Software Update

Launch System Profiler

The column on the left side is divided into four principal parts:

Hardware

Gives a detailed listing of the hardware in and connected to your system including memory, PCI and AGP cards, ATA and SCSI devices (such as hard drives), USB and FireWire connected devices, AirPort card, and modem. Each item displays all the information known to the system about the device such as model numbers and firmware revisions.

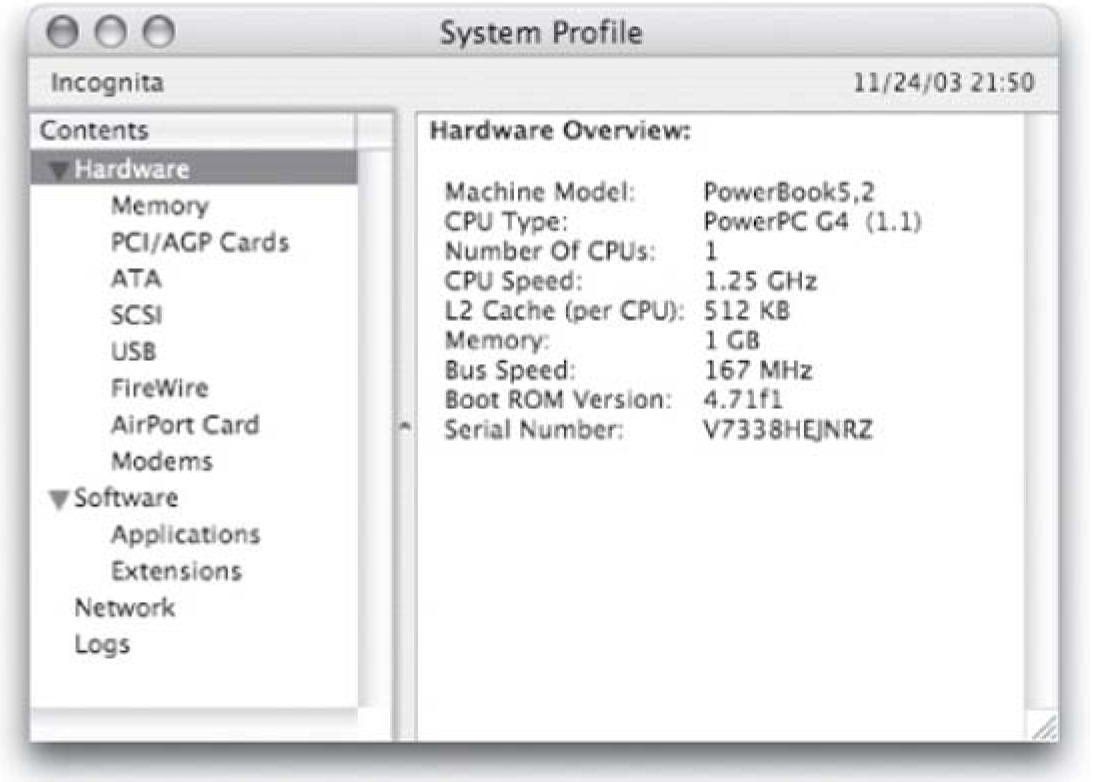
Running Software Update

There are three ways to run Software Update on your Mac:

- By clicking on the Software Update button in the About This Mac window
- By using the Software Update preference panel
- From the command line, using the *software_update* command

You can also configure Software Update to automatically check for and download updates from the Software Update preference panel.

Figure 7-2. The System Profiler main window



Hardware Overview:

```
Machine Model:        PowerBook5,2
CPU Type:             PowerPC G4  (1.1)
Number Of CPUs:       1
CPU Speed:            1.25 GHz
L2 Cache (per CPU):   512 KB
Memory:               1 GB
Bus Speed:            167 MHz
Boot ROM Version:     4.71f1
Serial Number:        V7338HEJNRZ
```

Software

Provides a detailed list of the applications in the */Applications* folder as well as the frameworks and kernel extensions on your system. Each item includes the name of the software component, its version number, and the date on which it was last modified.

Network

Gives a complete list of all the network connections on your machine and the configuration information, such as IP address, subnet mask, router address, and Ethernet address for each connection.

Logs

Gives access to the console and system logs for the system. The console gives information about events that are occurring for the currently logged-in user. The system log provides information about events at the system level.

The amount of information that the System Profiler gives you access to can be overwhelming. Thankfully, the interface does a much better job than its predecessor did in previous versions of Mac OS X by letting you drill down to the information you want to see.

7.2.1 Creating a System Profile Report

Occasionally you may be asked by Apple or some other software company to send a report from System Profiler so they can see the details of your system and troubleshoot a problem for you.

Refreshing the System Profiler View

A frequent "problem" that people have with System Profiler is when they connect or disconnect a USB or FireWire device and don't see a change in the System Profiler window. This is because System Profiler doesn't continuously poll the system to detect changes as they happen. Once it has collected the data, its job is to display it. If you have a System Profiler window open and make changes to your hardware configuration that you want to see, use the View→Refresh (⌘-R) menu.

There are a variety of ways to generate this information:

- Save an XML document containing all the information in a format that can be opened later with the System Profiler application. To do this, use the same File→Save (⌘-S) command that you would in any other application. This is Apple's preferred format for receiving system profiles.

- Export the data to a plain text or rich text file (RTF) using the File→Export menu.

- Print the file, but instead of sending it to your printer, save it as a PDF file using File→Print →Save as PDF.

To vary the detail level of the information in the report (as well as the size of the files generated), you can choose the level of detail you want to display using the View menu and then save, export, or print the report. When sending problem reports to Apple, it is best to err on the side of sending too much information rather than too little, however, you might want to be careful if you are actually printing out the reports—a full, extended report can run upwards of 75 pages.

7.2.1.1 System Profiler on the command line

The*system_profiler* command-line tool allows you to get the same information from the command line that System Profiler gives access to. When run with no arguments, system_profiler generates a full report, the beginning of which is shown in Example 7-1.

Example 7-1. The start of output from system_profiler

```
$system_profiler

Hardware:


   Hardware Overview:


        Machine Model: iMac

        CPU Type: PowerPC G4 (3.3)

        Number Of CPUs: 1

        CPU Speed: 1 GHz

        L2 Cache (per CPU): 256 KB

        Memory: 768 MB

        Bus Speed: 133 MHz

        Boot ROM Version: 4.59f1

        Serial Number: W831520KNHX
 ...
```

Submitting Bug Reports to Apple

One of the most valuable things you can do when something goes wrong with Mac OS X—and is a heck of a lot more productive than just complaining to your friends—is to file a bug with Apple. The easiest way to send feedback is to use the form on Apple's web site at *www.apple.com/macosx/feedback/*. Even better is to use the Apple Developer Connection's bug report site at *bugreport.apple.com/*. You'll have to have an ADC membership (free for the lowest tier) to use this form, but in return you'll be able to track the progress of your issue after you submit it.

The full report that *system_profiler* outputs is very long indeed. You might want to save it as a file so that you can view it using a text editor, or so that you can send it in to Apple or another software vendor as part of a bug report. To do this, use the following command:

```
$ system_profiler > ~/Desktop/MySystemProfile.txt
```

> You could take this tip a step forward and add it to your *crontab* (see Chapter 8,*Scheduling Tasks*) so that you always have an up-to-date inventory of your system. System administrators can use this tip to keep track of machines across their networks.

To limit the amount of information that *system_profiler* generates, you can pass a data type as an argument. You can get the list of data types for your system by using the *-listDataTypes* argument as shown in Example 7-2.

Example 7-2. Listing the data types that system_profiler can give data about

```
$ system_profiler -listDataTypes

Available Datatypes:

SPHardwareDataType

SPSoftwareDataType

SPNetworkDataType

SPMemoryDataType

SPPCIDataType
```

```
SPIDEDataType
SPSCSIDataType
SPApplicationsDataType
SPUSBDataType
SPFireWireDataType
SPAirPortDataType
SPFrameworksDataType
SPModemDataType
SPExtensionsDataType
SPLogsDataType
```

Armed with this information, if you wanted to see the status of the network connections of your Mac, you could use the *SPNetworkDataType* argument as shown in Example 7-3.

Example 7-3. Using system_profiler to get network connection data

```
$ system_profiler SPNetworkDataType
Network:

    Built-in Ethernet:

      Interface: en0
      Type: Ethernet
      Ethernet Address: 00:0a:95:99:e4:92

    AirPort:
      Interface: en1
      Type: AirPort
      IP Address: ("192.168.1.102")
```

```
Subnet Mask: ("255.255.255.0")

Broadcast Address: ("192.168.1.255")

Router Address: 192.168.1.1

DNS Servers: ("66.93.174.29", "66.93.87.2", "216.231.41.2")

Domain: x180.net

Ethernet Address: 00:03:93:ef:ba:a5
```

7.3 Monitoring System Activity

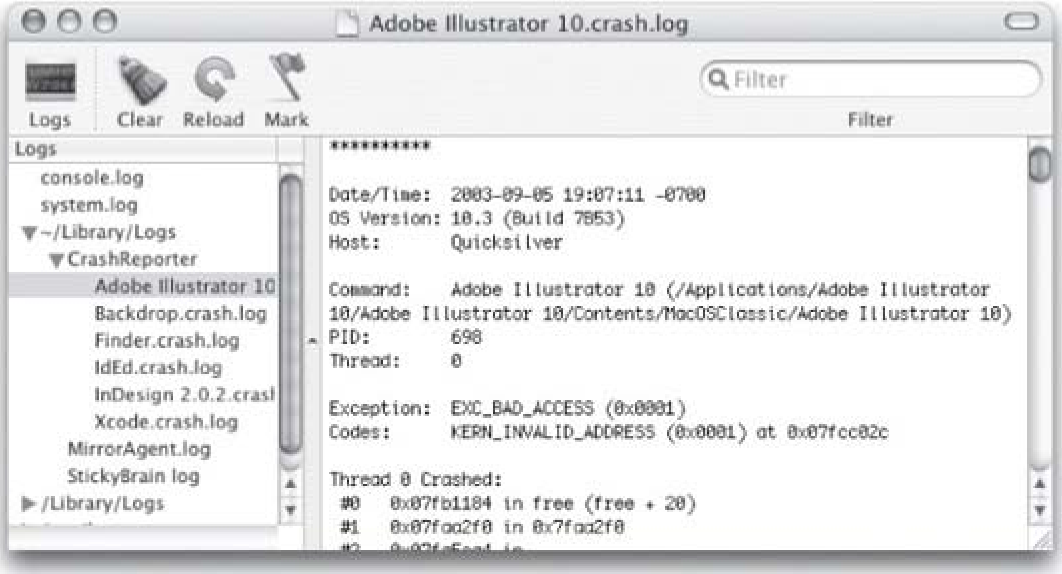
Once you know what is installed on your system, you need to be able to find out information about the system is running. The System Profiler report gives access to two logs (console and system) th provide a little bit of information, but two tools, Console and Activity Viewer, can provide quite a b information.

7.3.1 Console

The Console application (*/Applications/Utilities*), as shown in Figure 7-3, is the über utility to view log files on your computer. When you launch Console, it gathers the system logs and any logs from Home directory and displays a tree to navigate between the logs. If you don't see the left-hand tre click the Logs menu bar item (or use the View→Show Log List menu).

The first two logs you'll see are the same system and console logs that are available in System Pro that's just the warm up. The tree also gives you the following areas in which to see the logs:

Figure 7-3. The Console application



```
**********

Date/Time:  2003-09-05 19:07:11 -0700
OS Version: 10.3 (Build 7053)
Host:       Quicksilver

Command:    Adobe Illustrator 10 (/Applications/Adobe Illustrator
10/Adobe Illustrator 10/Contents/MacOSClassic/Adobe Illustrator 10)
PID:        698
Thread:     0

Exception:  EXC_BAD_ACCESS (0x0001)
Codes:      KERN_INVALID_ADDRESS (0x0001) at 0x07fcc02c

Thread 0 Crashed:
 #0   0x07fb1184 in free (free + 20)
 #1   0x07faa2f0 in 0x7faa2f0
```

~/Library/Logs

Contains the logs that are associated with a user. These include the various CrashReporter lo which give the gory details of any application crashes that you might have experienced. You' find the MirrorAgent log detailing the activities of synchronizing your iDisk.

/Library/Logs

Contains logs that are associated with the system at large. For example, you'll find CrashRep reports on components that are being run by the system (instead of applications that you rui if you are unlucky enough to experience a kernel panic, the log for it will be here.

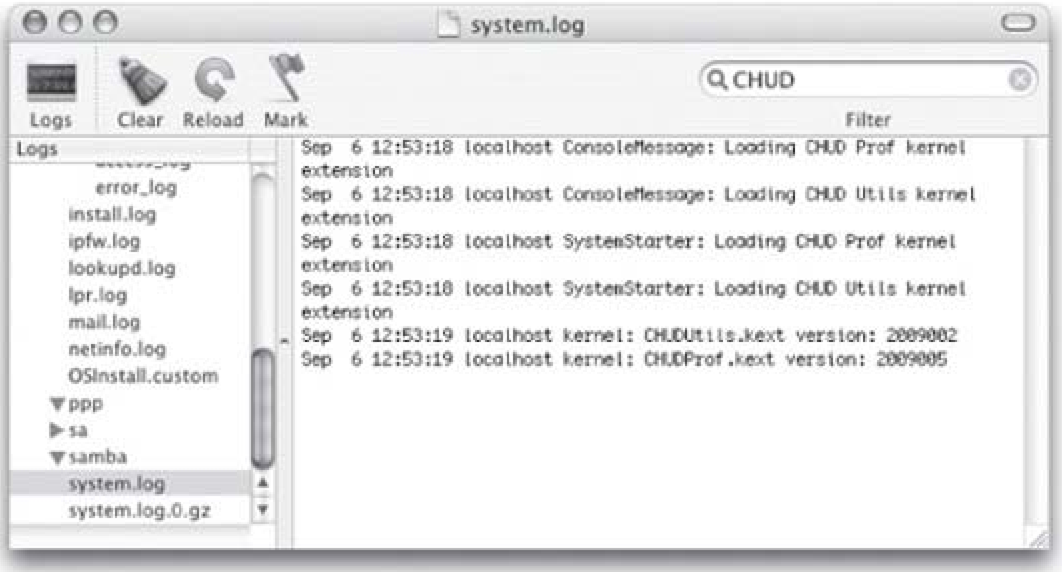
/var/log

Contains the logs of the various BSD Unix utilities running on your system. For example, you the logs for the Apache *httpd* web server and the *ipfw* firewall here.

With such an extreme amount of data at your fingertips, how can you sort through it all? This is wh Apple's use of the Filter box in the menu bar comes in handy. Figure 7-4 shows an example of the searching for any lines containing CHUD in the *system.log*. The quick filtering abilities of the Conso make it easy for you to look for needles in a haystack. A great use of this feature is to scan your we to search for requests from a particular IP address.

Another useful tool in the toolbar is the Mark button, which inserts a separator of text, along with t current date and time, into the log view. This allows you to bring up the console, make a mark in a something with another part of the system, and then come back to see what has happened since y the mark.

Figure 7-4. Filtering for CHUD in the system log



```
Sep  6 12:53:18 localhost ConsoleMessage: Loading CHUD Prof kernel
extension
Sep  6 12:53:18 localhost ConsoleMessage: Loading CHUD Utils kernel
extension
Sep  6 12:53:18 localhost SystemStarter: Loading CHUD Prof kernel
extension
Sep  6 12:53:18 localhost SystemStarter: Loading CHUD Utils kernel
extension
Sep  6 12:53:19 localhost kernel: CHUDUtils.kext version: 2009002
Sep  6 12:53:19 localhost kernel: CHUDProf.kext version: 2009005
```

7.3.2 Activity Monitor

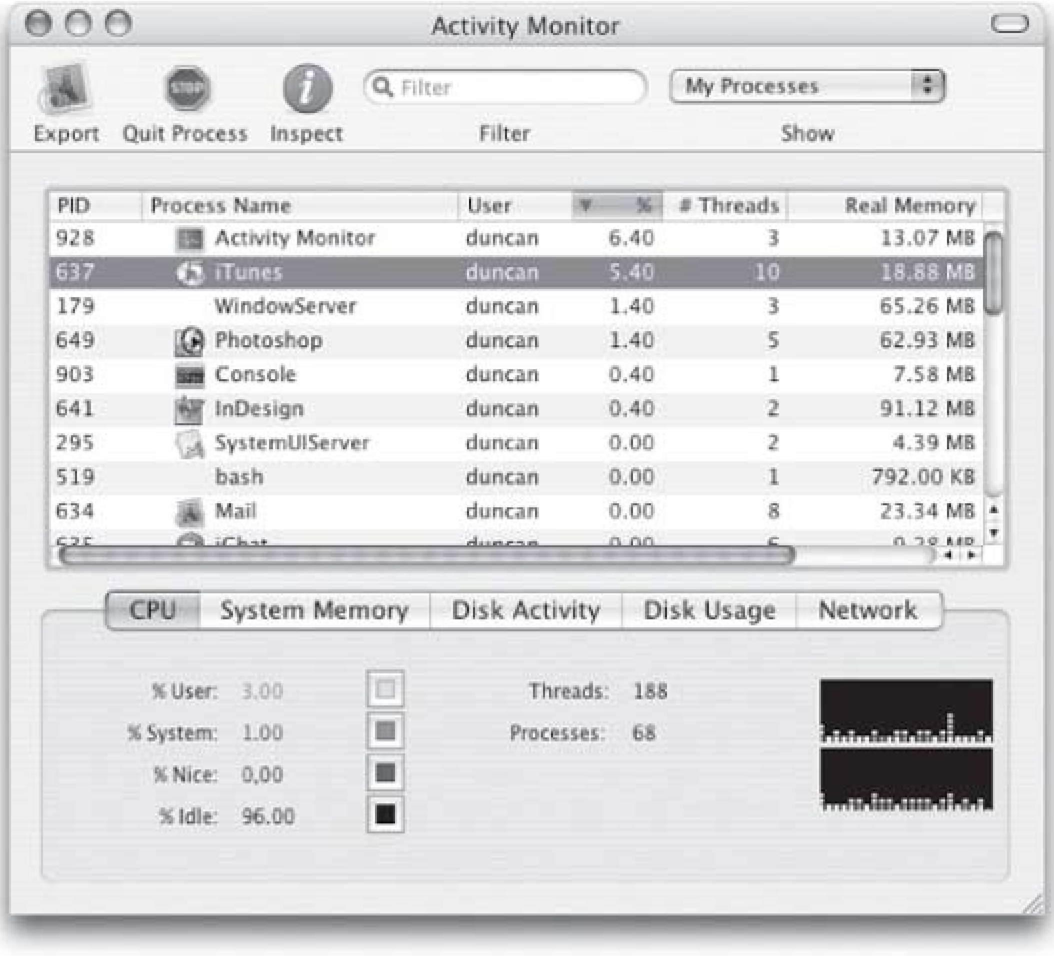
To view what's going on with your system right now, nothing beats the Activity Monitor (*/Applications/Utilties*) in Panther, shown in Figure 7-5. Activity Monitor gives realtime measureme resources in use on the system, the processes that are taking up the most memory or CPU cycles, a systems disk and network activity. The version of Activity Monitor in Panther consolidates the CPU and Process Viewer applications that shipped in previous versions of Mac OS X.

The top part of the Activity Monitor window contains a table list of the processes on your system al the various attributes of those processes. Unlike the Dock or the →Force Quit panel, here you ca every process running on your system. The processes in the main window can be filtered either wit text box or the pull-down menu in the menu bar.

The "All Processes, Hierarchically" option provides an interesting view. Using this view, you can see the processes on your system are owned by the *init* process discussed in Chapter 4,*System Startup Login*. You can also see how all the GUI applications that you have running are owned by the windo server processes and, if you have a Terminal window open, you can see the various processes own including the *bash* shell.

Not only can you monitor every application on the system from the Activity Monitor, but you can al terminate applications. When you select a program and then press the Quit Process button (or use Process→Quit menu), a dialog sheet will appear giving you the option of politely quitting the applic (the same as going to that application and using the Filre→Quit menu) or using Force Quit.

Figure 7-5. The Activity Monitor



Show: My Processes

PID	Process Name	User	%	# Threads	Real Memory
928	Activity Monitor	duncan	6.40	3	13.07 MB
637	iTunes	duncan	5.40	10	18.88 MB
179	WindowServer	duncan	1.40	3	65.26 MB
649	Photoshop	duncan	1.40	5	62.93 MB
903	Console	duncan	0.40	1	7.58 MB
641	InDesign	duncan	0.40	2	91.12 MB
295	SystemUIServer	duncan	0.00	2	4.39 MB
519	bash	duncan	0.00	1	792.00 KB
634	Mail	duncan	0.00	8	23.34 MB
635	iChat	duncan	0.00	6	0.38 MB

% User: 3.00
% System: 1.00
% Nice: 0.00
% Idle: 96.00

Threads: 188
Processes: 68

7.3.2.1 Figuring out what a process does

It's pretty easy to figure out what many of the processes that show up in Activity Viewer do, partic with applications like Safari or Photoshop which show up with their icon and name. Others such as *mach_init*, or *netinfod*, however, aren't as easy to figure out simply by looking at them. These are the processes that take care of various system functions. If you are curious about what a particula is, try looking at its manpage. For example, the manpage for *netinfod* will tell you that it is the Net database daemon.

7.3.2.2 Interpreting process information

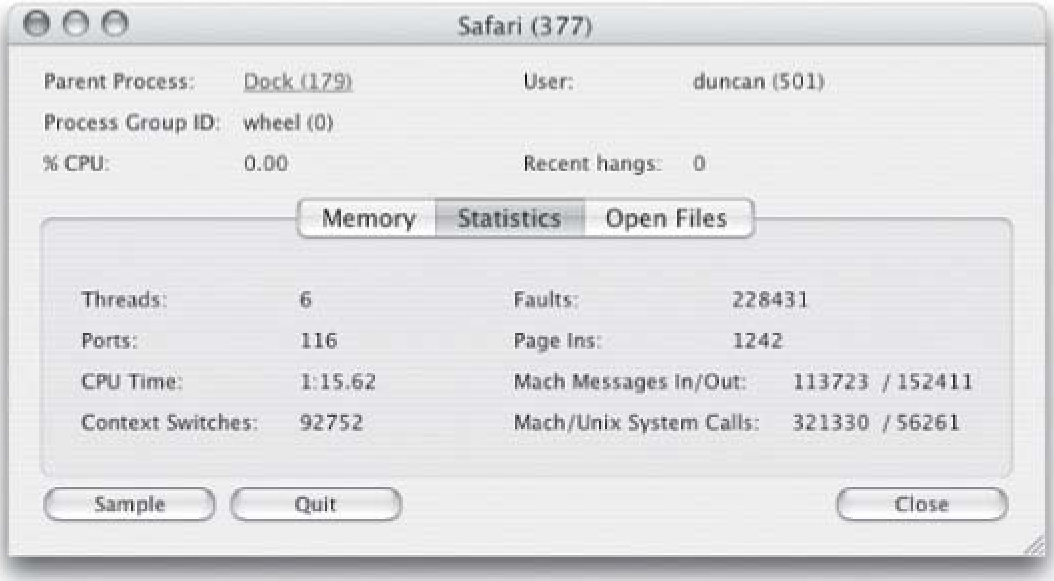
The impact of a process on your system can be in memory consumed, processor activity, or in the of disk I/O that it causes. Some processes can be running without using any processor time and ca consume large amounts of memory. Others can consume your processor without any impact on me usage. The process view gives quite a bit of this information to you all at once. Table 7-1 lists the columns in the top part of the window and what they mean. The most useful columns to use are th and Real Memory columns. If you notice your system is sluggish, click the %CPU heading to find o process is using the most processing power.

Table 7-1. Process attributes reported in the Activity Monitor main win

Attribute	Description
PID	The numerical id of the process
Process Name	The name of the process
User	The user that this program belongs to
% CPU	The amount of CPU time that the process is taking on your sys
# Threads	The number of threads that the process is running
Real Memory	The amount of physical memory used by the process
Virtual Memory	The total amount of memory addressed by the process

To get more detail on a process, select the process in the top part of the window and then hit the I button or use the Process→Inspect (⌘-I) menu. This brings up a detailed Inspector window that g more information about the process such as the Mach ports in use by the application, the number messages being sent via the Mach ports, the files that the application is using, and the various me statistics, such as faults and page-ins for the process, as shown in Figure 7-6. Many of these statis only of use to developers, but they can give you a good idea of the kinds of resources that your applications are using.

Figure 7-6. Inspecting the Safari process



Safari (377)

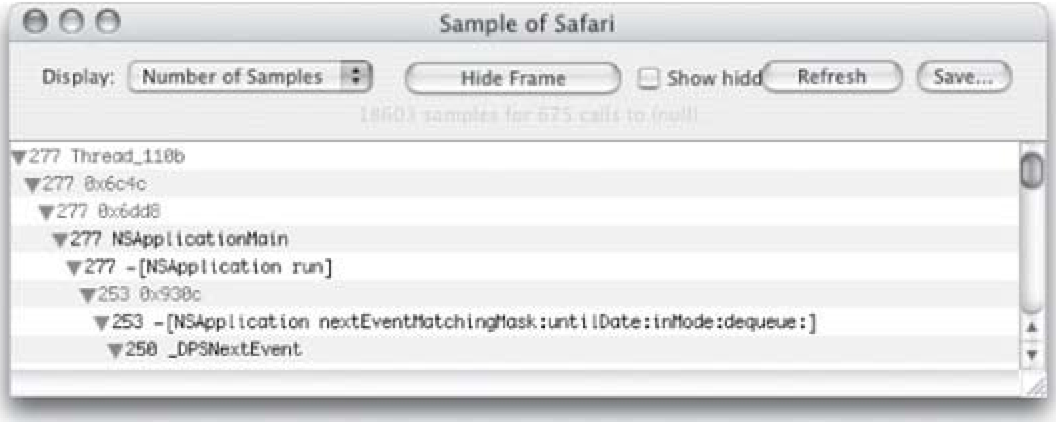
Parent Process: Dock (179)
User: duncan (501)
Process Group ID: wheel (0)
% CPU: 0.00
Recent hangs: 0

Threads: 6
Faults: 228431
Ports: 116
Page Ins: 1242
CPU Time: 1:15.62
Mach Messages In/Out: 113723 / 152411
Context Switches: 92752
Mach/Unix System Calls: 321330 / 56261

7.3.2.3 Sampling an application

The Process Inspector window gives access to a powerful feature that is of interest mainly to devel who want to take a peek into the inner operation of an application. From the Inspector, hit the San button and another window pops open (shown in Figure 7-7), allowing you to see the call stack of application, the amount of time spent in each function of an application, and other related informat only a developer could want to see. You can also sample a process from the main Activity Monitor by using the Process→Sample (Option-⌘-S) menu.

Figure 7-7. Sample of Safari running, as shown in the Process Viewer sar window



Sample of Safari

Display: Number of Samples

18603 samples for 675 calls to (null)

```
▼277 Thread_110b
 ▼277 0x6c4c
  ▼277 0x6dd8
   ▼277 NSApplicationMain
    ▼277 -[NSApplication run]
     ▼253 0x930c
      ▼253 -[NSApplication nextEventMatchingMask:untilDate:inMode:dequeue:]
       ▼250 _DPSNextEvent
```

7.3.3 Monitoring Processes from the Command Line

As good as the Console and Activity Monitor are, sometimes the only tool for a job is a command-li particularly when you are logged in remotely to a machine via *ssh* (the Secure Shell). Fortunately, are an abundance of good tools available on the command line, many of which served to inspire th tools in Mac OS X.

7.3.3.1 Working with log files

To work with log files on the command line, your primary tools of choice will be *grep* and *tail*. Nam "get regular expression," *grep* allows you to quickly search and return lines in a file that contain a of text. Think of it as the command-line version of the Filter box on the Console application (althou predates the Filter box by a few decades). Example 7-4 shows the use of *grep* to perform a search term CHUD in the system log (the same search as before, giving you a point of comparison).

Example 7-4. Searching the system log for the pattern CHUD

```
$ grep CHUD /var/log/system.log

Dec 8 12:53:18 localhost ConsoleMessage: Loading CHUD Prof kernel extension

Dec 8 12:53:18 localhost ConsoleMessage: Loading CHUD Utils kernel extension

Dec 8 12:53:18 localhost SystemStarter: Loading CHUD Prof kernel extension

Dec 8 12:53:18 localhost SystemStarter: Loading CHUD Utils kernel extension

Dec 8 12:53:19 localhost kernel: CHUDUtils.kext version: 2009002

Dec 8 12:53:19 localhost kernel: CHUDProf.kext version: 2009005
```

grep gives you quite a bit more power than just the ability to search using a simple term. It actuall you the ability to use any regular expression. For example, if you wanted to find all the lines that contained the words CHUD and kernel (in that order), you could use the `CHUD.*kernel` regex, whic human means "look for the term CHUD, then some number of any kind of characters followed by tl kernel".Example 7-5 shows the result of this regex. You can see that only the lines with both the w CHUD and kernel appear.

Example 7-5. Using a more detailed regexp with grep

```
$ grep CHUD.*kernel /var/log/system.log

Dec 8 12:53:18 localhost ConsoleMessage: Loading CHUD Prof kernel extension

Dec 8 12:53:18 localhost ConsoleMessage: Loading CHUD Utils kernel extension

Dec 8 12:53:18 localhost SystemStarter: Loading CHUD Prof kernel extension

Dec 8 12:53:18 localhost SystemStarter: Loading CHUD Utils kernel extension
```

To monitor what's changing in a log file, and instead of trying to read through the whole log file frc top, Unix gives you the *tail* command for looking at just the last part of a file. Example 7-6 shows t of*tail* on the system log after unplugging and plugging in a few USB devices.

Regular Expressions

Regular expressions (also known as *regex's*) is a small, highly specialized language for processing text strings. They can be a bit scary to use at first because of their extremely concise syntax which makes them look confusing and complicated to the uninitiated. Here are a few of the most commonly used characters in regular expressions:

Character	Description
.	Matches any single character
*	Matches zero or more occurrences of the character immediately preceding it
^	Matches the beginning of a line
$	Matches the end of a line
\	Used to quote the following character, for example to find a $ character

To learn all about regular expressions, you should pick up *Mastering Regular Expressions*,*2nd Edition*, by Jeffrey E. F. Friedl (O'Reilly & Associates, Inc., 2002).

Example 7-6. The tail command applied to the system log

```
$ tail /var/log/system.log
Dec 8 14:15:06 localhost kernel: USB Low Power Notice: The hub "Hub in Apple USB
Keyboard" cannot be used because there is not enough power for all its ports
Dec 8 14:15:06 localhost kernel: USBF: 169.316 AppleUSBHub[0x2755800]: insufficien
to turn on ports
Dec 8 14:15:06 localhost kernel: USBF: 169.316 AppleUSBHub[0x2755800]::ConfigureHu
PortPowerRequirements failed with 0xe00002e3
Dec 8 14:15:06 localhost kernel: USBF: 169.316 AppleUSBHub[0x2755800]::start Abort
```

```
startup: error 0xe00002e3
Dec 8 14:15:34 localhost kernel: USBF: 196.900 [0x2ae1000]::start - USB HID Interf
of device USB-PS/2 Mouse M-BA47 @ 7 (0x18222000)
Dec 8 14:15:55 localhost kernel: USBF: 217.924 [0x2564200]::start - USB HID Interf
of device USB-PS/2 Mouse M-BA47 @ 7 (0x18222000)
Dec 8 14:16:23 localhost kernel: USBF: 245.467 [0x239d600] USB Generic Hub @ 4
(0x18210000)
Dec 8 14:16:23 localhost kernel: USBF: 245.836 [0x2564000]::start - USB HID Interf
of device Apple Extended USB Keyboard @ 6 (0x18211000)
Dec 8 14:16:23 localhost kernel: USBF: 245.882 [0x23b9d00]::start - USB HID Interf
of device Apple Extended USB Keyboard @ 6 (0x18211000)
Dec 8 14:16:23 localhost kernel: USBF: 245.934 [0x2ae2800]::start - USB HID Interf
of device USB-PS/2 Mouse M-BA47 @ 7 (0x18212000)
```

By default, *tail*displays the last 10 lines of the file you apply it to. To change this behavior, you can the -*n* option to specify how many lines to go backwards into the file. Example 7-7 shows this in ac

Example 7-7. Using the -n argument of tail

```
$ tail -n1 /var/log/http/error.log
[Sun Nov 30 17:40:18 2003] [notice] Accept mutex: flock (Default: flock)
```

If you want to see the system log as it changes (for example, if you want to monitor the system lo you change USB devices and then execute a command with *sudo*), you can use the -*f* option to *tail*, shown in Example 7-8.

Example 7-8. Using tail -f on the system log

```
$ tail -f -n0 /var/log/system.log
Dec 8 14:19:09 localhost kernel: USBF: 411.832 [0x2ae2300]::start - USB HID Interf
of device USB-PS/2 Mouse M-BA47 @ 7 (0x18212000)
```

```
Dec 8 14:23:10 localhost sudo: duncan : TTY=ttyp2 ; PWD=/Users/duncan ; USER=root
COMMAND=/bin/ls
```

To exit from *tail* when running with the -*f* option, hit Control-C.

7.4 Working with Processes

The*ps* command (short for *process status*) displays information about the various processes on your system on the command line. However, if you just execute *ps* by itself on the command line, you'll see only the process information about the shell process you are running. For information about all the processes that belong to you, use the *ps -x* command as shown in Example 7-9.

Example 7-9. Listing all the processes that belong to a user

```
    $ ps -x
  PID  TT   STAT  TIME COMMAND
  177  ??   Ss    0:48.59 /System/Library/Frameworks/ApplicationServices.framew
  180  ??   Ss    0:00.77 /System/Library/Frameworks/ApplicationServices.framew
  184  ??   Ss    0:01.32 /System/Library/CoreServices/loginwindow.app/Contents
  266  ??   Ss    0:00.43 /System/Library/CoreServices/pbs
  290  ??   S     0:02.62 /System/Library/CoreServices/Dock.app/Contents/MacOS/
  297  ??   S     0:00.93 /System/Library/CoreServices/SystemUIServer.app/Conte
  298  ??   S     0:28.74 /System/Library/CoreServices/Finder.app/Contents/MacO
  315  ??   Ss    0:02.27 /System/Library/CoreServices/MirrorAgent.app/Contents
  340  ??   Ss    0:00.04 /sbin/mount_webdav -o noautomounted -o nobrowse -a7 -
  341  ??   S     0:02.19 /Applications/iChat.app/Contents/MacOS/iChat -psn_0_9
  342  ??   Ss    0:00.43 /System/Library/PrivateFrameworks/InstantMessage.fram
  345  ??   S     2:48.70 /Applications/Adobe InDesign 2.0/InDesign 2.0.2 /Appl
  347  ??   S     1:09.86 /Applications/iTunes.app/Contents/MacOS/iTunes -psn_0
  349  ??   R     0:11.25 /Applications/Utilities/Terminal.app/Contents/MacOS/T
  369  ??   S     0:00.44 /Applications/Backdrop.app/Contents/MacOS/Backdrop -p
  373  ??   S     0:22.34 /Applications/Adobe Photoshop 7/Adobe Photoshop 7.0/C
  351  std  S     0:00.11 -bash
```

When*ps* is combined with *grep*, you can find just the process you are looking for. Example 7-10 shows a command to return the process information for Photoshop.

Example 7-10. Looking for Photoshop using ps and grep

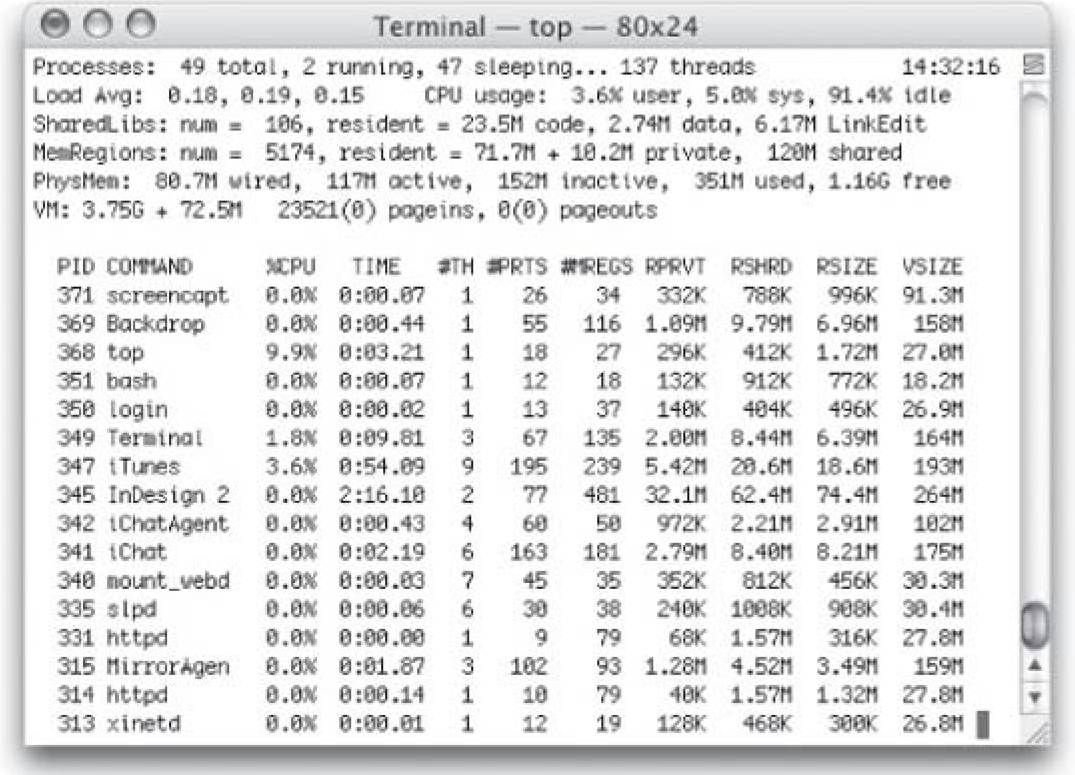
```
    $ ps -x | grep Photoshop
  373  ??  S    0:23.76 /Applications/Adobe Photoshop 7/Adobe Photoshop 7.0/C
  382 std  U+   0:00.01 grep Photoshop
```

To see all the processes running on the system, use the -*ax* option. This produces quite a bit of content, so it's best to use this option with *grep* to narrow down the results to only what you want to see. For example, if you want to see all the processes running as root, you would use:

```
      $ ps -aux | grep root
```

Another useful command for monitoring your system is *top*. In fact, it's almost certain that *top* provided the inspiration for the Activity Monitor's process view. When you execute *top*, your Terminal window will fill up with a list of processes as well as the percentage of processor time and memory they are using, as shown in Figure 7-8. Invariably, there will be too much information to fit on the window so you might want to resize your Terminal window to see more information before you issue the *top* command. When you are done with *top*, use Control-C to quit or use the Mac standard ⌘-. command.

Figure 7-8. top running in a Terminal window



```
Processes:  49 total, 2 running, 47 sleeping... 137 threads          14:32:16
Load Avg:  0.18, 0.19, 0.15     CPU usage:  3.6% user, 5.0% sys, 91.4% idle
SharedLibs: num =  106, resident = 23.5M code, 2.74M data, 6.17M LinkEdit
MemRegions: num =  5174, resident = 71.7M + 10.2M private,  120M shared
PhysMem:  80.7M wired,  117M active,  152M inactive,  351M used, 1.16G free
VM: 3.75G + 72.5M   23521(0) pageins, 0(0) pageouts

  PID COMMAND      %CPU   TIME   #TH #PRTS #MREGS RPRVT  RSHRD  RSIZE  VSIZE
  371 screencapt   0.0%  0:00.07   1    26    34   332K   788K   996K  91.3M
  369 Backdrop     0.0%  0:00.44   1    55   116  1.09M  9.79M  6.96M   158M
  368 top          9.9%  0:03.21   1    18    27   296K   412K  1.72M  27.0M
  351 bash         0.0%  0:00.07   1    12    18   132K   912K   772K  18.2M
  350 login        0.0%  0:00.02   1    13    37   140K   404K   496K  26.9M
  349 Terminal     1.8%  0:09.81   3    67   135  2.00M  8.44M  6.39M   164M
  347 iTunes       3.6%  0:54.09   9   195   239  5.42M  20.6M  18.6M   193M
  345 InDesign 2   0.0%  2:16.10   2    77   481  32.1M  62.4M  74.4M   264M
  342 iChatAgent   0.0%  0:00.43   4    60    50   972K  2.21M  2.91M   102M
  341 iChat        0.0%  0:02.19   6   163   181  2.79M  8.40M  8.21M   175M
  340 mount_webd   0.0%  0:00.03   7    45    35   352K   812K   456K  30.3M
  335 slpd         0.0%  0:00.06   6    30    38   240K  1008K   908K  30.4M
  331 httpd        0.0%  0:00.00   1     9    79    68K  1.57M   316K  27.8M
  315 MirrorAgen   0.0%  0:01.87   3   102    93  1.28M  4.52M  3.49M   159M
  314 httpd        0.0%  0:00.14   1    10    79    40K  1.57M  1.32M  27.8M
  313 xinetd       0.0%  0:00.01   1    12    19   128K   468K   300K  26.8M
```

7.4.1 Killing Processes

Unfortunately, there are times when you have to manually intervene and end the execution of a process. Maybe it is a buggy program that has stopped accepting user input or maybe the program is just consuming a huge dataset and you've decided you don't want to wait. Or as an administrator, you might need to log in to a remote system to kill a user's errant process or even reboot a server. Whatever the reason, here are the ways in which you can stop a process's execution.

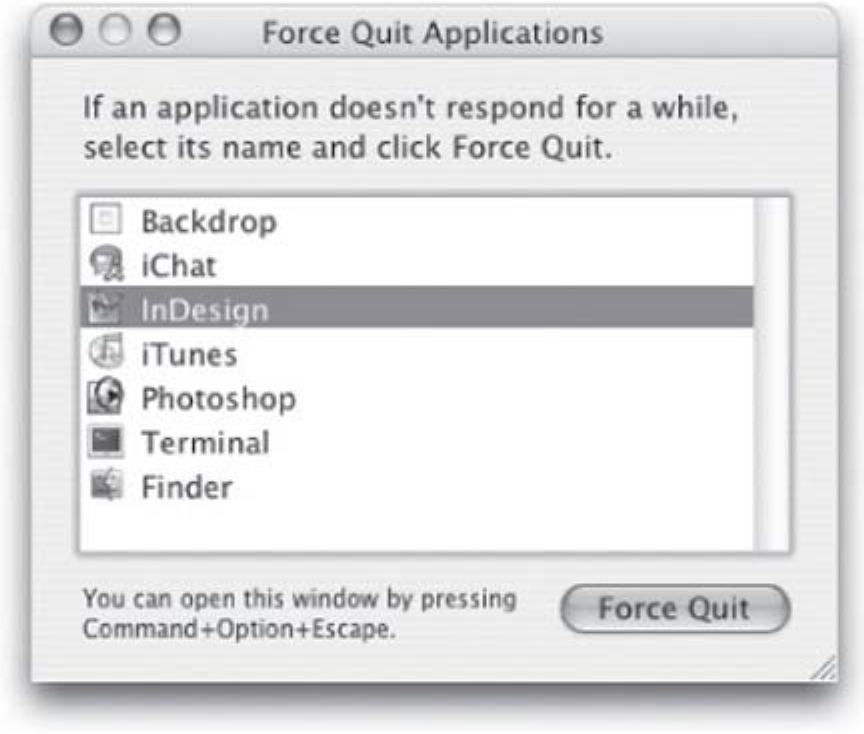
7.4.1.1 Force Quit

Using the Force Quit Applications window in Mac OS X, shown in Figure 7-9, is the easiest way to kill off an application while logged in to the system. You can get to it by using the →Force Quit menu or using the Option-⌘-Escape keystroke. Simply select the application that you want to kill off and hit the Force Quit button.

> Only GUI applications will show up in the Force Quit dialog box. To quit applications that aren't GUI applications, use the Activity Monitor as described in the next section.

You can also force quit an application using its icon on the Dock. Control-click the icon (or click and hold it down on for a second or two). A contextual menu will pop up. Press the Option key to turn the Quit menu item to Force Quit.

Figure 7-9. The Force Quit Applications window


Force Quit Applications

If an application doesn't respond for a while, select its name and click Force Quit.

- Backdrop
- iChat
- InDesign
- iTunes
- Photoshop
- Terminal
- Finder

You can open this window by pressing Command+Option+Escape.

7.4.1.2 Using the Activity Monitor

The Activity Monitor also provides a way to force applications to quit, including those processes that aren't visible on the display. To quit an application this way, simply highlight the process and then use the Quit Process button or the Process→Quit menu (Option-⌘-Q). This brings up a dialog box allowing you to quit or force quit an application.

7.4.1.3 Using the command line

As always, there is a command-line tool waiting for you if you can't use the GUI tools. Appropriately enough, it's named *kill*. To kill a process, you need to first get its process ID via the*ps* tool. Then you just pass that process ID to *kill* as shown in Example 7-11.

The Difference Between Quit and Force Quit

What's the difference between Quit and Force Quit? Quit makes a request to the application to terminate. This is essentially the same action you would take if you brought the application to the foreground and triggered the ⌘-Q keyboard shortcut.

Force Quit isn't so polite. It instructs the system to immediately terminate the process—with no-holds barred. This ensures that the process goes away, but does so without giving it a chance to save files or perform any other clean up.

Force Quit the Current Application

Many times, the application you want to force quit is the one that is running in the foreground. One way to quickly force quit the foreground application is to use Shift-Option-⌘-Esc. As soon as you do, the foreground application will be killed without prejudice. The downside: you won't be prompted to save any open documents—but if the application is truly stuck, then they were toast in any case.

This, however, will not work on the Finder. If you need to force quite the Finder, you'll need to use Option-⌘-Esc, select the Finder, and then click the Relaunch button.

Example 7-11. Using the kill tool

```
$ ps -x | grep emacs

  415 std S+ 0:00.01 grep emacs

  413 p2 S+ 0:00.13 emacs

$ kill 413
```

If a process is being unruly, you can tell the system to kill it without prejudice by using the *-KILL* option, which instructs the system not to be nice. Example 7-12 shows this in action:

Example 7-12. Using kill -KILL to terminate a process without prejudice

```
$ ps -x | grep emacs

  418 std S+ 0:00.01 grep emacs
  416 p2 S+ 0:00.04 emacs
$ kill -KILL 416
```

You should exercise great care when killing errant processes, especially when using the *kill* command, as it's easy to create problems. For example, every system administrator has a story about executing *sudo kill-KILL 1* instead of *sudo kill-KILL 128*. Unfortunately, the *init* process has the ID of 1, which will cause this command to hang the system.

7.5 Further Explorations

To get more information about the tools in this chapter, check out the following manpages:

- *softwareupdate*
- *system_profiler*
- *ps*
- *top*
- *grep*
- *tail*
- *kill*

Chapter 8. Scheduling Tasks

Computers are all about automation of tasks, and Mac OS X gives you several tools tohelp execute tasks at certain times of day and even on a regular and repeating basis.After all, if you want to copy a file from one place to another, or download a set ofweb pages at 5:25 a.m. on the dot, why should you have to get out of bed when thecomputer can do it for you?

To perform many of the tasks it has to, the system uses a set of utilities called *cron* and*periodic* to manage several housekeeping tasks. This includes such things as tidyingup log files and updating system databases like the one used by the *locate* command(discussed in Chapter 3.). This chapter shows you how to use these tools. First, however, you'll learn how to properly set the time on your computer to make sure the tasksyou're wanting to automate get triggered on time every time.

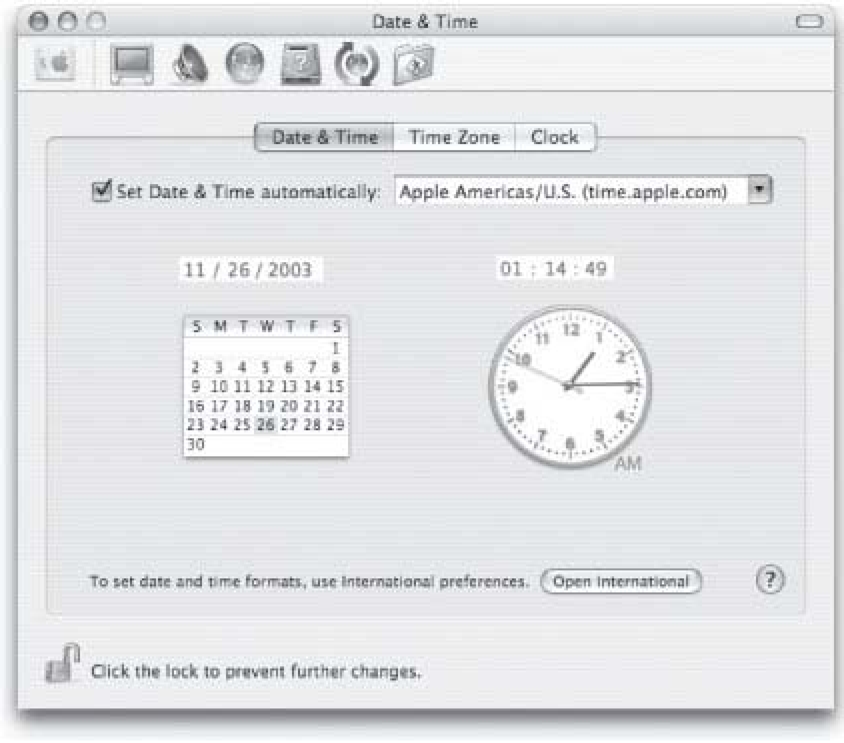
8.1 Setting the Time

Without having the time accurately set on your machine, it's pretty hard to scheduletasks. You can set the time and date on your computer yourself using the Date &Time preference panel (*/Applications/System Preferences*). If you spend any amountof time connected to the Internet (and who doesn't these days), you should bypass allthat and have your computer set its time from a network time server. Simply click the'Set Date & Time automatically' checkbox in the Date & Time preference panel, asshown in Figure 8-1, and select an appropriate time server near you.

A network time server is nothing more than a machine that has an accurate clock andthat understands the Network Time Protocol (NTP), which is designed to keep largenumbers of machines synchronized with an accurate clock; typically one of the atomicclocks that provides the most accurate time possible.

By default, the Date & Time preference panel allows you to set your time against oneof three servers provided by Apple. One of these servers is located in the U.S., anotherin Asia, and the third in Europe. You can also set your computer to synchronize its time with any NTP server you choose—such as a server on your local network that is set up by the network administrators. You can choose to synchronize against a number of publicly accessible time servers on the Internet. A list of network time servers isavailable at *www.eecis.udel.edu/~mills/ntp/servers.html*.

Figure 8-1. The Date & Time Preference panel



☑ Set Date & Time automatically: Apple Americas/U.S. (time.apple.com)

11 / 26 / 2003

01 : 14 : 49

To set date and time formats, use International preferences.

Click the lock to prevent further changes.

When you set a time server, the system does the following:

- Sets the time server that is being used in */etc/ntp.conf*.
- Makes sure that the *ntpd* process is running. This process will check the timeserver periodically and make sure your clock is set correctly.
- Set the `TIMESYNC` line in */etc/hostconfig* to -`YES`-, ensuring that *ntpd* will start when the system is rebooted.

8.2 Using iCal to Schedule Tasks

iCal is the personal calendaring application that comes with Mac OS X. iCal features multiple calenc duration. Events can be one-time occurrences, or they can repeat.

Each event can have an alarm that can display a notice on your computer screen, opena file, or eve */Applications/iCal.app/Contents/Resources*) to keeptrack of events and fire them off on schedule w

You can use iCal's alarms along with AppleScript to execute just about any kind oftask you'd like. T

1. Create an AppleScript application that performs the functionality you want andsave it somewh

> One logical place to store your scripts is in */Library/Scripts*. Anything you can store almostanything, including AppleScripts, shell scripts, a

2. Create a one-time or a repeating event in iCal.
3. Set the alarm properties on that event to open your AppleScript application.

For example, if you wanted to send a listing of all the files in your Home directorythrough email ev

Example 8-1. An AppleScript to list the files in the Home directory and se

```
set listing to (do shell script "/bin/ls -l $HOME")tell application "Mail"

    set the newMessage to (make new outgoing message with properties ¬        {sub
```

Once you have saved this script as an AppleScript application named *ListHomeDir*,you can set it up 8-2. Once set, as long as you are logged in to the computer at the time the event is scheduled for, go to the command line and use the Unix scheduling tools.

8.3 Using periodic

The*periodic* tool is designed to organize administrative tasks that need to be performed over and over again at regular intervals. The intervals that *periodic* supportsare: *daily, weekly*, and *monthly*. Mac OS X itself has a set of tasks that it runs using the *periodic* system, including:

- Tidying up log files and removing scratch files every day
- Rebuilding the *locate* database and rotating log files every week
- Performing log file rotation as well as login accounting every month

The tasks that *periodic* executes are a set of scripts in the */etc/periodic/daily, /etc/periodic/weekly*, and */etc/periodic/monthly* directories. To have *periodic* run your own script, simply add it to one of these directories. For example, if you have a batch of sales reports that you'd like to make a daily snapshot of, you could add the script inExample 8-2 to the */etc/periodic/daily* directory.

Example 8-2. A sample periodic script

```
#!/bin/bashecho Making daily backup of sales reports



DATE=`/bin/date +%Y-%m-%d`

/bin/mkdir -p /SalesBackups/$DATE

/bin/cp -R /Users/Shared/SalesData/* /SalesBackups/$DATE
```

periodic also gives you a way to control the order in which scripts run. If you look inthe */etc/periodic/daily* directory you'll notice that scripts that come with the systemstart with a number. To have your scripts execute in a particular order, simply prefixthem with a number and periodic will take care of ordering their execution, as shownin Example 8-3.

Example 8-3. Listing of the /etc/periodic daily directory

```
$ ls -l /etc/periodic/daily/

total 24

-r-xr-xr-x 1 root wheel 1389 30 Aug 20:36 100.clean-logs

-r-xr-xr-x 1 root wheel 3529 30 Aug 20:36 500.daily
```

The number in the filename controls the execution order of the scripts. The lower thenumber, the earlier it will be executed compared to other scripts in the directory. Forexample, to have the sales backup script in Example 8-2 execute after the rest of the daily tasks, you could save it as

/etc/periodic/daily/700.salesbackup.

8.3.1 Viewing the output from periodic

Since*periodic* runs in the background, any output produced by the scripts is hiddenfrom view. To see what happens, *periodic* saves the output into the */var/log* directory. Daily output is saved to*/var/log/daily.out*, weekly output is saved to */var/log/weekly.out*, and monthly output is saved to*/var/log/monthly.out*.

8.4 cron

The primary tool for scheduling tasks on the command line is the venerable *cron*. This tool is started automatically by *SystemStarter* at boot time and runs continuously inthe background. Every minute, *cron* wakes up and consults a set of tables to see ifthere is anything to be executed at that time, and if so, takes care of executing it. Thesetables, known as *crontab* files, are located in two places on the filesystem:

/etc/crontab

> The*crontab* file for the system at large. Each entry in this table represents a command that will be run by the root user and the time that it will be run. Anybodycan read this file, but only the root user can edit it.

/var/cron/tabs/

> This directory contains the user *crontab* files for each user on the system who is using *cron*. These files are hidden and are only visible to the root user, so that otherusers on the system can't look at each other's *crontab* files.

8.5 The System crontab File

Example 8-4 shows the system *crontab* as it appears in a default installation. While this file is for system tasks, you should always use the *crontab* file for your user.

Example 8-4. The system crontab file/

```
# /etc/crontab

SHELL=/bin/sh

PATH=/etc:/bin:/sbin:/usr/bin:/usr/sbin

HOME=/var/log

#

#minute   hour    mday    month    wday    who    command

#

#*/5      *       *       *        *       root   /usr/libexec/atrun

#

# Run daily/weekly/monthly jobs.

15        3       *       *        *       root   periodic daily

30        4       *       *        6       root   periodic weekly

30        5       1       *        *       root   periodic monthly
```

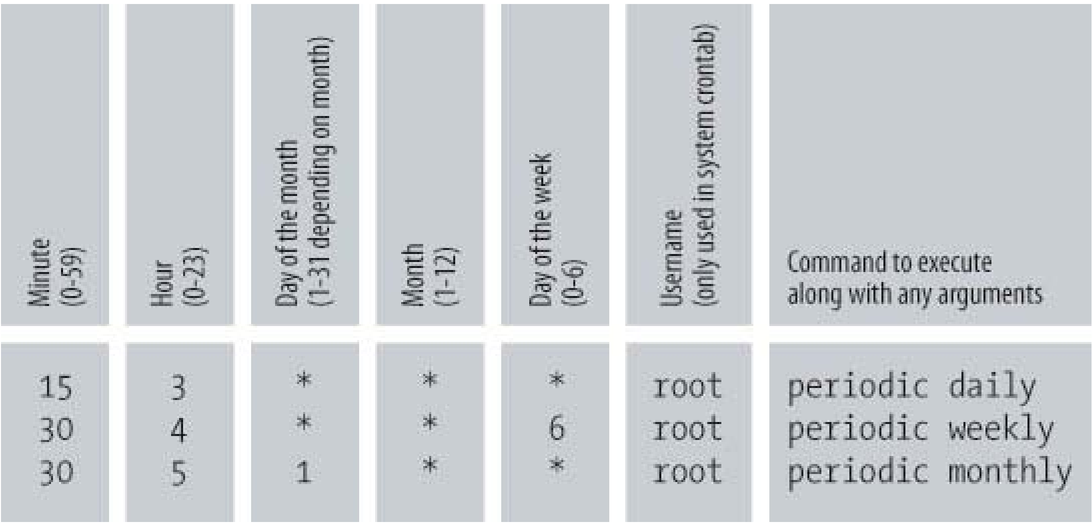
The*crontab* file format is similar to that of many other Unix utilities. Any line beginningwith the hash character (#) is a comment. The first three non-commented linesof the file set the environment that *cron* will run with. The remaining lines of the*crontab* file consist of five numbers defining the time pattern at which a particular taskis to run. The end of the line contains the command to run. In the case of the system*crontab*, the command also contains the name of the user under which to run the command.As the file itself indicates, each of the five numbers corresponds to a differenttime interval, arranged in order of finer to larger granularity. Figure 8-3 describes the settings for each of these fields. In addition to numbers, each field can contain an asterisk(*) character, which means match every possibility for that field.

To interpret the lines in the *crontab* file in Example 8-4 , read the fields for each line from left to right. For example, the first field (after the comments) in the system *crontab* indicates that `periodic daily` should be run on the fifteenth minute of the third houron any day of the month on any given month on any day of the week. This means that*periodic daily* will run at 3:15 a.m. every day. The second line indicates that the *periodic weekly* command will run on the thirtieth minute of the fourth hour of any Sunday;that is, it runs every Sunday at 4:30 a.m.

For each of the fields, you can also specify a list or range of numbers. For example, ifyou wanted to run a command every 15 minutes, you could use the following line:

```
0,15,30,60 * * * * command
```

Figure 8-3. The crontab file format



Minute (0-59)	Hour (0-23)	Day of the month (1-31 depending on month)	Month (1-12)	Day of the week (0-6)	Username (only used in system crontab)	Command to execute along with any arguments
15	3	*	*	*	root	periodic daily
30	4	*	*	6	root	periodic weekly
30	5	1	*	*	root	periodic monthly

8.6 The User crontab

To set up tasks that will get executed, you have to edit your own personal *crontab*.You can take a look at what you already have in your *crontab* file by using the *crontab* command:

```
$ crontab -l crontab: no crontab for duncan
```

This output means nothing has been scheduled yet. By default, a user account won'thave a crontab when it is first set up.

8.6.1 Editing a user crontab

There are two ways to edit your *crontab*. The first involves using whatever editor you've set up on the command line (for example, *vi, Emacs*, and *pico*). The second involvesusing any editor you want (such as *TextEdit* or *BBEdit*) and loading a text file as your *crontab*. To edit your file on the command line, use the following command:

```
$ crontab -e
```

For your first *crontab* entry, let's add a line that will make your computer say 'hello'every minute. To do this, add the following line to your *crontab* file:

```
* * * * * osascript -e 'say "hello"'
```

> If you get stuck in an editor that you are unfamiliar with, remember thatyou can get out of *vi* by typing `:q!` and out of *Emacs* by typing Control-X then Control-C. See Chapter 3,*The Terminal and Shell*, for more info about command-line editors.

Now, every minute of every day that your machine is on, it will say 'hello' to you,which could become annoying indeed. There are a couple things going on here:

1. The`osascript -e'say "hello"'` command is being issued by your system everyminute, based on the five preceding asterisks.
2. The command uses the default system voice set in the Speech preference panel tospeak the word 'hello' on cue.

But now that you've got a *crontab* file installed, you can use *crontab* to list the file:

```
$ crontab -l * * * * * osascript -e 'say "hello"'
```

The other way to create a *crontab* file is to use an editor like TextEdit or BBEdit. Toget the current*crontab* out in a form that you can open with any editor, save the file onyour hard drive, and then execute the *crontab* command as follows:

```
$ crontab mycrontabfile
```

This also gives a way to quickly reset the *crontab* file for a user. By passingthe*/dev/null* file into *crontab*, the user's *crontab* will be set to an empty file.

Using the *crontab* comment to specify a file is also a good way to accidentlylose any *cron* settings that you have in place. Be sure to checkyour*crontab* before loading in a new *crontab* file.

To retrieve your *crontab* for editing, you can direct the output using the followingcommand:

```
$ crontab -l > mycrontabfile
```

Running Virex from cron

If you've installed the McAfee Virex virus scanner from .Mac, you have the *vscanx*command-line virus scanner installed at */usr/local/vscanx/vscanx*. You can take advantageof this and have your disk scanned for viruses from *cron* instead of from theGUI. This is a bonus that the virus scanner will run no matter who's logged in to thecomputer or even if nobody is.

To enable this, delete the .*VirexLogin* item from list your Startup Items in theAccounts preference panel. Then add a line to your *crontab* file to execute*/usr/local/vscanx/vscanx* whenever you'd like.

Note that you need to give the full path to *vscanx*since it isn't installed in one of thestandard binary directories, such as */usr/bin*.

8.6.2 Additional configuration settings

The*cron* command on Mac OS X has been enhanced compared to those found onsome other Unix variants. For example, you can use the following more readable entriesin the time field:

- Days of the week can be indicated by their abbreviated name: *sun*, *mon*, *tue*, *wed*,*thu*, *fri*, *sat*.
- Months can be indicated by their abbreviated name: *jan*, *feb*, *mar*, *apr*, *may*, *jun*,*jul*, *aug*, *sep*, *oct*, *nov*, *dec*.
- You can indicate step values by using a fraction notation such as 8-17/2 which, ifit were in the hours field, would mean 'every two hours between the hours of 8a.m. and 5 p.m.'

Some special strings can be used in *crontab* files.Table 8-1 has a list of these strings.Except for the last one, all these strings replace the time fields. The last, `@AppleNotOnBattery`, can be used in front of a command to prevent it from running when yourlaptop is disconnected from AC power. This ensures that you don't run disk-intensivetasks when you need your battery the most. For example, if you write a script that copiesall your files from your *~/Documents* folder to some safe storage location that youwant to run only when your PowerBook is plugged in, you would use the following*crontab* entry:

```
0 * * * * @AppleNotOnBattery ~/bin/copyfiles
```

Table 8.1. Special strings that can be used in a crontab

String	Description	Equivalent To
@reboot	Run when the system reboots	
@yearly	Run on midnight of January 1	`0 0 1 1 *`
@monthly	Run at midnight on the first of the month	`0 0 1 *`
@weekly	Run at midnight each Sunday	`0 0 * * 0`
@daily	Run every day at midnight	`0 0 * * *`
@hourly	Run every hour at the top of the hour	`0 * * * *`
@AppleNotOnBattery	Prevents command from running if the system is on battery	

What About at, batch, atq, and atrm?

If you're coming to Mac OS X from another Unix system, you may be familiar withusing the *at, batch, atq*, or *atrm* commands for scheduling tasks. These commands exist in Mac OS X, but they have been disabled by Apple due to power managementconcerns. If you really need these commands, you must enable the *atrun* command in*/etc/crontab*, but do so at your own risk.

8.6.3 Sleep and cron

The*cron* system won't execute while your system is asleep. This is because the CPU is powered down and there's just enough happening in your machine to keep the contentsof memory ready when you want to wake the system up.

Sometimes this isn't a big deal. For example, if you use a *crontab* line to remind you tostretch every hour, then you won't mind it not running. However, for other tasks thatyou would like to have run, it can create a bit of a problem. The best piece of advice isto time tasks that need to be run when your system is less likely to be in sleep mode.

8.7 Changing periodic's Execution Time

By default, *periodic* runs daily tasks at 3:15 a.m., weekly tasks at 4:30 a.m., andmonthly tasks at you are sound asleep. If your system isn't on 24 hours a day, you might considerchanging the time idle. For example, if you wanted the daily tasks to run during yourlunch hour, the weekly tasks on a.m., you would edit the */etc/crontab* file to match Example 8-5. Since the system *crontab* doesn't

Example 8-5. The system crontab with the periodic tasks set at a more re

```
15    12     *    *    *    root    periodic daily0    10    *    *    2    root
```

8.8 Further Explorations

For more information about the technologies in this chapter, see the followingresources:

- *AppleScript: The Definitive Guide*, by Matt Neuburg (O'Reilly & Associates, 2003)

You may also be interested in consulting the following manpages:

- *periodic*
- *cron*
- *crontab*

Chapter 9. Preferences and Defaults

Whenever you customize the behavior of a Mac OS X application, such as changing the default font or colors or the windows that are visible, the various changes that you make are saved into a preference file for that application. This is in contrast with Windows where application preferences typically are saved into the monolithic registry. By having each application store its preferences in a separate file, the overall system is made more robust than if all the preferences were in one big file. If a preference gets corrupted, it is less likely to affect the system and typically will affect only the application that uses that preference. The designated location for preference data is the *Library/Preferences* folder in each of the filesystem domains. (Review Chapter 2,*Lay of the Land*, for more information about filesystem domains.)

Many Mac OS X applications, including all the applications Apple provides, go beyond just using the*Library/Preferences* directory and store their preferences in the *defaults system*. This system, which is often referred to as the *defaults database*, is made of each application's preferences stored in an XML-based property list (*plist*) file in the *Preferences* folder. By using the *defaults* system, applications can utilize code in the operating system to manage preferences instead of having to provide their own preference- handling code. Additionally, you can use the *defaults* command-line tool to read and write data into the database.

Before the transition to Mac OS X, users used to back up their preferences regularly to a floppy disk. This was done because the preferences would get corrupted at the drop of a hat. By having a stable backup floppy at the ready, it was pretty easy to get everything back to the way it was supposed to be simply by copying the preferences back into place. With Mac OS X, even though preferences don't tend to get corrupted as often, nor do they have as devastating an effect when they do, it is still easy to make quick backups of the *~/Library/Preferences* directory. For example, you can drag the *~/Library/Preferences* folder to a USB key fob, upload them to your iDisk, or even use Backup 2 (part of the .Mac set of tools) to back them up to your iDisk or a CD or DVD. In fact, you should do this regularly. Then, whenever you need to restore the preferences for an application, they are easy to find.

9.1 Property Lists

A property list (*plist*) file is a text file that can represent all kinds of data in a structured form. There are two formats for a property list file: an older ASCII-based property list format; and the newer XML-based property list format. We saw an example of the older ASCII property list format in Chapter 4 when we looked at Startup Items. This is one of the few places where you'll see the older format in use. In most places, including the defaults database, you'll see the newer XML format. A simple XML property list used by the battery menu extra is shown in Example 9-1. If you've ever looked at HTML or other XML dialects, the basic structure of this file should look somewhat familiar.

Example 9-1. The com.apple.menuextra.battery.plist file

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"

    "http://www.apple.com/DTDs/PropertyList-1.0.dtd">

<plist version="1.0">

<dict>

    <key>ShowPercent</key>

    <string>NO</string>

    <key>ShowTime</key>

    <string>YES</string>

</dict>

</plist>
```

The structure of a property list file contains a *plist* root element that contains one or more data elements. In Example 9-1, as in all property lists used by the defaults system, the child element of*plist* is named *dict* (which stands for *dictionary*—a mapping between names and values) and contains a set of key and string elements. Each set of key and string elements defines a name-value pair. In essence, the example file can be interpreted as "this file contains a dictionary in which the key *ShowPercent* is set to *NO* and the key *ShowTime* is set to *YES*".

> Unless you have modified the battery menu extra, you may not have this or many other *plist* files that are mentioned in this chapter. The *plist* files for a preference domain typically are created only on demand.

The XML elements that can appear in a property list file are listed in Table 9-1. One thing you might notice in Example 9-1 is that even though there are the true and false values that can appear as tags in the XML file, the ShowPercent and ShowTime keys have string values set to NO and YES respectively. It's unfortunate, but this inconsistency crops up in many of the defaults keys used by applications and is something you should be aware of.

Why Use XML?

Apple has jumped on the XML bandwagon with the property list file format. XML promises a standard format that is easy for tools to use and comprehend. In addition, it's a text-based format which means that it's relatively easy to edit (or at least easy to load into a text editor). However, not all XML documents, with all those angle brackets and tags and attributes along with schemas and namespaces, are easy for a *human* to edit. Nope. Not at all.

The primary advantage of the defaults system as it stands isn't that it makes preferences easy to read and edit. The reason is because it's easy to use tools (like the Property List Editor) to edit preferences in the defaults system without having to run the application that wrote them.

Table 9-1. Property list elements

Element	Description
plist	Root element of a property list.
dict	A dictionary to hold other data elemtents in key value form. The contents of this element are a set of <key> elements, followed by the data associated with the key.
array	Contains a set of data values in a particular order. The contents of these elements are simply a list of data elements.
string	Contains a data value as a string.
real	Contains a data value as a real number, such as 8.14.
integer	Contains a data value as an integer, such as 10 or -9.
date	Contains data that represents a date in ISO 8601 (a standard format for dates), such as 2003-10- 24T08:00:00Z which represents 8:00pm October 24th, 2003 (the time Panther was released to the public).
true	Indicates that the value is true.
false	Indicates that the value is false.
data	Contains arbitrary data.

9.2 Where Preferences are Stored

As mentioned earlier, the defaults database stores preference data in files located in the *Library/Preferences* folder of a filesystem domain. For example, when a preference applies to a single user, it is written to the *~/Library/Preferences* folder. If the preference applies to all users on a system, it is written to the */Library/Preferences* folder.

Each of the files in the defaults database takes a unique name, known as a *preferencedomain*, determined by the application that uses it. The Apple-recommended naming convention for preference domains is to use the reverse Internet domain name of an application's vendor, followed by the name of the application. For example, all the *plist* files used by the various Apple-supplied applications use filenames that begin with *com.apple*, followed by the name of the application, followed by the *.plist* extension. Example 9-2 shows a partial command-line listing of the *~/Library/Preferences* directory for a freshly created user:

Example 9-2. The ~/Library/Preferences directory

```
$ls -l

total 96

drwx------ 4 norman  norman   136  8 Dec 22:07 ByHost

drwx------ 3 norman  norman   102  8 Dec 22:06 Explorer

-rw------- 1 norman  norman   975  8 Dec 22:07 com.apple.Bluetooth.plist

-rw------- 1 norman  norman   475  8 Dec 22:06 com.apple.HIToolbox.plist

-rw------- 1 norman  norman   557  8 Dec 22:07 com.apple.MenuBarClock.plist

-rw------- 1 norman  norman  9795  8 Dec 22:07 com.apple.dock.plist

-rw------- 1 norman  norman  1294  8 Dec 22:08 com.apple.finder.plist

-rw------- 1 norman  norman  1040  8 Dec 22:08 com.apple.recentitems.plist

-rw------- 1 norman  norman   741  8 Dec 22:07 com.apple.scheduler.plist

-rw------- 1 norman  norman  5004  8 Dec 22:07 com.apple.sidebarlists.plist

-rw------- 1 norman  norman   456  8 Dec 22:07 loginwindow.plist
```

When you look into your own *~/Library/Preferences* directory, you'll see more files than this, but by applying the naming convention rules, you should be able to easily sort out which property list file belongs to which application.

Preferences are written into the *Library/Preferences* folder of a filesystem domain. When an application searches for the value of a preference it can be given a value from the user, local, network, or system filesystem domains. This means general settings that apply to all users can be defined at the local or network domain level. Settings that apply to a single user are written to the user domain so that they remain separate from the preferences of other users.

9.2.1 Host-based Preferences

Because Mac OS X can mount Home folders from a server and a user might log into several different machines with his account, the defaults database provides a mechanism for applications to store information on a per-machine basis. This allows applications, such as the one that provides iDisk synchronization, to keep distinct settings depending on which machine you are using. To keep these settings distinct, they are kept in a *ByHost* folder within the *Library/Preferences* folder. They too use a naming convention consisting of the reverse Internet domain name of the vendor, the application name, the Ethernet MAC address of the host, and then the *plist* file extension. Example 9-3 shows a list of the files you might find in the *ByHost* folder.

Example 9-3. The contents of the ByHost folder

```
$ls ~/Library/Preferences/ByHost

com.apple.HIToolbox.000a9599e492.plist

com.apple.iTunes.000a9599e492.plist

com.apple.idisk.000a9599e492.plist

com.apple.systemuiserver.000a9599e492.plist
```

9.2.2 Global Preferences

In addition to the preferences that you see in Example 9-2, a hidden file named. *GlobalPreferences.plist* in the *~/Library/Preferences* folder contains preferences used by all applications. It contains data that affects all applications equally, such as the locale of the system in use. Example 9-4 shows this file for a freshly created user.

Example 9-4. The .GlobalPreferences file

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"

    "http://www.apple.com/DTDs/PropertyList-1.0.dtd">

<plist version="1.0">

<dict>

        <key>AppleAntiAliasingThreshold</key>

        <integer>8</integer>

        <key>AppleLanguages</key>
        <array>
                <string>en</string>
                <string>ja</string>
                <string>fr</string>
                <string>de</string>
                <string>es</string>
                <string>it</string>
                <string>nl</string>
                <string>sv</string>
                <string>no</string>
                <string>da</string>
                <string>fi</string>
                <string>pt</string>
                <string>zh_CN</string>
                <string>zh_TW</string>
                <string>ko</string>
        </array>
        <key>AppleLocale</key>
        <string>en_US</string>
</dict>
</plist>
```

These preferences correspond to some of the settings you can apply in the System Preferences application. For example, Example 9-4 shows the array of strings associated with the *AppleLanguages* key. The strings in the array correlates to the language packages you can chose from in the International preference panel (shown in Figure 9-1).

9.2.3 Non defaults-based Preferences

Application preference files that aren't part of the defaults database can also be stored in the *~/Library/Preferences* folder. These preferences tend to be written in a proprietary format and are not easily readable except through the application that wrote them.

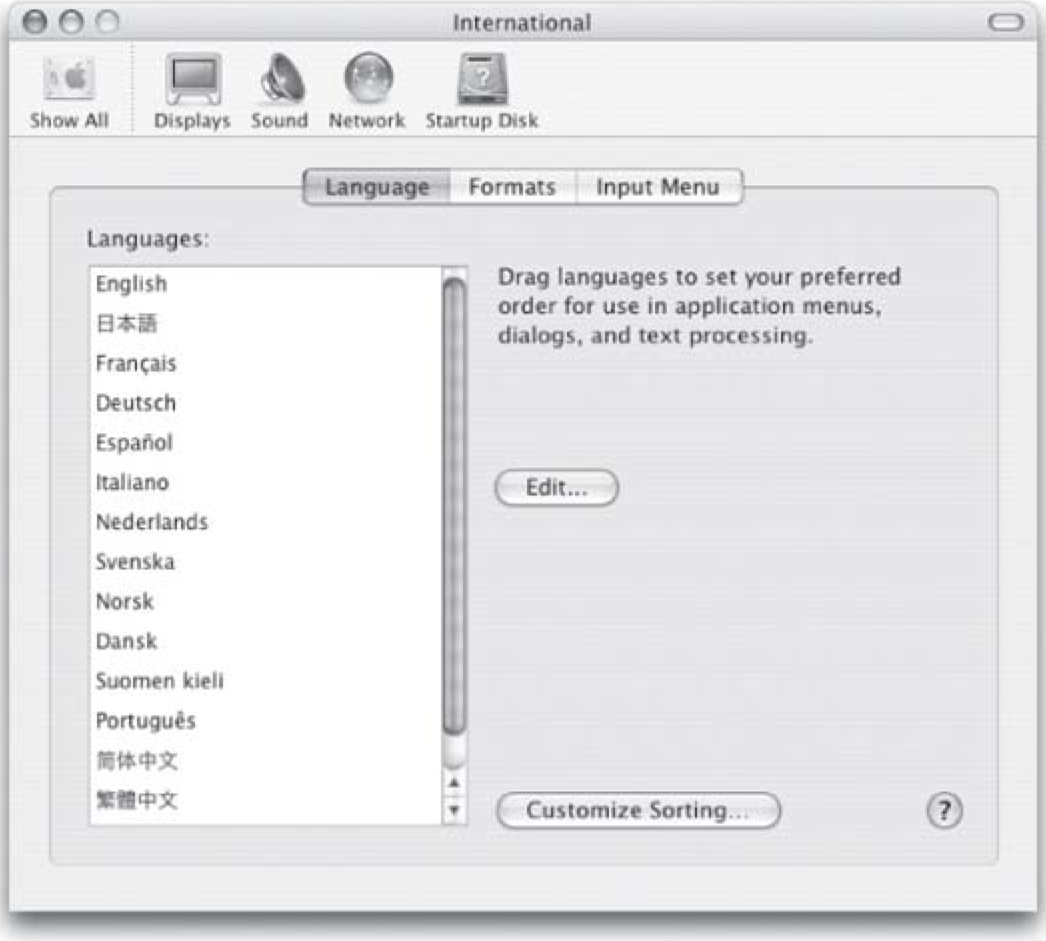
Typically, applications that write non-defaults database preferences are older Carbonbased applications that migrated to Mac OS X from Mac OS 9 and already have their own preference handling code. These applications are truly antisocial because they can only run in Classic mode, and they write their opaque preference files into Mac OS 9's */System Folder/Preferences* folder.

9.3 Reading and Writing Preferences

There are two ways in which to read and write information in the defaults database:

- With the Property List Editor (*/Developer/Applications*) installed as part of the Xcode Tools
- From the command line using the *defaults* command-line utility

Figure 9-1. The Language section of the International Preference panel



English
日本語
Français
Deutsch
Español
Italiano
Nederlands
Svenska
Norsk
Dansk
Suomen kieli
Português
简体中文
繁體中文

Drag languages to set your preferred order for use in application menus, dialogs, and text processing.

9.3.1 Property List Editor

The Property List Editor (*/Developer/Applications/Utilities*), shown in Figure 9-1, is a GUI application installed with the Xcode Tools that lets you view and edit property list files. When you first open a property list with the Property List Editor, you'll notice that the top root element is collapsed. Click the disclosure triangle next to it and you can drill through the various preferences.

As shown in Figure 9-2, there are three columns to the Property List Editor. These are:

Property List

Lists the keys of the property list; that is, the contents of the `<key/>` tags of a *plist* XML file.

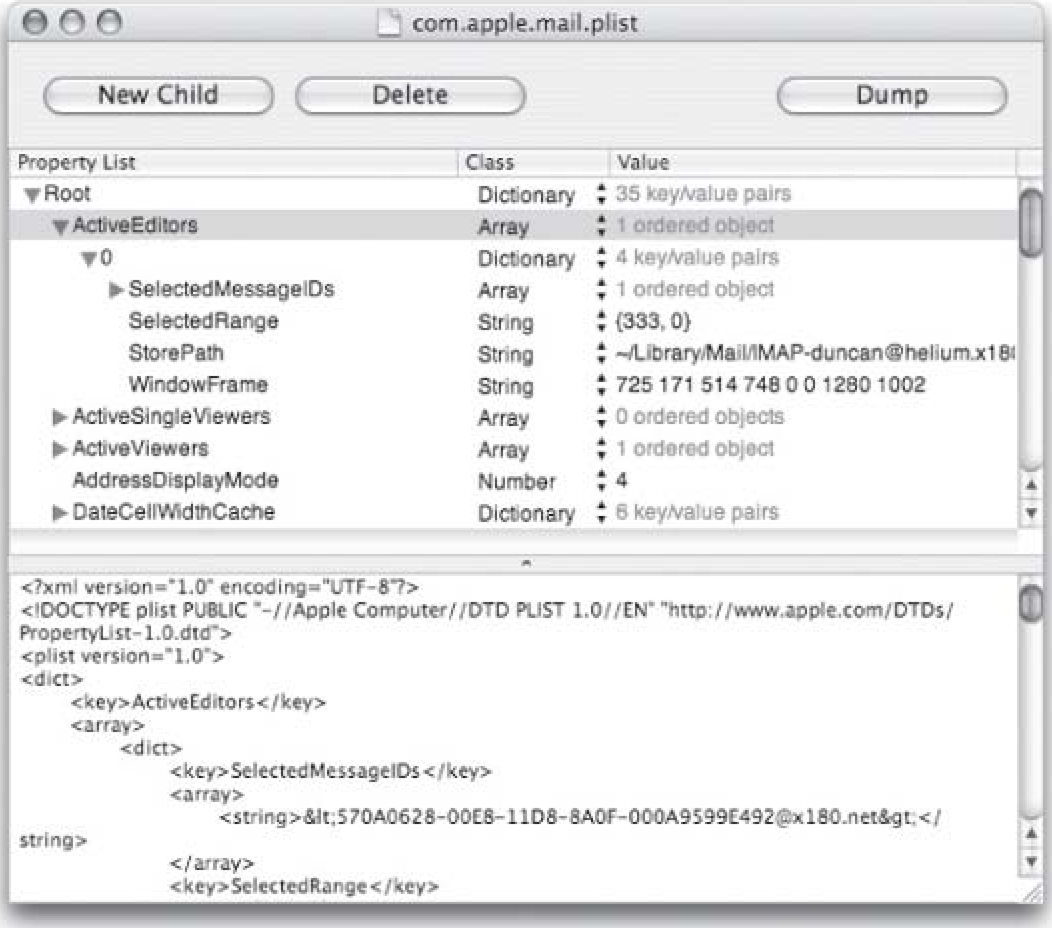
Class

Lists the classes available for each key definition. This affects the value tags in the XML file as defined in Table 9-1.

Value

Contains the value for the preference key.

Figure 9-2. The Property List Editor application



Property List	Class	Value
▼Root	Dictionary	35 key/value pairs
▼ActiveEditors	Array	1 ordered object
▼0	Dictionary	4 key/value pairs
▶SelectedMessageIDs	Array	1 ordered object
SelectedRange	String	{333, 0}
StorePath	String	~/Library/Mail/IMAP-duncan@helium.x18
WindowFrame	String	725 171 514 748 0 0 1280 1002
▶ActiveSingleViewers	Array	0 ordered objects
▶ActiveViewers	Array	1 ordered object
AddressDisplayMode	Number	4
▶DateCellWidthCache	Dictionary	6 key/value pairs

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/
PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>ActiveEditors</key>
        <array>
                <dict>
                        <key>SelectedMessageIDs</key>
                        <array>
                                <string>&lt;570A0628-00E8-11D8-8A0F-000A9599E492@x180.net&gt;</
string>
                        </array>
                        <key>SelectedRange</key>
```

To view the XML version of the *plist* file, click the Dump button in the upper-right corner of the Property List Editor's window. You can't edit the XML source here, but you can see the effects of the changes that you make in the top part of the window.

To edit an item, double-click the item's name that you want to edit, type in the new value, and then hit Return to accept the new value.

9.3.2 The defaults Command-Line Tool

The*defaults* tool gives you easy access to the defaults system from the command line, letting you read and write the preferences for any Mac OS X application on the system. As with any command-line utility, you can write shell scripts to run several defaults commands automatically, enabling you to set an application's behavior any way you'd like in an instant.

9.3.2.1 Reading preferences

The basic usage of the *defaults* command for reading an application's preferences is:

```
defaults read domain [key]
```

where*domain* is the property domain that you want to read the preference from, and *key* is an optional keyname that lets you see the value of a single key for a preference domain. Since a preference key can contain many different kinds of data, the output from this command will vary depending on the kind of data that is associated with a preference key. Example 9-5 shows the output from reading the preferences for a domain.

Example 9-5. Using the defaults command

```
$defaults read com.apple.menuextra.battery

{ShowPercent = YES; ShowTime = NO; }
```

Example 9-6 shows how to read a single key from a domain, in this case a boolean value.

Example 9-6. Using the defaults command specifying a key

```
$defaults read com.apple.menuextra.battery ShowPercent

NO
```

Strings and numbers will be shown like Boolean values. The only caveat is that a string containing a space will be surrounded by quotes. An array associated with a property key will be output as a list of items within a set of parentheses, and the items are comma-delimited.

Example 9-7 shows the most recently used applications as stored in the *com.apple.recentitems* preference domain.

Example 9-7. Recent applications stored in the com.apple.recentitems domain

```
$defaults read com.apple.recentitems apps

(

    "/Applications/Utilities/Terminal.app",

    "/Applications/Adobe InDesign 2.0/InDesign 2.0.2",

    "/Applications/System Preferences.app",

    "/Applications/Preview.app",

    "/Applications/QuickTime Player.app",

    "/Developer/Applications/Utilities/Property List Editor.app",

    "/Applications/Font Book.app",

    "/Applications/Safari.app",

    "/Applications/Utilities/StuffIt Expander.app",

    "/Applications/Can Combine Icons.app"

)
```

A dictionary associated with a key is output as a set of name/value pairs surrounded by curly braces.Example 9-8 shows a dictionary associated with the *InboxViewerAttributes* preference in the*com.apple.mail* preference domain.

Example 9-8. A dictionary of values associated with a preference

```
$defaults read com.apple.mail InboxViewerAttributes

{

    DisplayInThreadedMode = yes;

    SortOrder = "received-date";

    SortedDescending = NO;

}
```

You'll also see this format used when you list out all the defaults for a preference domain.

Example 9-9 shows the contents of the *com.apple.systemuiserver* domain, which controls the menu extras shown in the upper-right corner of the screen.

Example 9-9. A dictionary output of the preferences in the com.apple.systemuiserver domain

```
$defaults read com.apple.systemuiserver

{

    menuExtras = (

        "/System/Library/CoreServices/Menu Extras/Bluetooth.menu",

        "/System/Library/CoreServices/Menu Extras/AirPort.menu",

        "/System/Library/CoreServices/Menu Extras/Volume.menu",

        "/System/Library/CoreServices/Menu Extras/Battery.menu",

        "/System/Library/CoreServices/Menu Extras/Clock.menu",

        "/System/Library/CoreServices/Menu Extras/User.menu"

    );

}
```

You can also read the preferences for an application by using its name instead of trying to figure out its preference domain. The syntax for this usage is:

```
defaults read -app appname
```

For example, to read the preferences for Mail, you would use the following command:

```
$ defaults read -app Mail
```

9.3.2.2 Writing preferences

The basic syntax to write preferences to the defaults database is:

```
defaults write domain key value
```

where*domain* is the preference domain to write the preference to, *key* is the preference name, and*value* is the to content to associate with that key. For example, to set Terminal to use a blinking cursor, you would use the following command:

```
$ defaults write com.apple.terminal BlinkCursor YES
```

If you enter this command into a Terminal window and then open a new window, you'll see the cursor blinking. To reverse this setting, use the following command:

```
$ defaults write com.apple.terminal BlinkCursor NO
```

You can use two commands to write properties to an array. The first associates an array with a preference key and replaces any previous values associated with that key. This command has the syntax:

```
defaults write domain key -array element1 element2 element3...
```

To add new elements to the end of an array for a key without replacing all the elements in the array, use the -*array-add* option with the following syntax:

```
defaults write domain key -array-add element1 element2...
```

> You should always make sure that the application for which you are writing preferences isn't running when you use the defaults command. A running application might write something to the defaults database that will overwrite any changes you may have made.

9.3.2.3 Reading and writing host-specific preferences

To work with host-specific preferences, you can use the -*currentHost* or -*host* optionss to the *defaults* command. For example, to read the preference from the *com.apple.idisk* preference domain on the machine that you are working on, you would use the following command:

```
$ defaults -currentHost read com.apple.idisk
```

9.3.2.4 Reading and writing global preferences

To access the global preferences (those associated with all applications) use the -*g* options with the*defaults* command, as shown in Example 9-10.

Example 9-10. Reading global preferences

```
$defaults read -g
{
    AppleAntiAliasingThreshold = 4;
    AppleICUTimeFormatStrings = {
        1 = 'HH':'mm';
        2 = 'HH':'mm':'ss';
        3 = 'HH':'mm':'ss z';
        4 = 'HH':'mm':'ss z';
    };
    AppleKeyboardUIMode = 2;
...
```

9.4 Determining Preference Keys

Editing preferences in the defaults database is a chicken-and-egg problem. Without knowing the ke set a preference for, it's hard to customize an application's behavior. Setting preference keys rando won't accomplish much except to fill your defaults database with useless data. In order to know wh keys can be set, you have to do a bit of research. Three methods will help you determine the prefe keys an application might use:

- Looking at the preference *plist* files after you've customized an application
- Searching the web
- Digging into the application itself

The first of these methods is fairly self explanatory. Simply fiddle with the various settings of the application and watch the *plist* file to see what changes. Once you have an idea of the ways that ke named, you can use the *defaults find* command. This command searches through the defaults data and returns the preference file and data associated with any particular string. For example, if after tweaking the opacity of your Terminal windows, you searched for the string *Opaque*, you would fin key in the com.apple.Terminal domain, as shown in Example 9-11.

Example 9-11. Searching for a string in the defaults database

```
$defaults find Opaque

Found 1 keys in domain 'com.apple.Terminal': {TerminalOpaqueness = 0.9542592763900
```

If other applications used keys with the string *Opaque*, they would also be listed.

The second of these methods is also straightforward—using the web. The Mac OS X Hints web site *www.macosxhints.com/*) has a wealth of information on ways to hack the defaults database. Googl *www.google.com/*) searches can turn up quite a lot of useful information as well.

The third of these methods is a bit more difficult and requires looking inside the application's bundl using the *strings* command. The *strings* command examines a binary file and prints out all the strin finds in a binary. For example, to look inside the Dock's executable, you would use the command s inExample 9-12.

Example 9-12. Examining a binary for preference keys

```
$strings /System/Library/CoreServices/Dock.app/Contents/MacOS/Dock

__dyld_mod_term_funcs

__dyld_make_delayed_module_initializer_calls

The kernel support for the dynamic linker is not present to run this program.

showhidden
```

```
showshadow
DoesPointToFocusCursorUpdate
ClientMayIgnoreEvents
com.apple.finder
en_US
trashlabel
owensdock
dock
.dock
AppleShowAllExtensions
AppleShowAllFiles
notfound
trashfull
trashempty
finder
...
```

The result is a large number of strings, which probably don't have anything to do with preference s at all. But many of these strings are used as preference keys and values in the *plist* file. This metho becomes a sleuthing game that is boring for some and immensely interesting to others.

If you locate undocumented preferences, keep in mind that they may go away or be renamed in a version of the application, something that has been known to happen in the past (especially when hackers have unearthed settings that Apple would really rather keep private).

9.5 Further Explorations

For more information about the defaults system, see the PDF book titled *System Overview* and located at */Developer/Documentation/MacOSX/Conceptual/SystemOverview/SystemOverview.pdf.*

You may also want to consult the following manpages:

- *defaults*
- *strings*

Chapter 10. Disks and Filesystems

Mac OS X supports multiple device types that can be accessed as a disk. These include physical disks such as hard drives, FireWire and USB drives, multiple hard drives combined into a RAID, CDs and DVDs, various forms of flash memory, network disks residing on a server, and virtual disks that can exist in memory or be derived from files on another filesystem. Even though disks can take many forms, from the user perspective, a disk is a disk is a disk. Files can be moved and copied between them without too much thought.

To the system, however, two layers mediate between the operating system and disks:

- Device drivers that translate standard system file access calls into a form understood by the disk
- Filesystems that organize data on a drive into a form that can be accessed by a device driver

Mac OS X has device drivers, in the form of kernel extensions (*kexts*), for most of the devices that you will want to use as a disk including FireWire drives, USB flash memory cards, and SCSI disks.

This chapter starts out by introducing the kinds of filesystems that Mac OS X supports and shows you how to examine and work with these filesystems. It then shows you how to work with disks—both physical and virtual—including partitioning disks and moving data safely from one disk to another.

10.1 Filesystems

Each device stores data in a form that makes sense for that device. A hard drive usually writes data to the platters using a series of sectors and tracks. A CD writes data in one continuous track that spirals from the inside of the disk out. A compact flash card simply holds data in a matrix of memory cells on the chip. For efficiency reasons, the data that makes up a file may be scattered across a drive instead of being nicely organized in one lump. The role of a filesystem is to mediate between the world of the device where data is spread across the physical media of a disk and the world of the Finder where data shows up in an organized form of files and folders.

10.1.1 The Mac OS Extended Filesystem (HFS+)

The primary filesystem used by Mac OS X is the Mac OS Extended filesystem, also known as HFS+. Introduced with in Mac OS 8.1 and recently upgraded with journaling features for Panther, HFS+ allows long filenames with up to 255 Unicode characters, scales up to 2 TB of data on a filesystem, can handle 2 billion files, and allows for files up to 2 GB in size. In addition, each folder in an HFS+ filesystem can handle a maximum of 32,767 files. When HFS+ was first introduced, the size of these figures was way beyond what the state-of-the-art filesystems of the time could handle.

> Even though it looks like it will be good for the next couple of years, it's obvious that HFS+ (in its current form) only has a few more years left to it. Apple, in all their infinite innovative wisdom, no doubt realizes this and is probably hard at work at devising a new filesystem type for the future; something capable of keeping up with the rapidly growing file sizes we encounter when working with digital audio and video.

One of the quirks of HFS+ is that it is a *case-preserving* and *case-insensitive* filesystem. This means you can't have files named *Readme* and *README* in the same directory. This is similar to the way the filesystems work on Windows, but is different from the traditional Unix case-preserving filesystems where you can have both a *Readme* and *README* file in the same directory. In most cases this isn't a problem because people don't tend to place two files with the same name into a directory.

Another difference between the HFS+ filesystem and most others is that HFS+ supports the concept of *resource forks*. Resource forks were used on the old Mac OS to store all sorts of metadata for a file, such as icons. Although resource forks were a good idea, they didn't catch on with the rest of the world. Unix and Windows filesystems don't have an equivalent concept, so Apple adopted a similar policy and recommends that all applications that write files should avoid using resource forks. However, the filesystem still supports resource forks, mainly so older applications that rely on them can run just fine. In all likelihood, many applications will continue to use resource forks to some degree or another.

Case Insensitivity and Unix

When the Rhapsody team first brought up the system on top of the HFS+ filesystem, they were concerned that the case insensitivity would be a problem. It turned out that there was only one package in all the various Unix utilities where case insensitivity was a problem and that was in the Perl source code tree. The problem: there was both a *Makefile* and a *makefile* at the top level of the distribution package. Now that most of the core Perl developers run Mac OS X, this is no longer an issue.

10.1.1.1 Journaling

If something went wrong with the Mac OS X filesystem before Panther, a long and intensive *fsck* process would be run the next time the machine started up. For example, if the machine was powered off incorrectly or if the system crashed, it was pretty common to wait a long time at the initial gray boot screen. And if the filesystem truly got itself into a bad state, manual intervention was necessary to fix the filesystem.

Journaling, which was first introduced in Mac OS X Server 10.2.2 and is now the default filesystem type for Mac OS X Panther. It implements a scheme that keeps the filesystem structure of your disk safe even in the face of an unexpected shutdown. When your Mac reboots, disk repairs are made as needed. The filesystem does this by keeping a continuous record of changes to the files on a disk in a journal file in a designated area of the disk. If a computer starts up and the disk is in an inconsistent state, the journal is used to quickly restore the disk to its previous known state. This record keeping does come with a slight performance overhead. It typically takes 10 to 15 percent longer to write small files with journaling than without it, but in most cases, the slight performance loss is well worth the safety gained. Journalling also allows other optimizations to be made in the disk I/O system, which more than makes up for the performance penalty.

It is important to note that, in the face of unexpected shutdowns, journaling won't necessarily protect the data being written to disk. By protecting the filesystem, journaling protects all the data that is already on your disk from being lost because of a power outage at the wrong moment.

10.1.1.2 Fragmentation

In the past 20 years of personal computing, a common theme with hard disks has been the issue of file fragmentation. Early hard drive formats were extremely susceptible to performance problems manifested over time by having the data for the files split among too many locations on the drive. As files grew larger than their original allocation, the filesystem was forced to put parts of those files onto different parts of the disk. Even more modern formats such as HFS+ can exhibit performance slowdowns over time as a disk is used more.

Panther introduces the following two optimizations to the HFS+ driver when using a journaled filesystem:

Automatic file defragmentation

When opened, if a file has more than 8 fragments and is smaller than 20 MB in size, it is defragmented by simply moving the file to a new location on the drive where the file can be written in one contiguous block.

Adaptive Hot File Clustering

Over a period of time, the system keeps track of small files that are read frequently, but never written to. As the system learns which files are used most and which are least likely to change size, it moves them to the fastest part of the drive, where they can quickly be accessed. Files that don't meet the requirements for being in this "hot zone" are moved out to ensure that enough room exists for the files that should be there.

This means, at least for most files most of the time, a separate defragmentation program isn't needed. It also means you should always enable journaling on your drives so you can take advantage of these features. Fortunately, when installing Panther, the default filesystem type is Mac OS Extended (Journaled), so you should be set.

> These optimizations weren't part of the official advertised feature set of Panther. They were discovered by some programmers while reading the source code for the filesystem drivers available from the Darwin project.

10.1.2 Other Supported Filesystems

In addition to HFS+, Mac OS X supports several other types of filesystems, each of which has its own unique characteristics:

Mac OS Standard (HFS)

The standard Mac filesystem prior to the release of Mac OS 8.1, HFS is primarily supported so that older disks can still be accessed.

Unix File System (UFS)

A variant of the standard BSD Fast File System, UFS is a case-sensitive and casepreserving filesystem provided to ensure that applications needing a case-sensitive filesystem can be run. Some people recommend that Unix software developers use UFS, but experience has shown that HFS+ is a much faster filesystem and caseinsensitivity isn't the big problem many make it out to be. If you think you have a need for UFS, you'll want to consider your decision carefully because, under Mac OS X, this filesystem doesn't perform as well as HFS+.

Universal Disk Format (UDF)

The standard format for all DVD media formats including video, DVD-ROM, DVD-RAM, and DVD-RW as well as some writeable CD formats.

ISO-9660

The standard cross-platform file format for CD-ROM data disks.

Audio CD

The format used by standard audio CDs.

MS-DOS File Allocation Table (FAT)

The standard filesystem of MS-DOS; widely used by Microsoft Windows. Mac OS X supports both the 16- and 32-bit variants of FAT.

10.2 Network-based Filesystems

In addition to working with physical filesystems, Mac OS X also supports a set of network-based filesystem protocols. These are used when you mount a filesystem from a machine elsewhere on the network and, for the most part, make a remote filesystem appear as though it were local to your computer. The network filesystem protocols supported by Mac OS X are:

Apple Filing Protocol (AFP)

> The native network filesharing protocol for Mac-based computers. Originally designed to work over AppleTalk, it now operates well over IP-based networks. Most connections to Mac-based filesystems will be AFP-based.

Service Message Block (SMB)/Common Internet File System (CIFS)

> The native network filesharing protocol for Windows-based computers. Known as SMB for most of its life, Microsoft started standardization of the protocol under the CIFS name in the late 1990s but these efforts were never finalized. This protocol is implemented in Mac OS X by the Samba suite of software (www.samba.org/).

Network File System (NFS)

> An older, Unix-based network filesystem used by Linux, the various BSD systems, Solaris, AIX, and other Unix variants. NFS allows Unix machines to transparently share filesystems in such a way that the Unix user security model is preserved.

Web-Based Distributed Authoring and Versioning (WebDAV)

> A set of extensions to HTTP that allows you to collaboratively edit and manage files on remote web servers. WebDAV is the native protocol used for mounting iDisk shares from .Mac. You can find more information about WebDAV at www.webdav.org/.

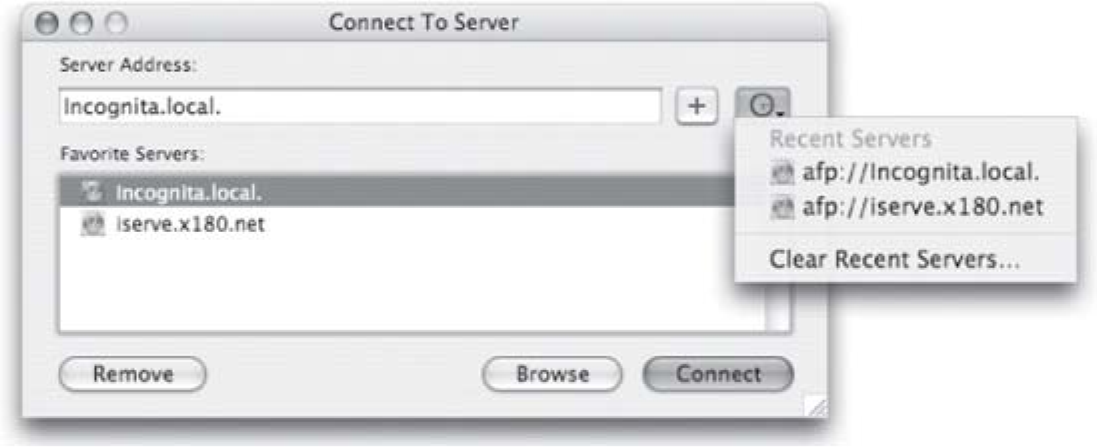
File Transfer Protocol (FTP)

> An older protocol used to transfer files between machines that still is in use on the Internet. Traditionally, FTP has been used with an FTP client (similar to how the web is browsed with a web browser), however, if you want you can mount an FTP site as a drive.

10.2.1 Mounting a Network File System

The Finder's Go→Connect to Server menu (⌘-K), shown in Figure 10-1, gives you a simple interface for mounting remote disks locally. There are four ways to connect to a remote server:

Figure 10-1. Connecting to a server with the Finder



Server Address

Here you can type the hostname or IP address of the server that you want to connect to. The Finder will assume that you want to connect via AFP if you simply enter a hostname or IP address, but you can also use a URL in this field to connect to AFP, FTP, SMB, and WebDAV filesystems. Table 10-1 gives the syntax for these URLs.

Favorite Servers

A list of servers that you can add by clicking the plus (+) button next to the Server Address field. Use this for servers that you connect to frequently.

Recent Servers

The Clock button gives you access to the servers you have recently connected to. You'll see these recent servers by URL, as shown in Figure 10-1.

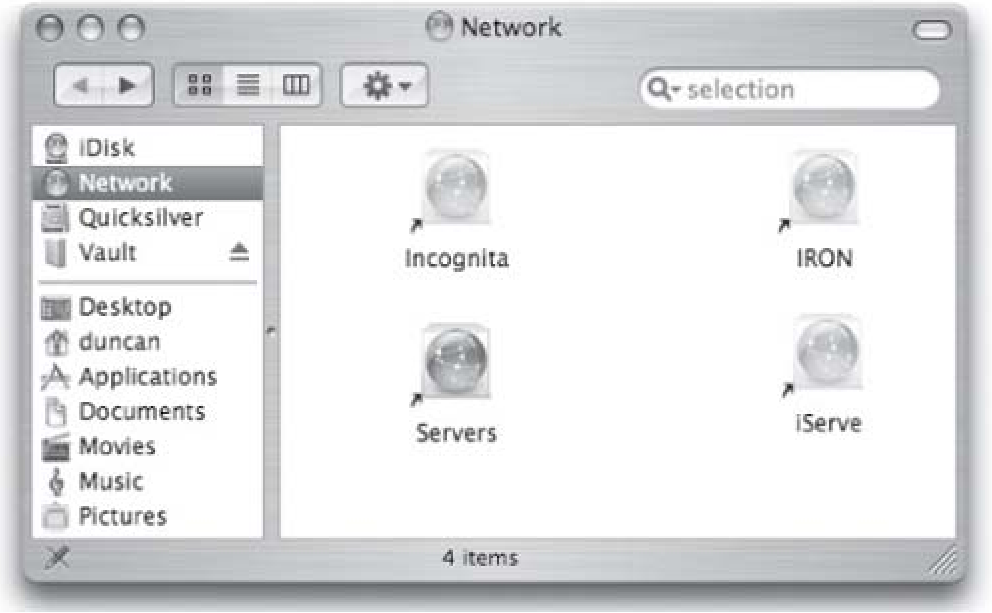
Table 10-1. Network file system types, URL syntax, and command-line mount utilities

Protocol	Filesystem Type	URL Syntax	Command-line Mount Utility
AFP	afp	afp://[*user*]@*host*/[*volumename*]	mount_afp
FTP	ftp	ftp://[*user*]@*host*/[*path*]	mount_ftp
NFS	nfs	nfs://[*user*]@*host*/[*path*]	mount_nfs
SMB/CIFS	smbfs	smb://[*user*]@*host*/[*sharename*]	mount_smbfs
WebDAV	webdav	http://[*user*]@*host*/[*path*]	mount_webdav

Browse

When you click the Browse button, a Finder view will open focused on the Network, as shown in Figure 10-2. This will provide you access to the various Mac- and Windows-based filesystems that are on the local network.

10-2. The Network browser in the Finder



When you connect to a network-based filesystem using Connect to Server, it will show up as a drive in the Finder and will be mounted into the filesystem in the */Volumes* directory. If you want to mount a network-based filesystem into a different directory than */Volumes*, you'll need to use the*mount* command-line utility. The mount command has the following general syntax:

```
mount -t filesystemtype url mountpoint
```

The —*t* option lets you specify the type of filesystem you want to mount. The *mountpoint* must be a directory that already exists in the filesystem. For example, to mount a .Mac user's Public folder, issue the following commands:

```
$ mkdir -r /Disks/iDisk

$ mount -t webdav http://idisk.mac.com/runningosx-Public /Disks/iDisk
```

The first command, *mkdir —r /Disks/iDisk* creates a directory and subdirectory in the filesystem that you can use for mounting a disk volume to. The second command actually mounts the iDisk Public folder, which belongs to the .Mac member named *runningosx*, into the */Disks/iDisk* directory.

There are also specific *mount* command variants for each protocol. These are listed in Table 10-1.

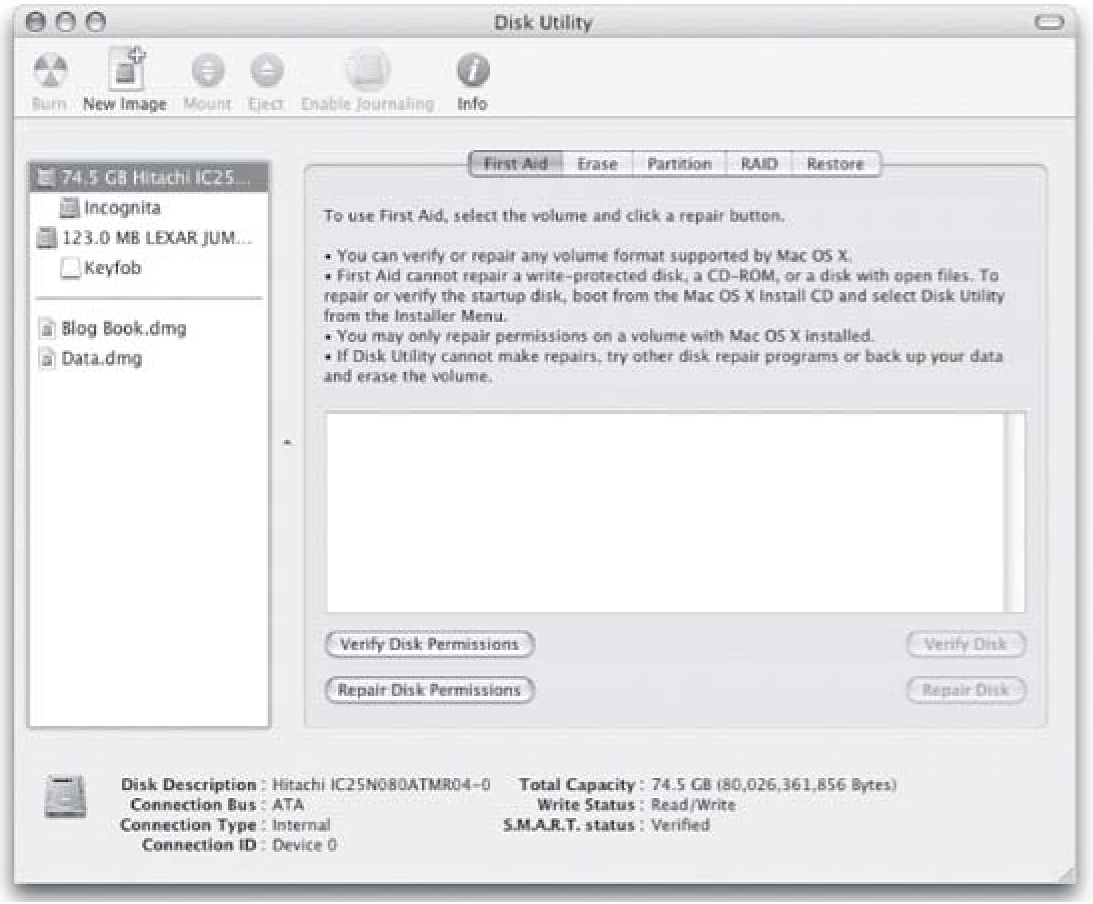
10.3 Disk Utility

The primary tool for working with disks and filesystems is the Disk Utility (*/Applications/Utilties*), shown in Figure 10-3. The Disk Utility can be used to configure, format, eject, and partition disks of all kinds. On the left side of the Disk Utility interface is a list of the disks attached to your machine and the volumes that exist on those disks. Additionally, if you have mounted any disk images, a list of the most recently accessed files appears at the bottom of the left-hand column. On the right side of the interface is a set of panels that give you access to the actions that you can accomplish with a drive. At the bottom of the window is a status display that gives you all sorts of information about the disk or volume you have selected.

Some of the tasks that can be performed with Disk Utility are:

- Mount and eject disks, including hard drives, CDs, and disk images.
- Get the type, format, capacity, and room available for any disk attached to your computer.

Figure 10-3. Disk Utiltity



To use First Aid, select the volume and click a repair button.

- You can verify or repair any volume format supported by Mac OS X.
- First Aid cannot repair a write-protected disk, a CD-ROM, or a disk with open files. To repair or verify the startup disk, boot from the Mac OS X Install CD and select Disk Utility from the Installer Menu.
- You may only repair permissions on a volume with Mac OS X installed.
- If Disk Utility cannot make repairs, try other disk repair programs or back up your data and erase the volume.

Disk Description : Hitachi IC25N080ATMR04-0
Connection Bus : ATA
Connection Type : Internal
Connection ID : Device 0

Total Capacity : 74.5 GB (80,026,361,856 Bytes)
Write Status : Read/Write
S.M.A.R.T. status : Verified

- Check and repair disks (other than the boot disk).
- Check and repair disk permissions.
- Erase a disk, including erasure by writing random data to a disk.
- Partition a disk into multiple volumes which subdivides the drive into multiple logical disks.
- Create and work with disk images.
- Restore the contents of a disk image to a disk.
- Set up a Redundant Array of Independent Disks (RAID).

A command-line interface tool called *diskutil* can be used to perform all the features of the GUI Disk Utility program and more. To get a list of the disks attached to your computer, execute the command*diskutil list* as shown in Example 10-1.

Example 10-1. Getting a list of drives using the diskutil command

```
$diskutil list
/dev/disk0
   #:                   type name          size      identifier
   0: Apple_partition_scheme               *74.5 GB  disk0
   1:    Apple_partition_map               31.5 KB   disk0s1
   2:         Apple_Driver43               28.0 KB   disk0s2
   3:         Apple_Driver43               28.0 KB   disk0s3
   4:       Apple_Driver_ATA               28.0 KB   disk0s4
   5:       Apple_Driver_ATA               28.0 KB   disk0s5
   6:         Apple_FWDriver               256.0 KB  disk0s6
   7:     Apple_Driver_IOKit               256.0 KB  disk0s7
   8:          Apple_Patches               256.0 KB  disk0s8
   9:              Apple_HFS Incognita     74.5 GB   disk0s9
/dev/disk1
   #:                   type name          size      identifier
   0: Apple_partition_scheme               *27.9 GB  disk1
   1:    Apple_partition_map               31.5 KB   disk1s1
   2:              Apple_HFS Backup        27.8 GB   disk1s3
```

This view gives you much more information than the list of drives and volumes on the left side of the Disk Utility application. In particular, it shows you that more than just a single volume is associated with a disk. In fact there are several parts, sometimes called *slices*, to a disk, each with an identifier. The first slice of the disk holds the partition map. This is a key to where everything is located on the rest of the disk. Then a group of other partitions follows. In the case ofExample 10-1, slices 2 through 8 of *disk0* hold disk drivers used by Mac OS 9, and slice 9 is the actual HFS+ volume of the drive. The Mac OS 9 drivers are installed by default when you a format a disk. If you format an internal boot drive without the OS 9 drivers, you'll see something like the output in Example 10-2, where only the partition map and the main HFS+ volume appear.

10-2. The list of partitions of a disk without OS 9 drivers

```
$diskutil list

/dev/disk0

   #:                   type name          size      identifier

   0: Apple_partition_scheme               *37.3 GB  disk0

   1:    Apple_partition_map               31.5 KB   disk0s1

   2:              Apple_HFS TiBook        37.1 GB   disk0s3
```

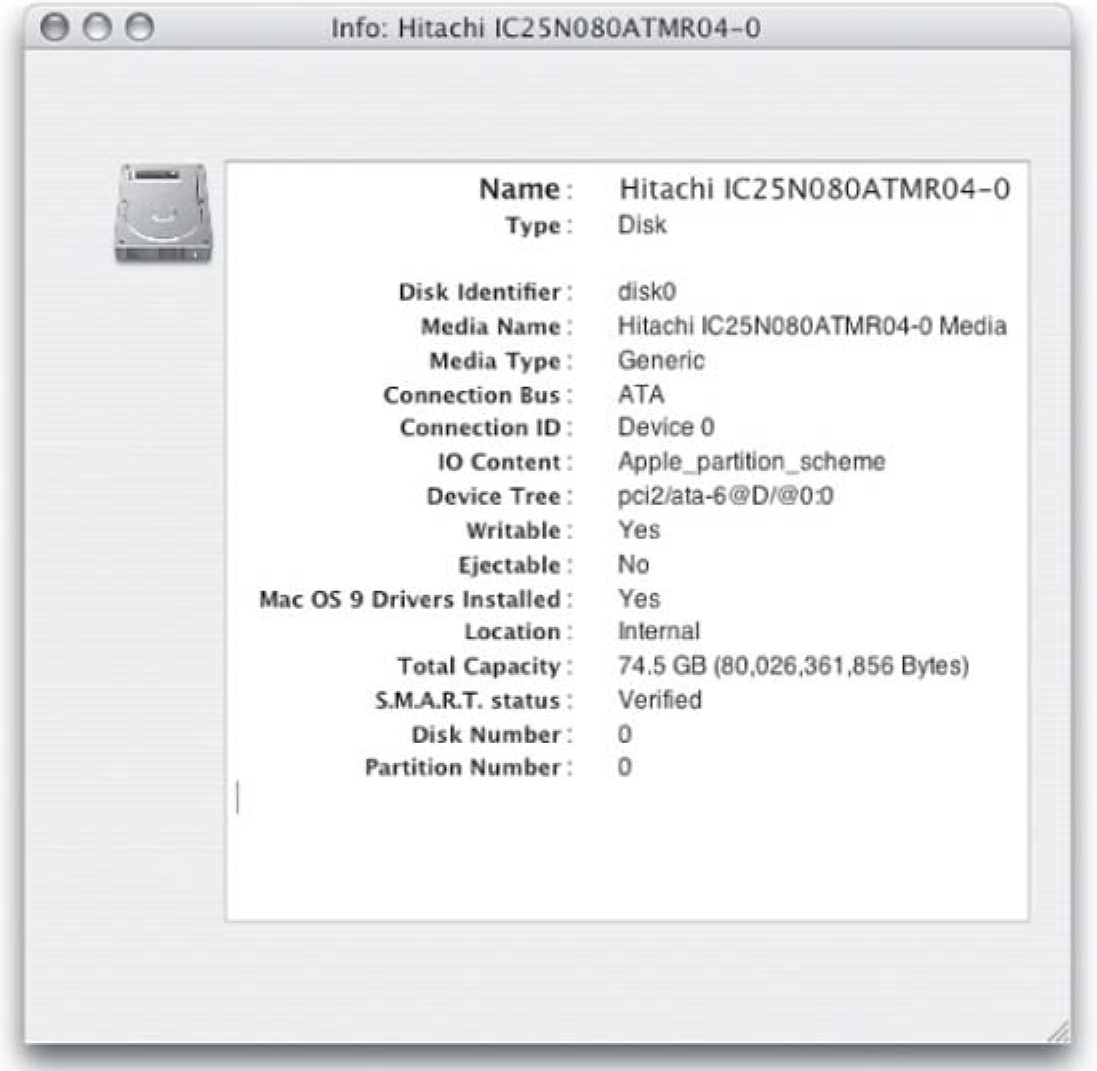
10.3.1 Examining a Disk

To take a closer look at a device attached to your computer or a volume associated with a device, select the disk or volume and then click the Info button. A window pops up listing all sorts of information about the device or volume, as shown in Figure 10-4. Some of the fields that you'll see in the Info window for a disk are:

Name

The name of the device or volume. In the case of a volume, this will be the name you gave the volume and is that which appears in the Finder. In the case of a device, this will be the hardware name of the disk, such as Hitachi IC25N080ATMR04- 0.

Figure 10-4. Disk Utility's Disk Info window



Info: Hitachi IC25N080ATMR04-0

Name : Hitachi IC25N080ATMR04-0
Type : Disk

Disk Identifier : disk0
Media Name : Hitachi IC25N080ATMR04-0 Media
Media Type : Generic
Connection Bus : ATA
Connection ID : Device 0
IO Content : Apple_partition_scheme
Device Tree : pci2/ata-6@D/@0:0
Writable : Yes
Ejectable : No
Mac OS 9 Drivers Installed : Yes
Location : Internal
Total Capacity : 74.5 GB (80,026,361,856 Bytes)
S.M.A.R.T. status : Verified
Disk Number : 0
Partition Number : 0

Disk Identifier

> The label by which the operating system refers to the disk. For example, the primary boot disk in most Macs is identified as *disk0*. A volume on the drive has an additional set of characters identifying the partition on the drive it occupies, such as *disk0s9*. The disk identifier is also the filename in the */dev* directory under which programs may access the drive directly.

Connection Bus

> The kind of bus that the device is attached to. The internal disk in a PowerBook will show *ATA* for this field while an external FireWire drive will show *FireWire*.

Device Tree

The identifier in the system's device tree that the device occupies. This is the same identifier that shows up in the Open Firmware device tree.

Writable

Indicates whether or not the device can be written to.

Ejectable

Indicates whether or not the device can be ejected.

Total Capacity

The amount of data that the device can hold.

S.M.A.R.T. Status

Many disks made in the last few years contain a set of on-board diagnostic functions known as S.M.A.R.T., which stands for *Self-Monitoring, Analysis, and Reporting Technology*. These disks monitor disk calibration, cyclic redundancy check (or CRC) errors, disk spin-up time, rotational speed of the disk, distance between the head and the disk, and the temperature of the drive. The values you see in this field are either *Verified*, which means everything is okay, *About to Fail*, which means the disk needs to be replaced immediately, or *Not Supported*, which means the disk doesn't support S.M.A.R.T.

If you examine a volume on a disk with the Info button, you'll see more information about the disk, including:

File System

The type of filesystem that the volume is formatted to use.

Capacity, Free Space, Used

The total amount of data that can be stored on the volume, the amount of additional data that can be stored on the volume, and how much space the current contents of the volume occupy.

Number of Files, Number of Folders

The number of files and folders on the volume.

Permissions Enabled, Can Turn Permissions Off, Can Repair Permissions

> Indicates whether or not the system keeps and enforces permission information on files. This setting will always be Yes for the internal boot drive of your machine and will usually be No for external drives where it is expected that you'll be moving data between machines. For disks that support permissions, the *Can Repair Permissions* setting indicates whether or not you can use Disk Utility to set the permissions of a disk to work with the current system.

Supports Journaling, Journaled

> Indicates whether or not the volume supports journaling and whether or not it is enabled.

You can also get information on a disk using the *diskutil* tool. Two commands than can be used with*diskutil* are:

```
diskutil info device

diskutil info mountpoint
```

The first command uses a disk identifier, such as *disk0*, to locate the disk to obtain the information of. The second command uses the location in the filesystem where the device is mounted, such as /. Example 10-3 shows the result of running *diskutil info* against the boot drive of a machine.

Example 10-3. Getting disk information for the boot disk drive using the disk identifier

```
$diskutil info disk0

   Device Node:        /dev/disk0

   Device Identifier:  disk0

   Mount Point:

   Volume Name:


   Partition Type:     Apple_partition_scheme

   Bootable:           Not bootable

   Media Type:         Generic
   Protocol:           ATA
   SMART Status:       Verified

   Total Size:         74.5 GB
   Free Space:         0.0 B

   Read Only:          No
   Ejectable:          No
   OS 9 Drivers:       Yes
   Low Level Format:   Not Supported
```

Example 10-4 shows the information about the volume mounted at /.

Example 10-4. Getting disk information for the root volume by using the mountpoint

```
$diskutil info /
   Device Node:        /dev/disk0s9
   Device Identifier:  disk0s9
   Mount Point:        /
   Volume Name:        Incognita

   File System:        Journaled HFS+
                       Journal size 8192 k at offset 0x831000
   Permissions:        Enabled
   Partition Type:     Apple_HFS
   Bootable:           Is bootable

   Media Type:         Generic
   Protocol:           ATA


   Total Size:         74.5 GB
   Free Space:         63.5 GB


   Read Only:          No
   Ejectable:          No
```

From this output, you can see that the filesystem is formatted as Journaled HFS+, and that the journal size is a mere 8192 KB. Even though it is modest in size, the journal has a large impact on the integrity of the filesystem and allows for many performance optimizations.

10.3.2 Verifying and Repairing Disk Permissions

One of the most common things that can go wrong with a disk is that the permissions for the various files and directories on it get set incorrectly. Although the permissions for a disk should never get out of whack, quite a few installers that require an administrator's password seem to set the permissions on various directories of your disk to what they think they should be. This is one reason why many system administrators abhor old-style software installers and prefer to install software distributed on disk images, which can be drag-and-dropped into the Applications folder without harm.

To see if the permissions on your disk are correct for your system, select the disk or volume you want to check and click the Verify Disk Permissions button in the First Aid panel of the Disk Utility interface. This determines what the correct file permissions are for your system and then checks your disk against those permissions. If your permissions are not set correctly, you should strongly consider repairing them by clicking the Repair Disk Permissions button.

To check permissions from the command line, use the following syntax to *diskutil*:

```
diskutil verifyPermissions [diskid | mountpoint]
```

For example, to verify the permissions of the boot volume, you would use the following command:

```
$ diskutil verifyPermissions /
```

To repair permissions, use the following:

```
$ diskutil repairPermissions /
```

10.3.3 Verifying and Repairing Disks

To ensure that the filesystem on a disk is working correctly, select the disk or volume that you want to check and click the Verify Disk button. This scans the various parts of the filesystem to make sure they are intact. If any problems are reported, you should use the Repair Disk button to correct the issues.

In order to verify or repair a disk, Disk Utility must first unmount the drive, removing the ability to access the files on the disk while it performs its work. This means you can't verify or repair the disk you booted from, which in most cases is the disk that you want to verify. There are two solutions to this problem, and they work equally well:

- Boot off Disk 1 of the Mac OS X install CD set. When the installer shows up, select Installer →Open Disk Utility from the menu bar so you can work on the local disk.
- Put your machine into FireWire target disk mode by booting the machine and holding down the T key. Then plug the machine into another Mac using a FireWire cable and use Disk Utility on the other machine to verify and repair the disk.

If Disk Utility runs into a problem it can't handle, you'll need to enlist the help of another application such as Drive 10 from Micromat (www.micromat.com/) or Disk Warrior from Alsoft (www.alsoft.com/DiskWarrior/).

To verify and repair a disk from the command line using *diskutil*, you can use one of the following:

```
diskutil verifyDisk [diskid | mountpoint]

diskutil repairDisk [diskid | mountpoint]
```

Example 10-5 shows the command you should use to repair a FireWire disk named *Backup*:

Example 10-5. Repairing a disk using the diskutil command

```
$diskutil repairDisk /Volumes/Backup
```

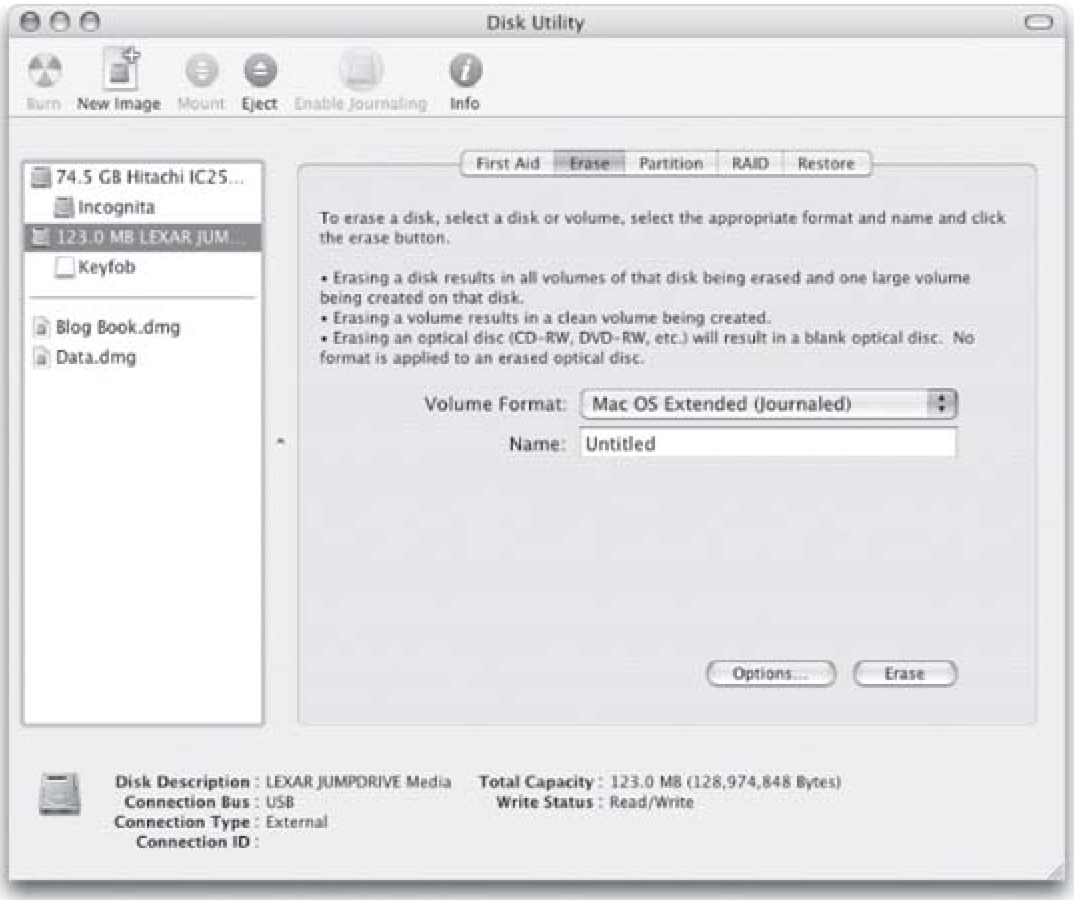
```
Started verify/repair on disk disk1s3 Backup

Checking HFS Plus volume.

Checking Extents Overflow file.

Checking Catalog file.

Checking Catalog hierarchy.

Checking volume bitmap.

Checking volume information.

The volume Backup Alpha appears to be OK.

Verify/repair finished on disk disk1s3
```

10.4 Erasing and Formatting Disks

There are many times when you want to "start over" with a drive, such as when you are upgrading your Mac OS X installation and would rather start from scratch, or when you want to make sure that all of your data is removed from a FireWire drive. Erasing a disk with the Disk Utility is very simple, almost too simple considering the fact that all the data on the disk will be obliterated.

The Erase tab cof Disk Utility, shown in Figure 10-5, lets you erase and format a disk with the following options:

Figure 10-5. Erasing a disk using Disk Utility



To erase a disk, select a disk or volume, select the appropriate format and name and click the erase button.

• Erasing a disk results in all volumes of that disk being erased and one large volume being created on that disk.
• Erasing a volume results in a clean volume being created.
• Erasing an optical disc (CD-RW, DVD-RW, etc.) will result in a blank optical disc. No format is applied to an erased optical disc.

Volume Format: Mac OS Extended (Journaled)
Name: Untitled

Disk Description : LEXAR JUMPDRIVE Media
Connection Bus : USB
Connection Type : External
Connection ID :
Total Capacity : 123.0 MB (128,974,848 Bytes)
Write Status : Read/Write

Volume Format

The Volume Format pull-down list lets you select the filesystem you want to place on the device. For most purposes you'll want to use Mac OS Extended (Journaled).

Name

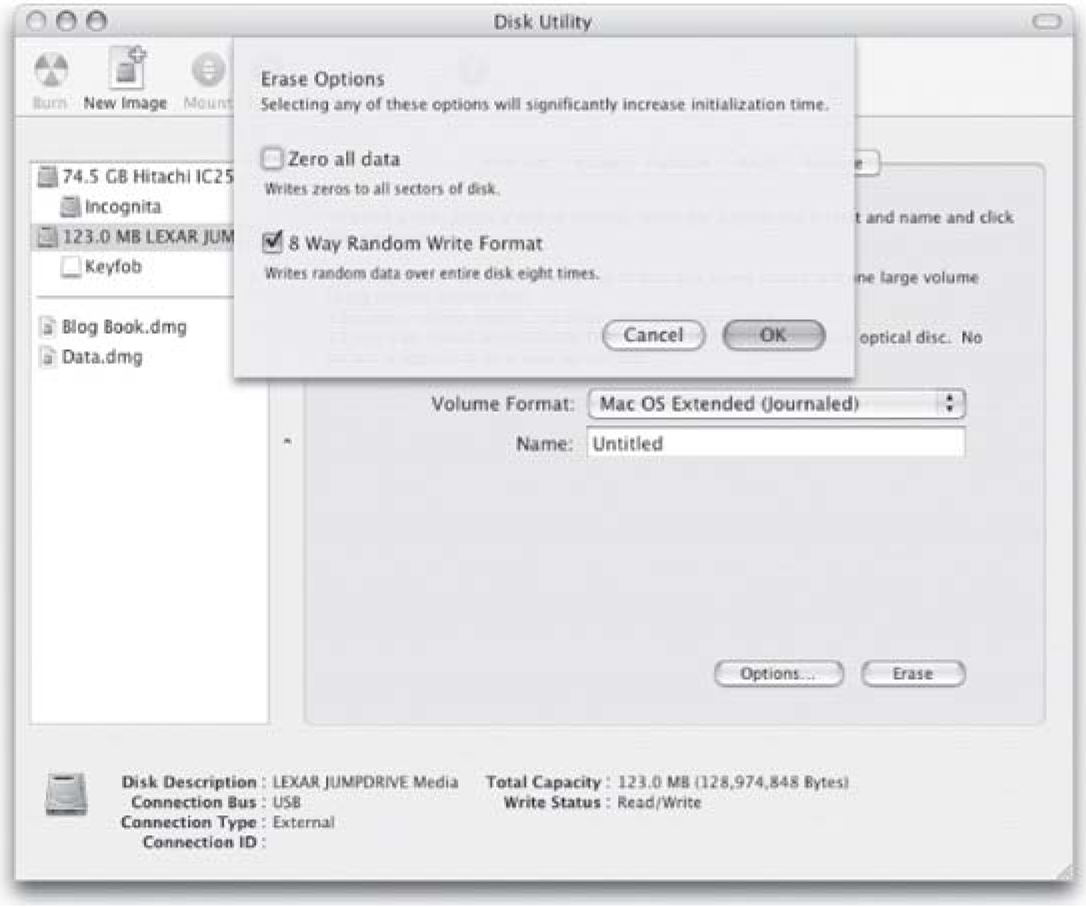
Lets you name the volume that will be created on the disk.

There are additional options hiding behind the Options button, shown in Figure 10-6. These options are:

Zero all data

This option forces Disk Utility to go beyond just setting up a new filesystem on the disk, resulting in every bit on the disk being set to 0. This ensures that the data previously on the drive can't be accessed. Because Disk Utility has to write a 0 to every part of the drive, this option isn't nearly as fast as just erasing and formatting a drive.

Figure 10-6. Erase options in Disk Utility



Erase Options

Selecting any of these options will significantly increase initialization time.

☐ Zero all data

Writes zeros to all sectors of disk.

☑ 8 Way Random Write Format

Writes random data over entire disk eight times.

Volume Format: Mac OS Extended (Journaled)

Name: Untitled

Disk Description : LEXAR JUMPDRIVE Media
Connection Bus : USB
Connection Type : External
Connection ID :

Total Capacity : 123.0 MB (128,974,848 Bytes)
Write Status : Read/Write

8 Way Random Write Format

This option is for the truly paranoid. If you select this option, Disk Utility randomly writes data to every bit of storage eight times, making it highly unlikely for anybody to recover data from the disk. Even a company like Drive Savers (www.drivesavers.com) would be hard-pressed to find anything usable on the disk, even after they disassembled it and used very expensive tools to analyze the magnetic structure of the disk. The downside is that an 8-way write takes a long time to format a large disk.

To use the *diskutil* command to erase a disk, use one of the following commands:

```
diskutil eraseDisk [format] newName [diskid | mountpoint]

diskutil zeroDisk [diskid|mountpoint]

diskutil randomDisk [numberOfPasses] [diskid|mountpoint]
```

For example, to format an external FireWire disk that had the device ID of disk1, use the following:

```
$ diskutil eraseDisk "Journaled HFS+" Backup disk1
```

Why Write Eight Times?

So, why would you want to take the time to write random data eight times over the surface of your hard drive? Well, when you write data to a disk, there is still a faint magnetic remnant of the original data on the drive. Very sophisticated equipment can detect these magnetic traces and recreate data that was once on the drive but erased. Each time data is written to a spot on a disk, the traces from previous data at that spot get covered up and fade. After eight writes, it should be close to impossible to resurrect data from any traces that remain.

To write zeros across the entire filesystem of a FireWire disk named *Backup*, use the following:

```
$ diskutil zeroDisk /Volumes/Backup
```

One thing to keep in mind when either zeroing or writing random data to a disk is that both operations take quite a while to perform on large disks. After issuing the command, you might want to go off and refinance your house; the commands might be done by the time you get back.

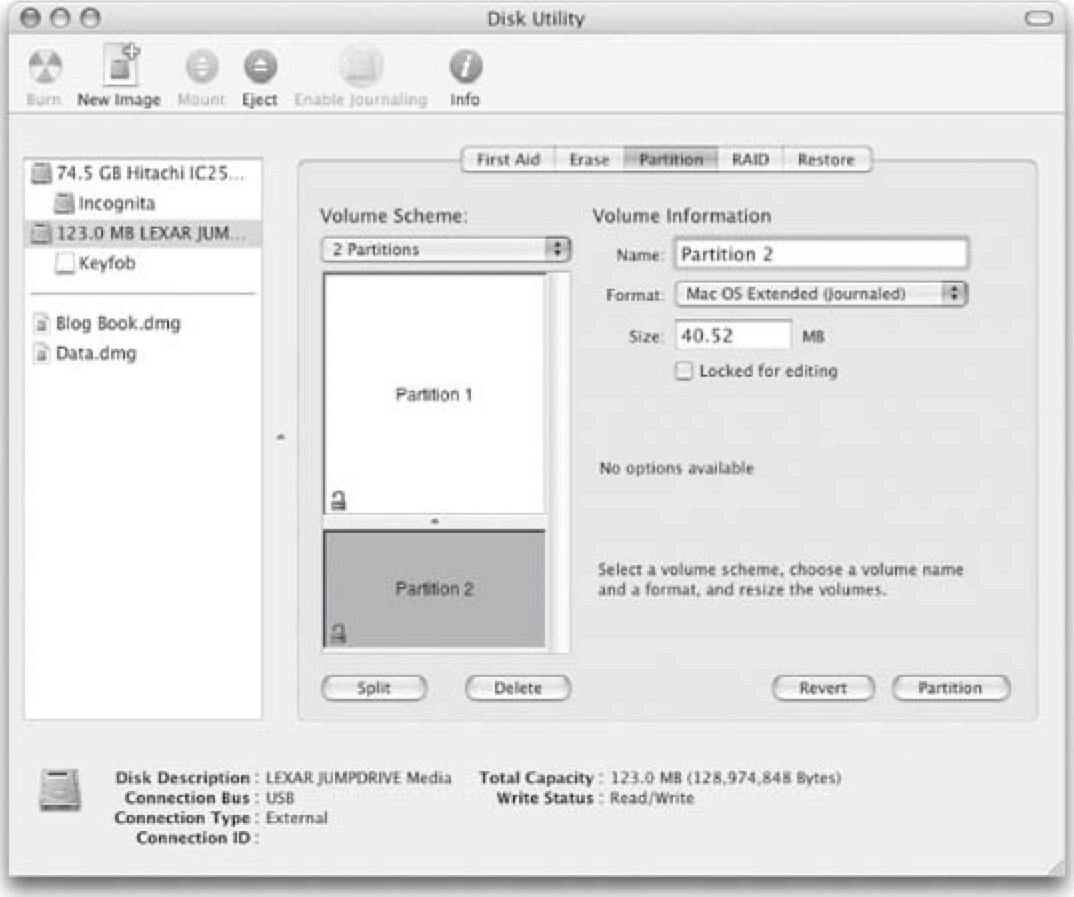
10.5 Partitioning Disks

For the most part, you'll want to assign a single volume to each of your hard drives. It's quick and easy and is usually the most efficient use of space. However, there are times when you might want to have more than one volume on a drive. For example, while working with the first few releases of a new version of Mac OS X, you might want to have a test partition for the new version while keeping a stable, known version available. Or you might want to keep an old copy of Mac OS 9 available on a separate partition from your Mac OS X installation for running Classic.

> The best and only time to partition your disk is when you're installing the operating system. Once the system is installed, you can't go back later and partition the drive or adjust the partition sizes. So, if you want to partition your drive, you must first back up all your critical data to another drive, then do a clean install of Mac OS X, using the Disk Utility on Install Disc 1. For details on partitioning and installing Mac OS X, see Appendix A, *Installing Panther*.

To create multiple volumes on a disk, select the disk that you want to split up and click the Partition tab. This brings up an interface that lets you configure how much space to give each volume on a disk as shown in Figure 10-7. From the volume scheme pop-up menu choose the number of partitions you want and size them however you want using either the graphical tool or typing sizes for each partition. Then click the Partition button to commit the changes. This erases all the data on the disk, creates the volumes, and then creates the filesystems for each volume.

Figure 10-7. Paritioning a disk using the Disk Utility



Volume Scheme: 2 Partitions

Volume Information

Name: Partition 2

Format: Mac OS Extended (Journaled)

Size: 40.52 MB

☐ Locked for editing

No options available

Select a volume scheme, choose a volume name and a format, and resize the volumes.

Disk Description : LEXAR JUMPDRIVE Media
Connection Bus : USB
Connection Type : External
Connection ID :

Total Capacity : 123.0 MB (128,974,848 Bytes)
Write Status : Read/Write

10.5.1 Adding a Disk Drive

For the most part, adding an internal disk drive to your computer is as easy as making sure that you get a drive with an interface that your computer supports (IDE drives for most G3 and G4 machines, Serial ATA for the newer G5 machines) and that will fit into your Mac. Adding an external FireWire drive is even easier; you simply plug it in. If the FireWire drive is formatted properly, it should automatically appear in your Finder and in Disk Utility. If the drive is not formatted, it won't appear in the Finder, but it will appear in Disk Utility.

When you first add a drive to a machine, you should make sure that it is formatted the way you want. Often, new hard drives arrive from the factory preformatted as MSDOS FAT. If you encounter this with a new drive, you will need to reformat the drive as Journaled HFS using the Disk Utility. The only exception to this rule is if you want to share an external FireWire drive with a Windows machine, in which case you'll want to leave the drive formatted as MS-DOS FAT.

Naming Your Partitions

All Macs come with their hard drives named *Macintosh HD*. Not only is this name boring, but when you enable file sharing on multiple computers, you don't really want to mount multiple drives all with the same name across the network. I recommend that you name your hard drive something distinctive. I usually name the boot drives of my machines with the same name I give the machine itself. That way, when I have a bunch of disks mounted from various machines, I know which is which.

10.6 Disk Images

Disk images (files with a *.dmg* extension) have become a frequent part of life on Mac OS X. Originally created to store the block-for-block contents of a floppy disk for ease of duplication, the disk images created and used by Panther are based on a format called Universal Disk Image Format (UDIF). UDIF allows the storage of the same partition tables, disk drivers, and volumes found on physical disks, which lets disk images to serve as an intermediary in the duplication of any kind of disk, including CDs and DVDs.

Beyond their origin as a disk duplication format and because of their ease-of-use, disk images are used by many software vendors, including Apple, as a distribution format instead of *.sit* (StuffIt), *.tar*, or *.zip* files. When you double-click a disk image (or use Disk Utility to open it), Mac OS X uses the contents of the image as the data for a disk that it mounts into the filesystem, which shows up in the Finder as shown in Figure 10-8;Figure 10-9 shows the same disk image in Disk Utility.

10.6.1 Types of Disk Images

Because they can be used for many different purposes, disk images have several different types. These are:

Read/Write disk image

> A fixed-size disk image that doesn't just contain data but also allows data to be written to it. These disk images are useful if you want to have an encrypted data storage point, or if you want to set up the contents of a disk image before converting it to one of the other types. This kind of disk image occupies the same amount of space on a disk as its capacity.

Read Only disk image

> A fixed-size disk image that only allows data to be read from it. This kind of disk image occupies the same amount of space on a disk as its capacity.

Figure 10-8. A disk image mounted in the Finder

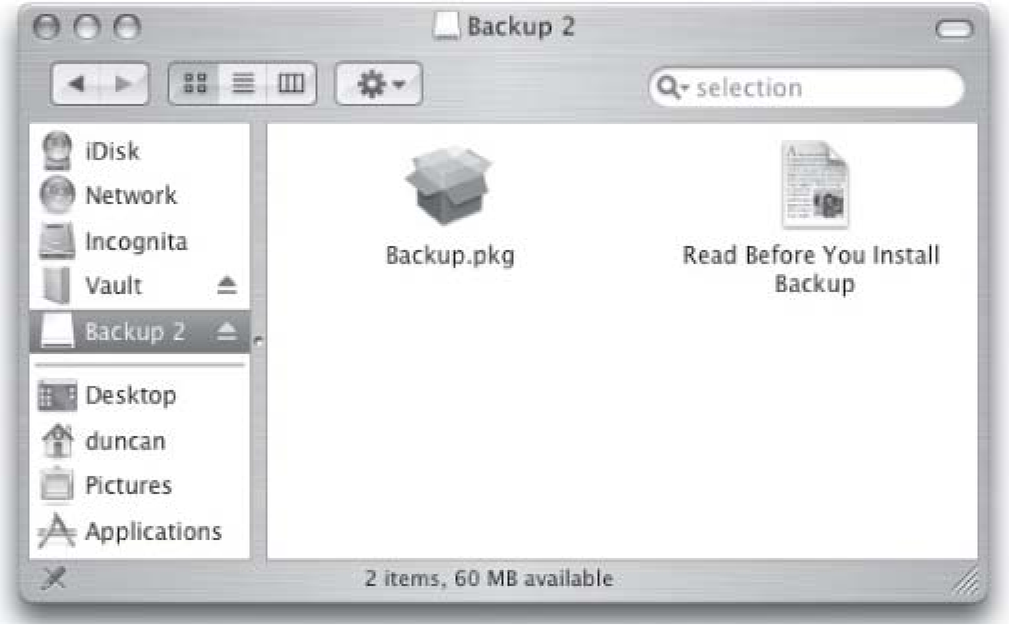
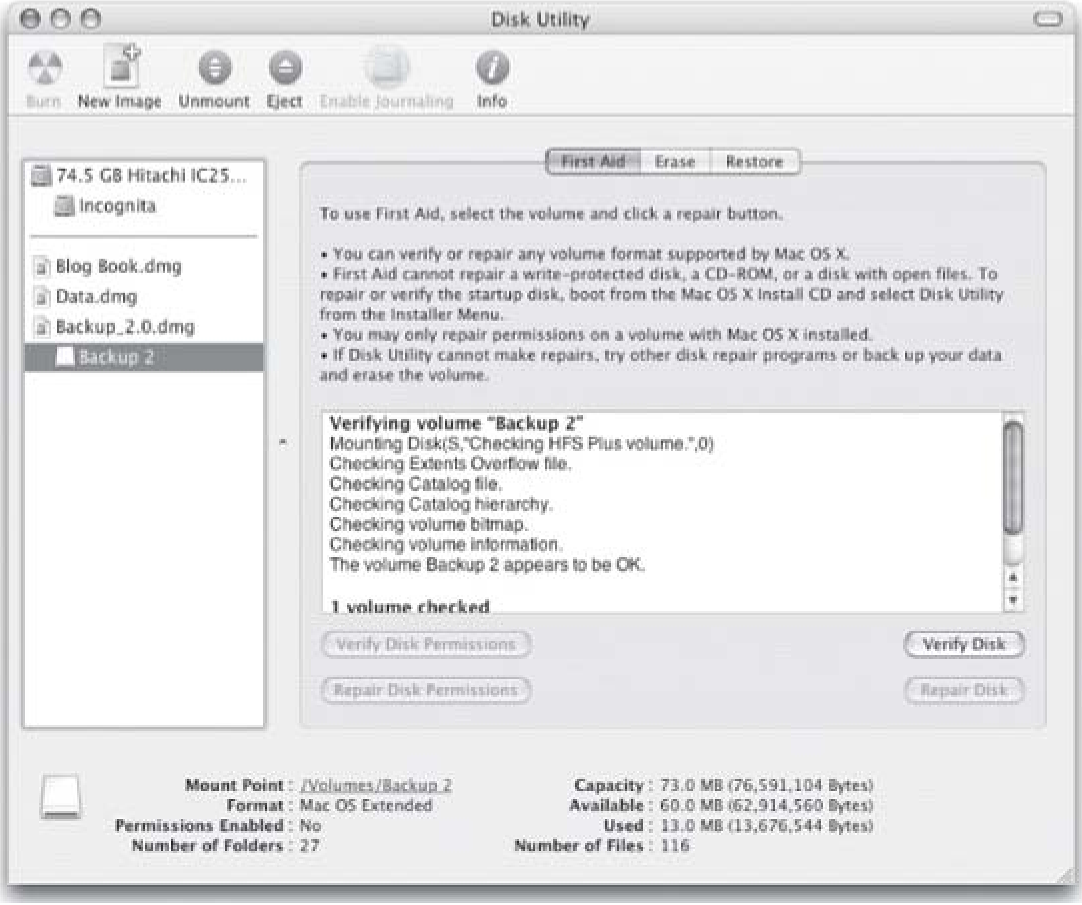


Figure 10-9. A disk image mounted in Disk Utility



To use First Aid, select the volume and click a repair button.

- You can verify or repair any volume format supported by Mac OS X.
- First Aid cannot repair a write-protected disk, a CD-ROM, or a disk with open files. To repair or verify the startup disk, boot from the Mac OS X Install CD and select Disk Utility from the Installer Menu.
- You may only repair permissions on a volume with Mac OS X installed.
- If Disk Utility cannot make repairs, try other disk repair programs or back up your data and erase the volume.

```
Verifying volume "Backup 2"
Mounting Disk(S,"Checking HFS Plus volume.",0)
Checking Extents Overflow file.
Checking Catalog file.
Checking Catalog hierarchy.
Checking volume bitmap.
Checking volume information.
The volume Backup 2 appears to be OK.

1 volume checked
```

Mount Point : /Volumes/Backup 2
Format : Mac OS Extended
Permissions Enabled : No
Number of Folders : 27

Capacity : 73.0 MB (76,591,104 Bytes)
Available : 60.0 MB (62,914,560 Bytes)
Used : 13.0 MB (13,676,544 Bytes)
Number of Files : 116

Compressed disk image

A read-only disk whose contents have been compressed, resulting in the size of the disk image being much less than its capacity. This is the preferred format for distributing software across the Internet.

CD/DVD Master disk image

A disk image whose internal format uses either ISO-9660 or UDF so its contents can be directly burned to a CD or DVD. CD and DVD master disk images end with the *.cdr* extension.

Sparse disk image

A read/write disk image format that starts out smaller on disk than its original capacity. The disk image grows in size as needed up to either its capacity or the size of the disk it is located on, whichever is less. Sparse disk images end with the *.sparseimage* extension.

10.6.1.1 Encryption and disk images

One attribute that disk images don't share with their physical cousins is that all the data on a disk image can be encrypted. This enables you to store data on public disks that can't be accessed by anybody who doesn't have the proper password. For example, you could store valuable data on a keychain USB flash drive using an encrypted disk image with the knowledge that if somebody gains access to your keys, he won't be able to gain access to the data stored on the USB drive. As well, these encryption features are used by FileVault to protect your Home directory, if you enable this feature in System Preferences.

10.6.2 Working with Disk Images on the Command Line

In addition to Disk Utility, Mac OS X provides the *hdiutil* command-line tool to work with disk images. This tool has a wealth of options to create and manipulate disk images, many of which venture into very arcane territory. I'm going to cover a few of the most commonly-used features of*hdiutil* so you can see how to manipulate disk images from the command line.

To mount a disk image, use the following command:

```
hdiutil mount imagename
```

This command mounts the contents of the disk image into the */Volumes* folder and it appears as a disk in the Finder's Sidebar. Example 10-6 shows a disk image named *test.dmg* being mounted.

Example 10-6. Mounting a disk image using hdiutil

```
$hdiutil mount test.dmg

Initializing...

Attaching...

Finishing...

Finishing...

/dev/disk2          Apple_partition_scheme

/dev/disk2s1        Apple_partition_map

/dev/disk2s2        Apple_HFS                     /Volumes/test
```

When you mount a disk image, you'll see several items that should be familiar from looking at disks. There are a series of slices to the disk including a partition map. The output of the *diskutil* command also gives you the filesystem location where the disk image was mounted.

You aren't limited to mounting disk images into the */Volumes* folder. The following command can be used to mount the contents of a disk image into any directory in the filesystem:

```
  hdiutil mount -mountpoint directory imagename
```

The only catch here is that the directory you are mounting the disk drive into has to exist in the filesystem. The contents of the directory that you are mounting the disk image into will also be hidden by the contents of the disk image. Example 10-7 shows an example of this command:

Example 10-7. Mounting a disk image into the filesystem

```
$hdiutil mount -mountpoint /Foo test.dmg

Password:

Initializing...

Verifying...

Checksumming DDM...

                                 DDM: verified  CRC32 $70362E53

Checksumming Apple (Apple_partition_map : 0)...

     Apple (Apple_partition_map : 0): verified  C
```

```
RC32 $DFA2282D
Checksumming (Apple_Free : 1)...

                    (Apple_Free : 1): verified  CRC32 $00000000
Checksumming Apple_HFS_Untitled_2 (Apple_HFS : 2)...
..............................................................................

Apple_HFS_Untitled_2 (Apple_HFS : 2): verified  CRC32 $2877B09E
Checksumming (Apple_Free : 3)...

                    (Apple_Free : 3): verified  CRC32 $00000000
Verification completed...
verified CRC32 $3CAEE635
Attaching...
Finishing...
Finishing...
/dev/disk5           Apple_partition_scheme
/dev/disk5s1         Apple_partition_map
/dev/disk5s3         Apple_HFS                        /Foo
```

To unmount a disk image use the following command:

```
  hdiutil unmount mountpoint
```

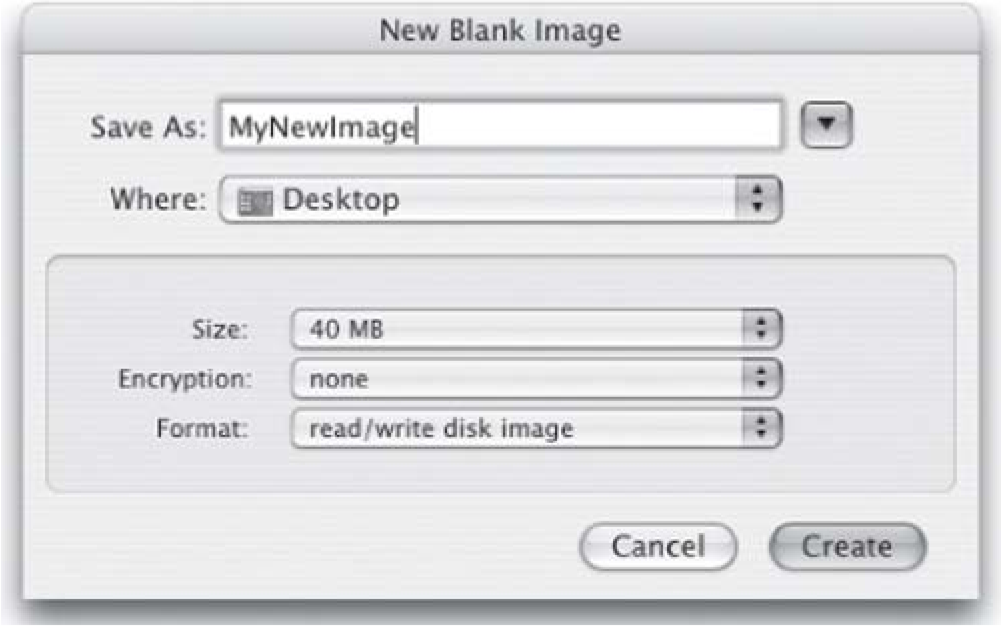
Here, the *mountpoint* is the path of the folder where the disk image is mounted.

10.6.3 Creating an New Disk Image

To create a new empty read/write disk image using Disk Utility:

1. Use Disk Utility's Images→New→Blank Image menu, or the New Image button on Disk Utility's toolbar. A dialog window opens, as shown in Figure 10-10.

Figure 10-10. Options for creating a disk image



New Blank Image

Save As: MyNewImage

Where: Desktop

Size: 40 MB

Encryption: none

Format: read/write disk image

2. Select the size of the disk image. This size governs the disk image's capacity. As with hard drives, keep in mind that the size of a disk image doesn't equal how much data it can actually hold because of filesystem overhead.

3. Select whether or not you want to encrypt the disk image.

4. Select the kind of disk image to create. When you create a new disk image, you have your choice of a read/write disk image or a sparse disk image. For most purposes, the standard read/write disk image is appropriate.

When you hit the Create button, a disk image will be created and mounted into the Finder and into the filesystem in */Volumes*. By default, the disk image contains a single HFS+ volume. To change the format of the volume to HFS or to UFS, simply highlight the volume in Disk Utility and use the Erase tab, just as you would with an actual physical disk.

To create a disk image on the command line, use the following *hdiutil* command:

```
hdiutil create -volname volumename -fs fstype -size size imagename
```

where*volumename* is the name of volume you want to create, *fstype* is one of of the filesystem types listed in Table 10-2, and *size* is the size of the disk image expressed as a number followed by either *b* (which stands for sectors, not bytes), *k* (for kilobytes), *m* (for megabytes), *g* (for gigabytes),*t* (for terabytes), *p* (for pentabytes), or *e* (for exabytes).

Table 10-2. Filesystem type codes used by hdiutil

Filesystem Type Code	Description
HFS+J	Journaled HFS+
HFS+	HFS+ (non journaled)
HFS	Original HFS
MS-DOS	The FAT filesystem used by MS-DOS
UFS	The Unix filesystem (Berkeley Fast File System)

Example 10-8 shows how to create a 10 MB disk image.

Example 10-8. Creating a new disk image

```
$hdiutil create -volname Foo -fs HFS+ -size 10m foo.dmg

Initializing...

Creating...

...............................................................................

Finishing...

created: /Users/duncan/foo.dmg
```

To create an encrypted disk image, add the -*encryption* argument to the command. Applied to the command in Example 10-8, it looks like:

```
  hdiutil create -encryption -volname Foo -fs HFS+ -size 10m foo.dmg
```

By default *hdiutil create* creates read/write disk image files. Creating other kinds of disk image files requires the use of the -*format* argument along with a four-letter code. The complete list of codes is given in Table 10-3. For example, to create a sparse disk image, you would use the following command:

```
  diskutil create -format UDSP -voname Foo -fs HFS+ size 10m foo.dmg
```

Table 10-3. Commonly diskutil disk image format codes

Code	Description
UDRW	Read/write
UDRO	Read only
UDZO	Compressed
UDTO	DVD/CD master

10.6.4 Creating a Disk Image from a Folder

To create a disk image that holds the contents of a folder using Disk Utility:

1. Use Disk Utility's Images → New → Image From Folder, or the New Image button on Disk Utility's toolbar. An Open dialog window appears giving you the chance to select a folder to create the disk image. Select the folder you want to create an image from and hit the Open button.

FileVault Implementation

The new FileVault in Panther, which encrypts all the data in your Home folder, uses disk images under the covers to perform its magic. When you enable FileVault for a user, all the user's data is placed into an encrypted sparse disk image. When a user logs in to a machine, the sparse disk image holding their data is mounted into the */Users* folder. When they log out, the disk image is unmounted, making any data in the user's Home directory inaccessible.

As of version 10.3.1, when you enable FileVault for a user, the data that was in their original Home folder is erased. It is, however, the same kind of erase that happens when you drag files to the trash. This means that any data that was in a user's Home folder before FileVault was enabled could potentially be recovered. The best time to enable FileVault for a user is before any sensitive data is placed into their Home folder.

2. A dialog window, confusingly titled Convert Image, opens, letting you set the options for the disk image, as shown in Figure 10-11.

3. Select the kind of image to create. Since you are creating an image with data, you can select from read-only, compressed, DVD/CD master, as well as read/write.

4. Select whether or not you want to encrypt the contents of the disk image.

When you hit the Create button, the disk image is created. Unlike when you create an image, it will not be mounted.

To create a disk image from a folder on the command line, use the following *hdiutil* command:

```
hdiutil create -srcfolder folder imagename
```

where*folder* is the path to the folder that you want to image and *imagename* is the name of the resulting image. Example 10-9 shows the creation of a disk image from the Documents directory in the Home directory:

Example 10-9. Creating a disk image with the contents of a directory

```
$hdiutil create -srcfolder ~/Documents Documents.dmg

Initializing...

Creating...

Copying...

................................................................................

Converting...

Preparing imaging engine...

Reading DDM...

   (CRC32 $FA1A8A19: DDM)

Reading Apple_partition_map (0)...

   (CRC32 $5BC1024A: Apple_partition_map (0))

Reading Apple_HFS (1)...

...............................................................................

   (CRC32 $B732C490: Apple_HFS (1))

Reading Apple_Free (2)...

   (CRC32 $00000000: Apple_Free (2))

Terminating imaging engine...

Adding resources...

................................................................................
```

```
Elapsed Time: 24.503s

   (1 task, weight 100)

File size: 31918290 bytes, Checksum: CRC32 $001F90E7

Sectors processed: 188034, 173057 compressed

Speed: 3.4Mbytes/sec

Savings: 66.8%

Finishing...

created: /Users/duncan/Documents.dmg
```

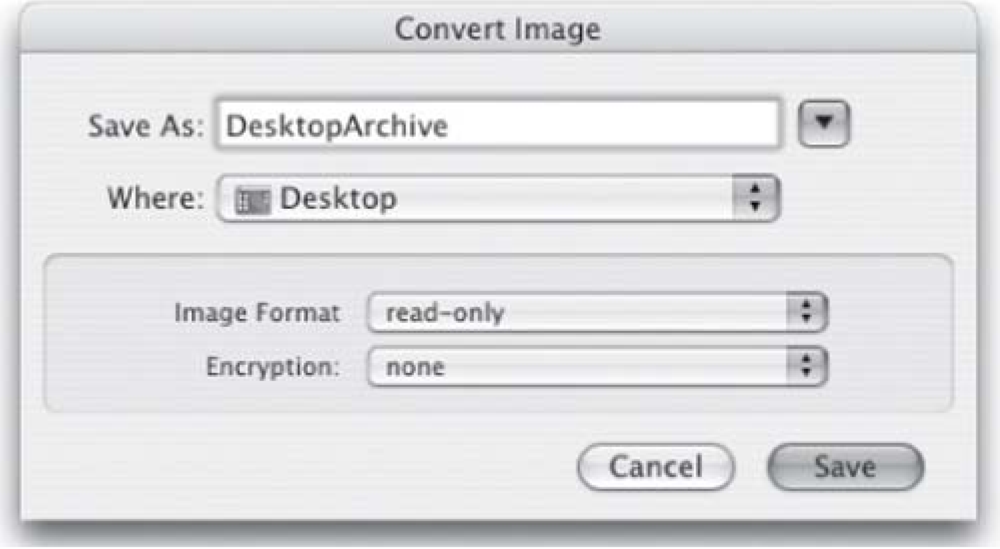
By default, the disk images that *hdiutil* creates from a directory are read-only. To create a read/write disk image, use the following syntax for the *hdiutil* command:

```
  hdiutil create -srcfolder folder -format UDIF imagename
```

10.6.5 Converting a Disk Image

Once you have worked with a disk image in one format, such as read/write, you may want to convert it to read-only format, so that it is compressed and ready to post to the Internet, or convert it into a CD master that is ready for duplication. To convert an image using Disk Utility, use the Images→Convert menu option, select the disk you want to convert, and then use the Covert Disk dialog box that appears to select the kind of image you want to convert it to. This dialog box is shown in Figure 10-11.

Figure 10-11. Options for creating a disk image from a directory



Convert Image

Save As: DesktopArchive

Where: Desktop

Image Format: read-only

Encryption: none

To convert a disk image on the command line, use the *hdiutil convert* command as follows:

```
hdiutil convert -format format -o outputfile imagefile
```

Where*format* is the four-letter code that the disk image will be converted to, *outputfile* is where to save the new disk image, and *imagefile* is the disk image to convert. Example 10-10 shows how to use this command to convert a disk image to a compressed disk image.

Example 10-10. Converting a disk image to a compressed disk image

```
$hdiutil convert -format UDZO -o Compressed.dmg Documents.dmg

Preparing imaging engine...

Reading DDM...

   (CRC32 $EEB3C0C1: DDM)

Reading Apple_partition_map (0)...

   (CRC32 $5A2C9F73: Apple_partition_map (0))

Reading Apple_HFS (1)...

..............................................................................

   (CRC32 $C7097C7F: Apple_HFS (1))

Reading Apple_Free (2)...
```

```
...............................................................................
   (CRC32 $00000000: Apple_Free (2))
Terminating imaging engine...
Adding resources...
...............................................................................
Elapsed Time: 23.061s
 (1 task, weight 100)
File size: 48394357 bytes, Checksum: CRC32 $79BEA1E4
Sectors processed: 258898, 238321 compressed
Speed: 5.0Mbytes/sec
Savings: 63.5%
created: Compressed.dmg
```

10.6.6 Creating a Disk Image from a Device

Since disk images can hold all the data on a disk drive, including the partition map, it's natural that you can create a disk image directly from either a disk drive or a volume on it. By creating a disk image from a drive, you can later restore that image back onto the drive or onto any other drive. You can use this for backup purposes or to create a master disk image for a department or company, making it easy for you to clone one installation onto every machine. To create a disk image from a disk drive using Disk Utility, select the drive and then select Images→New→Image from*diskname* from the menu bar. Here, *diskname* is the name of the volume you've selected in Disk Utility's left pane.

> Keep in mind that if you want to clone the drive or partition that contains your installation of Mac OS X, you shouldn't use Disk Utility to accomplish this task while the system is running. Instead, you should either boot from Panther's Install Disc 1, or put your Mac into target mode and mount its drive(s) via FireWire on another system.

Creating an image of a device from the command line is a bit trickier. There isn't a direct command in *hdiutil*, so you have to use the *convert* command and convert from the disk's device in the */dev* directory. Example 10-11 shows how to create a compressed disk image of an external FireWire drive mounted at */dev/disk1*. Since this command accesses the raw device of the disk, the command must be run by a user with administrative privileges.

Example 10-11. Creating a disk image from a device

```
$sudo hdiutil convert /dev/disk1 -format UDZO -o /FirewireBackup.dmg

Preparing imaging engine...

Reading DDM...

   (CRC32 $70362E53: DDM)

Reading Apple_partition_map (0)...

   (CRC32 $DFA2282D: Apple_partition_map (0))

Reading Apple_Free (1)...

   (CRC32 $00000000: Apple_Free (1))

Reading Apple_HFS (2)...

...............................................................................

   (CRC32 $15DC1C94: Apple_HFS (2))

Reading Apple_Free (3)...

................................................................................

   (CRC32 $00000000: Apple_Free (3))

Terminating imaging engine...

Adding resources...

................................................................................

Elapsed Time: 11.784s

 (1 task, weight 100)

File size: 1433395 bytes, Checksum: CRC32 $43A4EE39

Sectors processed: 58605120, 62161 compressed

Speed: 2.6Mbytes/sec

Savings: 100.0%

created: FirewireBackup.dmg
```

10.6.7 Burning a Disk Image to CD or DVD

Burning an image to CD or DVD from Disk Utility is simple; just select the disk image and click the Burn button. Make sure that the image doesn't contain more data than the disc you are burning to. CDs can hold up to 660 MB and DVDs can hold up to 4.7 GB.

It's just about as easy from the command line. Simply use the *hdiutil burn* command as shown in Example 10-12.

Example 10-12. Using the hdiutil burn command

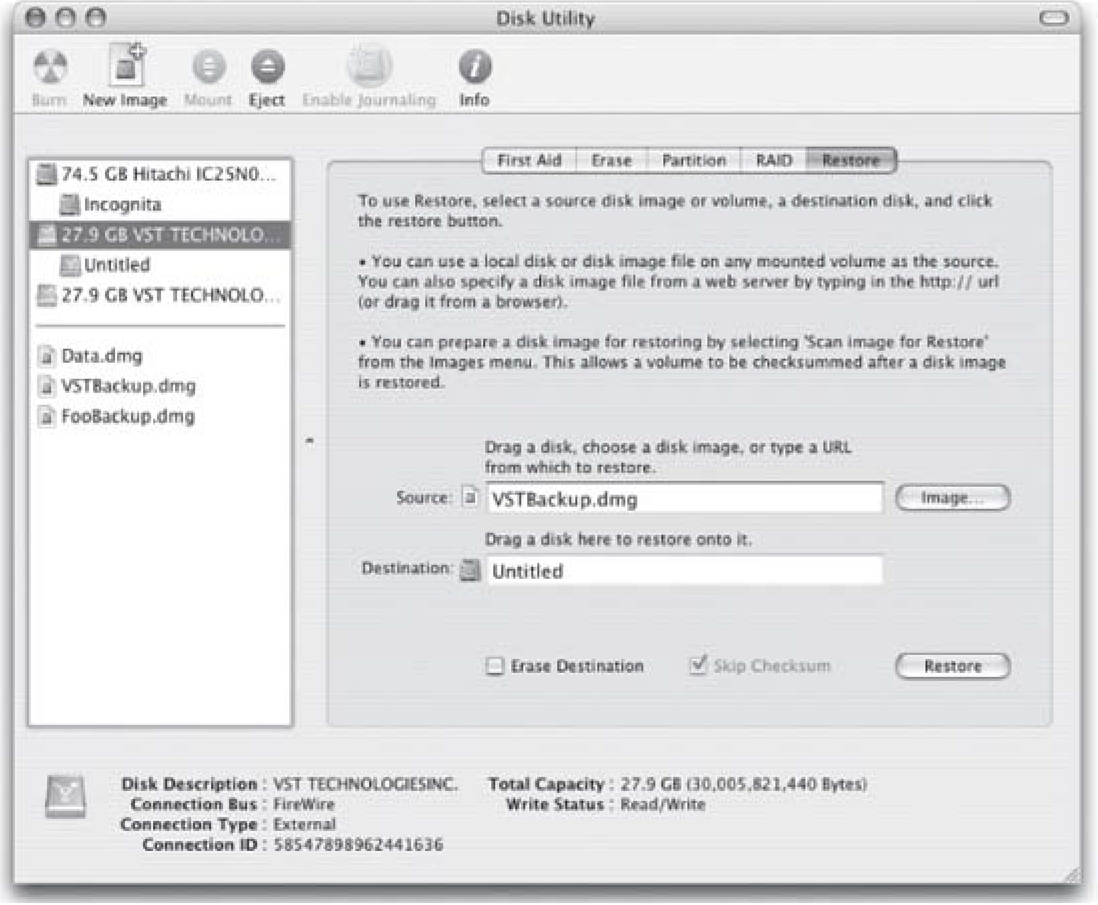
```
$hdiutil burn Documents.dmg
Please insert a disc:
Preparing data for burn
Opening session
Opening track
Writing track
...............................................................................
Closing track
Closing session
Finishing burn
Verifying burn...
Verifying
...............................................................................
Burn completed successfully
...............................................................................

hdiutil: burn: completed
```

10.6.8 Restoring a Drive Image to a Drive

To restore the contents of a disk from a disk image, use Disk Utility's Restore panel, as shown in Figure 10-12, and use the following process:

1. Either drag a disk image to the Source text field or click the Image button and select the image you want to restore from.
2. Drag the disk that you want to restore to from the left-hand column to the Destination text field.

Figure 10-12. Using Disk Utility to restore a disk from a disk image



To use Restore, select a source disk image or volume, a destination disk, and click the restore button.

• You can use a local disk or disk image file on any mounted volume as the source. You can also specify a disk image file from a web server by typing in the http:// url (or drag it from a browser).

• You can prepare a disk image for restoring by selecting 'Scan image for Restore' from the Images menu. This allows a volume to be checksummed after a disk image is restored.

Drag a disk, choose a disk image, or type a URL from which to restore.

Source: VSTBackup.dmg

Drag a disk here to restore onto it.

Destination: Untitled

Disk Description : VST TECHNOLOGIESINC.
Connection Bus : FireWire
Connection Type : External
Connection ID : 58547898962441636
Total Capacity : 27.9 GB (30,005,821,440 Bytes)
Write Status : Read/Write

3. Optionally, click the Erase Destination checkbox if you want to replace the contents of the disk with the disk image.
4. Click the Restore button.

> You should be aware that it is possible to restore the contents of a drive to another drive that is either larger or smaller than the original drive. As long as there is enough space on the target drive to hold the data being moved to it, Disk Utility will take care of the rest.

10.6.9 Moving the Contents of One Drive to Another

You can use the Restore panel of Disk Utility to copy one disk to another. This is useful when you want to move a perfect copy of your boot disk to another drive. For example, you could copy a disk to an external FireWire drive and then use that drive as a master disk image for restoring the disk onto other machines. To do this using the Restore panel, select a disk as the source instead of a disk image.

This feature is especially handy when you want to upgrade the boot drive in a machine. For example, if you wanted to upgrade the hard drive in your PowerBook, you could use the following process:

1. Boot your system using the Mac OS X Install Disk 1. Instead of proceeding with the install, use Installer⟶Open Disk Utility menu to launch Disk Utility.
2. Plug in an external FireWire drive (either one that you own or have begged or borrowed from someone). Make sure that the FireWire drive is large enough to hold the contents of your hard drive.
3. Click the Restore tab of Disk Utility (refer to Figure 10-10).
4. Set the Source of the restore to your internal boot drive.
5. Set the Destination of the restore to your external FireWire drive.
6. Click Restore and go have a coffee at your local coffee shop while Disk Utility copies the data from the internal drive to the external.
7. When Disk Utility is done, exit the installer and power down your system.
8. Replace the drive in your computer. You should refer to Apple's documentation for your model of Mac to help you.
9. Once the new drive is installed, boot your system with the Mac OS X Install Disk 1 again. And once again, launch Disk Utility.
10. Verify that the new disk drive appears in the drive list. Make sure that it is formatted to use Journaled HFS+.
11. Using the Restore tab, make the external FireWire drive the Source and the new internal hard drive the Destination.
12. Click Restore and go have some more coffee.

When Disk Utility is done, you're almost done. You just need to check to make sure that your disk is set up to boot. Usually Disk Utility does the right thing if each disk only has a single partition on it, but when you are moving data around on disks with multiple partitions, things might not always work out so well. You can check the boot status of your drive with the *bless* command, as shown in Example 10-13 (replacing NewDrive with the name of your drive).

Example 10-13. Checking your drive

```
$sudo bless -info /Volumes/NewDrive

finderinfo[0]:  2589 => Blessed System Folder is /Volumes/NewDrive/System/Library/
```

```
CoreServices

finderinfo[1]:     0 => No Startup App folder (ignored anyway)

finderinfo[2]:     0 => Open-folder linked list empty

finderinfo[3]:     0 => No OS 9 + X blessed 9 folder

finderinfo[4]:     0 => Unused field unset

finderinfo[5]:  2589 => OS X blessed folder is //Volumes/NewDriveSystem/Library/

CoreServices

64-bit VSDB volume id: 0xCE442C88EEA5FF45
```

If you don't have a blessed system folder (you'll see No Blessed System Folder in the output), you can bless it with the following command (replacing NewDrive with the name of your drive):

```
  $ sudo bless -folder /Volumes/NewDrive/System/Library/CoreServices
```

Check the disk again with the *bless —info* command. If everything is fine, you are ready to reboot your Mac. Your system will boot from the internal hard drive like nothing ever happened, except that the hard drive should now be either bigger or faster depending on what you replaced it with.

10.7 Creating a RAID

The last feature that Disk Utility and its related command-line tools enables is the use of a Redundant Array of Independent Disks (RAID), where multiple disks are combined into one virtual disk. In general, RAID solutions are designed for increased data availability and integrity. They trade the overhead of maintaining data across a set of disks for the fault tolerance that using multiple drives can provide.

There are several types of RAID in use today. These are:

RAID 0

Data is striped across two or more disks giving increased I/O performance; if each disk is on a separate disk controller, this allows for large virtual disks. This type of RAID provides no protection against disk failure. If a disk fails, all the data on the disk set is lost.

RAID 1

Data is mirrored across two drives giving complete data redundancy. If one disk fails, the other still has all the data needed to keep going.

RAID 3, RAID 4, RAID 5

Multiple disks (three or more) are used to both stripe data for performance and to keep parity information to provide for redundancy in case a disk fails. Because data is split and duplicated among many disks, this type of RAID requires significant computation power.

Mac OS X provides support for RAID 0 (striping) and RAID 1 (mirroring). Both types of RAID can be performed with acceptable performance in software. The other forms of RAID are not supported in software as the increased overhead would create too large a performance hit to be worthwhile. If you require RAID 3, 4, or 5, you should take a close look at hardware solutions such as Apple's Xserve RAID, which has dedicated processors to handle the extra overhead, or other third-party RAID solutions.

To create a RAID set using Disk Utility, use the RAID panel, as shown in Figure 10-13. Drag the disks that you want to be part of your RAID to the Disk table, select the RAID scheme, and then click the Create button.

To create a RAID set using the command line, use the *diskutil createRAID* command as follows:

```
diskutil createRAID [mirror|stripe]setname filesystemtype diskids...
```

Example 10-14 shows how to create a striped RAID set using two disks with the device IDs of *disk1* and *disk2*:

Example 10-14. Creating a RAID set using the command line

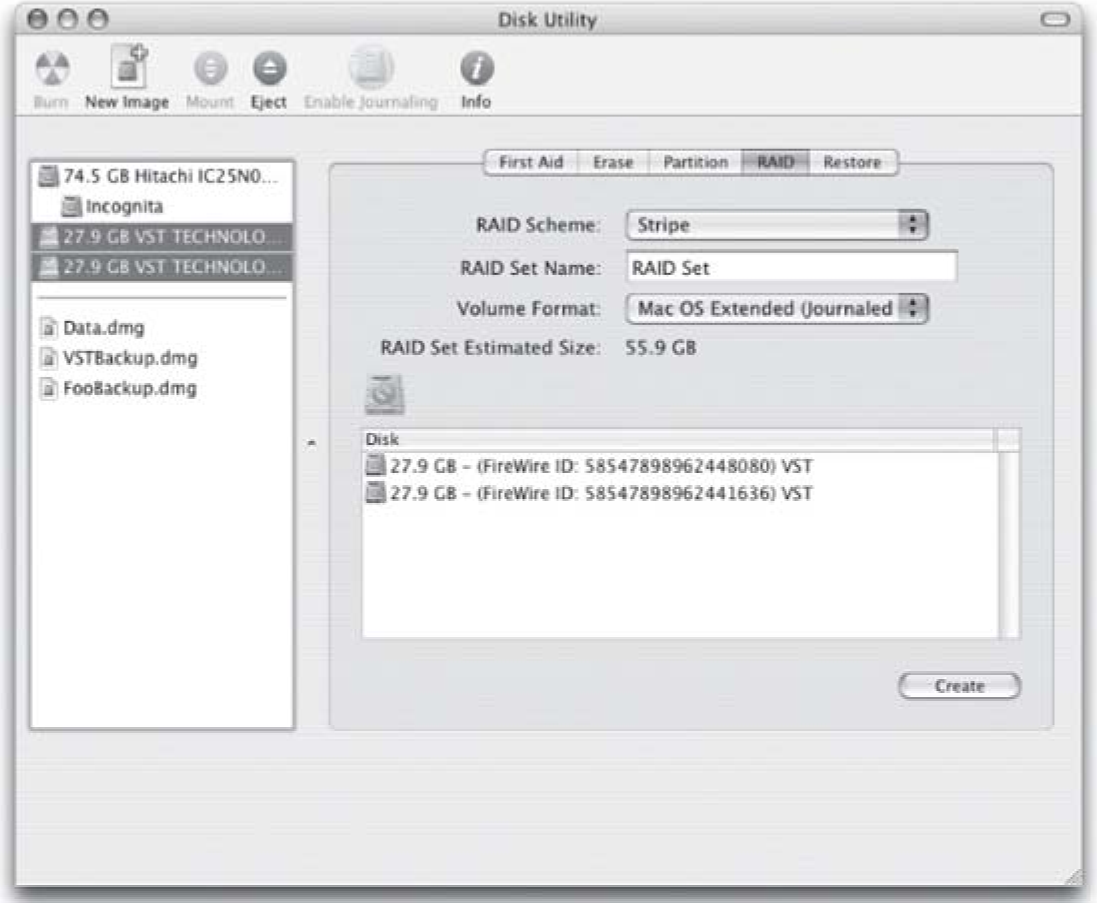
```
$diskutil createRAID stripe RAID "Journaled HFS+" disk2 disk3

The RAID has been created successfully
```

10.8 Further Explorations

If you find yourself using the tools in this chapter extensively, you should consult the following manpages for more information:

- *diskutil*
- *mount*
- *umount*
- *hdiutil*

Figure 10-13. Creating a RAID using Disk Utility



74.5 GB Hitachi IC25N0...
Incognita
27.9 GB VST TECHNOLO...
27.9 GB VST TECHNOLO...
Data.dmg
VSTBackup.dmg
FooBackup.dmg

RAID Scheme: Stripe
RAID Set Name: RAID Set
Volume Format: Mac OS Extended (Journaled
RAID Set Estimated Size: 55.9 GB

Disk
27.9 GB – (FireWire ID: 585478989624448080) VST
27.9 GB – (FireWire ID: 585478989624441636) VST

Part III: Advanced Topics

This part of the book covers advanced topics such as how to work with Directory Services, the printing system of Mac OS X, networking, and how the various network services work.

Chapters in this part of the book include:

Chapter 11. Open Directory

Open Directory, the directory services layer of Mac OS X (see Figure 11-1 for a detailed view), provides the essential service of providing information to the system about users, machines, printers, and much more. It's also something that most people know very little about. To many, it's mysterious because its name connotes a tie to the idea of a filesystem directory. In reality Open Directory doesn't have much at all to do with the filesystem other than the fact that its data is arranged in a hierarchical tree.

For the most part, however, Open Directory is an enigma because the role it plays is central enough to the system that it's hard to distinguish what it's doing. At Apple's 2003 Worldwide Developer's Conference (WWDC), an Apple employee retold a story about how management always wanted to see a demo of directory services. His response was simply, "Did you make it past the login window? [If so,] well, that's the demo."

Every time you log in, whether through a local or a network account, and every time you browse for Macintosh- or Windows-based file servers, you are using Open Directory. When you log in to your Mac using the login window, the login window consults Open Directory to see whether or not you have a valid username and password for the system. If Open Directory indicates that the username and password are okay, login proceeds. If not, you're challenged until you either get it right or your entry is refused. When you want to connect to a server, the Finder consults Open Directory for a list of server-based filesystems. In large networks, Open Directory can be used to configure printers, mail settings, and much more.

This chapter gives you an overview of what directory services are, where they came from, and what problems they solve. It also shows you how directory services are used in Mac OS X by Open Directory and how to connect to servers that provide directory information like Mac OS X Server, Active Directory, and NetInfo.

11.1 Open Directory in Action

To help explain how Open Directory is used in Mac OS X, let's look at a few examples. When you enter your username and password into the login window, the following steps happen:

1. The login window calls Open Directory with a request to authenticate the user.
2. Open Directory takes the username and password and if the user exists, looks up the authentication method.
3. Using the proper process governed by the authentication method, Open Directory attempts to validate the password.
4. Open Directory indicates whether or not the user was authenticated to the login window.
5. If the user was authenticated, the login window proceeds to create a GUI session for the user.
6. As the GUI session is created, Open Directory is queried to give the location of the user's Home folder.

This basic process of querying Open Directory for user information is followed by all parts of the system that either know how to use Open Directory or are using it behind the scenes by using the PAM (pluggable authentication modules) functionality built into many Unix-based applications. For example, when you log in to your computer remotely via SSH, the following steps occur:

1. *sshd* (the SSH server daemon) gets the username and password for the user requesting to log in.
2. *sshd* then makes a PAM call to authenticate the user. This is handled by Open Directory.
3. Open Directory takes the username and password and if the user exists, looks up the authentication method.
4. Using the proper process governed by the authentication method, Open Directory attempts to validate the password.
5. Open Directory indicates whether or not the user was authenticated to *sshd*.

In addition, the act of browsing the network for filesystems when you use the Finder's Go → Connect to Server (⌘-K) menu causes a lookup into Open Directory which then presents the information that it finds using LDAP, NetInfo, Rendezvous, SMB, SLP, and AppleTalk. Open Directory is also utilized by Terminal's File → Connect to Server (Shift-⌘-K) command, which allows you to create a connection to Rendezvous-enabled computers that advertise SSH and Telnet services.

11.2 Directory Services Defined

Historically, Unix systems have stored user and password information in "flat" text files located in the */etc* directory and let applications access that data directly. This works well enough when you only have to administer one machine. However, in the 1980s, with the advent of the Unix-based workstation, as well as the proliferation of personal computers, allowing mass deployments of computers in offices and universities, managing user information on each machine became a serious administration problem. The problem only became worse as administrators tried to cope not just with setting up users, but setting up printers, servers, and other network resources.

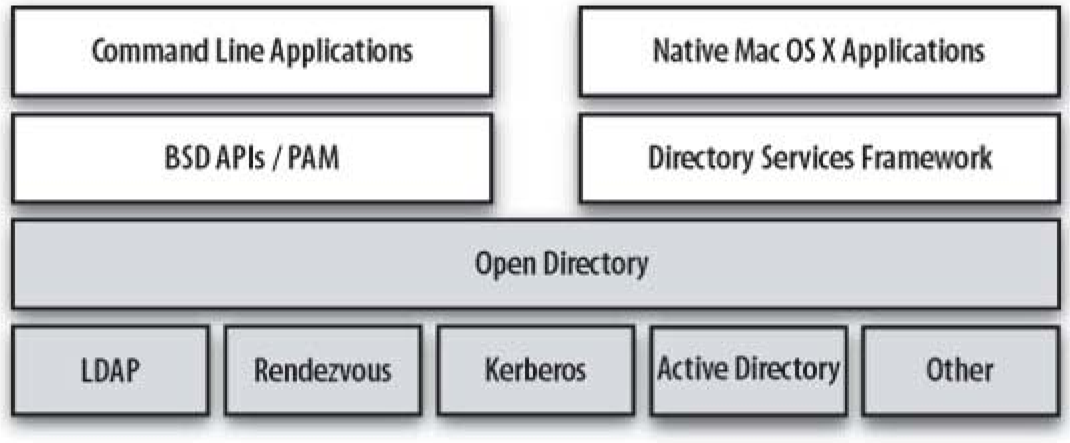
Various solutions have been invented over the years to alleviate this problem. Sun introduced Network Information Services (NIS, also known as "Yellow Pages"), NeXT developed NetInfo, Novell built a business around NetWare (and failed), and Microsoft ended up taking it away from Novell with Active Directory. All these solutions have a common goal: to centralize the information needed so that a group of users across a variety of machines can be managed effectively.

When development on Mac OS X started, Apple inherited the NetInfo code that came with NEXTSTEP. The company knew it was not going to be enough to just provide for management of homogenous networks of Macs, so Apple set out on a path to build a set of technologies collectively known as Open Directory. These technologies brought support for the Lightweight Access Directory Protocol (LDAP) and Kerberos (a network authentication standard developed at the Massachusetts Institute of Technology) to the operating system. With the release of Panther, Mac OS X interoperates with all the major directory services systems in use including Active Directory, as shown in Figure 11-1.

The way in which applications utilize Open Directory is illustrated in Figure 11-1. When a native Mac OS X application, such as the login window, needs to authenticate a user it calls into Open Directory. It then can authenticate the user information against whatever data store it is configured to use. To accommodate Unix-based command- line applications, the BSD authentication routines have been modified to use PAM, which then uses Open Directory to authenticate against. Likewise, whenever any system component needs to find a resource, such as a network file share, it uses Open Directory.

The end result is a system in which the applications that need authentication and configuration data can get it without knowing, or even caring, where that data is stored. It should be noted that it's not just end-user applications that use this abstracted data. The data about a user that's obtained from Open Directory is used by every part of the system. For example, the filesystem permissions scheme described in Chapter 6,*Files and Permissions*, relies on the user and group information stored in Open Directory.

Figure 11-1. How applications and Open Directory fit together


Command Line Applications
Native Mac OS X Applications
BSD APIs / PAM
Directory Services Framework
Open Directory
LDAP
Rendezvous
Kerberos
Active Directory
Other

11.2.1 Kinds of Directory Information

Open Directory can handle many different kinds of information. The following are the most common types of data that Mac OS X looks to Open Directory to provide:

- User identification data including real name, username, and user ID.
- Authentication information used to verify passwords given by a user
- Group identification data
- The location of a user's Home folder , either on a local machine or on a network server
- Network filesystems
- Quotas for disk, print, and mail usage
- Network-enabled printers that are either shared by other machines or are directly connected to the network

11.2.2 Sources of Directory Information

Open Directory is a flexible bridge allowing information to be obtained from many different sources. These sources fall into roughly three categories: configuration information, authentication services, and discovered services.

11.2.2.1 Sources of configuration information

LDAP

> The open standard used in many mixed environments and the native Open Directory database in Mac OS X Server 10.3.

NetInfo

As mentioned earlier, NetInfo is the legacy directory service picked up from NEXTSTEP and earlier versions of Mac OS X. It is the standard local directory service for Mac OS X.

Active Directory

This is Microsoft's LDAP-based directory service provided by Windows 2000 and Windows 2003 servers.

NIS

Network Information Services servers are still used by many Unix shops but less frequently since LDAP hit the street.

BSD flat files

These files are located in the */etc* directory.

11.2.2.2 Source of authentication information

To authenticate users, Open Directory supports the following technologies:

The Shadow Password Database

The default authentication mechanism used in Mac OS X. Passwords are stored in the */var/db/shadow* directory, which is only accessible by directory services and the root user.

Apple Password Server

This authentication server ships only as part of Mac OS X Server.

Kerberos

A network authentication protocol developed at MIT, Kerberos ships as part of Mac OS X Server as well as a stand-alone server implemented in many large government, educational, and corporate networking environments.

Active Directory

This is the directory services layer of Windows 2000 Server and Windows 2003 Server. It uses Kerberos to perform authentication.

11.2.2.3 Source of discovered services

Open Directory obtains information about network filesystems, printers, and other network devices from the following services:

Rendezvous

This is the Internet Engineering Task Force's (IETF) Zeroconf-based protocol for discovering services on the local network.

SMB

Service Message Block is used to announce and connect to Windows-based shared filesystems and printers.

SLP

Service Location Protocol is an open service discovery protocol used on some networks.

AppleTalk

Apple's legacy Mac OS-based protocol for discovering shared filesystems and printers.

11.3 Directory Domains

The advent of network-based directory services did not totally obviate the need for local configuration data. As much as it was necessary to make networks of machines work well together, it was also necessary for computers to be able to work well in standalone situations. Open Directory solves this problem by using a set of domains:

- The*local domain* that every installation of Mac OS X has. It is the first domain to be consulted by Open Directory
- Any number of *shared domains* for which an installation of Mac OS X can be configured to use so that it can participate in a networked environment

This allows Mac OS X to run just fine in stand-alone mode while allowing it to be configured to run in a large network environments. It can work well in enterprisewide configurations with multiple platforms that are woven together with various standards. Interoperability has been the holy grail for many network admins. With its range of directory integration, Mac OS X is a first-class citizen in such environments.

11.4 The Local Domain and NetInfo

The local domain is the default directory domain of the system. It consists of the following parts:

- The local NetInfo database that is created when the operating system is first installed. It is located in the */var/db/netinfo* directory.
- The local Shadow Password database, created when the operating system is first installed, located in the */var/db/shadow* directory.
- Rendezvous, SLP, and SMB for discovery of shared filesystems and other network services

The Shadow Password database is new in Panther. In previous versions of Mac OS X, the passwords were encrypted (using the *crypt* command-line tool) into a hash form and stored directly in the NetInfo database. However, because the information in the NetInfo database is available to anybody on the machine, the passwords were vulnerable to decryption attempts. All you had to do was dump out the NetInfo database (you'll see the commands to do so later in this chapter) to a flat file and then run any number of password-cracking utilities against that file. You could even do so on a separate machine once you had the flat file.

NetInfo is Dead, Long Live NetInfo

Since the first release of Mac OS X, Apple has been slowly putting together all the pieces to move away from NetInfo and use LDAP for Directory Services. With the release of Panther the migration on the server side is done. When you install Mac OS X server, the directory server in use is LDAP, and the only options in the Server Admin tools related to NetInfo are to migrate the data from a NetInfo database to LDAP.

However, on the client side, NetInfo is still the default local directory. So, the rumors of NetInfo's death are premature. At least in Panther.

The Shadow Password database changes this, locking the passwords into a directory where only the*root* user of the system can access them, thereby closing this security vulnerability. When the system needs to authenticate a user, Open Directory will look at the user's NetInfo record, see that the password is in the shadow password database, and then compare the information given to it against the user's password. Unless you have root privileges on the machine, there's no way to get to the data that Open Directory uses to authenticate a user.

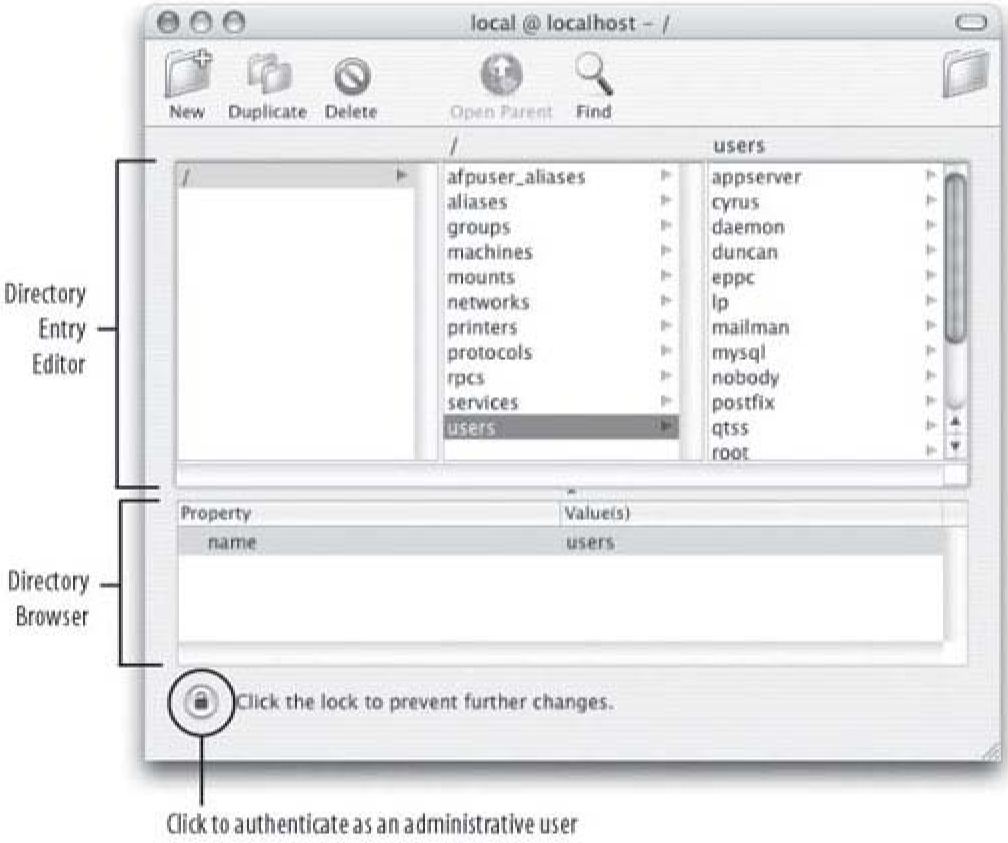
> The move to shadow passwords in Panther breaks some Unix-based tools that still expect to be able to find a crypt-based password. These tools need to be updated to use the PAM-based authentication libraries. The Unix tools that ship as part of the default Mac OS X installation use PAM. However if you are compiling your own tools, you'll need to be sure they use PAM in order to avoid authentication problems.

To modify information about users and groups in the local domain, typically you use the Accounts preference panel. Whenever you create or modify a user, the user information is stored in the local NetInfo database. Whenever you modify a password, either with the Accounts preference panel or the *passwd* command-line tool, that password is updated in the local Shadow Password database. In addition to these easy-to-use GUI tools in the System Preferences, Mac OS X provides the GUI-based NetInfo Manager (*/Applications/Utilities*) and a set of command-line tools to manipulate NetInfo data.

11.4.1 Examining NetInfo Data with NetInfo Manager

When you first launch NetInfo Manager, you see an interface with two parts, as shown in Figure 11-2. The top part of the interface is a browser that allows you to navigate through the tree hierarchy of the NetInfo database. The left-most column of the browser displays "/", which is the root of the NetInfo database tree. The column to the right of "/" is the set of top-level directories (not to be confused with filesystem directories!) that are in the database. Unlike other Apple applications, NetInfo Manager doesn't have a help system (⌘-?). This is a sign that this tool isn't for beginners, but not having a help system has given NetInfo a rap that it's a gnarly beast, when in fact it's rather easy to use, you just have to get used to it.

Figure 11-2. NetInfo Manager



Property	Value(s)
name	users

Click the lock to prevent further changes.

Directory Entry Editor

Directory Browser

Click to authenticate as an administrative user

Each directory node in the database can, in addition to holding subdirectories, hold a set of properties. These properties are displayed in the bottom part of the NetInfo Manager interface. As shown in Figure 11-2, the *users* directory contains subdirectories for each user on the system as well as a single property, *name*, that contains the name of the directory.

Because the NetInfo database is inherited from NEXTSTEP and is designed to serve as both a local and distributed directory, it contains many entries that you won't typically use on a single local system. The top-level subdirectories of greatest use on a local system are:

groups

Contains a listing of the *groups* and the *users* that belong to each group. The information in this subdirectory is analogous to the information stored in the classic Unix */etc/group* file.

users

Contains a listing of the users and the various properties associated with those users. This information serves the same purpose as the Unix */etc/passwd* file.

> By directly using NetInfo Manager to edit the local directory services domain, you could potentially create many problems and cause your system to start up incorrectly or not at all. Exercise great care before making changes.

11.4.1.1 Managing user information

When you create an account with the Accounts preference panel, all the properties about that user are stored in the local NetInfo database managed by Open Directory. By selecting the */users* directory in NetInfo, you'll be able to see all the users on your system, and when you select your username, you should see something similar to Figure 11-3. Every setting that controls how a user behaves on the system is stored right here. These settings are (in order that they typically appear for a user managed by the Accounts preference panel):

hint

The password hint that gets displayed when a user enters an incorrect password more than three times.

sharedDir

The name of the folder in the user's Home folder that will be shared with other users. By default this is the *~/Public* folder.

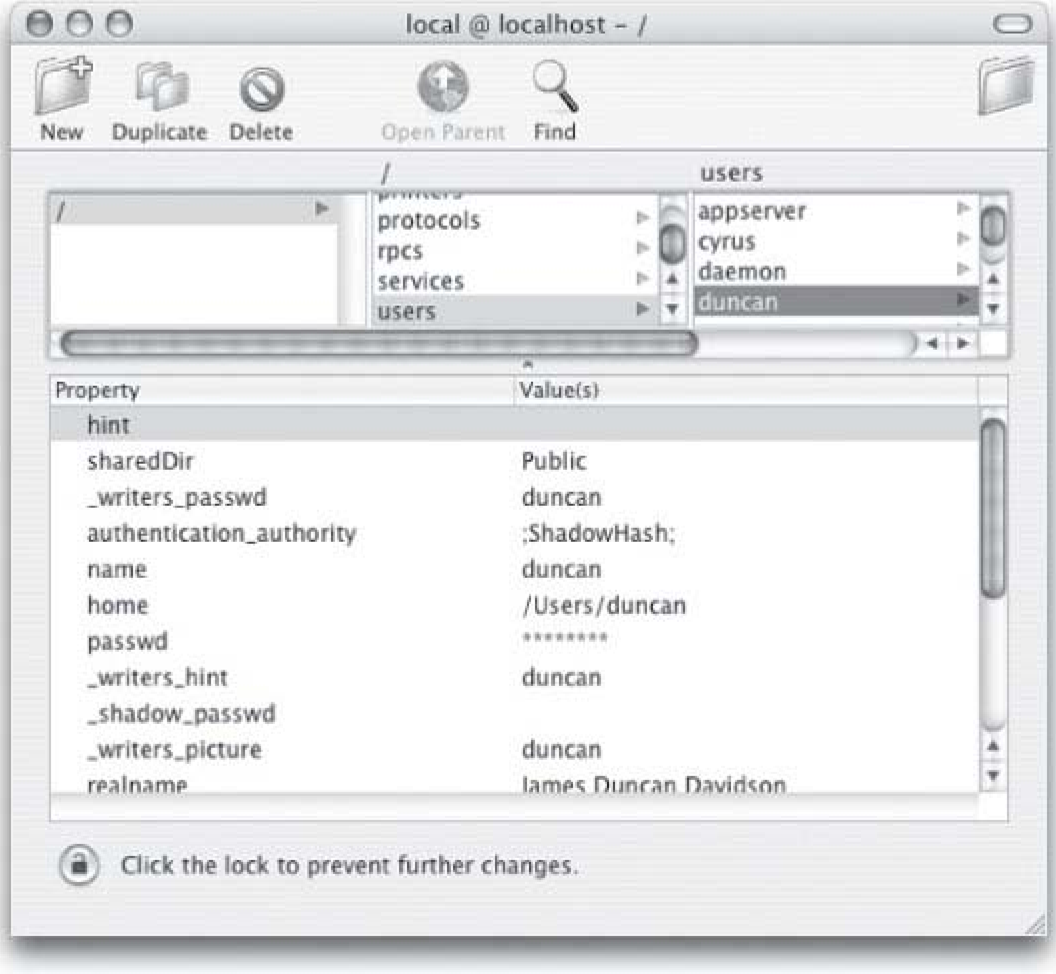
_writers_passwd

Lists the users that are allowed to change the password for this user. Typically you'll only see the username of the user record this setting applies to. However, an admin user can always change a password for a user in the Accounts preference panel.

authentication_authority

Indicates the authentication resource to use to authenticate the user against. By default in Panther this will be ;`ShadowHash`; indicating that the system should authenticate the user against the Shadow Password database in */var/db/shadow*. If you created an account with Jaguar or a previous release of Mac OS X and migrated it to Panther, this setting might be `basic`.

Figure 11-3. Examining a user subdirectory in NetInfo Manager



Property	Value(s)
hint	
sharedDir	Public
_writers_passwd	duncan
authentication_authority	;ShadowHash;
name	duncan
home	/Users/duncan
passwd	********
_writers_hint	duncan
_shadow_passwd	
_writers_picture	duncan
realname	James Duncan Davidson

Click the lock to prevent further changes.

name

The Unix username for the user. This is the same thing as the Short Name field in the Accounts preference panel.

home

The location on the filesystem of the user's Home folder.

passwd

By default when the *authentication_authority* is set to ;`ShadowHash`;, the property contains a set of asterisks. If the *authentication_authority* is set to `basic`, this property contains the password for the user in a hashed form.

_writers_hint

Lists the users that are allowed to change the hint of the user's password. By default this will only contain the name of the user's record.

_shadow_passwd

A legacy key left from older versions of Mac OS X and NEXTSTEP. Currently it's not used.

_writers_picture

Lists the users that can change the picture for this user.

realname

The long name of the user. This property corresponds to the Name field in the Accounts preference panel.

uid

The numeric user ID of the user.

shell

The default shell of the user; by default this will be */bin/bash* on Panther.

generateduid

A string that serves as a unique identifier for the user, not only on the local system but anywhere in the world. This string is used inside the Shadow Password database as a key to find the information to validate passwords.

gid

The primary group ID of the user.

_writers_tim_password

A legacy key left from older versions of Mac OS X and NEXTSTEP. Currently it's not used.

picture

The user's picture.

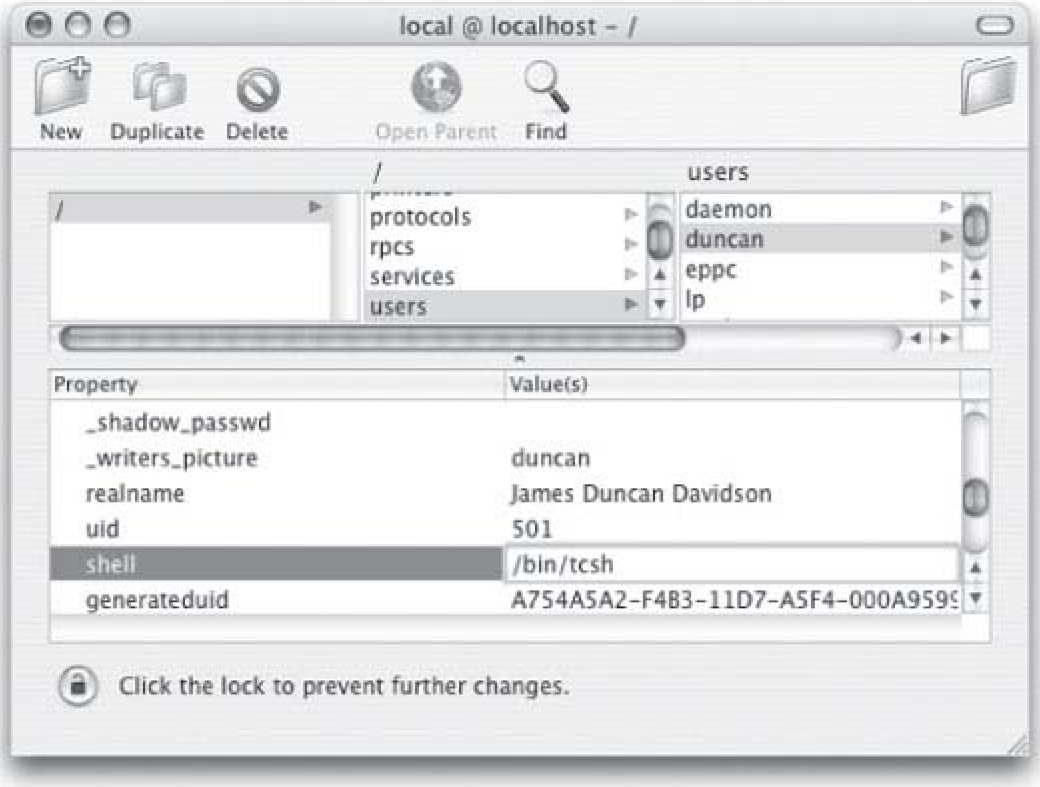
_writers_realname

Lists the users that can change the real name for this user.

Given access to this information, it becomes very easy to make substantial changes to the user records on your system. For example, if you want to change the default shell for a user from */bin/bash* to */bin/tcsh*, you can edit the shell property as follows:

1. First make sure you are authenticated by clicking the padlock at the lower-left corner of NetInfo Manager's window. If the padlock is closed, you'll need to authenticate as an administrative user.
2. Select the user that you want to modify.
3. Find the *shell* property and double-click the name of the shell (*/bin/bash* by default), as shown in Figure 11-4.
4. Change the value to */bin/tcsh*.
5. Click the padlock icon to prevent further changes from being made, or use Security→Deauthenticate.
6. Save the change with Domain→Save Changes (⌘-S).

Figure 11-4. Editing a user's shell in NetInfo Manager



Property	Value(s)
_shadow_passwd	
_writers_picture	duncan
realname	James Duncan Davidson
uid	501
shell	/bin/tcsh
generateduid	A754A5A2-F4B3-11D7-A5F4-000A959

Click the lock to prevent further changes.

11.4.1.2 Managing group information

The only way to affect the group settings for a user using the Accounts preferences panel is to grant the user administrator permissions. No other group manipulation is exposed. However, using NetInfo Manager, you can define new groups on the system and associate users with them. For example, to give the user *norman* administrative privileges, follow this process:

1. Navigate to the */groups/admin* subdirectory using the browser window at the top of the NetInfo Manager interface.

2. Select the *users* property, as shown in Figure 11-5.

3. Use the Directory→New Value (Option-⌘-N) menu to create a new value in the *users* property.

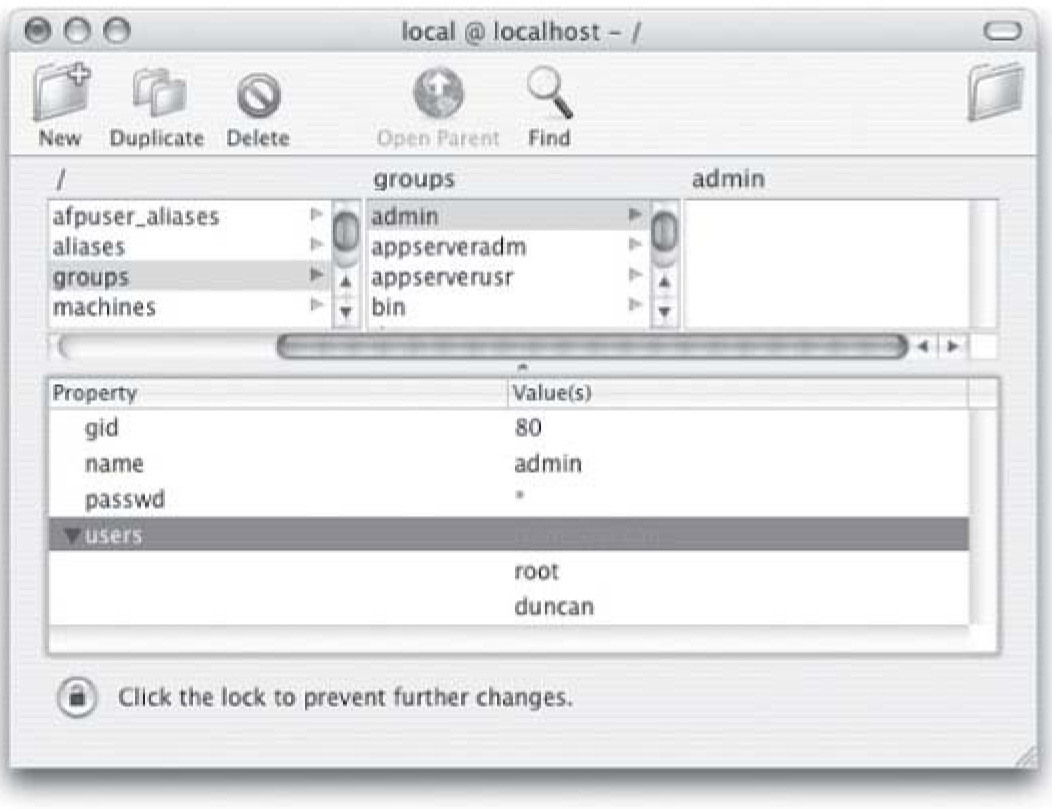
4. Change the *new_value* string that was inserted to *norman*.

5. Save the change by using Domain→Save Changes (⌘-S).

When you check the "Allow user to administer this computer" checkbox in the Accounts preference panel, the system is simply adding the user to the *admin* group (just as we've done here). Likewise, when you uncheck the administration checkbox in the Accounts preference panel, the system removes the user's name from the */groups/admin* list in the NetInfo database.

Figure 11-5. Editing the /groups/admin subdirectory in NetInfo

Manager



Property	Value(s)
gid	80
name	admin
passwd	*
▼users	
	root
	duncan

11.4.1.3 Creating a nonhuman user

If you need to create a nonhuman user, meaning an account that is used to run some server program securely (not an account for a dog or a fish), it's best not to create that user with the Accounts preference panel, as a nonhuman user doesn't need to have a Home folder created. Instead, you can create an account by directly editing the NetInfo database. Usually, its easiest to duplicate an existing user and then edit the various properties to give your new user the correct setup. Here's the step-by-step guide:

1. Launch NetInfo Manager and authenticate as an admin user by clicking the padlock or using Security→Authenticate. When prompted, enter your password and hit Return.

2. In the directory browser at the top of the window, click the *users* entry in the second column and then select the *unknown* user as shown in Figure 11-6.

3. Click the duplicate button in the toolbar or use Edit→Duplicate (⌘-D). A duplicate of the unknown user will be created after you accept the dialog box challenge.

4. In the property-value editor at the bottom of the application window, change the *uid* (user ID) property to a number that isn't already assigned to some other user. By convention, you should use an ID between 100 and 500. IDs less than 100 are reserved for use by Mac OS X, and IDs greater than 500 appear in the login window. Be sure not to use an ID already in use. To be safe, check the various other user entries to make sure you aren't using their ID before proceeding. Although NetInfo Manager is perfectly happy allowing you to create multiple user entries with the same user ID, the system itself won't be happy with

this situation.

5. Assign a group ID that is the same number as the user ID. You'll create the group for the user in a minute.

6. If the user needs a Home directory (not likely for a nonhuman user, but the documentation for the software that you are creating the nonhuman user for may indicate that it is needed), set the *home* property to the filesystem location of the directory. If you do assign a Home directory, you are responsible for creating that directory. You can always add or change the *home* property later.

7. If the user needs to use a shell (once again, not likely but consult the documentation for the software that you are creating the nonhuman user for), set the *shell* property to the shell you'd like to use (for example, *bash, tcsh, zsh*).*/usr/bin/false*, a program that doesn't do anything but return a non-zero exit code, is a good choice when you want to make sure that the user can't use a shell.

8. Set the *realname* property to something that makes sense.

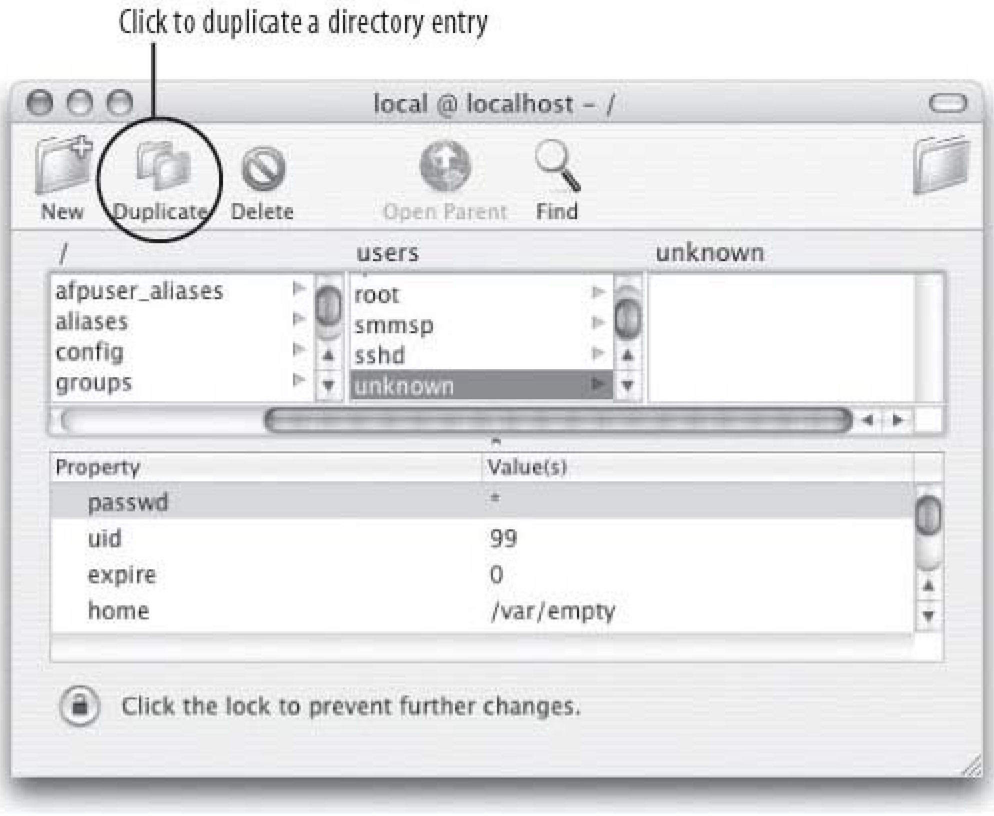
9. Move to the groups entry in the browser.

10. Duplicate the *unknown* group.

11. Change the name of the group to be the same as the name of the new user.

12. Change the *gid* (group ID) number to match the user ID number.

Figure 11-6. Duplicating a user entry



Click to duplicate a directory entry

Property	Value(s)
passwd	*
uid	99
expire	0
home	/var/empty

Click the lock to prevent further changes.

11.5 Configuring Shared Domains

For many users, especially those that use their Macs at home, the default local NetInfo-based domain is all that's needed. However, users in corporate settings will want to take advantage of the ease of administration and flexibility that using shared domains allows. When a shared domain is in use, any user can log into any machine that is part of the domain and access his Home folder from a network server. All the settings and data for all the computers on the network can be centralized. Also, because there is no user data stored on an individual machine, the data can be replaced or upgraded with ease. When the user logs in to the new machine, all of his data is just where he left it.

To enable a Mac to participate in a shared domain, you need to perform a two-step process. The first step is to set up the shared domain directory server that you want to use. The second is to set up the authentication rules for your system. Both steps can be performed with the Directory Access utility (*/Applications/Utilities*), shown in Figure 11-7.

11.5.1 Configuring Open Directory Sources

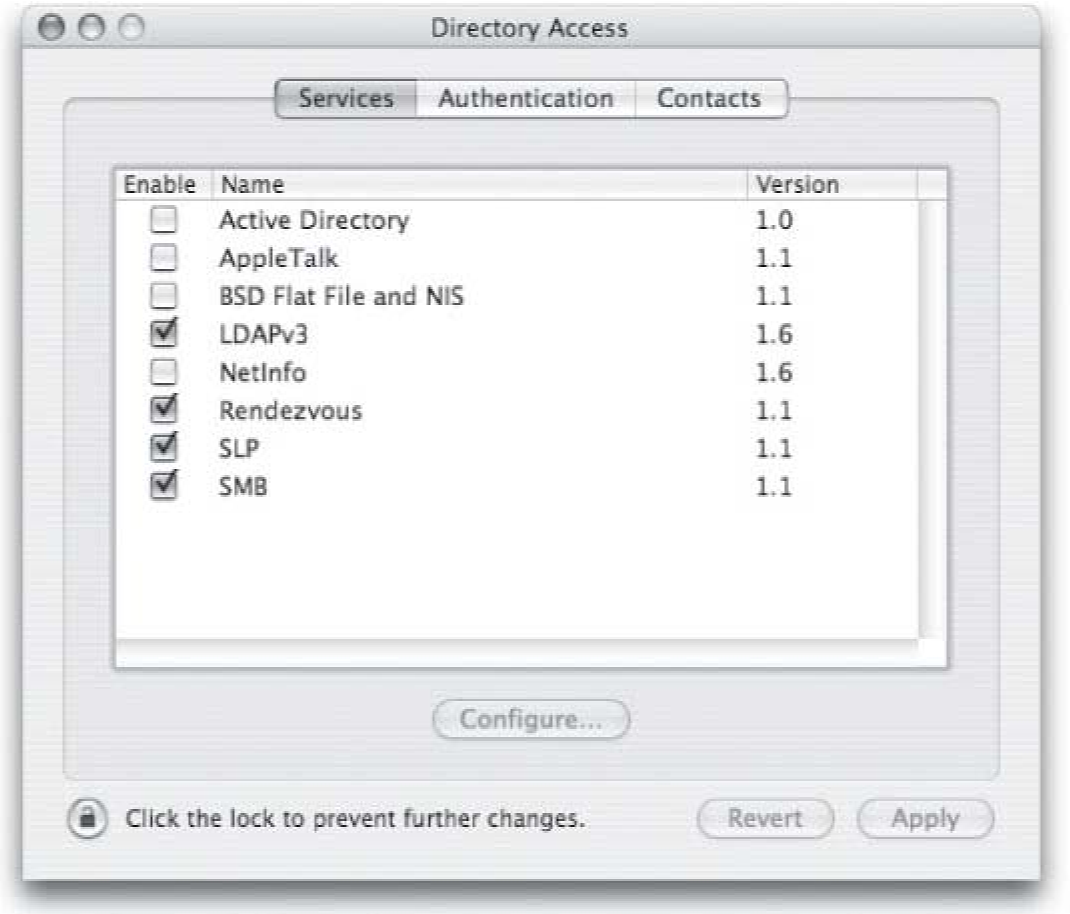
When you launch Directory Access, the first thing you'll see is the Services configuration, as shown in Figure 11-7. This gives you a list of directory sources that can be used with Open Directory. These are the same sources of information discussed earlier in the chapter. By using Directory Access, you can enable and disable these sources as well as configure them.

11.5.1.1 Using Mac OS X Server's Open Directory Server

To configure a client to use the LDAP-based directory services provided by Open Directory Server, select the LDAPv3 entry and click the Configure button. Even if the padlock icon at the lower-left of the window is unlocked, you may still be challenged for an administrator password. After you authenticate, you'll see the panel shown Figure 11-8.

The Use DHCP-supplied LDAP server box is checked by default. This means that if the system obtained an IP address from a DHCP server that also is set to provide information on where to find an LDAP server, it will go ahead and use it as a directory service. If you are on a network where an LDAP server isn't configured through DHCP, or if you are using a fixed IP address, you'll need to add your server manually. To do this, click the New button and enter the following information:

Figure 11-7. The Directory Access application used to configure Active Directory



Enable	Name	Version
□	Active Directory	1.0
□	AppleTalk	1.1
□	BSD Flat File and NIS	1.1
☑	LDAPv3	1.6
□	NetInfo	1.6
☑	Rendezvous	1.1
☑	SLP	1.1
☑	SMB	1.1

Click the lock to prevent further changes.

Configuration Name

This can be any name you want to assign to your LDAP configuration.

Server Name or IP Address

The hostname or IP address of the LDAP server.

LDAP Mappings

The type of LDAP mappings to use. For most purposes, the default "From Server" setting is appropriate. You'll see how to use the other options in this list in the next section.

SSL

Indicate whether or not to use SSL to contact the LDAP server. This will secure the LDAP connection using a the certificate credentials provided by the server.

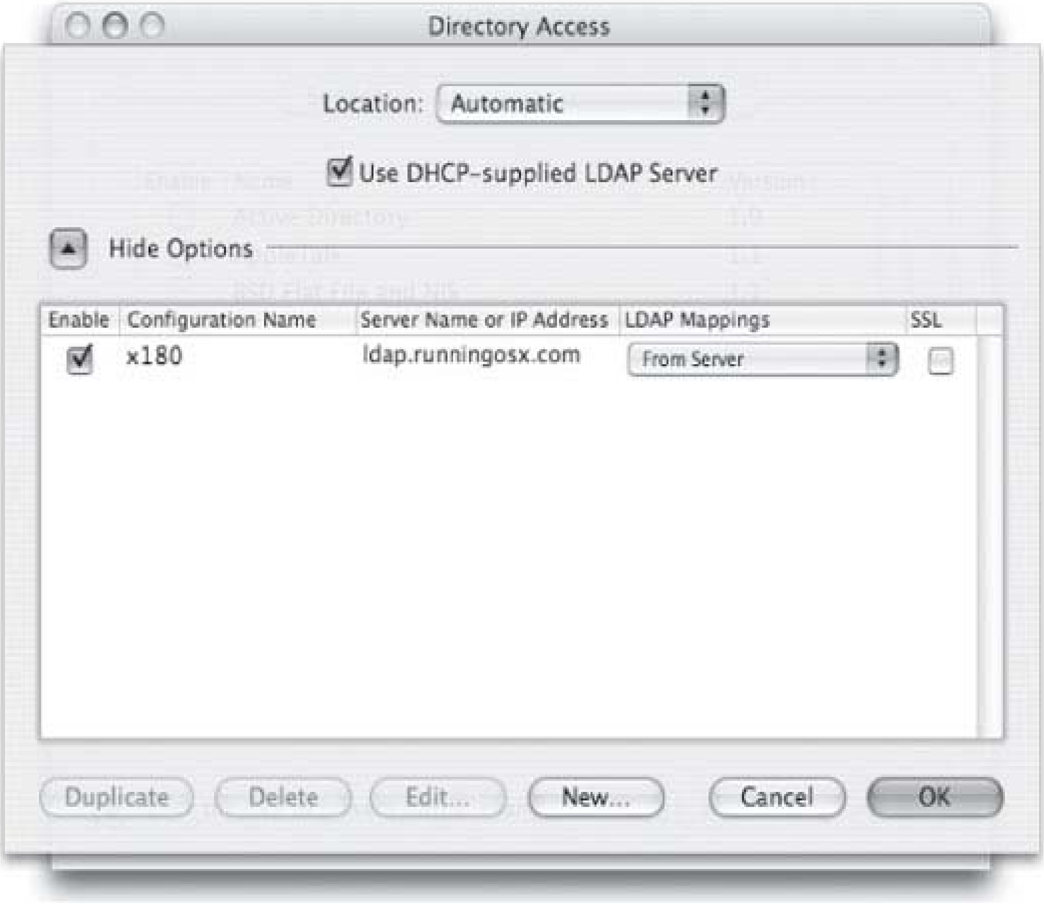
This procedure works for most network environments that use Mac OS X Server's LDAP services. However, it's a good idea to verify this information with the administrator of the Mac OS X Server you are trying to connect to.

11.5.1.2 Using other LDAP servers

Since Open Directory Server uses the standard LDAP protocol with no special modifications, the Open Directory client is already LDAP savvy. The only thing that Open Directory needs to know to use any other LDAP server is how data in the directory is stored. This is known as the server's data mappings. The LDAP servers you may use fall into three categories:

- The server is already configured to provide seamless integration with Mac OS X.
- The server is set up to use standard Unix RFC 2307 mappings.
- The server requires you to configure your mappings on Mac OS X.

Figure 11-8. Configuring the LDAP servers that Open Directory will use



Directory Access

Location: Automatic

☑ Use DHCP-supplied LDAP Server

Hide Options

Enable	Configuration Name	Server Name or IP Address	LDAP Mappings	SSL
☑	x180	ldap.runningosx.com	From Server	□

Trusting the Network

As this book was being prepared for publication, a story broke in the press about a security vulnerability in Mac OS X. It stated an attacker could take over a machine by exploiting Open Directory. In order to perform this attack, an attacker sets up a DHCP server for a network (shutting down any other operational DHCP servers), and then configures LDAP through DHCP to let him log in to any machine that boots on the network.

This isn't a new exploit, but rather is symptomatic of the changes in networking. It used to be that it was rare to move your machine between networks. You always knew the DHCP server or at least knew your admins took care of such things. Therefore picking up LDAP settings from DHCP settings aided easy system configuration. Now, with mobile computing being such a pervasive part of life, it's more questionable to make this assumption.

If you have a laptop that you carry between networks, uncheck the Use DHCP-supplied LDAP server box shown in Figure 11-8, and you won't be subject to this kind of attack.

Unless your setup falls into the first category, you'll most likely need to get some information from your system administrator to configure the LDAP directory service. If you do need to configure your own mappings, select the Custom option from the LDAP mappings pull-down menu. How you set up these mappings depends on your LDAP server and is beyond the scope of this book.

11.5.1.3 Configuring Active Directory domain servers

The ability to use Active Directory, the native directory service used by Microsoft Windows 2000 or Windows 2003 Server, is new in Panther. This support relies on the fact that Active Directory uses standard LDAP and Kerberos protocols. In fact, if you wanted to, you could connect to an Active Directory server simply by configuring it as an LDAP server. However, the mappings of data in an Active Directory server don't match up with the mappings needed by Mac OS X. Fortunately, Apple has provided Active Directory-specific functionality in Open Directory that seamlessly maps between the data mappings used by Microsoft Windows and the mappings that Mac OS X expects to see.

What is Active Directory?

Active Directory is Microsoft's directory that is provided as part of Windows 2000 Server and Windows 2003 Server. It performs the same kinds of duties for a Windows-based network that Apple's Open Directory in Mac OS X Server can provide for Macintosh systems. Since Active Directory is widely deployed in enterprise environments, Apple built in the ability to run from Active Directory into Mac OS X to make it easier for the admins of Windows-based networks to work with Macs. Hopefully it will inspire some of the employees of those companies to switch to the Mac.

At this time, it's a one-way street. Open Directory on Mac OS X can't provide all the functionality for a Windows-based machine that Active Directory can. Being able to run a Mac on an Active Directory-based network, however, is a great step forward for interoperability.

Active Directory comes with a slew of its own terms like *forest*, which refers to a group of Active Directory trees. It reuses terms like *domain* for its own purposes. For the most part, your system administrator will provide the information you need to know. You can also get more information from *Active Directory, 2nd Edition*, by Robbie Allen, et al. (O'Reilly & Associates, Inc., 2003).

To configure Open Directory to use an Active Directory server, select the Active Directory entry in Directory Access and click the Configure button. You'll be presented with the panel shown in Figure 11-9. Fill in the directory forest, domain, and computer ID fields with the values provided by your network administrator and then click the Bind button.

Depending on your setup, you may also want to set the following advanced options:

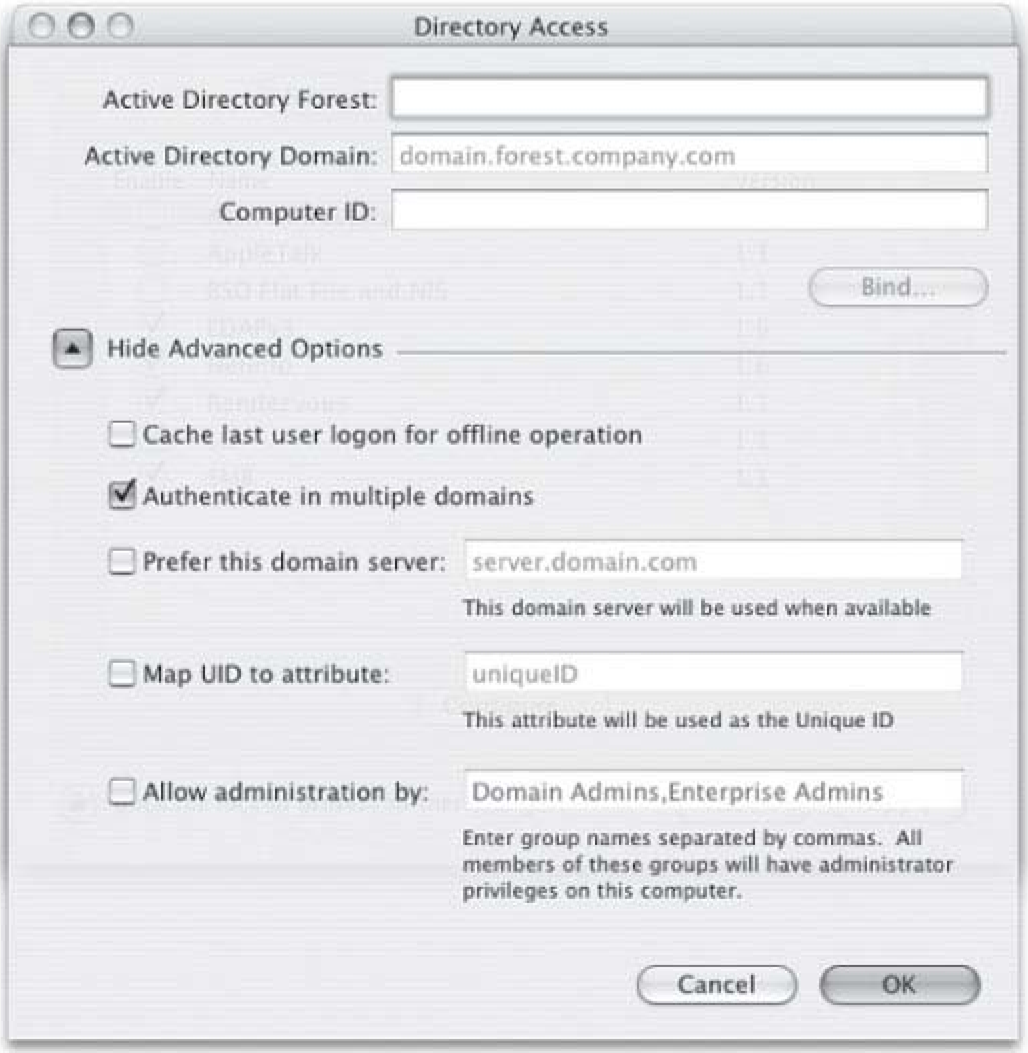
Cache last user log on for offline operation

Lets you log in to your machine even if the Active Directory server isn't available. This is handy when you have a laptop and need to use it while you are away from the office.

Authenticate in multiple domains

Lets users from any domain in the Active Directory system for your network log in to your computer.

Figure 11-9. Seting up Active Directory in Directory Access



Active Directory Forest:

Active Directory Domain: domain.forest.company.com

Computer ID:

□ Cache last user logon for offline operation

☑ Authenticate in multiple domains

□ Prefer this domain server: server.domain.com

This domain server will be used when available

□ Map UID to attribute: uniqueID

This attribute will be used as the Unique ID

□ Allow administration by: Domain Admins,Enterprise Admins

Enter group names separated by commas. All members of these groups will have administrator privileges on this computer.

Prefer this domain server

Lets you specify the hostname of the Active Directory server that you want to use by default. If this server is unavailable, Open Directory will automatically use another server that is part of the forest if available.

Map UID to attribute

By default, Active Directory doesn't use user IDs, but prefers to use longer GUIDs (Globally Unique ID). If your Active Directory server has been configured to store a user ID for each user (typically when Active Directory has already been configured to support Unix computers), you can specify the attribute within Active Directory that is used to store the ID. If you don't select this option, then a user ID is automatically generated for you based on the GUID attribute in Active Directory.

Allow administration by

Specifies a list of Active Directory groups whose members will be considered to have administrative privileges by Open Directory.

11.5.1.4 Configuring NetInfo-based domain servers

If your network directory services are based on NetInfo, you can configure Open Directory to use it by selecting the NetInfo service type and clicking the Configure button. You'll be presented with the NetInfo configuration panel shown in Figure 11-10. As with LDAP, Open Directory is configured to automatically discover any NetInfo server set in DHCP. In addition, you can set Open Directory to try to contact a NetInfo server via a network broadcast attempt, or you can configure it to contact a specific NetInfo server. Since directory information in NetInfo is always stored the same way, there's no further configuration to perform.

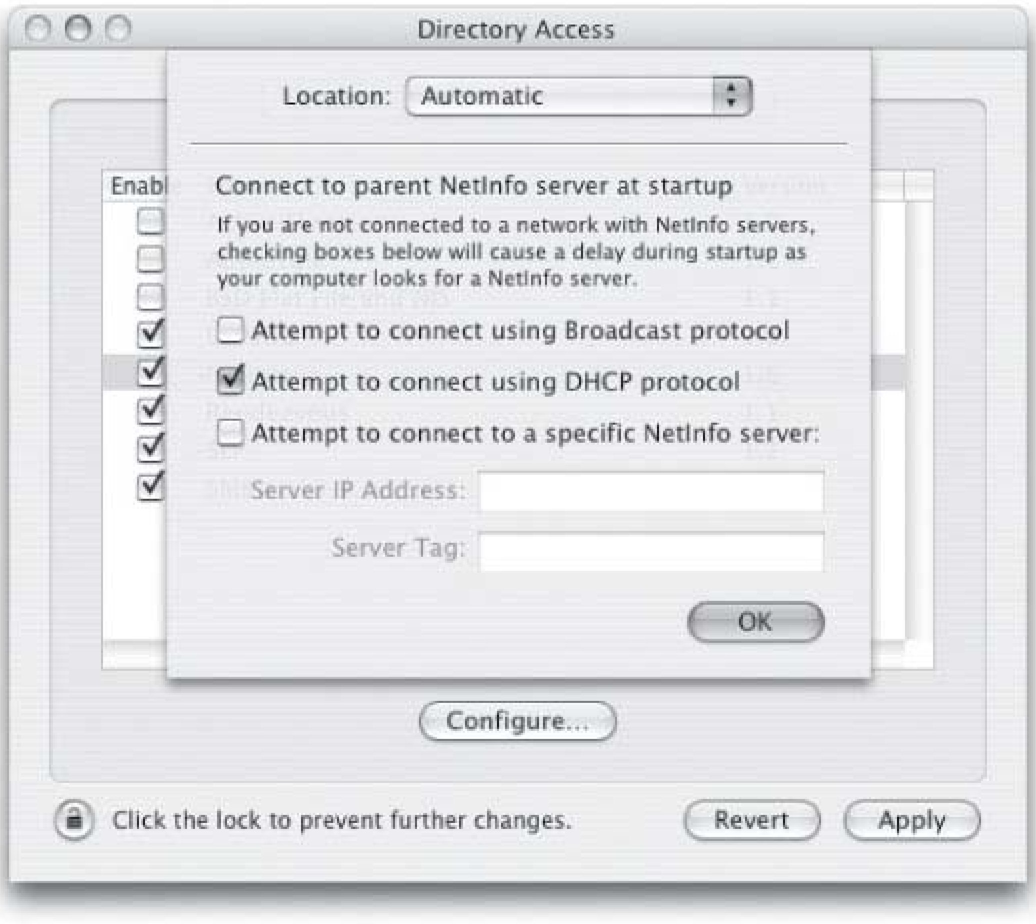
11.5.1.5 Configuring NIS domain servers

To configure the use of NIS-based directory services, select the checkbox next to the BSD Flat File and NIS entry in Directory Access (refer back to Figure 11-7). Next, click the Configure button. You'll be presented with a panel where you can enter the NIS domain name of your network and, optionally, a list of NIS servers. You can also configure Open Directory to attempt to locate a NIS server by using network broadcasts. As with NetInfo, since there is only one way to store data in a NIS server, there is no further configuration to be performed.

11.5.2 Configuring Shared Domain Authentication

Once you have set up the various servers that provide shared domain directory services to your machine, you'll be able to access the resources defined by those servers. However, to use those servers for authentication and to let users defined by those servers log in to your machine, you need to configure Open Directory's authentication settings. When you click the Authentication tab, you'll see the interface shown in Figure 11-11. By default, you'll see one entry: */NetInfo/root*. This indicates that your machine is set up to use only the local domain for authentication.

Figure 11-10. Configuring access to NetInfo based directory services



Location: Automatic

Connect to parent NetInfo server at startup

If you are not connected to a network with NetInfo servers, checking boxes below will cause a delay during startup as your computer looks for a NetInfo server.

☐ Attempt to connect using Broadcast protocol

☑ Attempt to connect using DHCP protocol

☐ Attempt to connect to a specific NetInfo server:

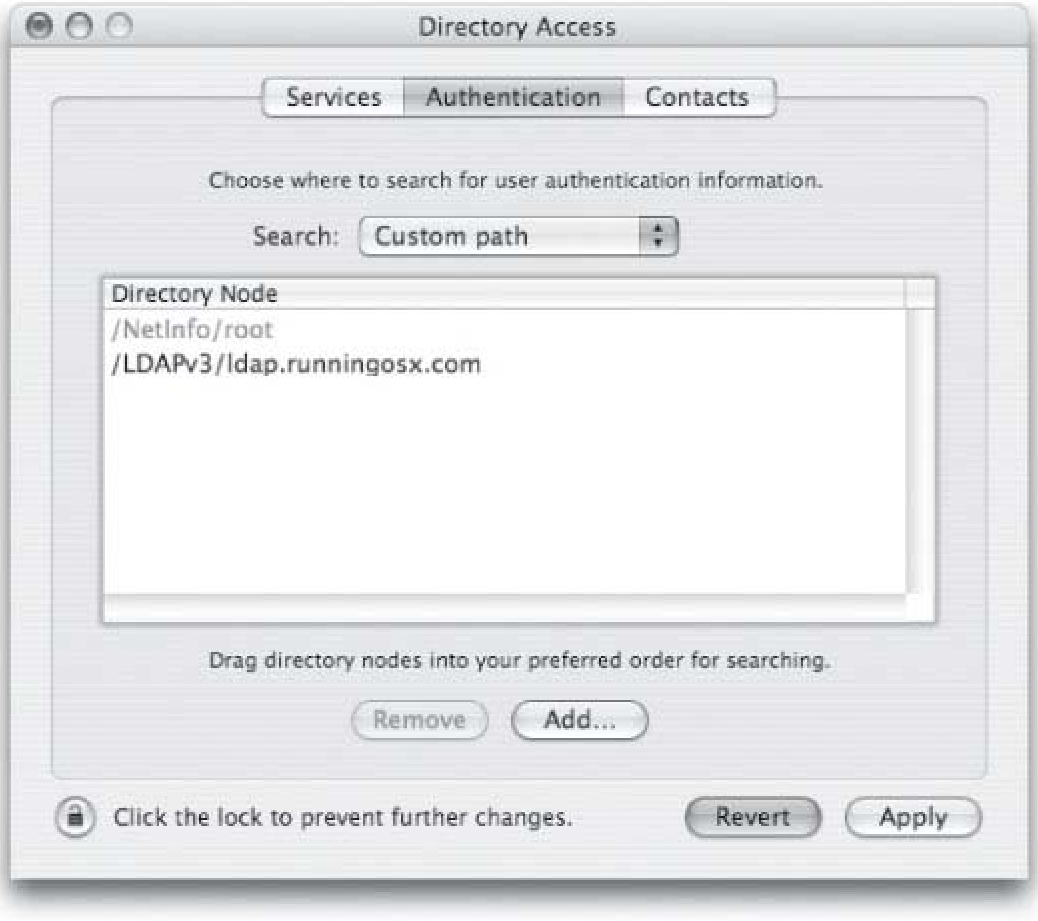
Server IP Address:

Server Tag:

To add a shared domain directory service to be used for authentication, change the search pull-down menu to Custom, as shown in Figure 11-11. You'll then be able to click the Add button to use any of the servers that you configured in the Services panel. In Figure 11-11 I've set up Open Directory to use the LDAPv3 server located at *ldap.runningosx.com* to serve as an authentication directory service. This means any user with an account defined by that server will be able to log on the machine. If you have configured multiple servers, you can drag them into the order in which you want them to be consulted for authentication purposes. Note that */NetInfo/root* stays at the top of the list. This ensures that the local domain always takes precedence over information in any shared domain.

When you click the Add button, you'll also notice an entry for */BSD/local*. This sets up Open Directory to use the classic Unix */etc/passwd* and */etc/group* files for authentication. It lets you use these files if you wish, but for all the reasons stated at the beginning of the chapter, you most likely don't want to use this option.

Figure 11-11. Configuring directory services used for authentication



Choose where to search for user authentication information.

Search: Custom path

Directory Node

/NetInfo/root

/LDAPv3/ldap.runningosx.com

Drag directory nodes into your preferred order for searching.

Click the lock to prevent further changes.

11.5.3 Configuring Shared Domain Contacts

Not only does Open Directory handle network services and authentication duties, but it can also provide contact information to the Address Book. To use shared domain servers for contact data, simply change the search pull-down menu to Custom Path (just as you did for Authentication) and add the servers you would like to use.

> Configuring shared contacts in Open Directory isn't the only way to take advantage of LDAP servers from the Address Book. You can set any number of LDAP servers in the Address Book preferences to use as sources of contact information.

11.6 Kerberos and Single Sign-on

Kerberos is a network authentication protocol that was developed at MIT to allow applications to identify users over open and insecure networks. It is used by governments, large corporations, and higher education. Kerberos is also the native authentication protocol of Active Directory. Since Jaguar, Apple has been moving aggressively to support Kerberos in both Mac OS X Server and Mac OS X—as well as all of the password-using applications in Mac OS X such as Mail, FTP, SSH, and Apple File Sharing. The reason Apple is making this push is to enable *single sign-on*.

Single sign-on means that after a user enters a name and a password in the login window, every application on the system that needs to authenticate itself for a network service, for example, Mail wanting to log in to the mail server, can do so automatically without requiring the user to enter a different username and password.

For users of Mac OS X, either Kerberos is configured for your network and it just works out of the box, or there is a bit of configuration work to be accomplished. If your network falls into the second category, you'll need to get some information from your system administrator.

11.7 Command-Line Open Directory Tools

Mac OS X provides a suite of command-line tools to view and manage the information in Open Directory. The most useful of these tools are:

dscl

A general-purpose interactive command-line tool for working with data in any Open Directory data source including LDAP, NetInfo, SMB, and Rendezvous. This tool is new in Panther and can also be used in single-shot mode.

nicl

An interactive command-line tool for working with data in a NetInfo database. Unlike *dscl*, this tool will not work with any other data source. This tool can also be used in single-shot mode.

nidump

Extracts data from a NetInfo database into either legacy Unix flat file formats (such as the files found in the */etc* directory) or into a NetInfo-specific raw format.

niload

Loads data from a flat file, either a Unix */etc* format file or a NetInfo raw format file.

niutil

A single-shot tool for reading and writing NetInfo database information. You should consider using *nicl* in single-shot mode instead of this tool.

nifind

Searches through a NetInfo database for directories that match a pattern.

nigrep

Searches through a NetInfo database for directories or properties that match a particular pattern.

Of these commands, the most useful are *dscl, nicl, nidump*, and *niload*.

11.7.1 Backing Up and Restoring a NetInfo Database

Before you get too creative with your NetInfo database, you should make sure you have a good backup of it. You can create a backup simply by copying the */var/db/netinfo/local.nidb* directory. For example, you can use the following to create a backup:

```
    $ sudo cp -R /var/db/netinfo/local.nidb /var/backups/backup.nidb
```

If you manage to get the NetInfo database into a state where it can't be used, you can boot into single-user mode by holding down ⌘-S as the computer starts up. When you are presented with the single-user prompt (#), execute the commands in Example 11-1.

Example 11-1. Restoring the NetInfo database in single-user mode

```
     # /sbin/fsck -y

# /sbin/mount -uw /

# mv /var/db/netinfo/local.nidb /var/backups/damaged.nidb

# cp -R /var/backups/backup.nidb /var/db/netinfo/local.nidb
```

The first command makes sure that the filesystem is safe. The second command mounts the filesystem in read/write mode. The third command moves the damaged database out of the way. The fourth copies the backup copy of the database back into place.

11.7.2 Using dscl

The*dscl* command-line tool is an interactive program like the shell, which means that when you run it, it stays active. And, like the shell, it features tab-completion and a history. This lets you navigate through the large amounts of data that can be present in Open Directory. The best way to learn how to use this tool is to go on a guided tour. Example 11-2 shows how to launch *dscl* so that it will connect to Open Directory running on your local system.

Example 11-2. Launching dscl

```
$ dscl .

/ >
```

The dot is important. It means connect to Open Directory on the local system. You'll notice that the prompt changes to the greater-than (>) symbol. This means that *dscl* is ready to accept your commands.Table 11-1 lists the various commands you can use. The top level of the directory tree that you are at when you start dscl contains the various directory sources that Open Directory has access to. To see these sources of directory information, use the *list* command as shown in Example 11-3.

Table 11-1. dscl commands.

Command	Description
`help`	Prints out the various commands for *dscl*.
`list [`*path*`]`	Lists the subdirectories of the given directory. If no path is given,the subdirectories of the current directory will be listed.
`cd`*path*	Changes the working directory to the path given.
`read [`*path* `[`*key*`]]`	Reads the properties in a directory. If no path is given, the current path is used. You can specify a specific key of a property to read. If no key is given, all the properties of the path will be printed.
`search`*path key val*	Searches for records that match a pattern at the given path.
`auth`*user* `[`*password*`]`	Authenticate to the system as a user. You can either specifiy a password or let *dscl* prompt you for one.
`create`*path* `[`*key* `[`*val*`]]`	Creates a path or a property with the given key and value at a path. This command will overwrite any existing path or property.
`append`*path key val*	Appends a property value to the property at the given path.
`merge`*path key val*	Appends a property value to the property at the given path if the value does not already exist.
`delete`*path* `[`*key* `[`*val*`]]`	Deletes a path or a property value at the given path.
`change`*path key oldval newval*	Changes the property value at the given path.
`changei`*path key index val*	Changes the property value at a given index for the given path .
`passwd`*userPath* `[`*newPassword*`]`	Changes the password for the user whose record is at a given path.
`quit`	Quits*dscl*.

Example 11-3. Using the dscl list command

```
/ > list
NetInfo
LDAPv3
PasswordServer
Rendezvous
SLP
SMB
Search
Contact
/ >
```

To take a look at the NetInfo database that serves as the local directory service domain, use the *cd* and then the *list* commands as shown in Example 11-4.

Example 11-4. Using the dscl cd command

```
/ > cd NetInfo/
/NetInfo/root > list
AFPUserAliases
Aliases
Config
Groups
Hosts
Machines
Networks
Users
```

```
/NetInfo/root >
```

You'll notice the exact same set of subdirectories in the NetInfo tree here as you saw in NetInfo Manager, with one key difference, the initial letter in each directory name is in uppercase. This reflects a mapping from the lowercase names that NetInfo uses to the new naming convention used by Open Directory. You can view the properties of a subdirectory record with the *read* command, as shown in Example 11-5.

To take a look at the NetInfo database that serves as the local directory service domain, use the *cd* and then the *list* commands as shown in Example 11-4.

Example 11-5. Using the dscl read command

```
/NetInfo/root > read Users/duncan

_shadow_passwd:

_writers_hint: duncan

_writers_passwd: duncan

_writers_picture: duncan

_writers_realname: duncan

_writers_tim_password: duncan

sharedDir: Public

AppleMetaNodeLocation: /NetInfo/root

AuthenticationAuthority: ;ShadowHash;

AuthenticationHint:

GeneratedUID: A754A5A2-F4B3-11D7-A5F4-000A9599E492

NFSHomeDirectory: /Users/duncan

Password: ********

Picture: /Library/Caches/com.apple.user501pictureCache.userImage

PrimaryGroupID: 501

RealName: James Duncan Davidson

RecordName: duncan

UniqueID: 501
```

```
UserShell: /bin/bash

/NetInfo/root >
```

Once again, you should notice that this is basically the same information that you saw in NetInfo Manager, but some of the names are different. You'll also notice the difference in capitalization due to a bit of translation behind the scenes between the older NetInfo style of naming and the newer Open Directory style. Table 11-2 provides a mapping of these properties. If you just want to read a single property from a user record, you can use the read command, as shown in Example 11-6.

Example 11-6. Reading a property from a user record

```
/NetInfo/root/Users > read duncan PrimaryGroupID

PrimaryGroupID: 501
```

So far, you've only been reading values from the database. To write a value, you'll first need to authenticate and then use the *create* command. For example, if you wanted to set the *AuthenticationHint* property, you would use the set of commands shown in Example 11-7.

Example 11-7. Writing a property to a user record

```
/NetInfo/root/Users > auth duncan

Password:********

/NetInfo/root/Users > create duncan AuthenticationHint "The Usual"
```

Now, when you take a look at the *AuthenticationHint* property, you'll see that it has changed, as shown in Example 11-8.

Example 11-8. Checking a modified property

```
/NetInfo/root/Users > read duncan AuthenticationHint
```

```
AuthenticationHint: The Usual
```

> One important thing to note about the *auth* command is that it will only authenticate you in the directory database that you are working in. For example, if you have an LDAPv3 directory configured in Open Directory, when you authenticate in the NetInfo database, you are not authenticated in the LDAPv3 database.

When you are ready to leave the *dscl* shell, use the *quit* command.

You can also use *dscl* in single-shot mode—this is where you can enter a command directly at the shell prompt and it won't result in an interactive shell. For example, from the command line we can read the *AuthenticationHint* property as shown in Example 11-9.

Example 11-9. Using dscl in single-shot mode

```
$ dscl localhost -read /NetInfo/root/Users/duncan AuthenticationHint

AuthenticationHint: The Usual
```

11.7.3 nicl

The*nicl* command works in much the same way as the *dscl* command. The major difference between the two is that *nicl* presents only data in the NetInfo database and presents property names in NetInfo format. To run *nicl*, use the command shown in Example 11-10.

Example 11-10. Starting nicl

```
$ nicl .

/ >
```

When you are at the *nicl* prompt, you can use the same commands as *dscl*, listed in Table 11-2. For example, to list the directories at the top of the NetInfo tree, use the *list* command as shown inExample 11-11.

Table 11-2. Mapping between NetInfo and Open Directory style user property names

Open Directory Property Key	NetInfo Directory Property Key
AuthenticationAuthority	authentication_authority
AuthenticationHint	hint
GeneratedUID	generateduid
NFSHomeDirectory	home
Password	passwd
Picture	picture
PrimaryGroupID	gid
RealName	realname
RecordName	name
UniqueID	uid
UserShell	shell

Example 11-11. Using the nicl list command

```
/ > list

1         users

2         groups

3         machines

4         networks

5         protocols

6         rpcs

7         services

8         aliases

9         mounts

10        printers

68        afpuser_aliases

72        config
```

You can see from the output in Example 11-11 that each directory in the NetInfo database has a number associated with it as well as a name. To read the properties associated with a user, use the*read* command as shown in Example 11-12.

Example 11-12. Using the nicl read command

```
/ > read /users/duncan

sharedDir: Public

authentication_authority: ;ShadowHash;

name: duncan

home: /Users/duncan

passwd: ********

realname: James Duncan Davidson

uid: 501

shell: /bin/bash

generateduid: A754A5A2-F4B3-11D7-A5F4-000A9599E492

gid: 501

picture: /Library/Caches/com.apple.user501pictureCache.userImage

hint: foo

_writers_passwd: duncan

_writers_hint: duncan

_writers_picture: duncan

_shadow_passwd:

_writers_tim_password: duncan

_writers_realname: duncan
```

11.7.4 nidump

The*nidump* command is primarily used for outputting the contents of the NetInfo database in standard Unix formats and for making backups of the database. For example, to output the user information in the database in the traditional format used by */etc/passwd*, you would use the *nidump* command as shown in Example 11-13.

Example 11-13. Using the nidump command

```
$ nidump passwd .
nobody:*:-2:-2::0:0:Unprivileged User:/var/empty:/usr/bin/false
root:*:0:0::0:0:System Administrator:/var/root:/bin/sh
daemon:*:1:1::0:0:System Services:/var/root:/usr/bin/false
unknown:*:99:99::0:0:Unknown User:/var/empty:/usr/bin/false
smmsp:*:25:25::0:0:Sendmail User:/private/etc/mail:/usr/bin/false
lp:*:26:26::0:0:Printing Services:/var/spool/cups:/usr/bin/false
postfix:*:27:27::0:0:Postfix User:/var/spool/postfix:/usr/bin/false
www:*:70:70::0:0:World Wide Web Server:/Library/WebServer:/usr/bin/false
eppc:*:71:71::0:0:Apple Events User:/var/empty:/usr/bin/false
mysql:*:74:74::0:0:MySQL Server:/var/empty:/usr/bin/false
sshd:*:75:75::0:0:sshd Privilege separation:/var/empty:/usr/bin/false
qtss:*:76:76::0:0:QuickTime Streaming Server:/var/empty:/usr/bin/false
cyrus:*:77:6::0:0:Cyrus User:/var/imap:/usr/bin/false
mailman:*:78:78::0:0:Mailman user:/var/empty:/usr/bin/false
appserver:*:79:79::0:0:Application Server:/var/empty:/usr/bin/false
duncan:********:501:501::0:0:James Duncan Davidson:/Users/duncan:/bin/bash
```

To make a backup of the NetInfo database, handy for when you want to make potentially harmful changes, use the following command:

```
    $ nidump -r / . > nibackup.txt
```

11.7.5 niload

The*niload* command is complementary to *nidump*. This command can accept a variety of Unix */etc* files as input to add data to a NetInfo database. For example, the following command loads a*/etc/passwd*-style file named users into the database:

```
    $ niload passwd . < users
```

To perform a full database restore from the output of a *nidump* command, you would use the following command:

```
    $ nidload -r / . < nibackup.txt
```

11.8 Further Explorations

Several resources are available to help you deepen your understanding of directory services and the technologies behind Open Directory. These are:

- *Mac OS X Server Open Directory Administration* by Apple Computer available at www.apple.com/macosx/server.
- *LDAP System Administration*, by Gerald Carter (O'Reilly & Associates, 2003)
- *Kerberos: The Definitive Guide*, by Jason Garman (O'Reilly & Associates, 2003)

Also, you may want to refer to the following manpages:

- *DirectoryService*
- *netinfo*
- *dscl*
- *nicl*
- *nidump*
- *niload*
- *fsck*

Chapter 12. Printing

The Mac's impressive printing capabilities date back almost 20 years to the introduction of the PostScript-based LaserWriter printer in 1985. During the transition to Mac OS X, Apple redesigned the printing system to make it easier for printers to develop drivers, and also made it more flexible and robust. This was by no means a seamless transition as it took time for printer manufacturers to rewrite their drivers. But with Panther, the payoff for this work has been realized: seamless printing from any application to any Mac- or Windows-based printer and beyond to PDF generation and built-in fax capabilities. What once was difficult, if not impossible, is now very easy to do.

This chapter gives you an overview of how the print system works, how to add and configure local and network printers, and how to work with printers from the command line.

12.1 Print System Overview

Mac OS X's printing system makes use of the following technologies:

- Quartz and PDF for creating print-ready output from applications.
- Common Unix Printing System (CUPS) for managing printers, printer drivers, and print jobs. CUPS also accepts print jobs using the Internet Printing Protocol (IPP). The CUPS project can be found on the Internet at *www.cups.org/*.
- Samba 3 for connecting to Windows SMB-based printers and for receiving print jobs from Windows clients. Additional information on Samba can be found at *www.samba.org/*.
- Open Directory for locating printers with LDAP, NetInfo, Rendezvous, and SMB.

Generally the printers that you will use come in two varieties:

Locally attached

Printers that are directly connected to your machine via USB or FireWire. You can always access these printers and, if you choose, share them with other computers on the network by going to System Preferences → Sharing → Services and enabling Printer Sharing.

Network accessible

Printers that are either connected directly to the network (such as many laser printers in office environments) or are attached to other computers that have enabled printer sharing. Because Mac OS X understands a variety of network protocols, it can connect to printers hosted by Windows- and Unix-based machines as well as by other Macs.

In addition there are two virtual types of printers that are built into the print system:

Fax

From any print dialog, you can press the Fax button and fax a document with the modem built into your Mac.

PDF

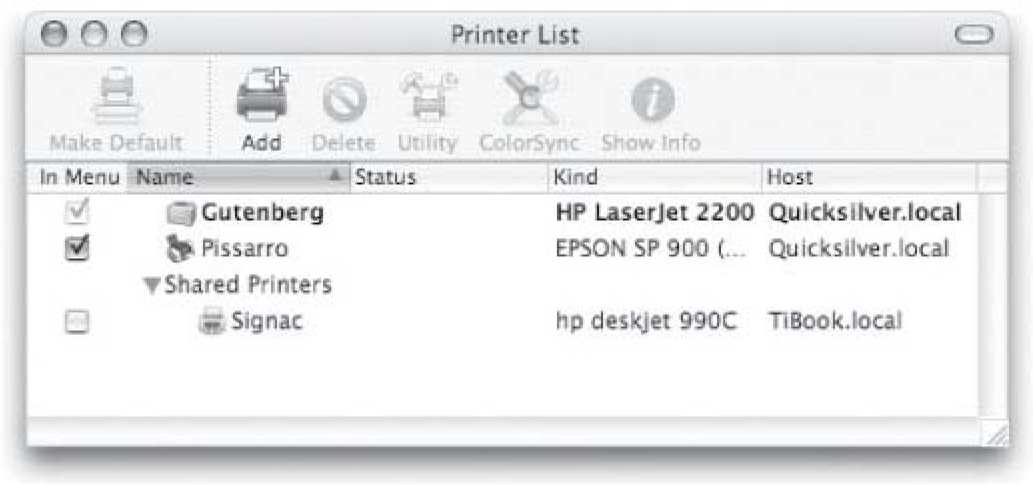
Instead of printing to hard copy, you can easily generate PDF files.

12.2 Managing Printers

The Printer Setup Utility (*/Applications/Utilities*) is the principal application used to define and manage printers from the GUI. You can either launch this application directly or from the Print & Fax preference panel. When launched, it displays a list of the printers that you can use, as shown inFigure 12-1.

Each printer in the list shows whether it appears in Print dialog's menu, the name of the printer, its current status, the kind of printer it is, and the host it is attached to.

Figure 12-1. The Printer Setup Utility window

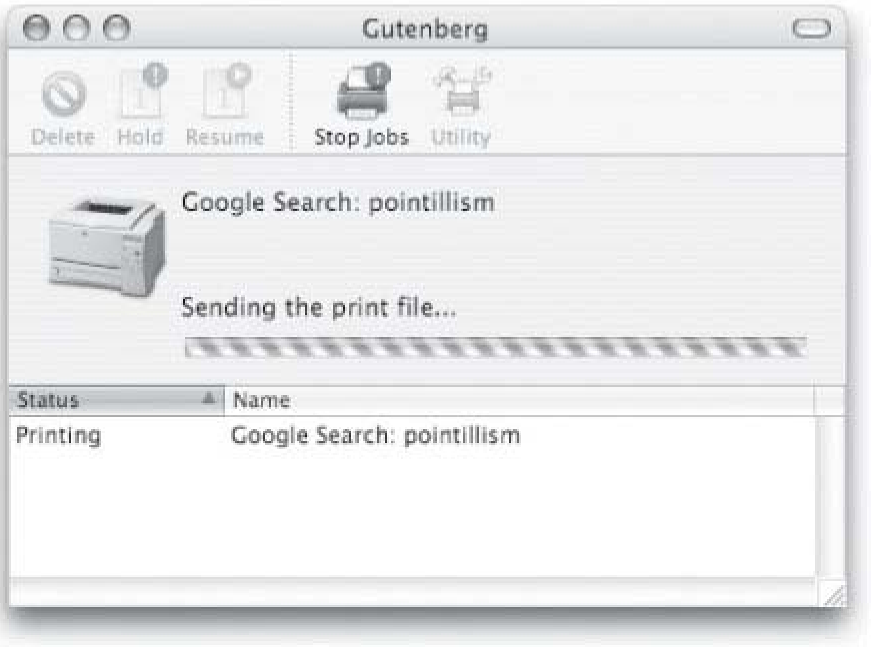


In Menu	Name	Status	Kind	Host
✓	**Gutenberg**		**HP LaserJet 2200**	**Quicksilver.local**
☑	Pissarro		EPSON SP 900 (...	Quicksilver.local
	▼Shared Printers			
□	Signac		hp deskjet 990C	TiBook.local

In the figure, two printers are local to the machine (*Quicksilver.local*), and one printer is being shared from another machine (*TiBook.local*). The default printer, which is shown in boldface in the Printer Setup Utility, is used automatically by the system whenever a document is printed. You can select the default printer by highlighting the name of the printer and clicking the Make Default button.

Double-clicking a printer in the printer list brings up a window showing the print queue (a list of pending print jobs) for that printer along with the status of the printer, as shown in Figure 12-2.

Figure 12-2. A printer queue status window shows a print job being sent to a printer



Google Search: pointillism

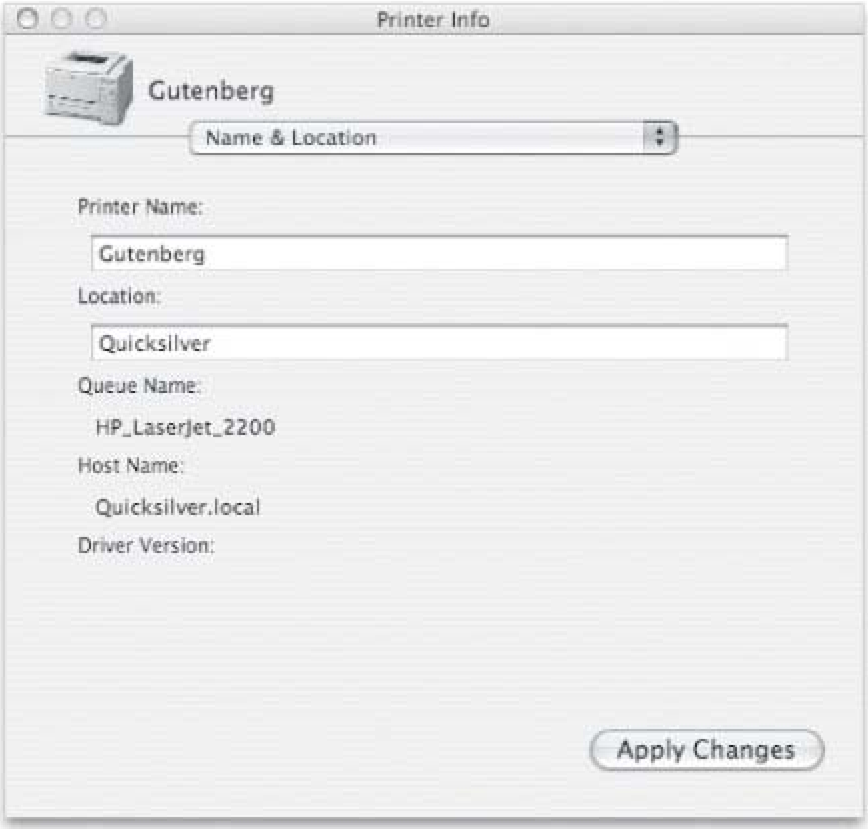
Sending the print file...

Status	Name
Printing	Google Search: pointillism

You can get more information about a printer by highlighting it in the Printer Setup Utility list and clicking the Show Info button, or by going to the Printers→Show Info (⌘-I) menu. This will bring up a dialog window as shown in Figure 12-3. The window shows the name and location of the printer (both fields can be edited for a local printer), the name of the print queue (used by the underlying CUPS system to route print jobs) for the printer, and the host the printer is located on. This window also contains a pull-down menu that gives you access to the printer model information as well as the installable options, such as duplexer, extra paper trays, and installable memory that are available to the printer.

To retrieve the list of printers that your Mac knows about, go to the command line and use the *lpstat -a* command as shown in Example 12-1.

Figure 12-3. The printer info window



Printer Name:
Gutenberg

Location:
Quicksilver

Queue Name:
HP_LaserJet_2200

Host Name:
Quicksilver.local

Driver Version:

Example 12-1. Seeing the computers attached to a Mac

```
$lpstat -a

DESKJET_990C@TiBook accepting requests since Jan 01 00:00

HP_LaserJet_2200 accepting requests since Jan 01 00:00

Stylus_Photo_900 accepting requests since Jan 01 00:00

Internal_Modem accepting requests since Jan 01 00:00
```

The output from the *lpstat -a* command lists the printers by queue name and, in the case of a printer located on a remote machine, the name of the machine that hosts the print queue for the printer.

> Two items may show up in the output from *lpstat -a* that don't appear in the Printer Setup Utility. These will be labeled *Internal_Modem* and *Bluetooth-Modem*. These devices are used by the underlying print system to handle faxing from applications via the Print Dialog box.

To see the default printer from the command line, use the *lpstat -d* command, as shown in Example 12-2.

Example 12-2. Seeing the default printer from the command line

```
$lpstat -d

system default destination: HP_LaserJet_2200
```

To set the default printer that is in use by the system from the command line, use the *lpoptions -d* command with the queue name of the printer to make the default queue. Example 12-3 shows the use of this command.

Example 12-3. Setting the default printer from the command line

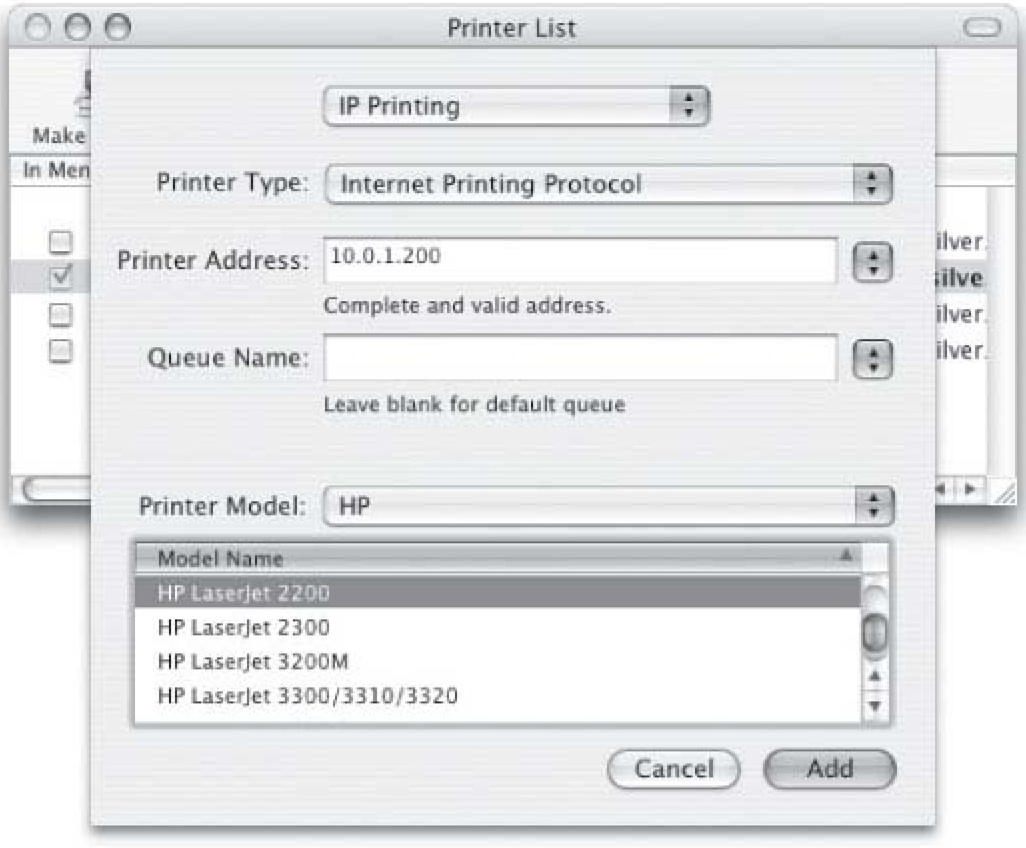
```
$lpoptions -d Stylus_Photo_900

job-sheets=none,none
```

12.3 Adding and Configuring Printers

Many printers, including those that are directly connected to your machine via USB or FireWire and those that are shared by other Macs, automatically appear in the list of printers in the Printer Setup Utility, making them ready to use almost immediately. However, if you want to use a network-based printer that isn't shared by a Mac or use a printer for which your Mac can't figure out the driver to use automatically, you'll have to add it yourself.

To add a printer, click the Add Printer button (or use the Printers→Add Printer menu). After a few moments, a dialog sheet (shown in Figure 12-4) pops up that allows you to add the printer and specify it as the default. The pull-down menu at the top of the dialog has the following options to choose from:

Figure 12-4. Adding a printer using Print Utility



IP Printing

Printer Type: Internet Printing Protocol

Printer Address: 10.0.1.200

Complete and valid address.

Queue Name:

Leave blank for default queue

Printer Model: HP

Model Name
HP LaserJet 2200
HP LaserJet 2300
HP LaserJet 3200M
HP LaserJet 3300/3310/3320

Can't Print to an LPD/LPR Printer?

The version of CUPS that ships in Panther implements a change which makes some older LPD/LPR print servers reject jobs. If you have this problem, you can find a workaround at *www.macosxhints.com/article.php?story=20031118180912371*.

AppleTalk

Lets you add and configure a printer using the legacy AppleTalk protocol. You can select from printers on the local AppleTalk zone and from any other zones that your computer knows about.

IP Printing

Lets you add and configure a printer connected to a server (either a standalone machine or an embedded server inside the printer) using an IP-based protocol. These protocols include: LPD/LPR, a Unix-based printing protocol that has become a de-facto industry standard; Internet Printing Protocol(IPP), a newer protocol based on HTTP that is intended to provide better interoperability over Internet- based networks; and HP's Jet Direct protocol, widely deployed in HP network- enabled printers. Because they are IP-based, these protocols can be used to communicate with printers beyond your local network.

Open Directory

Lets you add a printer defined in the various Directory Services that your computer has access to. For example, if your network administrator set up information about the computers on your network in an LDAP server that your machine connects to, that information shows up here.

Rendezvous

Lets you add Rendezvous-enabled printers that don't automatically show up in your printer list but that exist on the network.

> If you are using a USB printer attached to an AirPort Extreme base station, the printer will show up in the Rendezvous section.

USB

> Lets you add any USB printer that isn't already set up on your machine. Typically, only USB-based printers for which Mac OS X can't figure out the proper driver to use appear here instead of being automatically added to the list of printers.

Windows Printing

> Lets you connect to any printer that is being shared by a Windows or Windows Server machine. The list of printers is organized by workgroup or domain; you can select the workgroup or domain to look for printers.

Managing Printers over the Web with CUPS

The Printer Setup Utility isn't the only interface for managing the print system. You can access the CUPS layer of the print system directly by loading *http://localhost:631* into a web browser. This presents a web-based interface that allows you to directly interact with the different parts of the CUPS system.

The most useful part of the CUPS web interface is the Printers section. This lists the same kind of information, and lets you perform the same kinds of tasks, as the Printer Setup Utility. Its main advantage is that it is accessible from other machines on the network when printer sharing is turned on.

For more sophisticated control over how CUPS works and to exercise greater security over how your computer works as a network-accessible print server, you can edit the plain-text configuration files that CUPS uses that are located in the */etc/cups* directory. You can find full documentation for configuring and administering CUPS online at *www.cups.org*, or by following the *Documentation* link from the CUPS configuration web interface.

No matter which type of connection is used to connect to a printer, you'll need to select in the printer model by using the Printer Model pull-down menu at the bottom of the Add Printer dialog. This will set the print system to use the correct driver for the printer. Apple ships with drivers for printers from Apple, Brother, Canon, Epson, HP, Lexmark, Tektronix, and Xerox. For drivers for other printers, you'll need to check your manufacturer's web site.

> Support for many printers that were previously unsupported under Mac OS X has been added thanks to CUPS and Gimp-Print, a set of open source printer drivers for many older (as well as newer) printers. You can even find a driver for the original Epson 9-pin dot matrix printer by looking under ESP in the Printer Model pull-down in the Add Printer dialog. You can find more information on Gimp-Print at *gimpprint.sourceforge.net/*.

Once a printer has been added, you need to make sure that it is configured properly. There are many settings, such as whether or not a printer has a duplexer for two-sided printing, or how much memory it has installed; these items can't always be automatically configured for you. To examine the configuration for a printer, select it in the list and click the Info button, or select Printers→Show Info (⌘-I) from the menu bar.

12.3.1 Determining IP-based print queue names

To add an IP-based printer, you'll need to know the IP address or hostname of the server, possibly the name of the queue for the printer you want to use if the server handles more than one printer, and the model of the printer. To get the name of a queue for a printer, you can either ask a network administrator or use the *lpstat -a -h* command, as shown in Example 12-4.

Example 12-4. Using lpstat -h to list queue names on a machine

```
$lpstat -a -h TiBook.local

Internal_Modem accepting requests since Jan 01 00:00

DESKJET_990C@TiBook accepting requests since Jan 01 00:00

HP_LaserJet_2200@Quicksilver accepting requests since Jan 01 00:00

Stylus_Photo_9001@Quicksilver accepting requests since Jan 01 00:00
```

The output in Example 12-4 contains more printers than those connected to the machine that the information was requested from. It also includes the other shared printer queue names on the network and indicates the hostnames that those queues are on.

12.3.2 Vendor-specific connections

Depending on the drivers you have installed, there may be several additional items in the Add Printer pull-down menu. These are provided by printer manufacturers to handle situations that aren't covered by the built-in capabilities of Mac OS X.

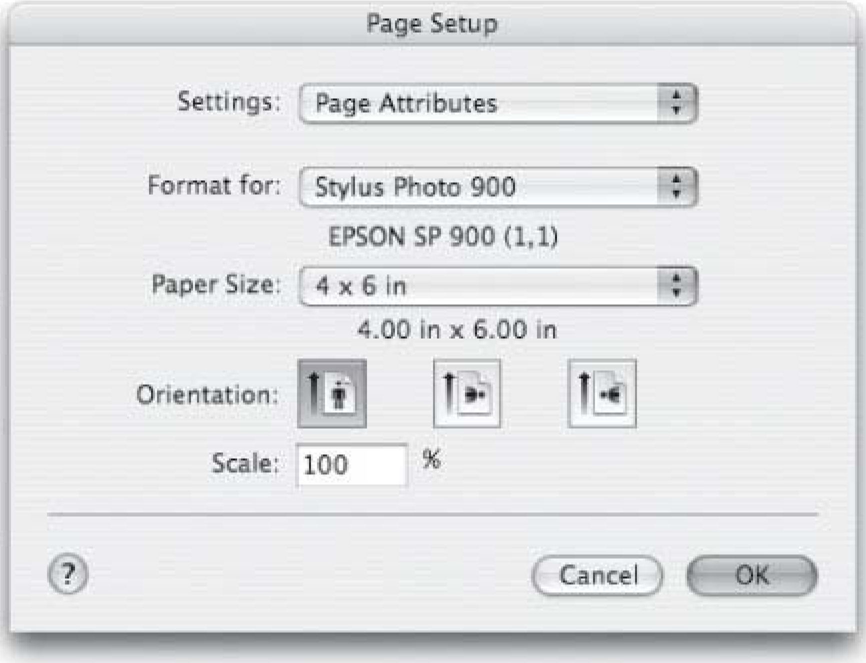
12.4 Anatomy of a Print Job

Every print job starts as output from an application. The first step, albeit one that is already performed for you, is to produce output from an application and tell it what size paper you want to use and the orientation of that paper. In most applications this is usually accomplished by using the standard File→Page Setup (Shift-⌘-P) command, which brings up the dialog shown in Figure 12-5. Most of the time you'll want to leave the default settings in this dialog or just change the orientation of your page. However, if you are printing a 4 x 6 photo or an envelope, you'll want to select a printer attached to your system which will give you access to the page sizes it can handle.

> Not all applications have a Page Setup dialog box, and those that do don't always have the Shift-⌘-P keyboard shortcut.

The second step in getting output from an application is to print a document using the standard File→Print (⌘-P) command, which brings up the dialog shown in Figure 12-6. From here you can print to a printer, save to a PDF file, or fax documents anywhere in the world. This rather simple dialog gives access to a wealth of options that can be performed by both the system and by individual printer drivers. The dialog lets you select the printer to print to, listing both local printers and remote printers. It also gives you the opportunity to use a set of preset settings and to customize the printer-, and possibly, application-specific settings.

Figure 12-5. The Page Setup dialog



Page Setup

Settings: Page Attributes

Format for: Stylus Photo 900

EPSON SP 900 (1,1)

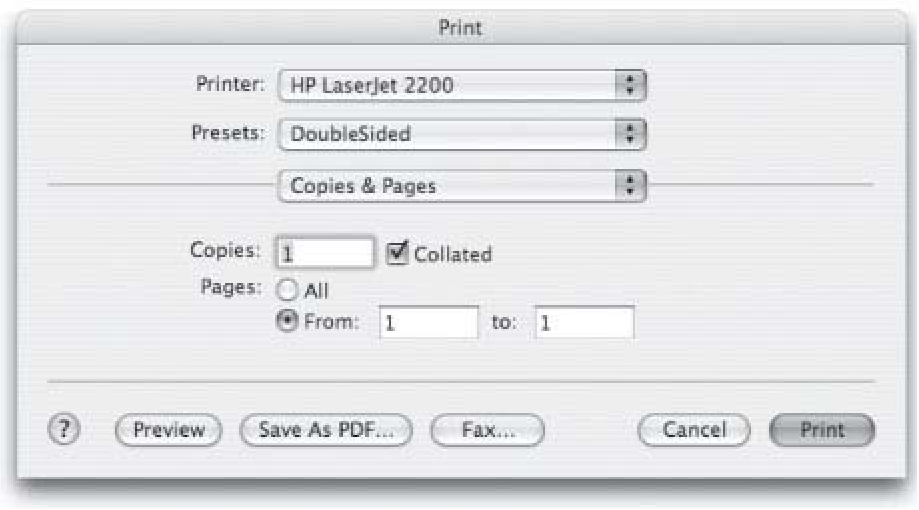
Paper Size: 4 x 6 in

4.00 in x 6.00 in

Orientation:

Scale: 100 %

Figure 12-6. The standard print dialog



Print

Printer: HP LaserJet 2200

Presets: DoubleSided

Copies & Pages

Copies: 1 ☑ Collated

Pages: ○ All

● From: 1 to: 1

The following settings are always available:

Copies & Pages

> Lets you specify how many copies of a document you want to print. Some applications put additional controls into this pane to allow you to specify what content to print in terms that make sense for the application. For example, when you print from Microsoft Excel, you have control over the sheets to print out of a workbook.

Layout

> Gives you the ability to print up to 16 pages on one side of a sheet of paper and the option to print on both sides of the sheet, if your printer supports duplex twosided printing. Confusingly enough, you can't control the orientation of printing here—you have to set that using the Page Setup dialog before you print.

Output Options

> Gives you the ability to save your print job as either a PDF or a PostScript file.

Scheduler

> Lets you schedule a print job to be printed at some time in the future. You can even place a job on the printer queue, and hold it indefinitely until a time when you are ready for it to be printed. This last option is useful for when you are traveling and don't have access to your printer back at the office, but want to create print jobs before arriving at your destination.

Paper Handling

Lets you control how the pages are ordered when printed, or to print only even- or odd-numbered pages. This is handy when you want to manually print two-sided documents on a printer that doesn't support duplex printing.

ColorSync

Sets whether or not you want the transformation from the document's color space to the printer's to be performed by the printer. Usually you want Mac OS X to perform this work unless you have already modified the colors in the document to print correctly on the target printer. This pane also gives you access to filters, such as applying a sepia tone or converting a grayscale document to black and white, that are performed by Quartz.

Summary

Gives a text representation of all the current settings for a print job.

In addition to these standard settings, many printer drivers add additional options to this menu. For example, the HP LaserJet printer driver adds panes to let you select which paper tray to use, whether or not to generate a cover page, and the resolution you want to print at. As another example, the Epson Stylus Photo printer drivers add panes allowing you to select the kind of paper you are printing on since different papers react differently to photographic printing.

Sometimes you have to look carefully here for a setting you want to use. For example, the driver for HP's InkJet printers that support duplex printing don't allow you to set up two-sided printing using the Layout pane; instead, a separate two-sided printing pane is provided and accessible via the lower pull-down menu. Of course we can't test and detail the quirks of each printer model here, but the point is you'll need to check out all the options that your printer driver gives you.

12.4.1 Saving Print Settings

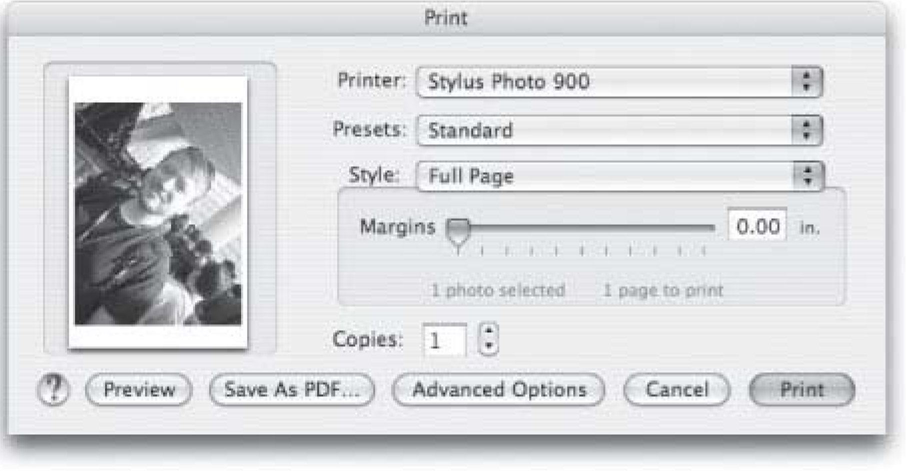
The Print dialog's Presets menu lets you create presets of your chosen settings across all the option panes, allowing you to quickly set up a print job. For example, you might want to provide a Double-Sided setting so you can quickly opt for duplex printing. Another good preset is a Photo preset that sets the paper type and ColorSync options for your particular printer.

To save a set of print settings, simply use the pull-down menu and select the Save As option. Once applied, the name that you choose for a group of print settings will always appear in this menu.

12.4.2 The Simplified Print Panel

Some applications that aim for an ease-of-use interface, such as iPhoto, can choose to present a simplified print panel as shown in Figure 12-7. This panel presents several of the settings located in the various sections in a single interface which many of the more complex settings of the regular print panel. You can get to all the options by clicking the Advanced Options button.

Figure 12-7. The simplified print panel



Print

Printer: Stylus Photo 900

Presets: Standard

Style: Full Page

Margins 0.00 in.

1 photo selected 1 page to print

Copies: 1

12.5 Printing from the Command Line

Since the Mac OS X printing infrastructure is built on top of various Unix tools, it should be no surprise that you can print from the command line. You can print plaintext, PDF, and PostScript files from the command line by using the following command:

```
lpfilename
```

You can print the following kinds of files from the command line to any printer:

- Plain-text files
- PDF
- PostScript

Because Mac OS X supports all these technologies in its print system, you don't have to worry about sending PostScript to a non-PostScript printer—even if the printer is an old dot matrix printer. For example, to print out the contents of a PostScript file named *requirements.ps*, you would use the command as shown in Example 12-5.

Example 12-5. Printing a text file to a printer

```
$lp ~/requirements.ps

request id is HP_LaserJet_2200-10 (1 file(s))
```

CUPS tells you that the job has been submitted and provides you with an *identifier* for the job. The identifier is composed of two parts: the name of the printer that the job will be printed on, in this case *HP_LaserJet_2200*, and a sequence number, in this case the number 10 indicates that this is the tenth job printed from this machine since CUPS was launched.

When called by itself, *lp* prints to the default printer on the system. To print to another printer, you have to specify its queue to the *lp* command. But how do you know the names of the printers attached to your computer? Use the *lpstat* command (refer back to Example 12-1 to see this in action).

Armed with this information, you can use the *lp* command using the *-d* option as shown in Example 12-6 The *-d* option tells *lp* which printer to select as its destination.

Example 12-6. Printing a PDF file to a specific printer

```
$lp -d Stylus_Photo_900 ~/test.pdf
```

```
request id is Stylus_Photo_900-14 (1 file(s))
```

12.5.1 Working with PostScript

Panther has added several features to work directly with PostScript files. For example, you can open PostScript files (which usually have the *.ps* extension) using Preview. When you do, Preview will convert the PostScript to PDF and then display the result.

There's also a command-line tool, *pstopdf*, which will convert a PostScript file to PDF. For example, to convert a PostScript file named *stats.ps*, you would use the following command:

```
$pstopdf stats.ps
```

This creates a *stats.pdf* file. To specify a different output filename, use the *-o* option:

```
$pstopdf stats.ps -o results.pdf
```

Another helpful utility is *enscript* which converts plain-text files to PostScript. This command uses the following syntax:

```
enscript -p outputfile filename
```

where the *outputfile* argument is the name of the file to save PostScript data to and the *filename* argument is the plain-text file to convert. For example, to convert your *.bash_profile* file to PostScript, you could use the following command:

```
$enscript -p profile.ps .bash_profile
```

If you call *enscript* without the *-p* and *outputfile* options, it will print the converted PostScript output directly to the default printer on the system.

12.5.2 Command-Line Printing to AppleTalk-based Printers

For the most part, printing to an AppleTalk printer from the command line is as easy as printing to any other printer, as long as you have set up the printer using the Printer Setup Utility so that it has a print queue on your system. If you don't have the AppleTalk printer set up on your system, the first thing you should do is either set it up or find a shared print queue for that printer on another system. Life is much simpler when CUPS takes care of things.

If you don't, or can't, set up the AppleTalk printer using the Printer Setup Utility, then you'll need to use the *atprint* command. Unlike *lp*, the *atprint* command doesn't work with files. Instead, it takes its input from the output of other commands and sends it either to a printer that you specify or to the default AppleTalk printer (which is not the same as the default printer for the system set in the Printer Setup Utility). You also have to be careful to send the right kind of data to the printer. In the case of most AppleTalk printers, this means that you have to send PostScript data. Example 12-7 shows how to send a PostScript file to a printer using atprint.

Example 12-7. Sending a PostScript file to an AppleTalk printer

```
$cat manuscript.ps | atprint

Looking for PET:LaserWriter@*.

Trying to connect to PET:LaserWriter@*.

atprint: printing on PET:LaserWriter@*.
```

If the data you want to send isn't in PostScript format, you'll need to convert it first (if possible) using*enscript*.Example 12-8 shows how to print the contents of a plain-text file to an AppleTalk printer named PET.

Example 12-8. Printing a plain-text file to an AppleTalk printer

```
$cat .bash_profile | enscript -p - | atprint PET

Looking for PET.

Trying to connect to PET:LaserWriter@*.

[ 1 pages * 1 copy ] left in -

atprint: printing on PET:LaserWriter@*.
```

The*at_cho_prn* command is used to set the default printer for use by *atprint*. You'll need to execute this command asroot, as shown in Example 12-9.

Example 12-9. Setting the default atprint printer

```
  $ sudo at_cho_prn

  Password:

  Zone:*???????????^@????????????`?????????` ?

    1: ff01.04.9dtPET:LaserWriter


  ITEM number (0 to make no selection)?1

  Default printer is:PET:LaserWriter@*
  status: idle
```

All in all, while the *atprint* command is handy in a pinch, the right thing to do is to set up the printer in the Printer Setup Utility and leverage the power of CUPS and the rest of the Mac OS X printing system to your advantage.

12.6 Further Explorations

To learn more about CUPS, you can access the CUPS documentation on your Mac by opening Safari and browsing *http://localhost:631/documentation.html*. You may want to explore the following manpages for more information about how to use the printing system from the command line.

- *lp*
- *lpstat*
- *lpoptions*
- *pstopdf*
- *enscript*
- *atprint*
- *at_cho_prn*

Chapter 13. Networking

Networking has always been easy on the Mac. The original Mac shipped with Apple-Talk, which made it easy to connect a group of computers and printers. Historically, other systems have had a harder time, using a variety of standards that were sometimes proprietary and that did not always work well together. The rise of the Internet, however, has meant that for all practical purposes there is now one primary network standard that all machines use: the suite of protocols based on the Internet Protocol, more commonly known as IP. In the development of Mac OS X, Apple has gone to great lengths to make IP as easy to use as possible, approaching the ease of use of AppleTalk. For the most part, the system will try to autoconfigure itself to work with whatever network is available, making it easy to use in this day of cafe-computing. (Would you like Wifi with your latte?)

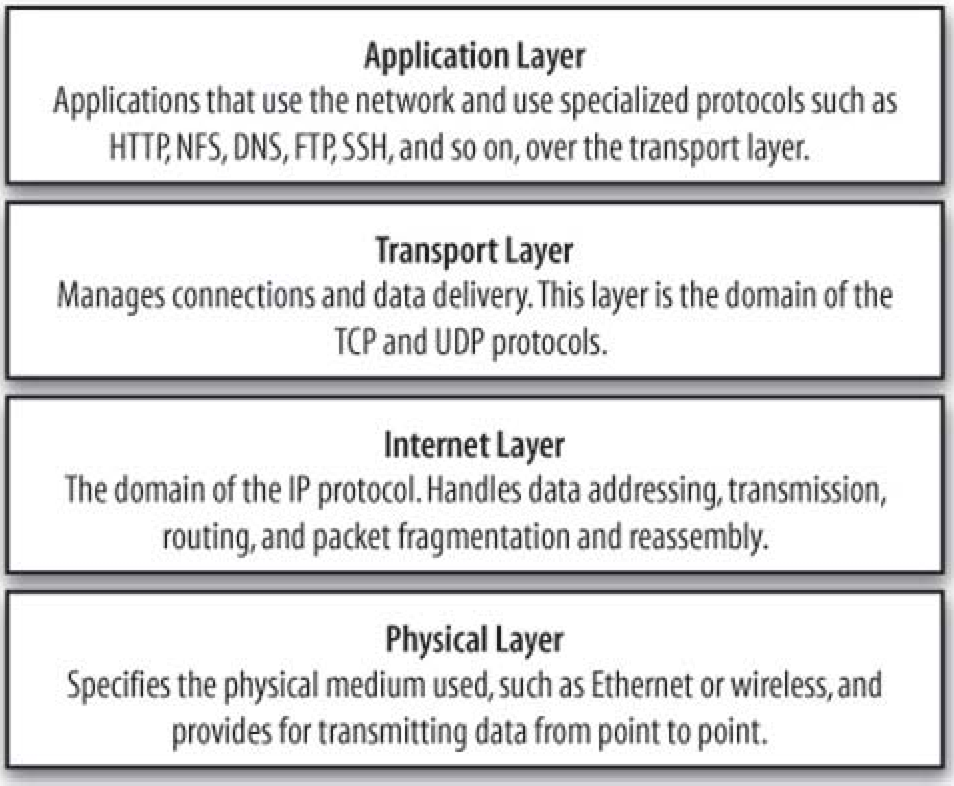
This chapter gives a fundamental view of how IP works and how to examine the various networking settings as well as monitor your network from both the command line and the GUI. Also, you'll see how dial-up networking, virtual private networks (VPNs), and firewalls can be configured.

13.1 The Internet Protocol

IP is the dominant network standard in use today because its namesake, the Internet, was success connecting the world's various networks together. IP is built on a set of assumptions that break the of networking into a set of layers, as shown in Figure 13-1. Each layer acts independently of the on and below it, allowing flexibility in how each layer is implemented.

Figure 13-1 is actually a simplified view of the abstract 7-layer networking model t you'll find in many books on IP networking. In practice, however, most people usua simplify things to the 4-layer view presented here.

Figure 13-1. A conceptual view of the networking stack



Application Layer
Applications that use the network and use specialized protocols such as HTTP, NFS, DNS, FTP, SSH, and so on, over the transport layer.

Transport Layer
Manages connections and data delivery. This layer is the domain of the TCP and UDP protocols.

Internet Layer
The domain of the IP protocol. Handles data addressing, transmission, routing, and packet fragmentation and reassembly.

Physical Layer
Specifies the physical medium used, such as Ethernet or wireless, and provides for transmitting data from point to point.

In addition to this layered approach, IP breaks data into separate self-contained pieces, known as , which are transmitted individually from their point of origin to their destination. Each packet can, i take a different route from its origin to its destination. This lets information flow even in the face o network topologies. The connections between networks can go down or change, but as long as the can find a new route, packets will eventually find their way through.

By using a layered model, and by using packets that can flow over virtually any kind of transmissio infrastructure, IP can and does run over a variety of mediums including wired Ethernet, wireless 80 fiber-optic lines, fireWire, and more. It's been joked that IP would work just fine over slips of pape by carrier pigeon (you can read the full joke at *www.ietf.org/rfc/rfc1149.txt*).

IP provides one other feature that is essential in today's networking world: it's *routable*. This mean networks can connect to each other and that packets can be routed across those networks to reach destination. This is how the Internet was built in the first place, as a collection of networks at unive government laboratories that had interconnections placed between them .

Above IP are two protocols that the applications on your machines use to access the network:

Transmission Control Protocol (TCP)

> Known as TCP/IP, or TCP over IP, this protocol specifies how packets of data should be trans over IP in a reliable fashion. TCP makes sure that all the data transmitted between two mach presented to an application in the order in which it was transmitted and received. This is very important because IP doesn't guarantee that packets will be transmitted in any particular ord usually they are sent in the order in which they are generated, but because of changing netw conditions, they may arrive in any order. TCP also performs error checking. If a packet is cor doesn't arrive, TCP automatically requests retransmission of the data from the sender withou intervention of the application. This means application writers don't have to worry about the which data is transported across the network. To an application, a network connection can be as a consistent stream of data when in fact that data is being split into several parts and mov the network in individual units.

User Datagram Protocol (UDP)

> UDP is used to send data between computers with a minimum of overhead. Unlike TCP, it do guarantee that the order in which packets arrive will be the same as when they were transmi it doesn't request redelivery of bad packets. Because of this, UDP is considered an unreliable but don't let the negative connotation of the word unreliable fool you. UDP is used extensivel performance needs are paramount, and the data errors being transmitted can be worked aro even ignored. For example, UDP is used by many video conferencing applications, such as iC Because of its low overhead and the fact that if a data frame representing a moment in time received, there is no need to retransmit it as it is no longer valid.

In conjunction with IP, two other important protocols work at the lower layers of the protocol stack

Internet Control Message Protocol (ICMP)

> ICMP is used by the network, at the Internet layer, to send messages to itself. For example, parts of a network can inform each other that errors have occurred or that a particular netwo segment is congested with too much traffic. It can also determine if a particular network add being used or not. ICMP is also used to test the connections between networks.

Address Resolution Protocol (ARP)

At the physical layer, an entirely different set of network addresses is used. For example, all cards have a unique Media Access Control (MAC) address to identify it. ARP is used to map th addresses used by the Internet layer of the network stack into the physical addresses used b equipment on the local network segment.

13.1.1 Network Addressing and Masks

Every device, or *host*, that uses the IP networking stack has an address so that packets can be sen address has to be unique to the network that the computer is part of. A traditional Internet networ (also known as an IPv4 address) is a sequence of 32 bits (4 bytes), usually written as a series of fo separated values in the form #.#.#.#, where each # is a decimal integer between 0 and 255 repre one of the four bytes of the address. For example, 66.93.174.29 is a valid IP address.

An IPv4 address is also composed of two parts, albeit indirectly. The first part identifies the local n while the second part identifies the host in that network. Confusingly, the boundary between these in an IPv4 address varies depending on the network. A network mask is used to delineate between network and host parts of the address. Network masks can come in many forms, but typically you'l following three network masks in use (listed both in dotted notation and in the hexidecimal form th see in some of the command-line utilities):

255.255.255.0 or 0xffffff00

Known as a Class C mask, this indicates that the first three bytes of the address are the netw address. There can be 254 hosts on a Class C network. This is the most common mask you'll day-to-day use.

255.255.0.0 or 0xffff0000

Known as a Class B mask, this indicates that the first two bytes of the address are the netwo address. There can be up to 65,000 hosts on a Class B network.

255.0.0.0 or 0xff000000

Known as a Class A mask, this indicates that the first byte of the address is the network addr the remaining bytes identify the host on the network. There can be up to 16.7 million hosts o A network.

Network addresses for hosts connected directly to the Internet must always come from an official s your Internet Service Provider (ISP). Your ISP owns the rights to use a block of addresses on the In and assigns your computer an IP address, or block of IP addresses, as part of your connection to th Internet.

13.1.1.1 IPv6 Addresses

The IPv4 addressing scheme was created at a time when there were relatively few computers runn protocol. The original designers didn't imagine that the Internet would expand to the point it is tod you can easily access it from most of the countries on the planet. Because of this expansion, as we anticipation that in the near future there will be an explosion of devices such as mobile phones and entertainment centers that will access the Internet, a new addressing scheme known as IPv6 is sta deployed.

IPv6 addresses are 128 bits long and are usually written as a series of 8 colon-separated, 16-bit va written in hexadecimal form. An example of an IPv6 address is:

```
fe80:0000:0000:0000:0203:93ff:feef:baa5
```

Unlike IPv4 addresses, IPv6 addresses don't require a separate network mask.

IPv6 isn't in wide use yet, at least in most of the networks that you are likely to find yourself using. to the limited number of IPv4 addresses, many networks, especially in Asia where the IPv4 address allocations have already run out, are starting to move to IPv6. In addition, many of the primary In backbone service providers are moving their networks to IPv6 in preparation for a full deployment.

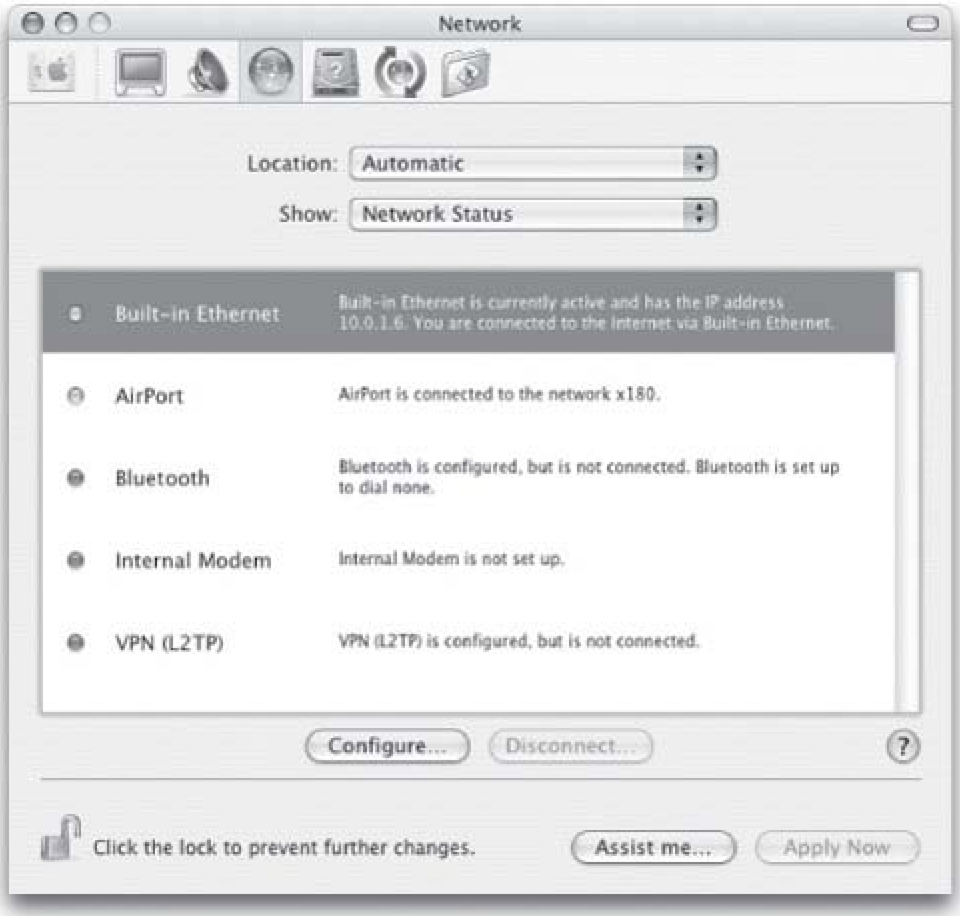
Mac OS X natively supports IPv6 addresses and, even though you might not be using an IPv6 netw few more years, you'll see these addresses as you work with the various network tools.

13.1.2 Examining Network Settings

There are several ways to see what IP address (or addresses if you have multiple interfaces active) using. The first, but not the most convenient, place most people will go to find this information is tl Network preference panel (System Preferences → Network), shown in Figure 13-2 This gives a head of the various network settings and the order in which network connections will be attempted.

A better place to look is the Network Utility (*/Applications/Utilities*) application, as shown in Figure Network Utility displays the information about the network interface that is currently being used to the network, also known as the *default interface*. You can select from the various network interface the popup menu.

Figure 13-2. Network Preference Pane status display



Location: Automatic

Show: Network Status

● Built-in Ethernet — Built-in Ethernet is currently active and has the IP address 10.0.1.6. You are connected to the Internet via Built-in Ethernet.

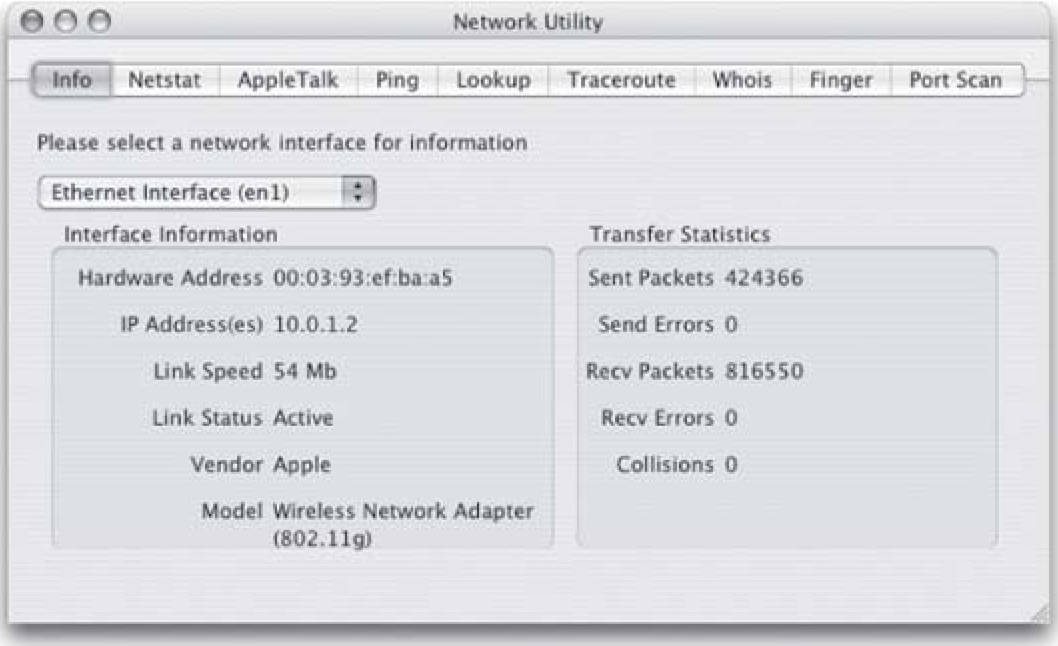
○ AirPort — AirPort is connected to the network x180.

● Bluetooth — Bluetooth is configured, but is not connected. Bluetooth is set up to dial none.

● Internal Modem — Internal Modem is not set up.

● VPN (L2TP) — VPN (L2TP) is configured, but is not connected.

Figure 13-3. Network Utility showing information on an active interf



Please select a network interface for information

Ethernet Interface (en1)

Interface Information

Hardware Address 00:03:93:ef:ba:a5

IP Address(es) 10.0.1.2

Link Speed 54 Mb

Link Status Active

Vendor Apple

Model Wireless Network Adapter (802.11g)

Transfer Statistics

Sent Packets 424366

Send Errors 0

Recv Packets 816550

Recv Errors 0

Collisions 0

The information you'll see in this display consists of the following:

Hardware Address

Displays the physical MAC address of the interface. This is the address used at the physical la move data to and from the network.

IP Address

Displays the IP address of the interface.

Link Speed and Link Status

Provides the speed of the connection as well as whether it is active or not.

Vendor and Model Version

Provides the maker of the physical interface as well any version information in the firmware interface.

Transfer Statistics

Provides some information on how the interface is performing. Typically as you use the netw see the sent and received packets counters increasing. You shouldn't see any errors or collisi something is wrong with your network.

On the menu, you'll notice designators like (en0), (en1), or (fw0). The operating system uses these identifiers to track the various interfaces. The identifiers you might see are:

lo0

This is the loopback interface, which allows network-aware programs to communicate with o programs on the system without sending packets onto the network. This interface always has (inet) address of 127.0.0.1 and the IPv6 (inet6) address of ::1.

en0

The primary Ethernet interface of your machine. This will always be the built-in Ethernet con your Mac.

en1

An Ethernet interface. Typically en1 will be the Airport card in your computer (if you have on second Ethernet card. In addition to the IPv6 (inet6) and IPv4 (inet) addresses, the MAC add given using the *ether* parameter.

fw0

The fireWire interface for your machine. This interface is disabled until you manually turn on as a networking option in the Network preference panel.

gif0

This is the generic interface tunnel which is used to tunnel network packets from one machin another, most often to connect two IPv6 networks over a link that uses the IPv4 protocol or t two IPv4 networks over a link that uses the IPv6 protocol. By default, this interface is not act *man gif* for more information.

stf0

The 6to4 tunnel interface is used to let IPv6 hosts on one network communicate transparentl IPv6 hosts on another network over an IPv4 link. By default, this interface is not enabled anc used when a machine is used as a router between an IPv6 and IPv4 network. See *man stf* fo information.

To get more detailed information about the current state of all the network adapters connected to y use the *ifconfig* command line tool, as shown in Example 13-1.

Example 13-1. Using ifconfig to look at the status of the network connect

```
$ifconfig

lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384

        inet6 ::1 prefixlen 128

        inet6 fe80::1 prefixlen 64 scopeid 0x1

        inet 127.0.0.1 netmask 0xff000000
```

```
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280

stf0: flags=0<> mtu 1280

en1: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500

        inet6 fe80::203:93ff:feef:baa5 prefixlen 64 scopeid 0x4

        inet 10.0.1.2 netmask 0xffffff00 broadcast 10.0.1.255

        ether 00:03:93:ef:ba:a5

        media: autoselect status: active

        supported media: autoselect

en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500

        ether 00:0a:95:99:e4:92

        media: autoselect (none) status: inactive

        supported media: none autoselect 10baseT/UTP <half-duplex> 10baseT/UTP <fu

10baseT/UTP <full-duplex,hw-loopback> 100baseTX <half-duplex> 100baseTX <fullduple

100baseTX <full-duplex,hw-loopback> 1000baseTX <full-duplex> 1000baseTX <fullduple

hw-loopback> 1000baseTX <full-duplex,flow-control> 1000baseTX <full-duplex,flowcon

hw-loopback>

fw0: flags=8822<BROADCAST,SMART,SIMPLEX,MULTICAST> mtu 4078

        tunnel inet -->

        lladdr 00:0a:95:ff:fe:99:e4:92

        media: autoselect <full-duplex> status: inactive

        supported media: autoselect <full-duplex>
```

The output from *ifconfig* lists the information for each interface using its identifier.

13.1.2.1 Looking at addresses on the local network

The system keeps a table of the MAC addresses of the various computers that are on the local netw segment and their IP addresses. To see this table, use the *arp -a* command as shown in Example 1

Example 13-2. Using arp to display the local IP addresses in use

```
$ arp -a

gateway.x180.net (192.168.1.1) at 0:c:41:4d:0:cc on en1 [ethernet]

dhcp-105.x180.net (192.168.1.105) at 0:3:93:50:e2:74 on en1 [ethernet]

? (192.168.1.255) at (incomplete) on en1 [ethernet]
```

There are three lines in *arp*'s output in Example 13-2. The first two are hosts whose IP addresses r
name and that have MAC addresses that allow them to receive packets. The last entry is a special
because 255 is the broadcast address on most networks and is primarily used to send packets to al
on a network. Since it is a network address and isn't associated with a host, there isn't a MAC addr
corresponds to it.

13.2 Configuring IP Addresses

When IP was first used, every computer on a network had to be configured with the correct IP address and netmask in order to be able to communicate with other machines. Fortunately, most networks provide Dynamic Host Configuration Protocol (DHCP) services. These services allow machines connected to the network to be configured automatically with an IP address and netmask for that network. DHCP also helps conserve IP addresses by letting a potentially large number of computers, that may only on the network infrequently, share a smaller number of active IP addresses. DHCP services can be provided by a separate server on the network or can be built into the hardware, known as a gateway router, providing access to the Internet via an ISP.

When you plug your computer into a DHCP-enabled network, your machine sends out a low-level request for configuration information. The DHCP server responds by sending an IP address, netmask, router, and DNS information that the computer needs to connect to the network. Your computer uses this information to configure itself and becomes a fully functional node on the network.

DHCP only works when the network has been configured properly. In the absence of a DHCP server, Mac OS X uses a link local address, which is part of Rendezvous. This is a private network address in the range 169.254.1.0 to 169.254.254.255. By assigning itself an address when no other address information is present, the system can interact with other hosts on the network that also have self-assigned addresses. This is perfect for setting up ad hoc networks—such as when you want to connect two laptops together with an Ethernet cable. This address won't work for accessing the Internet, but it works for connecting a group of computers together to share data, compile code, and the like.

13.2.1 Manually Configuring IP Information

If you need to manually configure your IP information so that you can access the Internet, the place to do so is in the Network preference panel, as shown in Figure 13-4. You'll need the following pieces of information from your network administrator or ISP:

- The IP address assigned to your computer

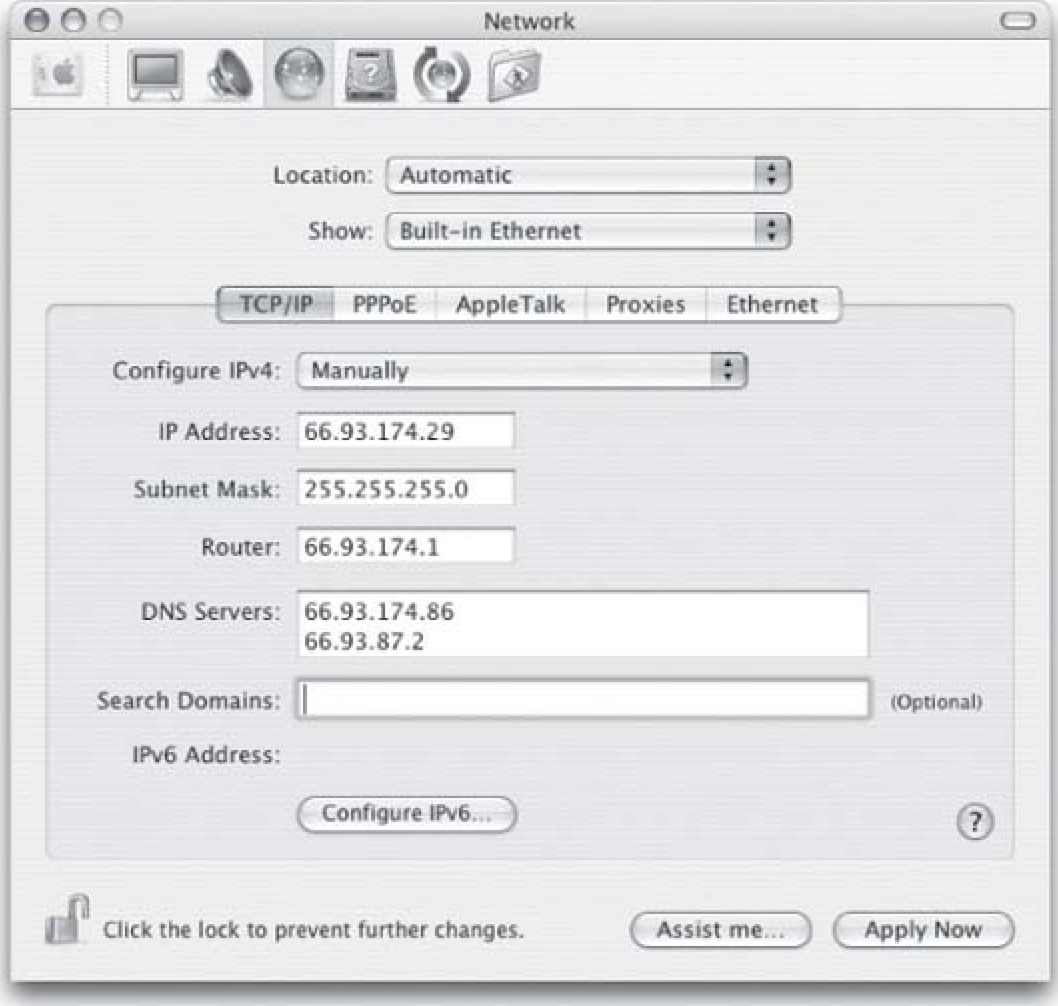
- The subnet mask for the address

- The router, also called a *gateway* by many admins, that your computer should use to communicate with the Internet

- The DNS servers your computer should use

While it is possible to configure your IP information using *ifconfig*, it is not recommended. Settings made with *ifconfig* are not reflected in the Network preference panel and will be overridden by any changes made in the GUI. If you want to use a command-line tool, you should take a look at *ncutil* by Jeff Frey. You can find it online at *deaddog.duch.udel.edu/~frey/darwin/ncutil.php*.

Figure 13-4. Configuring IP using the Network preference panel



Location: Automatic

Show: Built-in Ethernet

Configure IPv4: Manually

IP Address: 66.93.174.29

Subnet Mask: 255.255.255.0

Router: 66.93.174.1

DNS Servers: 66.93.174.86
66.93.87.2

Search Domains: (Optional)

IPv6 Address:

Click the lock to prevent further changes.

13.3 Naming and DNS

IP addresses, especially IPv6 addresses, aren't something that you ever really want to deal with directly; hostnames are much more convenient to use. However, every operation that involves a hostname has to involve looking up an IP address for that hostname. For example, when you point your web browser to *www.runningosx.com*, the system has to translate that hostname into an IP address. It does this by using the Domain Name System (DNS), a distributed naming system used to resolve a hostname to an IP address.

For all intents and purposes, DNS is fairly transparent. All you need to have is a DNS server defined in your network configuration and you can use hostnames in your applications. Without it, all you'll be able to do is surf the Net by IP address, which is not a very fun task at all.

If your computer obtained its IP address through DHCP, it should also be configured with the correct DNS server. However, if you have to configure your IP address manually, you'll need to provide valid values for a DNS server. If you have to configure your own DNS servers, make sure you use a DNS server that is close to you on the network. After all, every connection to a host on the Internet requires the resolution of hostname to IP address. The closer you are to the server, the better.

The DNS servers that your machine is currently using are shown in the Network preference panel. You can also find them in the */etc/resolv.conf* file, as shown in Example 13-3.

Example 13-3. Examining the contents of the resolv.conf file

```
$cat /etc/resolv.conf

domain x180.net

nameserver 66.93.174.29

nameserver 66.93.87.2

nameserver 216.231.41.2
```

On other Unix systems, you can directly edit the *resolv.conf* file to change your nameserver configuration. However, this file is automatically updated by the networking system in Mac OS X so any changes you make to it will be lost the next time you change networks.

13.3.1 Looking up DNS Information

To look up an IP address for a hostname yourself, or to find the hostname associated with an IP address, you can use the Lookup tab of Network Utility, as shown in Figure 13-5. Simply enter the host or IP address that you want to look up and click the Lookup button. Other options are available to you through the information pop-up menu. However, for most purposes, the default information setting should give you all the information you need.

On the command line, you can use the *host* command to determine the IP address for a host or vice versa, as shown in Example 13-4.

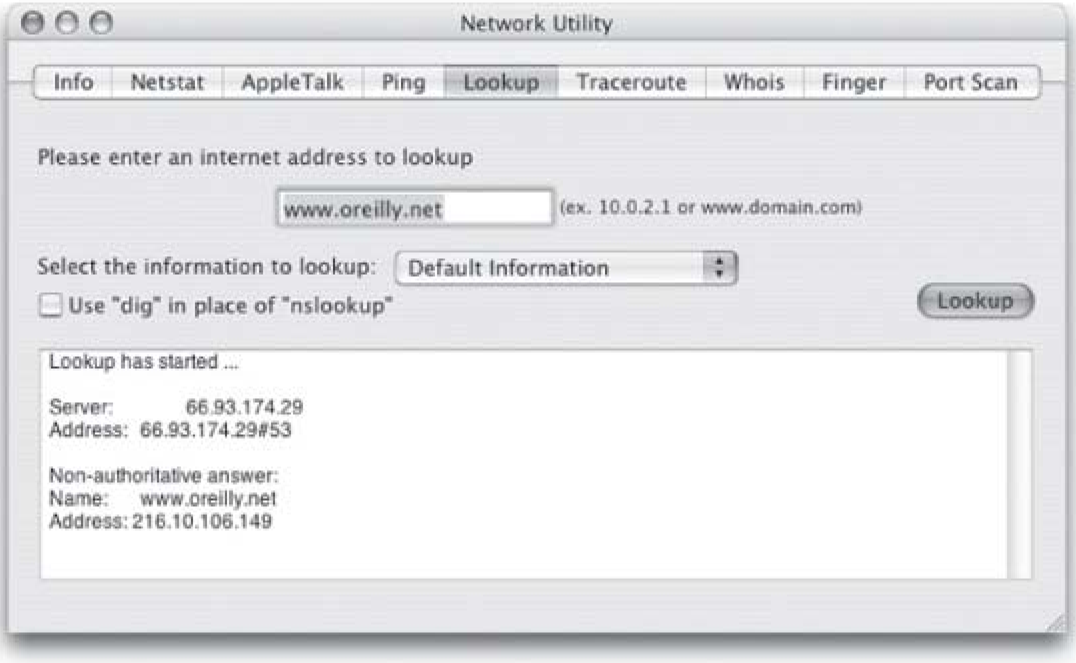
Example 13-4. Using host to look up DNS information

```
$host oreillynetwork.com

oreillynetwork.com has address 208.201.239.36

$host 208.201.239.36

36.239.201.208.in-addr.arpa domain name pointer www.oreillynet.com.
```

13.3.2 Multicast DNS

Even without a DNS server configured, there is another component of Rendezvous known as *multicast DNS* (mDNS). mDNS lets computers on a local network know each others' names so that you can access a computer using a human readable name instead of an IP address. For example, for a machine named Incognita, you should be able to browse a web server running on it from another machine on the local network by entering into Safari *Incognita.local*.

Figure 13-5. Using Network Utility to lookup host information



Please enter an internet address to lookup

www.oreilly.net (ex. 10.0.2.1 or www.domain.com)

Select the information to lookup: Default Information

□ Use "dig" in place of "nslookup"

```
Lookup has started ...

Server:         66.93.174.29
Address:  66.93.174.29#53

Non-authoritative answer:
Name:     www.oreilly.net
Address: 216.10.106.149
```

> By default when you install Mac OS X, the name of your machine will be based on the full name given when setting up the first account. For example, the default name of my machine after installing Panther was "James Duncan Davidson's Computer". You should change this to be something a bit shorter and more personalized (throughout this book you'll see Incognita and Quicksilver, the names of two of my machines) in the Sharing preference panel.

The combination of multicast DNS and self-assigned IP addresses provides the foundations for Rendezvous to work its magic. With multicast DNS, Rendezvous-enabled computers not only let each other know their names, but they can also advertise the services they offer. When you enable iTunes music sharing or iChat, a description of that service is broadcast via multicast DNS allowing any other Rendezvous machine to see it.

13.3.3 Ports and Services

The various services provided by a host, such as HTTP for serving web pages or SMTP for handling email, are each exposed to other computers on a separate port—a network connection endpoint in the IP stack identified by a number. For example, HTTP is defined to run on port 80. This means whenever you browse to a page on the server, your web client is opening a connection to port 80 of the server to make its request.

Most services use ports numbered less than 1024; these are known as the *well-known ports*. Table 13-1 lists some common services and their ports. You can also find an extensive listing of well-known services in the */etc/services* file.

Table 13-1. Some well known services and their ports

Service	Port(s)	Description
FTP	21, (990 secure)	file Transfer Protocol
SSH	22	Secure Shell Protocol for remote login
Telnet	23	Unsecure protocol for remote login
SMTP	25	Simple Mail Transfer Protocol for mail handling
DNS	53	Domain Name Service
HTTP	80, (443 secure)	Hypertext Transfer Protocol
POP3	110, (995 secure)	Post Office Protocol for mail handling
IMAP	143, (993 secure)	Internet Mail Access Protocol
LDAP	389, (636 secure)	Lightweight Directory Access Protocol
AFP	548	Apple Filing Protocol
IPP	631	Internet Printing Protocol (CUPS)

13.4 Private Networks and NAT

Because IPv4 addresses are relatively scarce, each address typically costs money to use. You may not pay directly for an IP address, but your ISP has a limited supply of them and will use some mechanism to control their use. To allow multiple machines to run behind a limited supply of IP addresses, many networks use a block of private addresses as well as Network Address Translation (NAT).

Supported by most routers, NAT processes all packets bound for the Internet from a private network, transforming the original IP addresses into an address that is reachable from the Internet, before sending the packets on. In effect, packets from the private network appear to be coming from the router running NAT. When packets arrive back at the router that corresponds to a connection already in progress, the NAT translates them into a form usable on the private network and sends them back to the original host.

You can tell if you are on a private network when your IP address is in one of the following ranges:

- 10.0.0.0 through 10.255.255.255
- 169.254.1.0 through 169.254.254.255
- 172.16.0.0 through 172.31.255.255
- 192.168.0.0 through 192.168.255.255

Using a NAT creates two follow-on effects:

- Since a NAT only relays packets from the Internet that were requested by a computer on the private network, a NAT acts as a basic firewall concealing the machines on the private network.
- With no way to directly address a machine on a private network, it's harder to set up a machine on the private network to act as a server. Some NATs will let you forward specific ports from their external IP address to various machines on the private network.

You'll need to consult the information about your NAT to be sure.

13.5 Routing

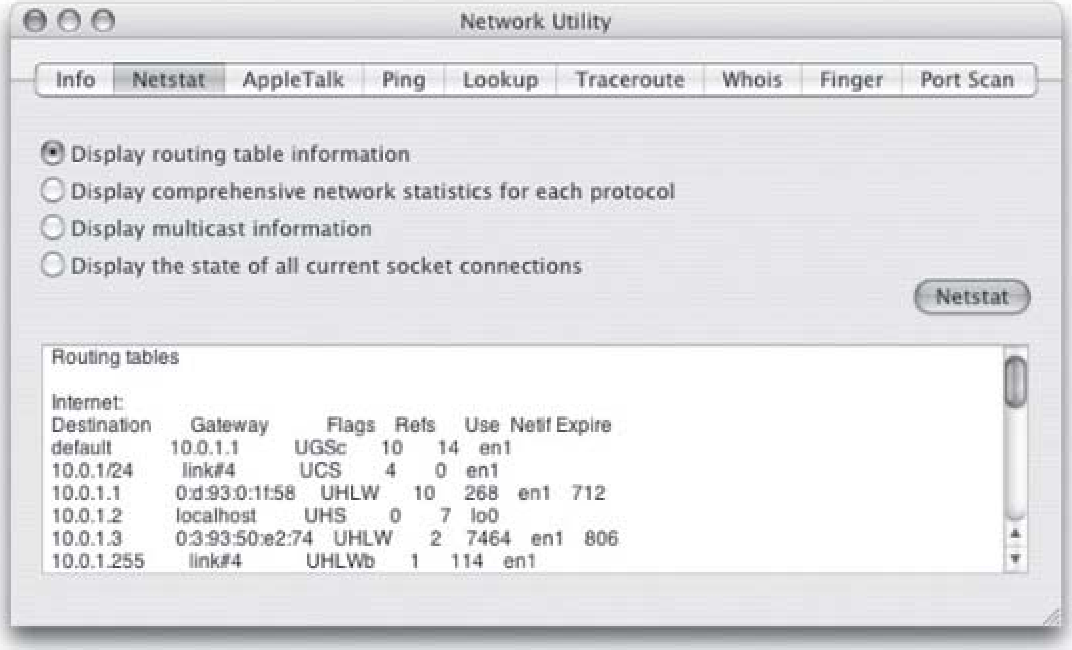
To see the routing table used by the system, you can use the Netstat tab of the Network Utility application, click the Display routing table information radio button, and then hit the Netstat button, as shown in Figure 13-6. This will output the same routing table information as the *netstat -r* command on the command line, shown in Example 13-5.

Example 13-5. The routing table as displayed by netstat -r

```
$netstat -r
Routing tables

Internet:
Destination        Gateway            Flags    Refs      Use    Netif Expire
default            10.0.1.1           UGSc       10       15    en1
10.0.1/24          link#4             UCS         3        0    en1
10.0.1.1           0:d:93:0:1f:58     UHLW       10      205    en1   1049
10.0.1.2           localhost          UHS         0        0    lo0
10.0.1.3           0:3:93:50:e2:74    UHLW        3     6059    en1   1032
10.0.1.255         link#4             UHLWb       1       97    en1
127                localhost          UCS         0        0    lo0
localhost          localhost          UH        392   479493    lo0
169.254            link#4             UCS         0        0    en1

Internet6:
Destination        Gateway            Flags      Netif Expire
localhost          localhost          UH          lo0
fe80::%lo0         fe80::1%lo0        Uc          lo0
fe80::1%lo0        link#1             UHL         lo0
fe80::%en1         link#4             UC          en1
```

```
fe80::203:93ff:fe5 0:3:93:50:e2:74   UHLW        en1
fe80::203:93ff:fee 0:3:93:ef:ba:a5   UHL         lo0
fe80::%fw0         link#6            UC          fw0
ff01::             localhost         U           lo0
```

Figure 13-6. Using Network Utility to examine routing information



● Display routing table information
○ Display comprehensive network statistics for each protocol
○ Display multicast information
○ Display the state of all current socket connections

```
Routing tables

Internet:
Destination     Gateway         Flags   Refs    Use  Netif Expire
default         10.0.1.1        UGSc     10      14   en1
10.0.1/24       link#4          UCS       4       0   en1
10.0.1.1        0:d:93:0:1f:58  UHLW     10     268   en1   712
10.0.1.2        localhost       UHS       0       7   lo0
10.0.1.3        0:3:93:50:e2:74 UHLW      2    7464   en1   806
10.0.1.255      link#4          UHLWb     1     114   en1
```

There are two parts to the routing table: the first is the Internet table for routing packets to IPv4-based networks; the second is the *Internet6* table for routing packets to IPv6-based networks. Each part contains a set of entries. Here's what the first few entries in the Internet table mean:

- The first line indicates that the default destination for all packets is 10.0.1.1, a router to the Internet. The packets should be sent via the `en1` interface.
- The second line indicates that packets for the 10.0.1 network should use the `en1` interface.
- The third line indicates that packets to 10.0.1.1 (the address of the router) should be sent to the device with the MAC address 0:d:93:0:1f:58 using the `en1` interface.
- The fourth line indicates that packets to 10.0.1.2 (in this case, the local machine) should be sent to the `localhost` on the `lo0` interface. The system puts this route into place so that any packets for the local machine that use the external IP address aren't sent to the network.

The*Internet6* table, shown in Example 13-5, contains the same sort of information but with IPv6 instead of IPv4 addresses.

The most useful entry in these tables is the `default` route. When you are having problems connecting to the network, you can quickly check to see if the system has a `default` route or not.Example 13-6 shows the output of *netstat -r* when all the network connections on a machine are down. As you can see, the only machine that packets can get to is `localhost`.

Example 13-6. The routing table with no external routes

```
$netstat -r
Routing tables

Internet:
Destination        Gateway           Flags    Refs      Use  Netif Expire
127                localhost         UCS         0        0    lo0
localhost          localhost         UH        536   499827    lo0
224.0.0/4          localhost         UCS         1        0    lo0
224.0.0.251        localhost         UHW         2       25    lo0

Internet6:
Destination        Gateway           Flags      Netif Expire
localhost          localhost         UH          lo0
fe80::%lo0         fe80::1%lo0       Uc          lo0
fe80::1%lo0        link#1            UHL         lo0
fe80::%en1         link#4            UC          en1
fe80::203:93ff:fe5 0:3:93:50:e2:74   UHLW        en1
fe80::%fw0         link#6            UC          fw0
ff01::             localhost         U           lo0
ff02::%lo0         localhost         UC          lo0
ff02::%en1         link#4            UC          en1
ff02::%fw0         link#6            UC          fw0
```

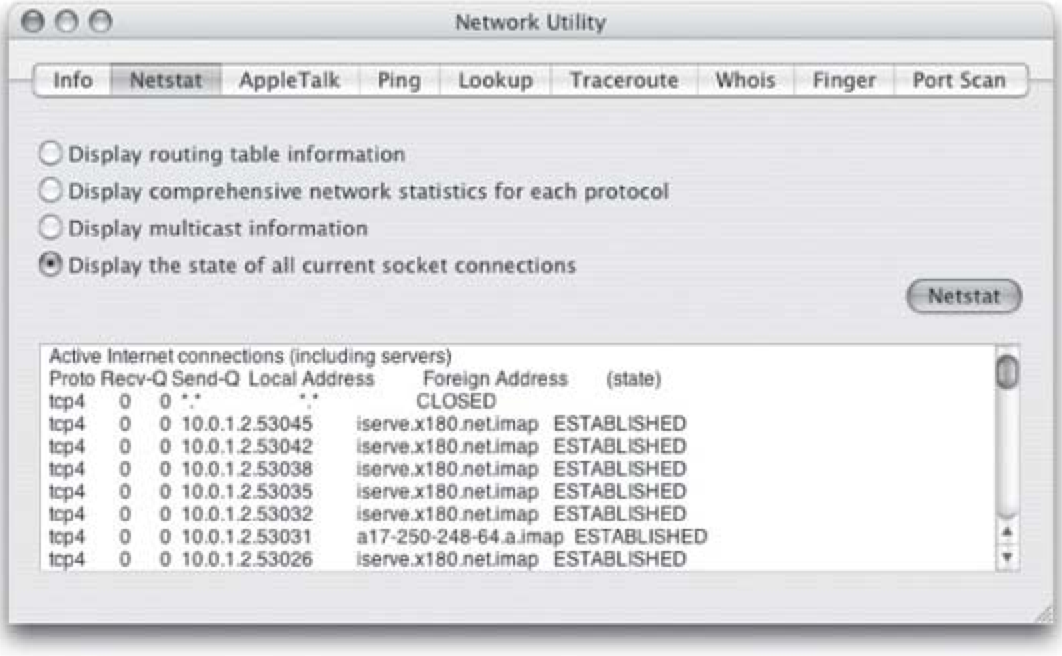
13.6 Active Network Connections

To dig a bit deeper, you can take a look at all the active network connections on your machine. To do this, use the same Netstat tab in Network Utility, click the radio button next to "Display the state of all current socket connections," and the hit the Netstat button. It usually takes a few seconds to generate the result, which is shown in Figure 13-7. This displays the same information as typing *netstat -a* into a Terminal window, shown in Example 13-7.

Example 13-7. Some output from netstat -a

```
$netstat -a

Active Internet connections (including servers)

Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)

tcp4       0      0  10.0.1.2.54007         swscan.apple.com.http  ESTABLISHED

tcp4       0      0  10.0.1.2.54135         dsl081-178-038.s.http  ESTABLISHED

tcp4       0      0  localhost.53195        localhost.ipp          CLOSE_WAIT

tcp4       0      0  *.*                    *.*                    CLOSED

tcp4       0      0  10.0.1.2.53045         iserve.x180.net.imap   ESTABLISHED

tcp4       0      0  10.0.1.2.53042         iserve.x180.net.imap   ESTABLISHED

tcp4       0      0  10.0.1.2.53031         a17-250-248-64.a.imap  ESTABLISHED

tcp4       0      0  10.0.1.2.53025         a17-250-248-64.a.imap  ESTABLISHED

tcp4       0      0  10.0.1.2.53020         mail.speakeasy.n.imap  ESTABLISHED
```

Figure 13-7. Using Network Utility to examing network connections



○ Display routing table information
○ Display comprehensive network statistics for each protocol
○ Display multicast information
● Display the state of all current socket connections

```
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address         Foreign Address        (state)
tcp4     0      0  *.*                   *.*                    CLOSED
tcp4     0      0  10.0.1.2.53045        iserve.x180.net.imap   ESTABLISHED
tcp4     0      0  10.0.1.2.53042        iserve.x180.net.imap   ESTABLISHED
tcp4     0      0  10.0.1.2.53038        iserve.x180.net.imap   ESTABLISHED
tcp4     0      0  10.0.1.2.53035        iserve.x180.net.imap   ESTABLISHED
tcp4     0      0  10.0.1.2.53032        iserve.x180.net.imap   ESTABLISHED
tcp4     0      0  10.0.1.2.53031        a17-250-248-64.a.imap  ESTABLISHED
tcp4     0      0  10.0.1.2.53026        iserve.x180.net.imap   ESTABLISHED
```

Each item in the list shows the local and remote addresses and ports in the form *host.port* where the port name is translated, if known, into a service name such as *http* or *imap*. In Example 13-7, the first connection listed is from the local machine to a server at apple.com. The connection is using the HTTP protocol.

13.7 AppleTalk

When the Mac was developed in the early 1980s, an easy-to-use networking protocol called AppleTalk was developed so that multiple users could share files and resources like printers. After its release, AppleTalk went through several changes including the ability to route AppleTalk packets between networks (called *zones* in AppleTalk parlance) and run AppleTalk over Ethernet. Because AppleTalk has been around since the introduction of the Mac, large numbers of networks still use AppleTalk and many AppleTalk-only devices still exist.

Mac OS X continues to support AppleTalk even though IP is its primary networking protocol. When you have an Ethernet or AirPort connection, AppleTalk is automatically running by default so any available AppleTalk devices on the network can be used. You can control whether or not AppleTalk is active for a network connection by using the Network preference panel.

From the command line, you can check on the status of AppleTalk on your machine by running the*appletalk -s* command, as shown in Example 13-8.

Example 13-8. Getting the status of AppleTalk

```
$appletalk -s

       AppleTalk interface.............. en1
       Network Number .................. 65420 (0xff8c)
       Node ID ......................... 122 (0x7a)
       Current Zone .................... *
       Bridge net ...................... 0 (0x0)
       Bridge number ................... 0 (0x0)

   DDP statistics:

       Packets Transmitted ............. 64
       Bytes Transmitted ............... 2751
       Best Router Cache used (pkts) ... 0
       Packets Received ................ 79
       Bytes Received .................. 3644
```

```
        Packets for unregistered socket . 0
        Packets for out of range socket . 2
        Length errors ................... 0
        Checksum errors ................. 0
        Packets dropped (no buffers) .... 0
```

The output from *appletalk -s* shows the network interface AppleTalk is active on, the numeric network and node identifiers, and some statistics about the packets being transmitted via AppleTalk.

You can see the AppleTalk hosts and services on the network by using the *atlookup* command, as shown in Example 13-9.

Example 13-9. Looking up AppleTalk hosts and services using atlookup

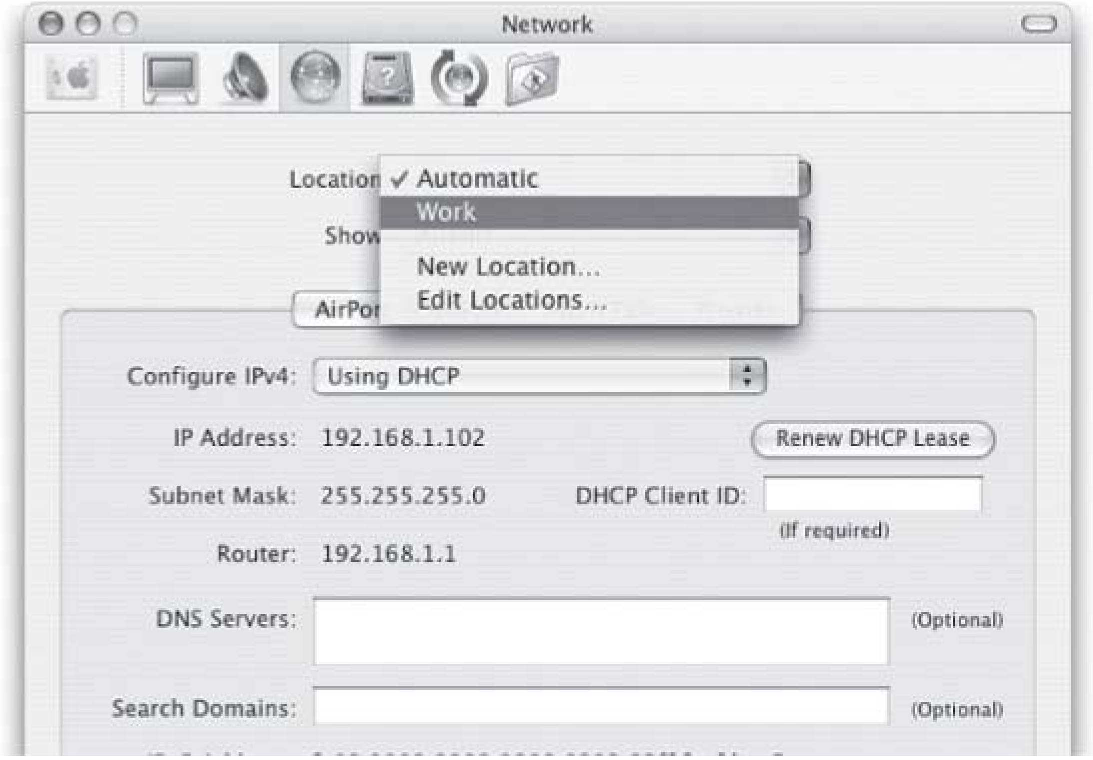
```
 $ atlookup
Found 3 entries in zone *
ff8c.7a.80      Incognita:Darwin
ffee.44.80      Quicksilver:Darwin
ffee.44.81      Quicksilver:AFPServer
ff01.04.08      PET:SNMP Agent
ff01.04.9d      PET:LaserWriter
ff01.04.9e      PET:LaserJet 2200
```

The first column in the output is composed of three dot-separated hexadecimal numbers. The first two numbers identify the host's network number and ID. The third number identifies the service being provided. The second column is composed of the name of the host as well as the name of the service being provided. In Example 13-9, the first two lines indicate that there are two Darwin-based (Mac OS X) machines on the network named Incognita and Quicksilver. The third line indicates that an Apple filing Protocol (AFP) server is running on Quicksilver which means that you can connect to filesystems shared by that machine.

13.8 Locations

You can use the Network preference panel to create and manage several independent profiles of network configuration information, known as *Locations*. This is handy when you have to manually configure your network settings for the office but want to use your computer easily on DHCP-based networks at home or at the local coffee house. Locations are managed using the Location menu in the Network preference panel, as shown in Figure 13-8.

Figure 13-8. Using the Location pull down menu in the Network preference panel



Configure IPv4: Using DHCP

IP Address: 192.168.1.102

Subnet Mask: 255.255.255.0

DHCP Client ID:

(If required)

Router: 192.168.1.1

DNS Servers: (Optional)

Search Domains: (Optional)

By default, all Mac OS X systems have a location named *Automatic*. To create a new location, select New Location from the Location menu. After naming the location, it appears in the pop-up menu. When you choose a location, you will be able to configure it as needed. When you hit the Apply Now button at the lower-right area of the Network preference panel, your changes are applied and your Mac attempts to connect to that network location.

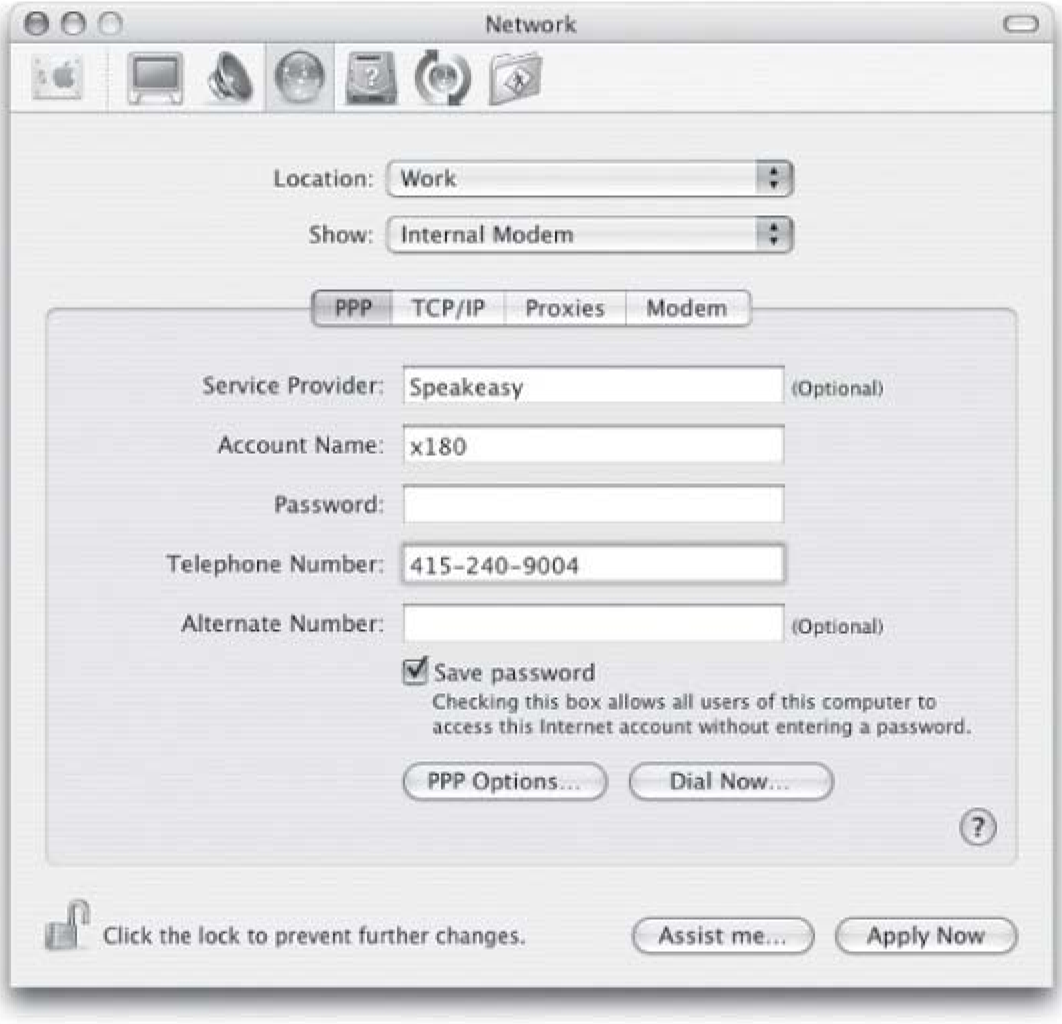
Once you have defined multiple locations, you can switch between them without going into the Network preference panel using the →Location menu. This menu also gives you a quick shortcut to the Network preference panel.

If all the networks you move your Mac between use DHCP, you can probably get away with not setting different locations and just using the Automatic location.

13.9 Using a Dial-Up Connection

Compared to using Ethernet or AirPort, connecting to a network through a modem requires the additional step of having your computer dial into a server. This can be done either on-demand when your computer wants to access the network, or only when you wish to. In order to use a dial-up network, you'll have to set up some additional information in the PPP tab of the Network preference panel for a modem, as shown in Figure 13-9. For most dial-up networks, using the information given to you by your ISP will work here.

Figure 13-9. Configuring dial up networking using the Networking preference panel



Location: Work

Show: Internal Modem

Service Provider: Speakeasy (Optional)

Account Name: x180

Password:

Telephone Number: 415-240-9004

Alternate Number: (Optional)

☑ Save password

Checking this box allows all users of this computer to access this Internet account without entering a password.

If you need to set up additional configuration items , you can click the PPP Options button and get the sheet displayed in Figure 13-10. Three options of interest here:

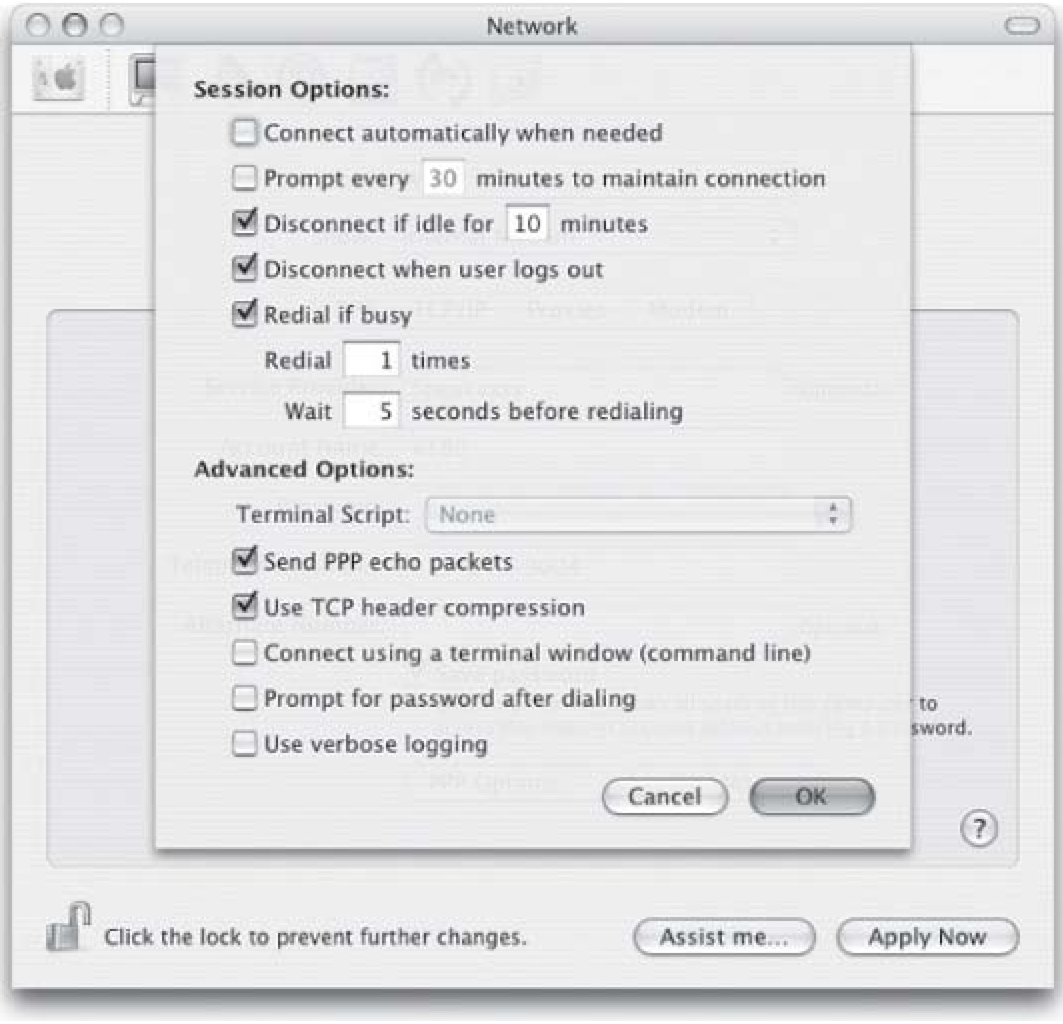
Connect automatically when needed

If this box is checked, your computer will try to dial your network provider when any program makes a request that needs connectivity. For example, if you were offline but opened Safari, the system would automatically connect so that you could use the application. This will also happen with the Backup utility included with .Mac memberships. If this box is checked and you have Backup scheduled to run a backup to your iDisk and your Mac isn't online, your Mac will attempt to connect to your ISP so Backup can get its job done.

Disconnect if idle

This setting controls how long your connection will be idle, that is how long the connection will be held if you aren't sending or receiving any packets.

Figure 13-10. Advanced PPP options



Session Options:

☐ Connect automatically when needed

☐ Prompt every 30 minutes to maintain connection

☑ Disconnect if idle for 10 minutes

☑ Disconnect when user logs out

☑ Redial if busy

Redial 1 times

Wait 5 seconds before redialing

Advanced Options:

Terminal Script: None

☑ Send PPP echo packets

☑ Use TCP header compression

☐ Connect using a terminal window (command line)

☐ Prompt for password after dialing

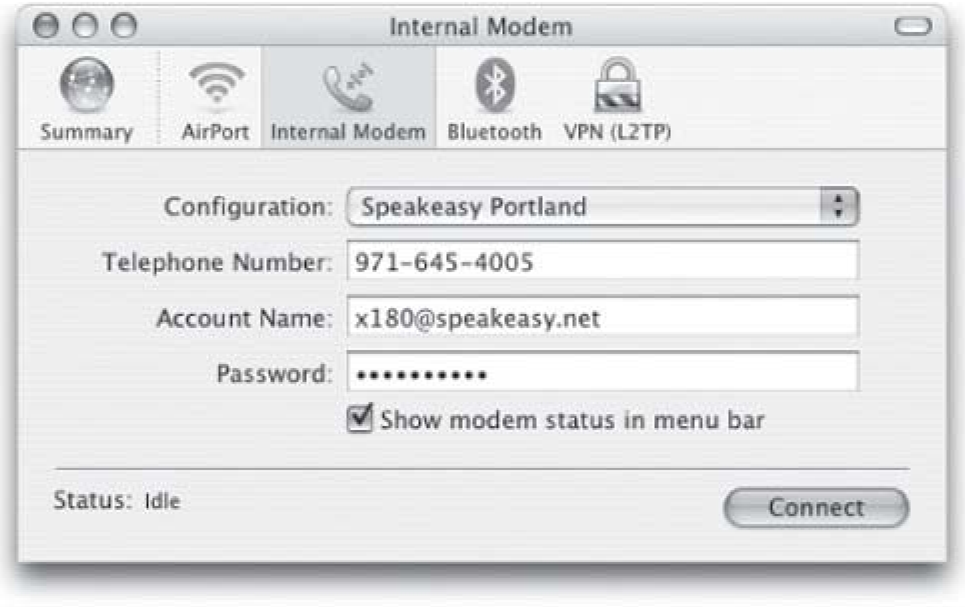
☐ Use verbose logging

Connect using a terminal window (command line)

> If checked, this causes a simple terminal window (not related to the Terminal application) to open so you can manually dial in to the PPP server. Usually this isn't required as most PPP servers use a standard login process that your Mac can deal with seamlessly. However, some older networks may require the use of this option.

In addition to using the Network preference panel, you can use Internet Connect (*/Applications*), shown in Figure 13-11, to connect to a dial-up networking provider. In addition, Internet Connect allows you to manage multiple numbers, which is handy if you are travelling between cities and want to easily connect to the right dial-up number. To add a configuration, use the Configuration pop-up menu and select Edit Configurations.

Figure 13-11. Using Internet Connect to manage dial up accounts



Configuration: Speakeasy Portland

Telephone Number: 971-645-4005

Account Name: x180@speakeasy.net

Password: •••••••••••

☑ Show modem status in menu bar

Status: Idle

Observant readers will note that Internet Connect also sports a few other options in its toolbar. One is AirPort, which provides you details on any wireless networks within range, including a signal strength meter. Another is Bluetooth, which allows you to connect to the network via a Bluetooth device such as a cell phone. Unfortunately, connecting via Bluetooth devices is still a bit of a mysterious art, with wildly different procedures depending on your device. Your best bet is to dig on the Internet for information on how to configure your device.

The last item is a VPN tab, allowing you to access a Virtual Private Network, which I'll discuss next.

13.10 Virtual Private Networks

In a nutshell, a VPN is an encrypted logical network connection, also known as a *tunnel*, that runs over a physical connection such as the open Internet. When you establish a VPN connection with a server, all the network packets between your computer and the server are encrypted and remain safe from prying eyes. This means you can access resources on another network, such as your corporate network, from anywhere in the world without compromising your corporate network's security.

Mac OS X supports two types of VPNs:

Point-to-Point Tunneling Protocol (PPTP)

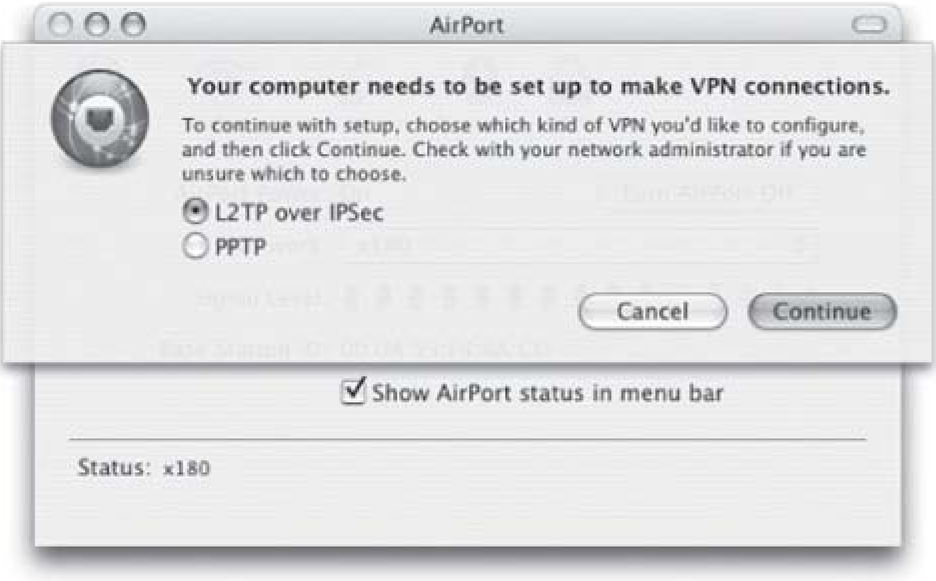
> A VPN standard developed by Microsoft and supported by many manufacturers of networking equipment. All you need to connect via PPTP is the address of the server, an account name, and a password or an RSA Secure ID card (a gizmo that displays a random number and is synchronized with a similar gizmo on the server side).

Level 2 Tunneling Protocol (L2TP)

> L2TP is a newer VPN standard that utilizes IPSec, a standard for encrypting data over an IP connection. To make an L2TP connection, you'll need the address of the server, an account name, a password or an RSA SecurID card, and a shared secret. The *shared secret* is a key that the sysadmin of the L2TP server gives you along with the account name and password. It is used in the initial set up of the tunnel.

To make a VPN connection, open Internet Connect, then click the VPN button. The first time you configure a VPN, you'll have to make a choice as to whether you want to configure your computer to use L2TP or PPTP. Unfortunately, Mac OS X can only support one kind or another, so you'll need to make sure to select the right kind for your connection, as shown in Figure 13-12. Once you've selected the type of connection, you'll see the Internet Connect VPN window, shown inFigure 13-13. This window works the same way the dial-up window does, in that you can configure Internet Connect so you can select from one of many different VPN configurations.

Figure 13-12. Configuring the type of VPN to use


AirPort

Your computer needs to be set up to make VPN connections.

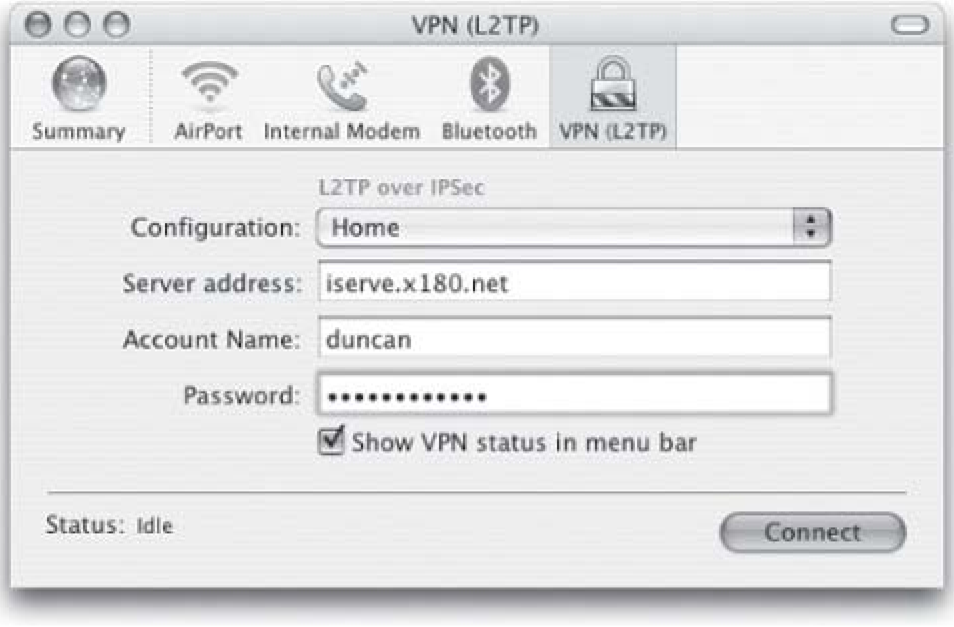
To continue with setup, choose which kind of VPN you'd like to configure, and then click Continue. Check with your network administrator if you are unsure which to choose.

● L2TP over IPSec
○ PPTP

☑ Show AirPort status in menu bar

Status: x180

Figure 13-13. Managing VPN connections with Internet Connect



L2TP over IPSec

Configuration: Home

Server address: iserve.x180.net

Account Name: duncan

Password: •••••••••••••

☑ Show VPN status in menu bar

Status: Idle

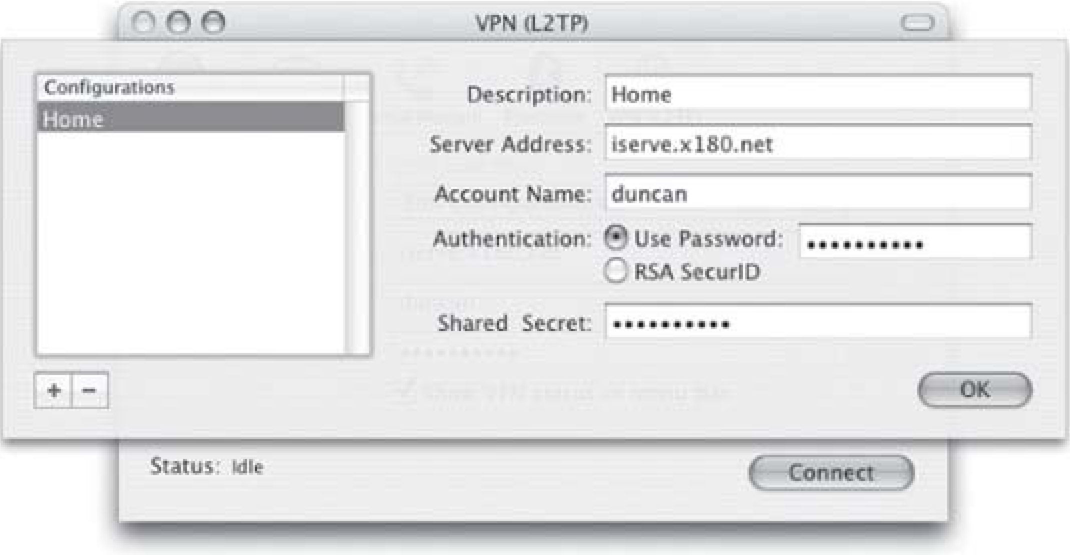
In order to enter in your shared secret for a L2TP connection, you'll need to edit the configuration, as shown in Figure 13-14.

13.11 Firewalls

Unlike many other operating systems, Mac OS X ships in a secure state with all network services disabled. This means you can be fairly certain that no matter what network you find yourself on, the likelihood of somebody cracking into your machine is very low. However, as you turn on various services, such as file or web sharing, the ports used to support those services on your computer are opened up, which means they can receive data from the network. For the most part, Apple does a good job releasing security updates, making sure that these services are patched as soon as vulnerabilities are discovered.

If you are truly paranoid and want take every step possible to control access to your Mac, you can enable Mac OS X's built-in firewall, based on *ipfw*, which performs packet filtering at the kernel level. You can turn on the firewall by using System Preferences→Sharing→firewall, as shown in Figure 13-15.

Figure 13-14. Managing VPN configurations with Internet Connect



Home

Description: Home

Server Address: iserve.x180.net

Account Name: duncan

Authentication: ● Use Password: •••••••••••
○ RSA SecurID

Shared Secret: •••••••••••

Status: Idle

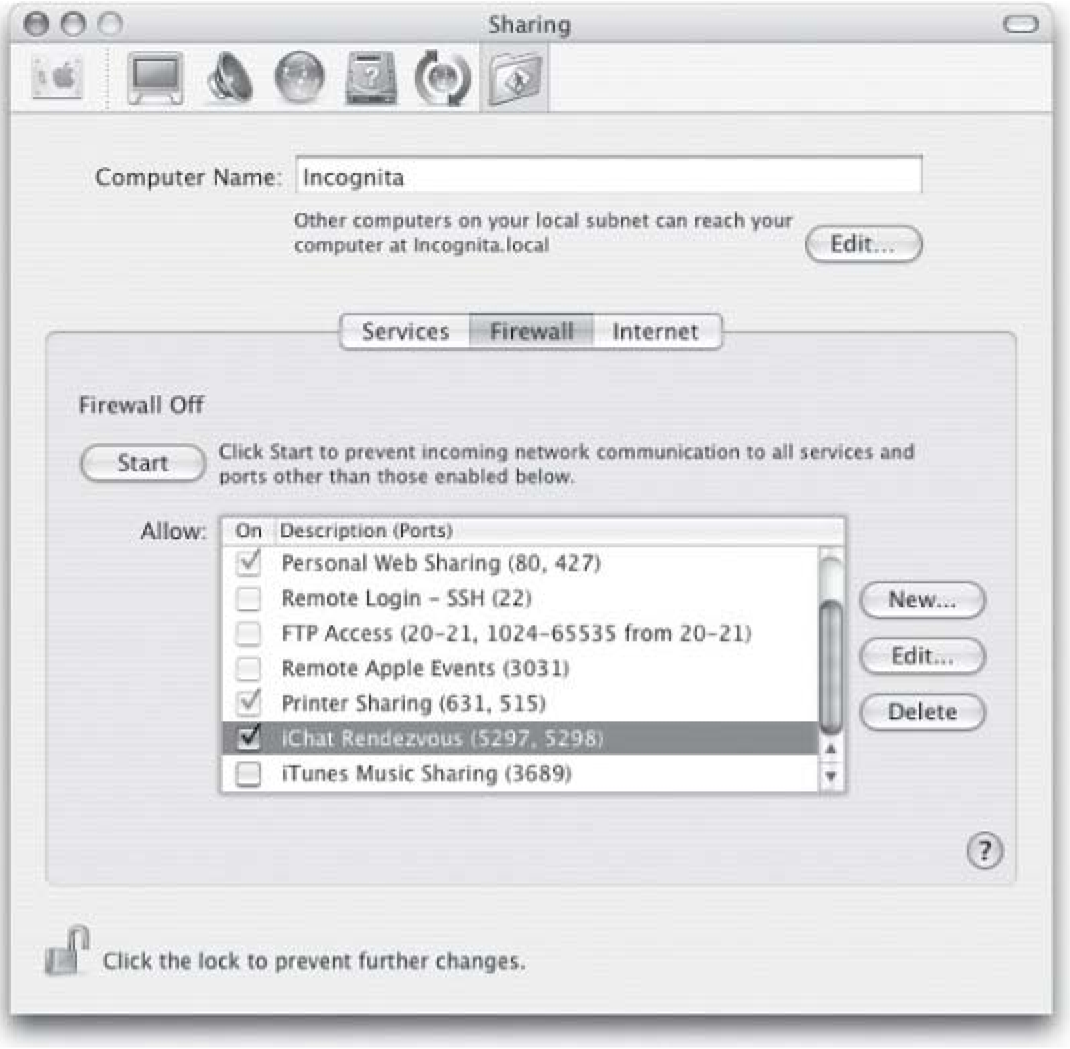
When you enable the firewall, only packets that correspond to the rules that you set up in the Allow list are allowed into your machine. All other packets are dropped. The default rules are set up so that any services that you share are allowed. However, other ports, such as those needed to use iChat over Rendezvous, are closed by default. To allow these ports, you can select the corresponding rule for the service you want to allow.

To open ports for services not listed, you'll need to create rules. Click the New button in the firewall pane. This brings up the sheet shown in Figure 13-16 and allows you to add a service. Several default services are listed in the pull-down menu where you can select and add. However, only a limited number of services that you might want to enable are listed. For example, the ports used by iChat during a voice or video chat aren't listed in the default rules or in the list of rules to add. You'll have to add the ports yourself using the Other option in the pull-down menu.

Information about which ports to open for iChat AV can be found in Apple Knowledge Base article 93208 (docs.info.apple.com/article.html?artnum=93208). A list of the well-known ports used by Apple software is contained in Apple Knowledge Base article 106439 (docs.info.apple.com/article.html?artnum=106439).

Since the firewall is based on *ipfw*, it is possible to manipulate the firewall and its rules from the command line. However, doing so is dangerous. It is easy to craft rules that, look secure, yet can make things worse than they were to begin with. This will give you a false sense of security. It is also easy to lock yourself out of your computer when editing rules remotely or even to put your computer in an unusable state by tweaking the wrong rule. The bottom line is that even though you can go into the depths of firewall configuration using *ipfw*, I strongly urge you not to. It's an area where it is way too easy to do more harm than good.

Figure 13-15. The Firewall configuration panel



Computer Name: Incognita

Other computers on your local subnet can reach your computer at Incognita.local

Firewall Off

Click Start to prevent incoming network communication to all services and ports other than those enabled below.

Allow:

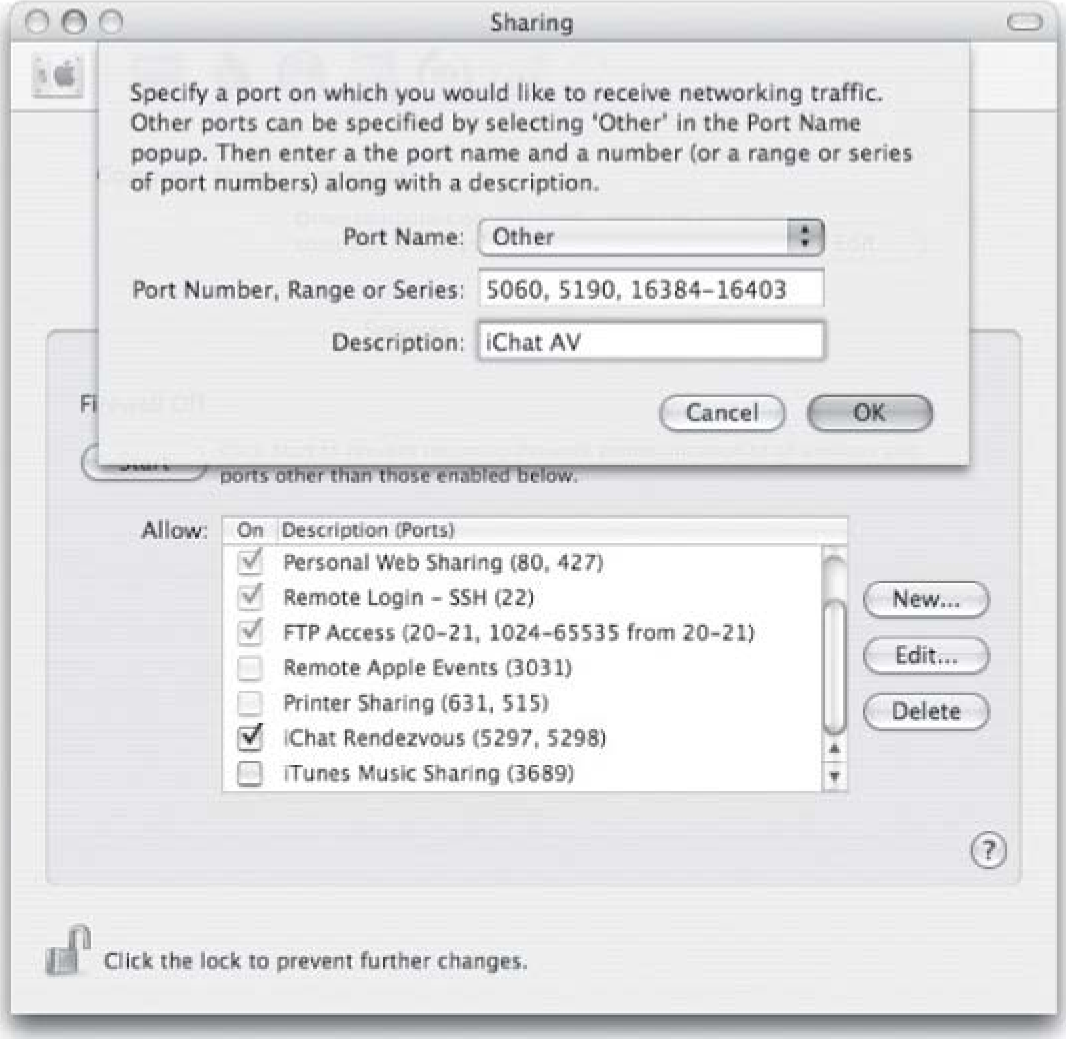
On	Description (Ports)
☑	Personal Web Sharing (80, 427)
□	Remote Login - SSH (22)
□	FTP Access (20-21, 1024-65535 from 20-21)
□	Remote Apple Events (3031)
☑	Printer Sharing (631, 515)
☑	iChat Rendezvous (5297, 5298)
□	iTunes Music Sharing (3689)

Click the lock to prevent further changes.

If you need more flexibility in your firewall than the GUI gives you, you should be using an external firewall. In fact, if you are worried about the security of your system enough to turn on the built-in firewall, you really should be using an external firewall. It is much more effective to have network security performed in a dedicated external device than it is to configure a piece of software on the machine which, if compromised, can give access to the internals of the machine. This is as true of third-party, add-on firewall products as it is of *ipfw*.

When you want to secure your machine in a network environment that you don't control, such as in a cafe, you should turn off all the services that your machine has on in the Sharing preference pane, make sure that iChat's Rendezvous mode is turned off, and make sure that you aren't sharing your iTunes music library. By turning off these services, you've done more to secure your machine than any firewall can do.

Figure 13-16. Adding a rule to the firewall



Specify a port on which you would like to receive networking traffic. Other ports can be specified by selecting 'Other' in the Port Name popup. Then enter a the port name and a number (or a range or series of port numbers) along with a description.

Port Name: Other

Port Number, Range or Series: 5060, 5190, 16384-16403

Description: iChat AV

ports other than those enabled below.

Allow:

On	Description (Ports)
☑	Personal Web Sharing (80, 427)
☑	Remote Login - SSH (22)
☑	FTP Access (20-21, 1024-65535 from 20-21)
□	Remote Apple Events (3031)
□	Printer Sharing (631, 515)
☑	iChat Rendezvous (5297, 5298)
□	iTunes Music Sharing (3689)

Click the lock to prevent further changes.

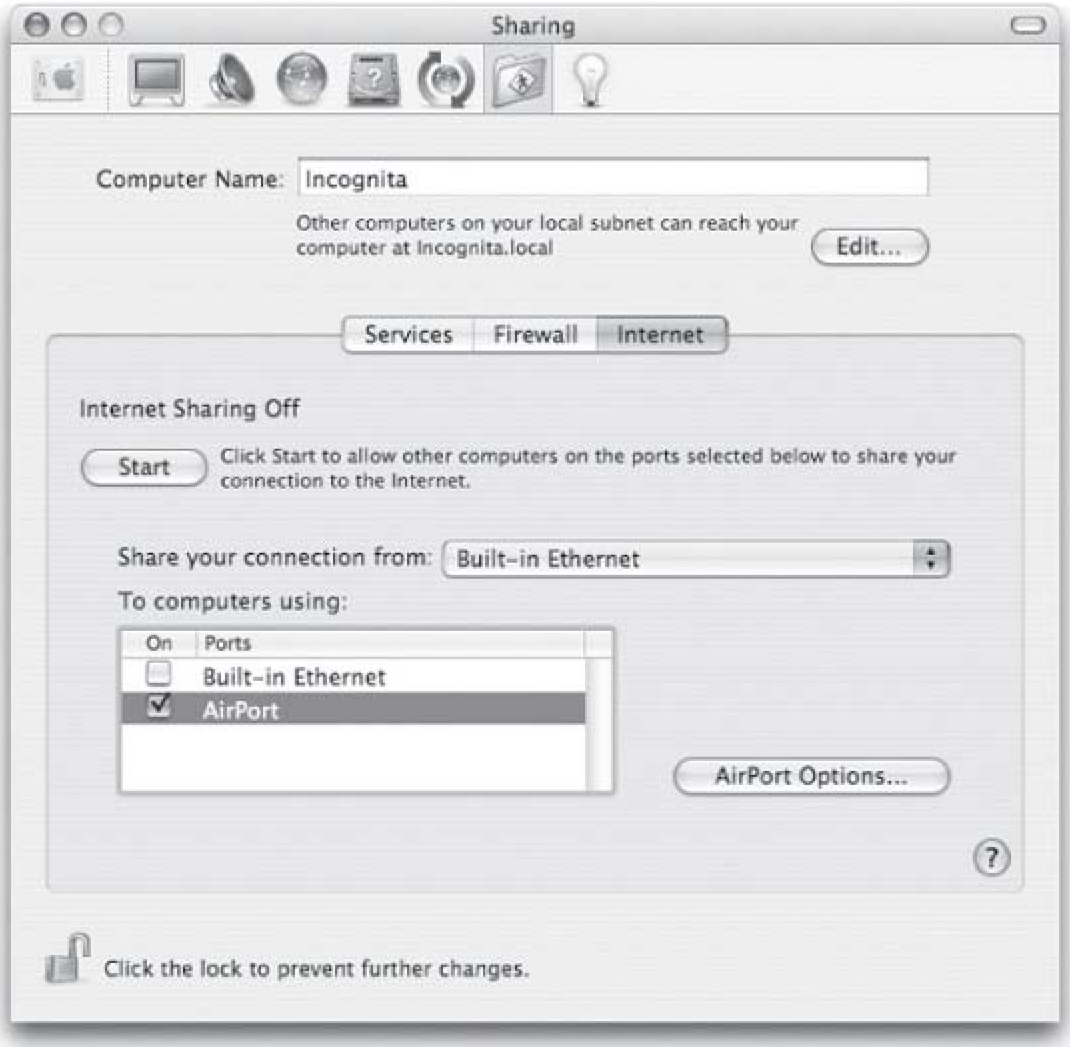
13.12 Internet Connection Sharing

Building on top of the various tools and concepts, such as DHCP and NAT, covered in this chapter, Mac OS X provides an easy-to-use mechanism for sharing one of its network connections to networks on any of its other network connections. For example, if you are in a meeting room and only have one Ethernet jack, but everyone has Wifi cards in their laptops, you can plug your Mac into the Ethernet and then share that connection with everyone else in the room over a private wireless network managed by your computer—regardless of whether they are running Mac OS X, Windows, or Linux. Essentially, you can turn your Mac into the equivalent of an Internet gateway.

> You shouldn't activate Internet Sharing on a network segment that already has an operational DHCP server. If you do, your Mac will interfere with the DHCP server on the network which will result in confusion among users and will draw the ire of the network administrator. In short, don't share to existing networks. Always make your own private network that to share your public network connection with.

To enable Internet Connection Sharing, simply use the Internet tab of the Sharing preference panel, shown in Figure 13-17. Use the pull-down menu to select which of your network connections you want to share—that is the connection that is connected to the Internet. Then click the checkboxes to select which network connections that you want your machine to act as a router for. If you want to share via AirPort, you'll want to click the AirPort Options button to configure the wireless network that will be set up. Once everything is configured to your liking, click the Start button and you'll be the star of the room.

Figure 13-17. Setting up Internet Connection Sharing



Computer Name: Incognita

Other computers on your local subnet can reach your computer at Incognita.local

Internet Sharing Off

Click Start to allow other computers on the ports selected below to share your connection to the Internet.

Share your connection from: Built-in Ethernet

To computers using:

On	Ports
☐	Built-in Ethernet
☑	AirPort

Click the lock to prevent further changes.

13.13 Further Explorations

If you would like to get a deeper knowledge of how networking works, you should investigate the following:

- *TCP/IP Network Administration, 3rd Edition*, by Craig Hunt (O'Reilly & Associates, Inc., 1992)
- *Virtual Private Networks, 2nd Edition*, by Charlie Scott, et al. (O'Reilly & Associates, Inc., 1998)
- *Building Internet Firewalls, 2nd Edition*, by Elizabeth D. Zwicky, et al. (O'Reilly & Associates, Inc., 2000)

Also, you can investigate the following manpages on your system:

- *ip*
- *ifconfig*
- *netstat*
- *appletalk*
- *ipfw*

Chapter 14. Network Services

Mac OS X comes well-equipped not only for using network services, but also for providing these services to other machines. With a click of a checkbox in the Sharing preference panel, you can share files, set up a web site, allow others to log in to your machine, provide an FTP server, and even share your printers. When activated, each of these services kicks into action a piece of software, known as a *server*, that responds to requests from other machines for access to data on your machine.

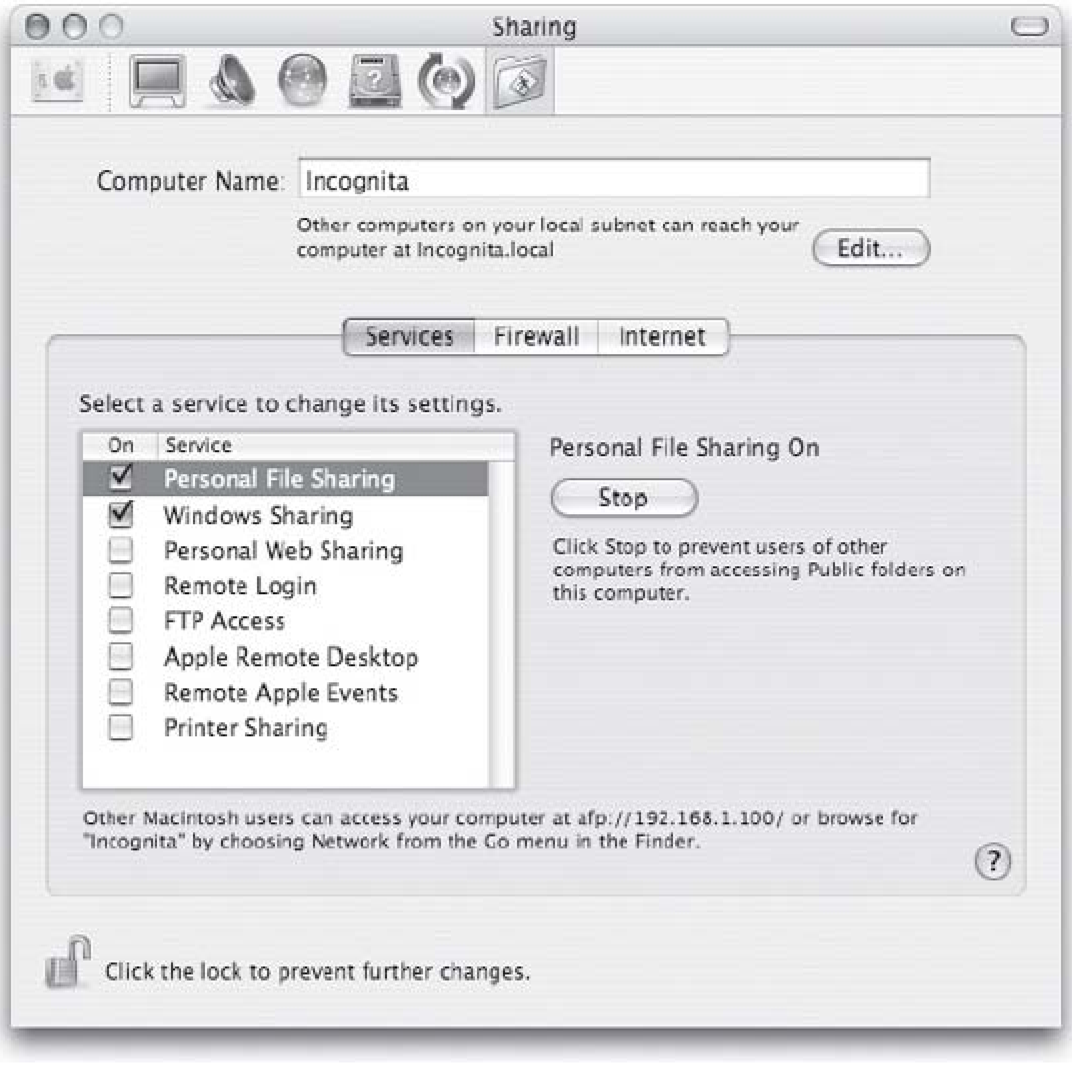
This chapter shows you what happens behind the scenes when you activate these services and how you can modify the way that they work to suit your own needs. Also, you'll see how to configure the mail server on your system, a service that's not exposed in the Sharing preference panel.

> A word of warning is called for whenever activating any service on your computer. By starting a service, any computer that has access to your Mac can potentially access your data. Server vulnerabilities, while usually not easy to exploit, can be and are discovered, and this means somebody could take control of your Mac from afar. Therefore, you should be careful to activate only those services that you need to have operational. You should also be aware of the number of computers that can connect to your machine. For example, if your machine is publicly accessible from the Internet, you probably don't want to enable Printer Sharing as anybody on the network could then print to your printer.

14.1 File Sharing

Two primary mechanisms can be used to share files from Mac OS X: the first is the native Apple Filing Protocol (AFP) that has been long used by the Mac. This is enabled when you click the Personal File Sharing checkbox in the Sharing preference panel. The second is the Service Message Block (SMB) protocol that is used to share your files with Windows machines. This is enabled when you click the Windows Sharing checkbox in the Sharing preference panel. Both settings are shown in Figure 14-1.

Figure 14.1. Enabling Personal File Sharing and Windows Sharing in the Sharing preference panel



Computer Name: Incognita

Other computers on your local subnet can reach your computer at Incognita.local

Select a service to change its settings.

On	Service
☑	Personal File Sharing
☑	Windows Sharing
□	Personal Web Sharing
□	Remote Login
□	FTP Access
□	Apple Remote Desktop
□	Remote Apple Events
□	Printer Sharing

Personal File Sharing On

Click Stop to prevent users of other computers from accessing Public folders on this computer.

Other Macintosh users can access your computer at afp://192.168.1.100/ or browse for "Incognita" by choosing Network from the Go menu in the Finder.

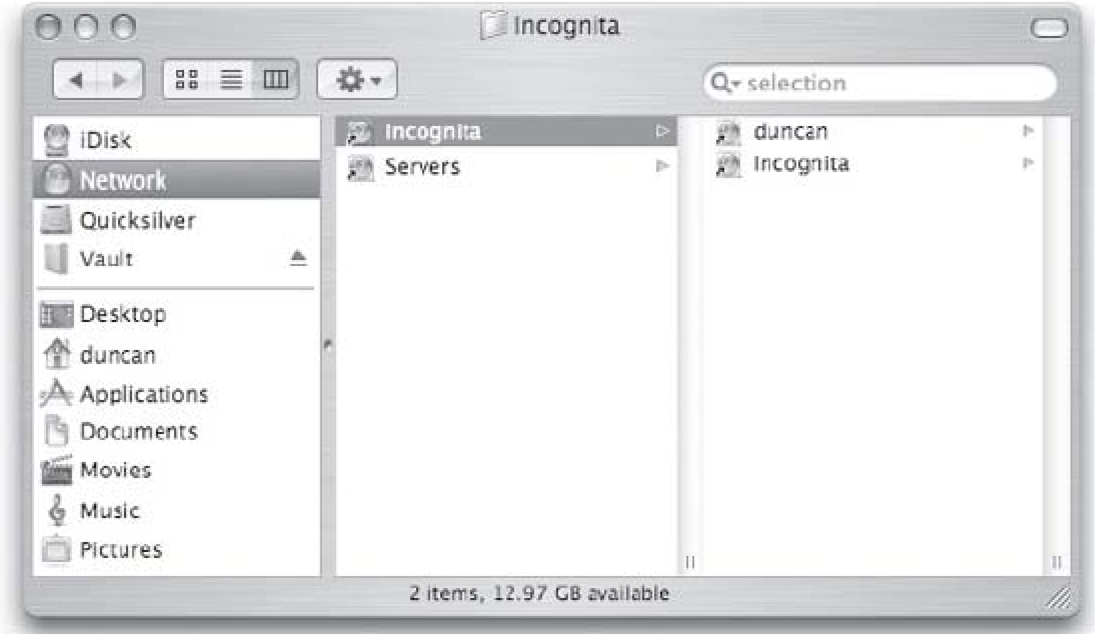
Click the lock to prevent further changes.

14.1.1 Sharing with Other Macs

When you enable Personal File Sharing in the Sharing preference panel, Mac OS X starts the *AppleFileServer* process and sets up Rendezvous so that other machines on the local network will be able to see that your machine will accept AFP requests. This will make your machine appear in the */Network* folder in the Finder view of the other Macs on your local network and allow them to connect to your machine, as shown in Figure 14-2.

The shares (locations in the filesystem that can be accessed by others) that your Mac exposes to other machines will vary depending on how users log in to your machine when they connect and will fall into one of the following scenarios:

Figure 14-2. Browsing AFP shares via the Network folder



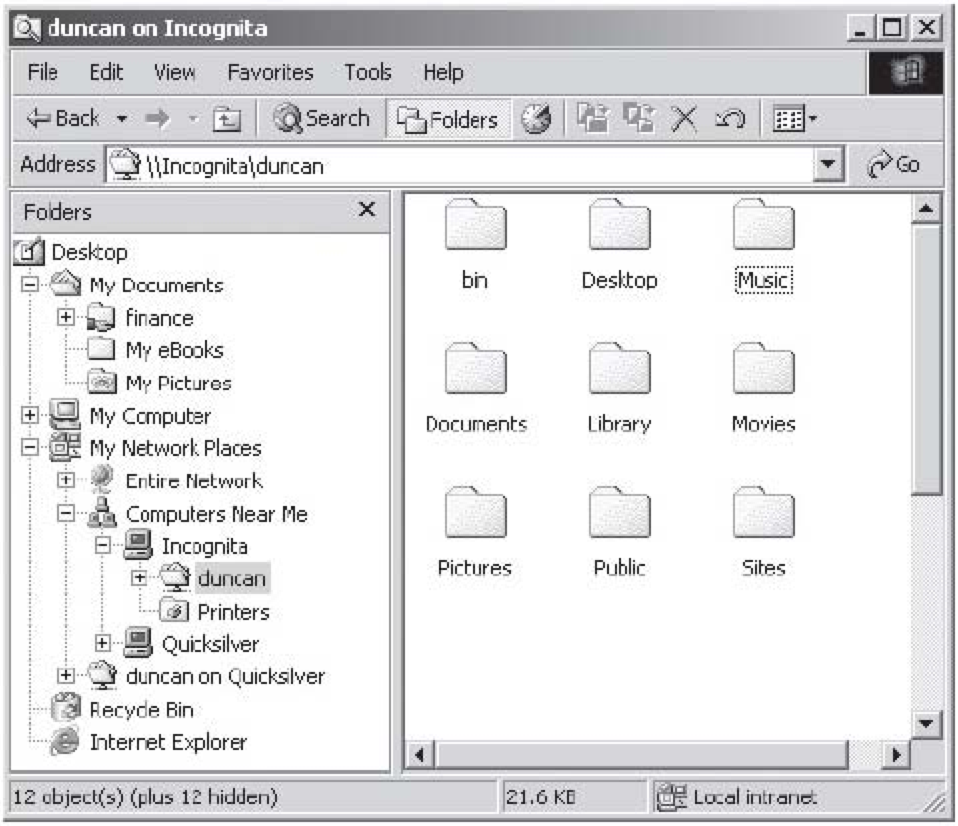
- If a user logs in to a machine as a guest, he will see a share for each user on the machine. Each user's share holds the contents of that user's *Public* folder. He won't be able to see any other files on the system.
- If a user logs in to a machine as a user with administrative privileges, he will be able to see a share for that user's Home folder as well as a share for each disk attached to the machine, including the boot disk.
- If a user logs in to a machine as a normal user (without administrative privileges), he will only see a share for that user's Home folder.

> In older versions of Mac OS X, the configuration for the AppleFileServer process was kept in NetInfo. With Panther, the configuration information is kept in */Library/Preferences/com.apple.AppleFileServer.plist*. As of the writing of this book, there is very little information, and no information from Apple, on the configuration keys that can be added to this file.

14.1.2 Sharing with Windows Machines

When you click the Windows Sharing checkbox in the Sharing preference panel, your Mac will appear in the Network Neighborhood view of Windows Explorer, as shown in Figure 14-3. To connect to a Mac from a Windows machine, simply double-click the machine in Explorer and then enter the name and password of a user on the Mac.

Figure 14-3. Browsing a Mac's shared folder from Windows Explorer



Address \\Incognita\duncan

12 object(s) (plus 12 hidden) 21.6 KB

By default, when you enable Windows Sharing, only the Home folders of the users on the system will be shared and made available to other machines. You can easily modify this by editing the Samba configuration file located at */etc/smb.conf* on your Mac. For more information on configuring Samba, see either *Using Samba, 2nd Edition*, by Jay Ts, et al. (O'Reilly & Associates, Inc., 2003) or *Samba Pocket Reference, 2nd Edition*, by Jay Ts, et al. (O'Reilly & Associates, Inc., 2003). You can also find a wealth of information on Samba, including documentation and sample configuration files, at www.samba.org/.

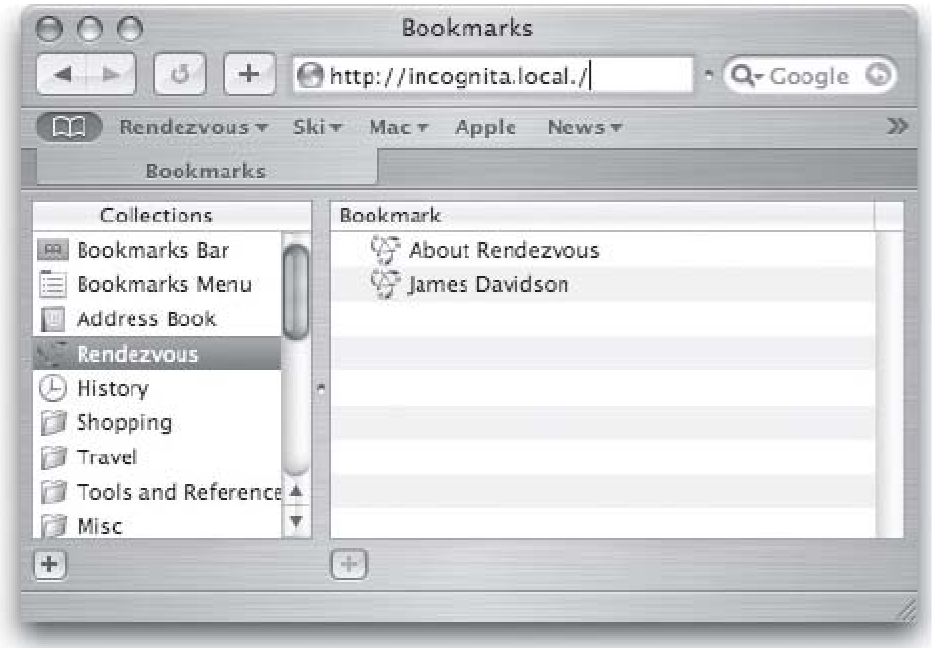
14.2 Web Sharing

The understated Personal Web Sharing checkbox in the Sharing preference panel enables the world-famous powerhouse Apache web server, the same web server that powers more web sites than any other. When you enable web sharing, the *httpd* web server daemon is started and will serve pages from the following locations:

- Pages in the */Library/WebServer/Documents* folder will be served when you make a request to your machine. For example, to browse the pages on the machine that you are serving from, you can point your web browser at *http://localhost*.
- Pages in each user's *Sites* directory will be served for requests to *http://localhost/* *~username*.

In addition, each user that has modified the pages in their *Sites* folder will have their site registered with Rendezvous so that it is visible to other Mac OS X users on the local network. You can see Rendezvous-advertised sites in Safari, as shown in Figure 14-4.

Figure 14-4. Browsing Rendezvous-Advertised sites in Safari



The Apache configuration file is located at */etc/httpd/httpd.conf*. For the most part, this is a standard Apache configuration file, with just a few changes made to adapt it to Mac OS X. Any of the standard Apache configuration directives will work in this file. To access the locally installed documentation on how to configure Apache, browse to *http://localhost/manual/* on your machine after turning on Personal Web Sharing. You can also find this documentation in the */Library/Documentation/Services/apache* directory.

The only item in the configuration file that isn't well documented is the last configuration block.

This enables Rendezvous advertisements of the web sites located on your Mac. This block is shown in Example 14-1.

Example 14-1. The Rendezvous configuration block in /etc/httpd/httpd.conf

```
<IfModule mod_rendezvous_apple.c>

    # Only the pages of users who have edited their

    # default home pages will be advertised on Rendezvous.

    RegisterUserSite customized-users

    #RegisterUserSite all-users



    # Rendezvous advertising for the primary site is off by default.

    #RegisterDefaultSite

</IfModule>
```

This block can contain the following directives:

RegisterUserSite

> This directive can be followed either by `customized-users` or `all-users`. When followed by`customized-users` (the default), only users that have edited their *~/Sites/index.html* file will have their site advertised through Rendezvous. When followed with `all-users`, every user's site on the system will be advertised, even if the *~/Sites/index.html* file hasn't been edited.

RegisterDefaultSite

> This directive causes a Rendezvous advertisement to be made for the server using your Mac's name. When enabled, users can easily browse the web content located in the */Library/WebServer/Documents* folder. This directive is disabled by default.

RegisterResource

> This directive, which doesn't appear by default in *httpd.conf*, lets you advertise a particular section of your site and must appear with the syntax `RegisterResource`*`resourcename path`*. For example, to advertise a Rendezvous link to the built-in Apache manual, you could add the following to your *httpd.conf*:`RegisterResource "Apache Manual" /manual`.

For more information on configuring Apache, see *Apache: The Definitive Guide, 3rd Edition*, by Ben Laurie, et al. (O'Reilly & Associates, Inc., 2002), or the *Apache Pocket Reference*, by Andrew Ford (O'Reilly & Associates, Inc., 2000).

> When adding configuration directives to Apache's configuration, consider creating a file in the */etc/httpd/users* directory that uses the *.conf* extension. Files in this directory that match the pattern **.conf* are automatically included by *httpd.conf*. By making your edits in subfiles you'll keep your configuration files neater as well as avoid potential conflicts from updated*httpd.conf* files that may be installed as part of future system updates.

14.3 Remote Login

When you enable the Remote Login checkbox in the Sharing preference panel, you are turning on the Secure Shell server (*sshd*). SSH is a protocol for using key-based encryption to allow secure communication between machines. To connect to a machine running *sshd*, simply use the following command:

```
ssh machinename
```

For example, to connect to a machine named *Quicksilver.local*, you would use the following:

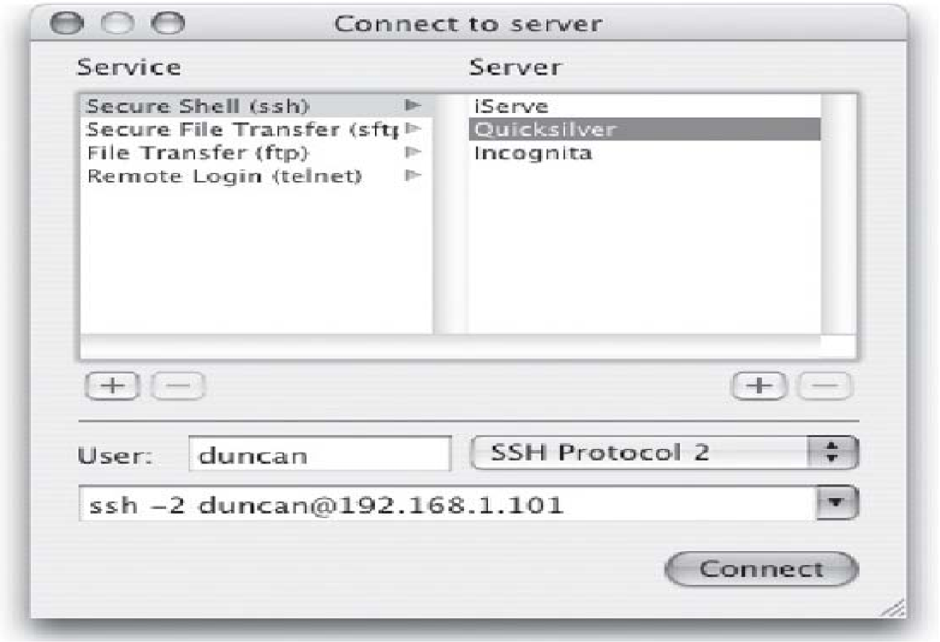
```
$ ssh Quicksilver.local
```

The*ssh* program will use the username you are logged in to the Terminal with to connect to the server. If you want to use a different username, prepend the machine name with `username@`. For example:

```
$ ssh norman@Quicksilver.local
```

You can also connect to *ssh* servers on your local network easily thanks to Rendezvous, by using the File→Connect To Server (Shift-⌘-K) menu in the Terminal. This brings up a dialog from which you can see the local machines you can connect to, as shown in Figure 14-5.

Figure 14-5. The Terminal's Connect to Server dialog



User: duncan

SSH Protocol 2

ssh -2 duncan@192.168.1.101

If you need to customize your installation of *sshd*, you can find its configuration file at */etc/sshd_config*. For more information about SSH, see *SSH*,*The Secure Shell: The Definitive Guide*, by Daniel J. Barrett, et al. (O'Reilly & Associates, Inc., 2001).

14.4 FTP Access

File Transfer Protocol (FTP), the old file-moving workhorse of the Internet before HTTP was invented, is the service that Apple should have left out. It's an insecure protocol that can compromise the security of your passwords by transmitting them in the clear. Luckily, FTP is disabled by default on Mac OS X, and you should leave it that way. However, if you want to turn it on, you can. When you do you can connect to your machine with any FTP client and you'll be able to access the contents of your entire system.

Instead of using FTP, you should enable SSH and use Secure FTP (SFTP). SFTP is supported by many modern FTP clients and is part of the SSH distribution as the *sftp* command-line tool. You don't need to do any other configuration than turn on the *ssh* server. You should also look into using*rcp* and *rsync*, two other file-copying tools that work well with *ssh*.

> One thing that *sftp, rcp*, and *rsync* don't provide that FTP does is the ability to provide anonymous access to your machine.

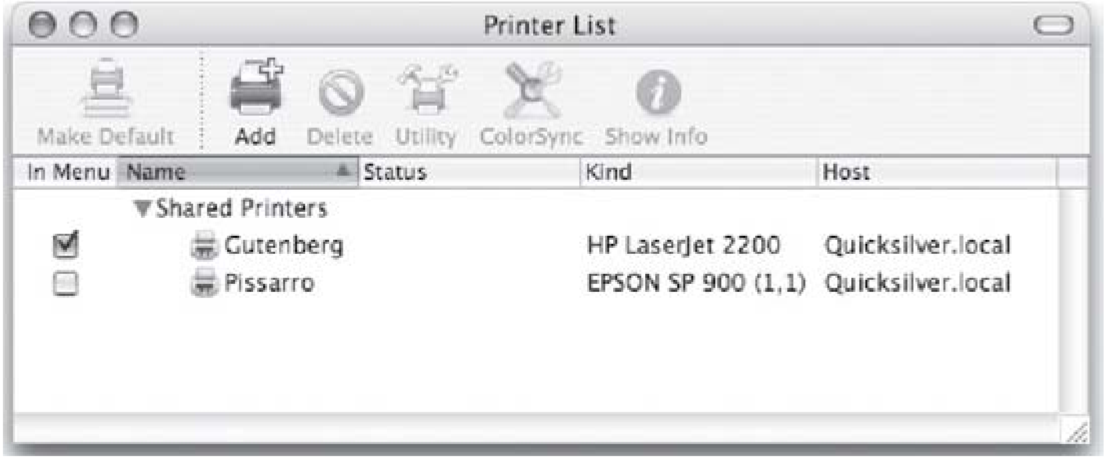
14.5 Sharing Printers

Since the print system is built using CUPS and SMB, any Mac-, Unix-, or Windowsbased system can use the printers attached to your Mac. To activate this feature, simply enable the Printer Sharing checkbox in the Sharing preference panel. This automatically enables both remote CUPS-based printer sharing via the IPP and LP protocols and sets up Samba to announce your printers to Windows-based machines.

14.5.1 Connecting to a Shared Printer on Mac OS X

When you enable print sharing, your printers will show up on other Macs in the shared printers section of the printer list in the Printer Setup Utility, as shown in Figure 14-6, as well as the list of printers in the print dialog box. All the print driver settings as configured on the machine that host the printers will be propagated to any machine using the printer. This means any user with a printer will be able to take full advantage of any special features the printer has, such as duplex printing.

Figure 14-6. Examining shared printers on the network in Printer Setup Utility

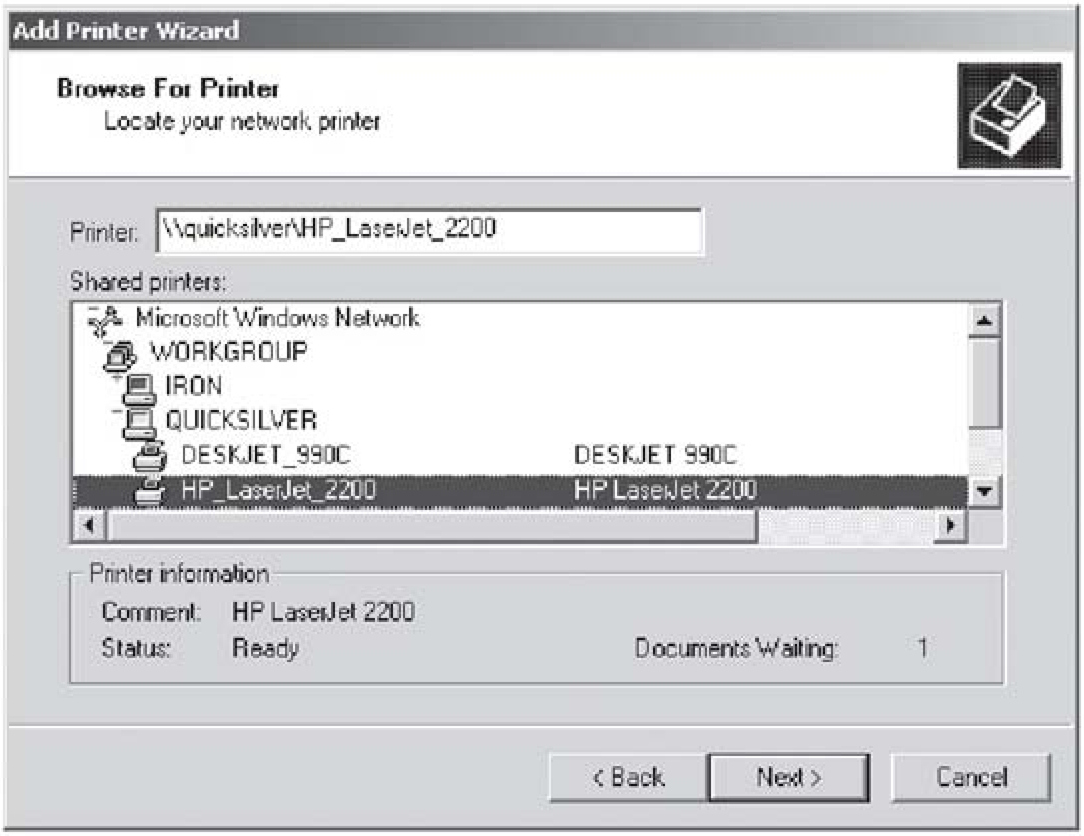


In Menu	Name	Status	Kind	Host
	▼Shared Printers			
☑	Gutenberg		HP LaserJet 2200	Quicksilver.local
□	Pissarro		EPSON SP 900 (1,1)	Quicksilver.local

14.5.2 Connecting to a Shared Printer Using Windows

Printers shared by a Mac will show up in the Network Neighborhood in Windows as well as when you browse for printers in the Add Printer Wizard. As an example, Figure 14-7 shows the printers shared throughout this chapter in the Add Printer Wizard. Unlike when other Macs connect to a shared printer, a Windows machine can't reuse the driver and settings. Instead, you'll have to install the printer driver on the Windows machine. Usually the driver comes on a disk with the printer, but you'll be better off grabbing the latest version of the Windows printer driver from the manufacturer's web site.

Figure 14-7. Connecting to a Mac based printer from Windows



Add Printer Wizard

Browse For Printer

Locate your network printer

Printer: \\quicksilver\HP_LaserJet_2200

Shared printers:

```
Microsoft Windows Network
  WORKGROUP
    IRON
    QUICKSILVER
      DESKJET_990C              DESKJET 990C
      HP_LaserJet_2200          HP LaserJet 2200
```

Printer information

Comment: HP LaserJet 2200

Status: Ready   Documents Waiting: 1

14.6 Mail

Even though it doesn't have a setting in the Sharing preference pane, one more server is on Mac OS X that many people find useful is the Postfix mail server. You can set up Postfix to handle mail duties for your machine in two ways:

- As a local mailer, which lets you send mail from your machine (including from the command line).
- As a first-rate SMTP server that can accept mail from the world at large as well as let you send messages.

The first, as a local mailer, is extremely useful, if for no other reason than being able to send yourself messages from *cron*. The second involves a bit more work and is full of responsibility. Running a first-class mail server on the Internet today, with all the problems with viruses and spam, is no small undertaking.

14.6.1 Enabling Local Mail

Your system is configured, as much as it can be, by Apple to be able to send mail from the command line. By default, the */etc/hostconfig* file contains the following line:

```
MAILSERVER=-AUTOMATIC-
```

This line causes the Postfix installation on your Mac to run a program called *postfixwatch*. This program watches the local mail queue. When a message is added to the queue, for example by using the *mail* command, it will cause the rest of the Postfix system to launch and deliver mail. However, unless you are using an IP address that resolves to a legal hostname in DNS, other mail servers on the Internet will reject mail from your machine.

In order to correct this, you'll need to modify the following settings in the */etc/postfix/main.cf* file by uncommenting the following:

`#myhostname`

> Set this parameter to the name that you want *postfix* to use. *postfix* uses this setting to identify itself to other mail servers. This name should be, but does not need to be, resolvable to an IP address as long as the domain part of the hostname is resolvable.

`#mydomain`

> Set this parameter to the domain name that you want *postfix* to use. The domain name you use here must exist and must be resolvable to an IP address. It should be set to a domain that you own and can receive mail for.

`#myorigin`

> Set this parameter to the domain that you want your email sent from. The domain name you use here must exist and must be resolvable to an IP address. This will result in email sent from the command line being addressed as from *user@domain.com*. You can leave this unset if you've set `myhostname` to a name that resolves to a valid IP address.

Example 14-2 shows the changes made to my */etc/postfix/main.cf* file in order to send mail. You'll want to replace *domain.com* with your own domain.

Example 14-2. The settings that need to be edited in /etc/postfix/main.cf

```
myhostname = powerbook.domain.com

mydomain = domain.com

myorigin = domain.com
```

> Once you've sent mail, causing *postfix-watch* to fire up the rest of Postfix, you'll need to instruct Postfix to reload its configuration. Do this by entering the following at the command line: `sudo postfix reload`

Once you make these changes, you'll be able to send email from the command line using the *mail* command. When you use the *mail* command, you finish a message by entering a period on a line by itself. Example 14-3 shows a sample mail session.

Example 14-3. Sending mail from the command line

```
$mail duncan@runningosx.com

Subject:Testing



This is a test. This is only a test. If this was an actual message....

.

EOT
```

If you need to troubleshoot, Postfix logs messages to the */var/log/mail.log* file.

14.6.2 Enabling the SMTP Server

Once you've made the configuration changes to send mail from your machine, it's a short hop to setting it up to be a SMTP server accessible from other machines. Before you do so, you should consider whether or not you want the responsibility of operating a mail server on your network or even on the public Internet. If you misconfigure your server, you can create what's known as an open relay, which allows spammers to use your machine to bombard millions of people with junk mail. This will ultimately land you on the black list of many other mail server administrators. Most people are better off having their ISP handle mail for them.

That said, if you decide to press on, here's how to do it:

- Edit the */etc/hostconfig* file and change the `MAILSERVER` line to read `MAILSERVER=-YES-`.
- Edit the */etc/postfix/master.cf* file and uncomment the line that begins with `#smtp.` This part of the file is shown in Example 14-4.

Example 14-4. Part of the /etc/postfix/master.cf file

```
# ==========================================================================
# service type  private unpriv chroot wakeup  maxproc command + args
#               (yes)   (yes)  (yes)  (never) (100)
# ==========================================================================
smtp      inet  n       -      n      -       -       smtpd
```

After making the edits, restart the Postfix system with the following command:

```
$sudo postfix reload
```

After you execute this command, Postfix will be listening to port 25 to accept email. Once you have the mail server up and running, you'll want to finish configuring it for your needs. Refer to *Postfix: The Definitive Guide*, by Kyle D. Dent (O'Reilly & Associates, Inc., 2003) or see the Postfix web site at *www.postfix.org/*for more information.

14.6.3 Mail Delivery Agents

Postfix delivers email to your local machine into the */var/mail* directory. There will be a separate file for each user that receives mail on the server. In order to retrieve messages from the mail server, you'll need either a Post Office Protocol (POP) or an Internet Message Access Protocol (IMAP) server. Unfortunately, Mac OS X doesn't ship with either a POP or an IMAP server.

You can, however, get both of these servers from the UW IMAP distribution, available from *www.washington.edu/imap*. If you would like to obtain a prebuilt version of UW IMAP, there is a shareware utility called Postfix Enabler, which allows you to enable Postfix. It also contains a copy of both the UW IMAP and POP servers. You can download Postfix Enabler from *www.roadstead.com/weblog/Tutorials/PostfixEnabler.html*.

Part IV: Appendixes

This section of the book contains quick reference material:

Appendix A, *Installing from Scratch*

Appendix B, *Boot Command Keys*

Appendix C, *Other Sources of Information*

Appendix A. Installing From Scratch

If you're lucky, you'll never need to know the information in this Appendix. If, however, you're always tweaking your system, sooner or later you're going to either want or need to reinstall Mac OS X Panther on your computer. This appendix provides you the details you need to install Panther on your Mac the way *you* want it and gives you tips for things to think about along the way—including the all-important step of backing up your data.

A.1 Preparing to Install Panther

While Apple's installer for Mac OS X is pretty straightforward, there are some things you need to think about (quite seriously) before you gut your system and install Panther. If you have a new Mac that's fresh out of the box, your slate's pretty clean. However, if you have a Mac that you've been using for some time, you need to think about backing up your existing system, and more importantly, how you want to reconfigure your system for running Panther in the future.

A.1.1 Backing Up Your Life

When you back up your computer, you're not just backing up raw bits of data; you're backing up your life. Why do I put it that way? Well, think about what's on your computer. Email, bookmarks to your favorite web sites, contacts in your Address Book, your music, pictures, documents, and projects you're working on, and most importantly passwords stored in your Keychains file that you use to access everything. If you don't have a regular backup ritual, you should really think good and hard about implementing something now. Not tomorrow, or next week. Now. You never know when your Mac is suddenly going to get hit with stray solar radiation and go all wacky like Screwball Squirrel in a Tex Avery cartoon (and that's pretty darn wacky).

A.1.1.1 Where to back up

If you have a .Mac account, you can only back up about 100 MB of data to your iDisk, assuming you haven't expanded your storage capacity. Sure, you can expand your online storage to hold more data, but to expand your iDisk at the price point Apple wants, you may as well go out and buy an external FireWire drive. FireWire drives are relatively inexpensive, but do your homework before you run out to pick something up. Things I look for in a drive are:

Size

> Because the need for space is insatiable, you should always get the largest drive you can afford.

Portability

> Think hard on this one. Do you want or need to take your FireWire drive around with you? Or is a larger drive that sits on your desk but that doesn't fit in your backpack okay? If you want a portable drive, make sure to get a drive that doesn't require an external power source. I've ended up with a few of each: a couple of small 30 GB drives that fit in my backpack and don't require a power cord; and a huge 200 GB monster that sits behind my desktop computer.

Speed

> For a portable drive, you'll have to settle for a slower disk. If, however, you've opted to

get a bigger desktop-sized drive, be sure to get something nice and fast. 5400 rpm is okay, 7200 rpm is much better. When you're storing as much data as modern drives can hold, you'll want to be able to get to it right away. The faster, the better.

Backup software and other utilities

Some FireWire drives come with tools that you can use for backing up data from your Mac to the drive. There are also some third-party backup utilities, but again, I won't recommend one over the other because that's really not my place. Frankly, buy a drive for the drive and not for any packaged extras it might come with.

Take the time to read the reviews of various drives, their specs, and any software that might come with them before you make your final decision. The best advice I can give when it comes to buying a backup drive is: don't be cheap. Remember, this is your life you're backing up. I've owned cheap FireWire drives before and regretted it.

A.1.1.2 What to back up

Okay, so now that I've made it pretty clear that you need to pay attention to what you're backing up your data to, it's time to look at what you should really back up before you gut your existing system and install Panther.

For the purpose of gutting and installing a new system, you won't need to worry about backing up the System folder and all the Unix utilities that came with Mac OS X.

Again, focus on what's most important to you: stuff in your Home folder, any shell scripts or AppleScripts you may have written, configuration files, and application preferences.

If you have a .Mac account, you should consider using iSync for part of your backup needs. Huh, iSync, you say? Well, consider that iSync lets you store the contacts in your Address Book, calendar and to-do items from iCal, and bookmarks from Safari on your iDisk. These are three things that you won't need to back up if you have a .Mac account. Before you perform your backup, sync your data to your iDisk (or iPod) with iSync. That will store the latest copy of your data in a remote location where you can later retrieve it by performing another sync after you install Panther.

Here is a list of things you should seriously consider backing up before you thrash your system:

- Address Book data (*~/Library/Application Support/AddressBook*)
- iCal data (*~/Library/Calendars*)
- Safari bookmarks (*~/Library/Safari/Bookmarks.plist*)
- Preferences (*~/Library/Preferences*)
- Keychains (*~/Library/Keychains*)
- Email, message sorting settings, and junk-mail database rules from Mail.app (*~/Library/Mail*)
- Stickies (*~/Library/StickiesDatabase*)
- Fonts you've added to the system, either in your local domain (~/Library/Fonts) or for

global use on your system (/Library/Fonts)

- Any shell scripts you have created
- Any AppleScripts you have created
- Any databases you run and access frequently, including FileMaker Pro and MySQL databases

In addition to these specific items, you'll also want to look for anything stored in these folders in your Home folder:

- Documents (files you've created and saved locally)
- Movies (movies you've saved or created with iMovie, Final Cut Express, or Final Cut Pro)
- Music (music stored in iTunes)
- Pictures (pictures stored in iPhoto)
- Sites (any local web site you've created and are serving from your Mac)
- Files saved to your Desktop

Also, you'll want to make sure that you have the following information somewhere safe:

- QuickTime Pro's registration number (you can find this by going into the QuickTime preference panel)
- The registration codes for all your software (don't forget to have copies of your software or make sure you know where to go to download it from the Internet)

Again, this is just a rough list. You should use this as a guide, and then look good and hard at all the data stored on your Mac before you back it up. Once you've started installing Mac OS X on your Mac, even if you use one of the Archive and Install options that you can select, you should assume that there's no going back to recover the data. Make sure you have a good, solid backup before you pop in the first install disc and restart with the C key held down.

A.2 Installing Panther

Assuming that you've taken the time to back up your data and check it to make sure that everything you need is there, it's now time to think about installing Panther on your Mac. The important word there is "think"; there's nothing wrong with a default Panther installation, it's just that you should really think about how you're going to use your Mac.

- Are you going to run Classic? If so, you should consider setting up a separate partition to install Mac OS 9 in.
- Are you going to run more than one version of Mac OS X on your Mac for testing purposes? If so, you'll need separate partitions for them, too.
- Are you running an application that can benefit from a scratch disk, such as Final Cut Pro or Photoshop? Consider setting aside part of your hard drive as a partition just for that purpose. It won't be as good as a dedicated separate drive, but it's better than nothing.

The next thing you'll need to decide is how you will install Panther: clean or archive?

- A clean install is recommended, since it wipes your drive and checks it for errors (and attempts to fix said errors) before installing the operating system.
- If you opt to archive and install, all the data in the */Users* directory will be archived and retained in a buffer during the install, then dropped back into the */Users* directory once the operating system has completed. Then it's up to you to go back and pull what you want out of the archive and trash the rest.

The archive and install option works well in many cases. However, I'm just paranoid enough about software, and how various versions of software interact with each other, that I always go for the clean install myself.

A.2.1 Partitioning Your Hard Drive

If you're planning to partition your hard drive into more than one big partition, you should jot down how big you want those partitions to be. Keep in mind that the largest partition should be used for the system you boot into most.

For example, say you have a 40 GB drive, and you want to have separate partitions for your primary Panther installation, a test Mac OS X installation (for playing with other versions of the system—something that happens all too often in the *Running Mac OS X Panther* labs), Classic, and a scratch disk for Photoshop. If you aren't going to be using the test installation and Classic for much more than testing, you can get away with devoting a bare minimum of space on this partition. Classic will fit in under 2 GB, and 3 GB will do in a pinch for a Mac OS X installation. You'll also want to size your scratch disk appropriately; for Photoshop work 2 GB is usually sufficient unless you are working with poster-sized images. So, for this example, the partition scheme for a 40 GB drive might look like that shown in Table A-1.

Table A-1. A sample partition scheme.

Partition	Size	Use
1	33	Panther (Mac OS X 10.3) for primary use
2	3	Test Mac OS X installation
3	2	Classic
4	2	Scratch Disk

Once you have decided that you need to partition your drive, you'll need to do it before or during the installation process. You can't wait till after you finish and repartition your drive without loosing the data on your drives.

To partition your drive during the installation process, follow these steps:

1. Insert Mac OS X Install Disc 1 and reboot your Mac holding down the C key as it starts up.

2. From the menu bar select Installer ⟶ Open Disk Utility; this launches the Disc Utility program from the installation CD.

3. In the left pane, select the hard drive you want to partition.

4. Click the Partition tab button to the right to examine the partitioning scheme for the hard drive.

5. In the Volume Scheme section, the pop-up menu should probably be set to Current. From this, select the number of partitions you want to create on your hard drive. You can have up to 16 partitions on a single drive.

6. Set up your partitions by either grabbing the slider bar between the partitions in the Volume Scheme side or in the Volume Information section to the right.

7. In the Volume Information section, make sure you install the Mac OS 9 drivers if you plan to install Mac OS 9 to run Classic. Depending on how much space you need for Classic, give yourself at least 2 GB of space (and maybe more if you can) for installing Mac OS 9 and the applications you'll need to run in Classic mode. If you think you'll need more space for Mac OS 9, allocate the amount of space the apps require.

8. When you click on a partition in the Volume Scheme section, details about that partition show up in the Volume Information section, including its Name, Format, and Size. For example, when you set up new partitions, the partitions will have a name of Untitled 1, Untitled 2, and so on. If you want to change the name of a partition, click the partition block in the Volume Scheme section and then give the partition a new name by typing something into the Name field (for example, Panther).

9. For Mac OS X partitions, set the Format to "Mac OS Extended (Journaled)".

10. For Mac OS 9 partitions, set the Format to "Mac OS Extended".

11. When you're done changing the information in the Volume Information section, click the Partition button.

12. A warning sheet pops up, letting you know that by partitioning, you will destroy all the information on the drive. If you're certain that you have a good backup to reload your data from, click the Partition button on the sheet to split up your drive.

Your hard drive will be erased and the drive will be reformatted with the number of partitions you selected. You'll know Disk Utility is done when you see the partitions show up in the left side of the window.

Now that your drive has successfully been partitioned, quit Disk Utility with ⌘-Q to resume the installation process.

A.2.2 Step-by-Step Installation

Okay, so you've backed up your data, you've figured out how you're going to install Panther (clean, right?), and you've evaluated how you need to slice up your hard drive during the install. Now it's time to roll up your sleeves and get on with the process of installing Panther on your Mac.

A.2.2.1 Phase One: Installing the System

Once you've decided how you want to partition your drive—or even that you want to leave it all on one big partition—the next step is to actually break out the install discs and do the deed. Here's the step-by-step process, including pointers to the places in the installation process where you'll want to make some decisions.

1. With your Mac running, insert Install Disc 1 in your CD drive.

2. Restart your computer.

> A faster way to restart your Mac is to hold down the Option key and use the Apple→Restart menu. By holding down the Option key, you're forcing your Mac to restart without it prompting you first.

3. As your Mac is starting up (in other words, when you hear the famous *booonnnnnng!* sound), hold down the C key. This forces your Mac to boot using the Install Disc and starts you on your way to installing Panther on your Mac.

> If you've decided to repartition your drive, here's the point at which you'll want to follow the step-by-step directions from the previous section,*Partitioning Your Hard Drive.*

4. After your Mac starts up, you will be welcomed to the Mac OS X Installer. The installer uses a set of screens to help you configure your system and then the Mac OS X installation. You'll encounter the following screens:

Select Language

The first screen you see asks you to select the primary language you'll use on your

Mac. This sets the language defaults for the system, as well as the applications you later install.

Welcome to the Mac OS X Installer

This is just a warm little greeting to let you know that you're on your way to installing Mac OS X; just click the Next button to advance.

Important Information

This screen contains a lot of information, including details about the minimum system requirements for Panther, a firmware notice, advice about hardware compatibility, and advice about installing Mac OS X. You should take the time to read through all of this, since you're bound to encounter something that you haven't thought of yet.

The big gotcha for most people is the firmware. If you're not sure if your firmware is up-to-date, you should download the latest patch from Apple's web site and install it.

Software License Agreement

This is Apple's standard license agreement for Mac OS X. If you have the time, you should read through this before clicking the Continue button, but chances are you could read through (and understand) *Moby Dick* faster. After clicking the Continue button, a sheet will flop out of the window's titlebar asking you to confirm that you've read the license agreement and you agree to be persecuted to the fullest extent of the law should you be found in violation of said agreement. If you agree to be banished to the Land of Misfit Toys, click the Agree button to proceed with the installation process.

Select a Destination

Everyone wants a destination to go to, and so does Panther. Here, you need to decide where you're going to install Panther. If you haven't yet partitioned your hard drive, now's the time to do that before you select a drive to install Mac OS X on. For more information on how to partition your hard drive, see the section *Partitioning Your Hard Drive* later in this appendix.

After selecting which drive or partition you're going to install Mac OS X on, you will see that the Options button is now clickable. Clicking the Options button gives you, well, the option to choose how you want Mac OS X to be installed on the drive. The three options you have to choose from are:

Upgrade Mac OS X

This allows you to upgrade an earlier version of Mac OS X that exists on the disk to the new version you're installing.

Archive and Install

This option moves all the system files into a folder named Previous System and then installs the new version of Mac OS X. Beneath this option is a checkbox with the label, Preserve Users and Network Settings. If you click this checkbox, all the user data and any network settings will be saved as well. This is a handy option to use if you don't want to gut the entire system and want to make it (sort of) easier for users to get up and running (sort of) quickly after the install.

Erase and Install

This final option completely erases the hard drive or partition before installing Mac OS X. This is known as a clean install, and is recommended for any type of major release upgrade.

Also on the Options page is a pop-up menu that lets you select the filesystem type for Panther. Here you have just two options to select from, and the one to choose is Mac OS Extended (Journaled). Unless you really know what you're doing, do not select Unix File System from this menu.

Easy Install on HardDiskName

This is where the proverbial fork in the road comes in during the install. If you take the right fork, you can go with the default Easy Install; if you take the left fork, you can click the Customize button to pick and choose the items that will be installed.

If you select the Easy Install, the standard applications like iCal, iSync, iPhoto, iMovie, Internet Explorer, and all the possible language support packages will be installed on your Mac. The one thing that won't get installed on your Mac with the Easy Install method is Apple's version of X11; if you want that, you'll need to click the Customize button (or install it later from Install Disc 3).

If you click the Customize button, you'll be taken to another screen that's labeled, "Customize Install on *HardDiskName*". Below that, you'll see a list of the packages that can be installed. Items that have a checkmark in their box will be installed as part of the Easy Install. These items include:

- Essential System Software (this item is grayed out and cannot be unchecked)
- BSD Subsystem
- Additional Applications; these include Internet Explorer, StuffIt Expander, iTunes, iMovie, iPhoto, iCal (which is selected but grayed out), and iSync
- Printer Drivers (all but Epson Printer Drivers 2 are selected)
- Additional Speech Voices
- Fonts (all but Fonts for Additional Languages are selected)

    - Language Translations

    The following items are deselected by default; if you want to install these, you need to place a check in their checkbox:

    - Epson Printer Drivers 2
    - Fonts for Additional Languages
    - X11

    Things you can uncheck include:

    Language Translations

    The language you've selected at the beginning of the installation process will be installed by default, but why install Dutch, Japanese, and French (among a few) if you don't need them? By deselecting this item, you will free up 695 MB of hard drive space for other things.

    Printer drivers you won't need

    If you only have one brand of printer attached to your Mac, there really isn't a need to install the other brands. However, if you are installing on a laptop, you might want to keep all the drivers so that you can print to any random printer you might come across in your travels.

    Any of the Additional Applications

    Uncheck any applications you don't use. For example, Internet Explorer. After all, Mac OS X comes with Safari, so why install Microsoft's browser when you really don't need it?

5. When you've finished selecting the items you want to install, click the Install button to begin the installation process. A progress bar will appear with messages below, telling you what's being installed.
6. Insert Disc 2 when prompted.
7. Insert Disc 3 if prompted; you will only be prompted to insert Install Disc 3 if you're installing Apple's X11 package.

> If you didn't opt to install X11 from the Customize install screen, you can always install it later by inserting Install Disc 3 in your Mac and going to the Packages folder. Once there, double-click the *X11User.pkg* package to install X11.

8. Quit the installation when prompted and click the Restart button.

A.2.3 Configuring the System

Once you've made it through the install phase of Mac OS X Panther, your Mac will restart and you'll be presented with a series of set up screens to help you configure your Mac at a very basic level. You'll set up the first user account, configure network settings, and more.

The following list describes the configuration screens you'll encounter:

Welcome

> Based on information you provided back in Phase One on the Select Language screen, this screen asks you to select the region or country you live in. For example, if you selected English as your language, you will be asked to choose one of the following: United States, Canada, Australia, Ireland, or the United Kingdom. There is also a checkbox on this page to reveal other countries you can select from.

Personalize Your Settings

> Based on the country you selected on the Welcome screen, you'll be asked here to choose a keyboard layout to match your language needs. Like the previous screen, there is a checkbox to reveal all the other countries.

Your Apple ID

> If you have an Apple ID, you can enter that here along with its password. If you have a .Mac account, your .Mac email address (for example, *runningosx@mac.com*), as well as its password, will be your Apple ID by default. If you don't have a .Mac account, you can select the option "Create an Apple ID for me" and the installer will build an ID for you based on your username.

Registration Information

> Here you get to enter your name, address, phone number, and email address. This information gets transmitted back to Apple when you register Panther with Apple at the end of the installation process. The information you provide is only used by Apple, and if you have any concerns about how the company might use this information, you should click the Privacy button to reveal Apple's Privacy Policy.

A Few More Questions

> There are three options for you to select from here to provide additional information about yourself (or more importantly to Apple, how you intend to use Mac OS X). The first two options are found in pop-up menus, where you're asked to answer where you will use your computer and to describe what you do for a living. The third option just asks whether you'd like to receive Apple news and other information about its products, as well as

services from other companies.

Thank You

Now that you've filled out the basic information about who you are, click the Continue button to transmit that data to register your Mac and Panther with Apple.

Create Your Account

The first and last name you entered on the Registration Information screen will be combined and placed in the Name and Short Name fields on this screen. The only exception is that your Name appears as it normally would (for example, Norman Cook), and your Short Name appears as all lowercase text, run together (for example, norman). Fortunately, these fields are editable, so you can go back and tweak these to your heart's content. There are also spaces to enter and verify a password for your account, as well as to provide a Password Hint, and to select a picture that will be used on the login screen if you set up your account to require a login password.

Since this is the first user you're setting up on your Mac, this user will have administrator privileges by default. Other users can be set up later using System Preferences ⟶ Accounts, and they too can be assigned administrator privileges if you think they're worthy.

Get Internet Ready

This screen gives you two simple options to select from: to either use your existing Internet service, or to not configure your Mac to connect to the Internet. If you select the second option, you'll be asked to verify that you really don't want to connect your Mac to the Internet. If this is really what you want to do, you'll skip the next four steps and find yourself at the Select Time Zone screen, described later.

How Do You Connect?

Depending on the connection capabilities of your Mac, you will have one of five options to choose from:

- Telephone modem
- Cable modem
- DSL modem
- Local network (Ethernet)
- Local network (AirPort wireless)

Select the connection you'll use most and click the Continue button to configure your Mac to connect to the Internet.

If you select Cable modem or DSL modem, you will be taken to the Your Internet

Connection page, where you will need to fill in information about the network you will be connecting to. For the most part, leaving the TCP/IP Connection Type menu set to Using DHCP should be all you need to do here before clicking the Continue button, but check with your ISP first to see if it has anything specific for you to enter in the following fields:

- DHCP Client ID
- DNS Hosts
- Domain Name
- Proxy Server

If you selected Local network (either Ethernet or AirPort), you will be taken to the Your Local Area Network screen. If the network you're connecting to uses DHCP, your Mac will obtain its IP address from the DHCP server. If you want to use this configuration, make sure that the radio button next to the Yes option is selected and then click the Continue button. If you don't want to use DHCP (maybe you get to have a static IP address?), click the radio button next to the No option and then click Continue to go back and configure your network settings.

Now You're Ready to Connect

With all your settings in place, it's now time to establish a connection with Apple's servers to send off your registration information. Click the Continue button to register and proceed. You'll see a Connecting... screen with a twirling progress meter as your registration information is sent along to Apple.

Set Up Mail

If you have a .Mac account, the fields on this screen will be filled in to use your mac.com email account. If you have another email account you'd like to use, click the radio button next to "Add my existing email account", and then fill in the blanks for your other email account.

Select Time Zone

Here you're shown a map of the world, from which you can select the city nearest you for establishing the date, time, and the time zone your Mac is located in.

Thank You

If you've made it to this screen, your job is done. Panther has been successfully installed on your Mac and you're ready to start using it. Well, as soon as you click the Go button.

Finally, after about an hour or so of installing Panther on your Mac, you're ready to embark on using your Mac. It's up to you to set up and configure your Mac's preferences (through the System Preferences application) however you'd like. If you're in need of information on how to tweak and use Mac OS X Panther (other than the information you'll find in this book), you should check out the following books:

Mac OS X Panther Pocket Guide, Third Edition, by Chuck Toporek (O'Reilly & Associates, Inc., 2003).

> Revised and expanded to cover Panther, this handy little book covers all the basics you need to know about using Mac OS X Panther. The book includes keyboard shortcuts and a 35-page Task and Setting Index that you can use to further con- figure your Mac. The book's small size makes it easy to carry around in your computer bag and doesn't clutter up your desk either.

Mac OS X: The Missing Manual, Panther Edition, by David Pogue (O'Reilly & Associates, Inc./Pogue Press, 2003)

> David Pogue's Missing Manuals have shown Mac users the light, and this newly revised edition continues to lead the pack. If you're new to Mac OS X, you should read this book before you read the book you now hold in your hands.

Mac OS X Panther in a Nutshell, by Chuck Toporek, et al. (O'Reilly & Associates, Inc., 2004)

> More of a geek's treasure trove, *Mac OS X Panther in a Nutshell* contains a quick and dirty overview of Mac OS X from the 50,000 foot level, and then goes right down to the core, drilling into Mac OS X's Unix heart. About a third of the book is the Unix command reference, listing over a third of the approximate 1000 shell commands you can issue under Mac OS X. This book makes a great companion to *Running Mac OS X Panther*, and should be by your side as you explore the depths of your Mac.

Appendix B. Boot Command Keys

It seems like the people who truly know how their Mac works inside and out can do odd things with it by holding down a key, or a combination of keys, as the machine boots. Table B-1 lists the known boot keys that can be used when Mac OS X is starting up.

Table B-1. Boot keys to be used when Mac OS X is starting up

Key	Description
C	Boot from the CD or DVD drive installed in the computer
D	Boot from the internal disk drive
N	Boot from a network server
R	Resets the display on PowerBooks and iBooks to factory settings
X	Boot into Mac OS X
Shift-Option-⌘-Delete	Ignore the configured boot device setting and scan for an alternate boot device
T	Put the computer into FireWire target mode so it can be used as a hard drive
Option	Have Open Firmware run the OS Picker app to choose a disk to boot from
Shift	Boot into Safe Mode
⌘-V	Perform a verbose boot allowing you to see all the BSD output as the system boots
⌘-S	Perform a boot into single-user to perform some sort of system maintenance
Option-⌘-O-F	Access Open Firmware directly without booting into an operating system
Option-⌘-P-R	Reset Open Firmware's NVRAM, also known as "zapping" the PRAM
Hold down mouse button	Eject any media, such as a CD, in the system's removable drive

Appendix C. Other Sources of Information

The following is a list of resources for Mac OS X users, including books, magazines, mailing lists, and web sites.

C.1 Books

The following books are available for Mac users, administrators, and developers:

C.1.1 User and Administrator Focus

- *iMovie 3 & iDVD: The Missing Manual*, by David Pogue (Pogue Press/O'Reilly &Associates, Inc., 2001)
- *iMovie 2 Solutions: Tips, Tricks, and Special Effects*, by Erica Sadun (Sybex, 2002)
- *iPhoto 2: The Missing Manual*, by David Pogue, Joseph Schorr, and Derrick Story (Pogue Press/O'Reilly & Associates, Inc., 2003)
- *iPod: The Missing Manual*, by David Pogue (Pogue Press/O'Reilly & Associates, Inc., 2003)
- *Learning the bash Shell*, by Cameron Newhan and Bill Rosenblatt (O'Reilly & Associates, Inc., 1998)
- *Learning Unix for Mac OS X Panther*, by Dave Taylor and Brian Jepson (O'Reilly & Associates, Inc., 2004)
- *The Little Mac OS X Book*, by Robin Williams (Peachpit Press, 2001)
- *The Macintosh Bible, Eighth Edition*, by Clifford Colby, Marty Cortinas, et al. (Peachpit Press, 2002)
- *Mac 911*, by Christopher Breen (Peachpit Press, 2002)
- *Mac OS 9: The Missing Manual*, by David Pogue (Pogue Press/O'Reilly & Associates, Inc., 2000)
- *Mac OS X: The Missing Manual, Panther Edition*, by David Pogue, (Pogue Press/O'Reilly & Associates, Inc., 2003)
- *Mac OS X Disaster Relief*, by Ted Landau and Dan Frakes (Peachpit Press, 2002)
- *Mac OS X Panther for Unix Geeks*, by Brian Jepson and Ernest E. Rothman (O'Reilly & Associates, Inc., 2002)
- *Mac OS X Panther in a Nutshell*, by Chuck Toporek, Chris Stone, and Jason McIntosh (O'Reilly & Associates, Inc., 2004)
- *Mac OS X Panther Killer Tips*, by Scott Kelby (New Riders Publishing, 2003)
- *Mac OS X Panther Pocket Guide, Third Edition*, by Chuck Toporek (O'Reilly & Associates, Inc., 2003)
- *Mac OS X Panther Unleashed*, by John Ray and William C. Ray (Sams, 2003)
- *Office X for Macintosh: The Missing Manual*, by Nan Barber, David Reynolds, Tonya Engst (Pogue Press/ O'Reilly & Associates, Inc., 2002)

- *Secrets of the iPod*, by Christopher Breen (Peachpit Press, 2002)
- *Switching to the Mac: The Missing Manual*, by David Pogue (Pogue Press/O'Reilly & Associates, Inc., 2001)
- *Using csh & tcsh*, by Paul DuBois (O'Reilly, 1995)

C.1.2 Developer Focus

- *AppleScript: The Definitive Guide*, by Matt Neuburg (O'Reilly & Associates, Inc., 2003)
- *AppleScript for Applications: Visual QuickStart Guide*, by Ethan Wilde (Peachpit Press, 2001)
- *Carbon Programming*, by K. J. Bricknell (Sams, 2001)
- *Cocoa Programming*, by Scott Anguish, Erik Buck, and Donald Yacktman (Sams, 2002)
- *Cocoa Programming for Mac OS X*, by Aaron Hillegass (Addison-Wesley, 2001)
- *Cocoa Recipes for Mac OS X: The Vermont Recipes*, by Bill Cheeseman (Peachpit Press, 2003)
- *Learning Carbon*, by Apple Computer, Inc. (O'Reilly & Associates, Inc., 2001)
- *Learning Cocoa with Objective-C, Second Edition*, by James Duncan Davidson and Apple Computer, Inc. (O'Reilly & Associates, Inc., 2002)
- *Objective-C Pocket Reference*, by Andrew M. Duncan (O'Reilly & Associates, Inc., 2002)
- *Programming Objective-C*, by Stephan Kochan (Sams, 2003)
- *REALbasic: The Definitive Guide, Second Edition*, by Matt Neuburg (O'Reilly & Associates, Inc., 2001)

C.2 Magazines

The following print magazines are available for Mac users:

MacAddict

Published monthly, MacAddict is a magazine for users and power users. Each issue contains hardware and software reviews, and is accompanied by a CD containing free and shareware applications, as well as demo versions of games and many popular graphics applications.*www.macaddict.com*

Mac Design

Published monthly, *Mac Design* is a magazine for Mac-based graphic designers. *www.macdesignonline.com*

macHOME

Published monthly, *macHOME* is a magazine for home-based Mac users. Each issue contains articles and tutorials on how to use your Mac. *www.machome.com*

MacTech

Published monthly, *MacTech* is a magazine for Macintosh developers. Each issue contains articles and tutorials with code examples. *www.mactech.com*

Macworld

Published monthly, each issue of *Macworld* contains hardware and software reviews, as well as tutorials and how-to articles., *www.macworld.com*

C.3 Mailing Lists

The following mailing lists can help you learn more about the Mac.

C.3.1 Apple-Run Mailing Lists

The following mailing lists are run by Apple. For information on how to subscribe to these and other Apple-owned mailing lists, see *lists.apple.com*.

> Apple also maintains a listing of miscellaneous Mac-related mailing lists at *lists.apple.com/cgi-bin/mwf/forum_show.pl*; click the "Non-Apple Mailing Lists" link.

applescript-studio

For scripters and developers who are using AppleScript Studio to build AppleScriptbased applications for Mac OS X.

applescript-users

For scripters who are working with AppleScript.

carbon-development

For Carbon developers.

cocoa-dev

For Cocoa developers.

java-dev

For Java developers.

mac-games-dev

For Mac-based game developers.

macos-x-server

For network and system administrators who are running the Mac OS X Server.

rendezvous

Discussions on how to develop applications and devices that use Rendezvous.

scitech

Discussions on Apple's support for science and technology markets.

studentdev

For student developers.

weekly-kbase-changes

Keep informed of weekly changes to Apple's Knowledge Base (KB).

C.3.2 Omni Group's Mailing Lists

The following mailing lists are run by the Omni Group. For more information on how to subscribe to these and other Omni lists, see *www.omnigroup.com/developer/mailinglists*.

macosx-admin

A technical list for Mac OS X system administrators.

macosx-dev

A moderated list for Mac OS X application developers.

macosx-talk

A list for general discussions about the Mac OS X operating system.

C.4 Web Sites

These are just a few of the many URLs every Mac user should have bookmarked.

C.4.1 Apple Sites

Web site	Address
Apple's Mac OS X page Software Downloads page Apple's Support page Apple's Knowledge Base Apple Developer Connection (ADC) Apple Store Locator Bug Reporting .Mac page	www.apple.com/macosx www.apple.com/downloads/macosx www.info.apple.com kbase.info.apple.com developer.apple.com www.apple.com/retail developer.apple.com/bugreporter www.mac.com

C.4.2 Developer

Web site	Address
Cocoa Dev Central Cocoa Dev Wiki Mac DevCenter (by O'Reilly Network) Stepwise	www.cocoadevcentral.com www.cocoadev.com www.macdevcenter.com www.stepwise.com

C.4.3 Discussions and News

Web site	Address
Applelust Apple Slashdot MacCentral MacInTouch Mac Minute Mac News Network MacSlash	www.applelust.com apple.slashdot.org www.maccentral.com www.macintouch.com www.macminute.com www.macnn.com www.macslash.org

C.4.4 Rumor Sites

Web site	Address
Apple Insider Fly on the Mac MacRumors RumorTracker SpyMac Think Secret	www.appleinsider.com www.flyonthemac.com www.macrumors.com www.rumortracker.com www.spymac.com www.thinksecret.com

C.4.5 Software

Web site	Address
AquaFiles Bare Bones Software Fun with Fink The Omni Group Version Tracker	www.aquafiles.com www.barebones.com www.funwithfink.com www.omnigroup.com www.versiontracker.com/macosx

C.4.6 Tips, Tricks, Advice

Web site	Address
Mac OS X FAQ Mac OS X Hints Google Apple/Macintosh Search	www.osxfaq.com www.macosxhints.com www.google.com/mac.html

About the Author

James Duncan Davidson is a freelance author, speaker, and software developer, focusing on Mac OS X, Objective-C, Cocoa, Java, and related technologies. He is also the author of *Learning Cocoa with Objective-C*, coauthor of *Cocoa in a Nutshell*, and a contributor to *Mac OS X Hacks*—all published by O'Reilly & Associates, Inc.

In a previous life, Duncan was the creator of Apache Tomcat and Apache Ant and was instrumental in their donation to the Apache Software Foundation by Sun Microsystems. While at Sun, he led the expert groups for two versions of the Java Servlet API and served as one of the architects of the J2EE (Java 2, Enterprise Edition) Platform. It was also during his tenure at Sun that Wilfredo Sánchez showed him Java running on an early build of Mac OS X and started Duncan down the path to switching to a Mac from Solaris, Linux, and Windows.

Duncan regularly presents at conferences all over the world on topics ranging from open source and collaborative development to programming Java more effectively. He didn't graduate with a computer science degree, but sees that as a benefit in helping explain how software works. His educational background is in architecture (the bricks and mortar kind), the essence of which he applies to every problem that finds him. Duncan lives in the Pearl District of Portland, Oregon—a city widely regarded for its success in urban planning.

Colophon

Our book is the result of reader comments, our own experimentation, and feedback from distribution channels. Distinctive covers complement our distinctive approach to technical topics, breathing personality and life into potentially dry subjects.

The animal on the cover of *Running Mac OS X Panther* is a German shepherd. The model for this picture was Vinny, a search and rescue dog for the King County (Washington) sheriff's department. The German shepherd was hand-drawn from photographs of Vinny by his aunt, Lorrie LeJeune, a former editor at O'Reilly.

Search and rescue dogs are in quite a stressful field of work. In order for a dog to perform well, it must adapt to many different things—for example, modes of travel, new people, all kinds of weather, and various types of terrain. Often, search and rescue dogs are medium to large in size. They are expected to be intelligent, strong, and generally even-tempered. The German shepherd is by no means the only breed of dog that takes on this line of work. Ultimately, search and rescue dogs must have a strong nose and be physically fit. It is a difficult job that requires the dedication and commitment of both the dog and its owner/partner.

Claire Cloutier was the production manager for this book. Jill Steinberg was the copyeditor and Kellie Robinson was the proofreader. Chuck Toporek, Claire Cloutier, Darren Kelly, and David Futato provided quality control. Julie Hawks wrote the index.

Emma Colby designed the cover of this book, based on a series design by Edie Freedman. The cover image is an original illustration created by Lorrie LeJeune. Emma produced the cover layout with Quark XPress 4.1, using Adobe's ITC Garamond font.

David Futato designed the interior layout. James Duncan Davidson implemented the layout in InDesign. The text font is Linotype Birka; the heading font is Adobe Myriad Condensed; and the code font is LucasFont's TheSans Mono Condensed. The tip and warning icons were drawn by Christopher Bing.

The initial drafts of the early chapters of the book were written using a variety of applications from Microsoft Word to BBEdit. OmniOutliner and OmniGraffle were invaluable for organizing random thoughts into coherent form. Once it became apparent that a toolset change was in order to speed the production process, Adobe InDesign and Adobe InCopy became the tools of choice for writing and enabled a high degree of collaboration between the author, editor, and copyeditor. The illustrations were produced using Adobe Photoshop and Adobe Illustrator. The layout of the text and illustrations into final form was performed with Adobe InDesign.

The final production of the book was accomplished over a 48-hour weekend in Portland, Oregon with James Duncan Davidson, Chuck Toporek, and Kellie Robinson collaborating over proofs and PowerBooks with Jill Steinberg joining in from Seattle via telephone and iChat. Adobe InDesign was used to generate the final PDF files that went to press after a brief stop in Cambridge, Massachusetts for a final look over.

Every word was written on a Mac.

The Running Mac OS X Panther Lab

Writing a book like this requires a lot of gear—and a willingness to abuse it. You have to be able to experiment with the system, try out interesting ideas, and generally tear into the guts of the system to see how it works. And yet you need a stable platform on which to actually write the book. Because of these two opposing needs, as well as the need to try out lots of networking configurations, a total of six machines were used. These machines were:

- A 15-inch Aluminum PowerBook with a 1 GHz G4, 1 GB memory, 80 GB hard drive, SuperDrive, light-up keyboard, and spots in the screen (and which will be sent in to get fixed once the book has gone to press).
- A 15-inch Titanium PowerBook with a 768 MHz G4, 768 MB memory, 40 GB hard drive, and DVD drive. At one point, this machine was displaying garbage on its screen and was feared to be dead because the author had poked at one too many Open Firmware settings, but luckily a PRAM reset or three brought it back from the abyss.
- A PowerMac G4 with Dual-800 MHz G4s, 1.5 GB memory, 40 GB hard drive, 200 GB external storage, SuperDrive, and a 17-inch Studio LCD display. This was the primary desktop machine used and suffered the bulk of the experimentation needed.
- A 17-inch iMac with a 1GHz G4, 768 MB memory, 40 GB hard drive, and SuperDrive running Mac OS X Server 10.3. This machine served as the primary file repository for the book as well as an Open Directory server for the network.
- A 15-inch Pismo PowerBook with 400 MHz G3, 768 MB memory, 6 GB hard drive, and DVD drive. Even though it's old, it still runs great and is a great test bed for the really scary stuff.
- A Gateway desktop with a 350 MHz Pentium II, 192 MB memory, 4 GB hard drive, and CD-RW drive running Windows 2000 Service Pack 4. This machine was used for Windows compatibility testing.

In addition to these machines, an original NeXT color slab, kindly donated by Dr. Carl Williams of Detroit, Michigan, kept its more modern comrades company.

The computers were networked together using switched 100BaseT Ethernet, 802.11b (AirPort) wireless, and 802.11g (AirPort Extreme) wireless. Connected to the Macs, as well as the AirPort Extreme Base Station, at various times were the following printers:

- Hewlett-Packard LaserJet 2200d with internal JetDirect Ethernet print server
- Hewlett-Packard DeskJet 990Cse
- Epson Stylus Photo 900
- Epson Stylus Photo 1270

Creative juice provided by iTunes, the iTunes Music Store, and the incomparable iPod. No bits were harmed during the creation of this book, although many reams of paper were sacrificed.