A solid introduction to programming on the Mac OS X Snow Leopard platform

The Mac OS X Snow Leopard system comes with everything you need in its complete set of development tools and resources. However, finding where to begin can be challenging. This book serves as an ideal starting point for programming on the Mac OS X Snow Leopard platform. Step-by-step instructions walk you through the details of each featured example so that you can type them out, run them, and even figure out how to debug them when they don’t work right. Taking into account that there is usually more than one way to do something when programming, the authors encourage you to experiment with a variety of solutions. This approach enables you to efficiently start writing programs in Mac OS X Snow Leopard using myriad languages and put those languages together in order to create seamless applications.

Beginning Mac OS X Snow Leopard Programming:
- Teaches you where to find current resources for the developer tools that come with your copy of Mac OS X Snow Leopard
- Explores Xcode®, the application used to build Mac OS X programs
- Walks you through designing a graphical user interface with Interface Builder
- Shows you how application resources are stored and how applications work in multiple languages
- Explains writing applications using the Cocoa® frameworks, Xcode, and Dashcode
- Addresses how various scripting languages extend Mac OS X’s command-line interface

Michael Trent is a technical reviewer for numerous books and magazine articles and the coauthor of Beginning Mac OS X Programming with Drew McCormack.

Drew McCormack is an experienced computational scientist, founder of the “The Mental Faculty”—an independent company developing software for the Mac and iPhone—and the coauthor of Beginning Mac OS X Programming with Michael Trent.

Wrox Beginning guides are crafted to make learning programming languages and technologies easier than you think, providing a structured, tutorial format that will guide you through all the techniques involved.

Beginning Mac OS X Snow Leopard Programming: Your guide to programming on the Mac OS X 10.6 Platform

Michael Trent, Drew McCormack
Related Wrox Books

Beginning iPhone SDK Programming with Objective-C
ISBN: 9780470500972
Learning to develop iPhone applications doesn’t need to be an overwhelming undertaking. This book provides an easy-to-follow, example-driven introduction to the fundamentals of the Apple iPhone SDK and offers you a clear understanding of how things are done when programming iPhone applications with Objective-C. When you reach the end of the book, you will be prepared to confidently tackle your next iPhone programming challenge.

Professional Xcode 3
ISBN: 9780470525227
The steady increase in sales of Apple computers has triggered a strong interest in gaining a thorough understanding of Xcode and its tools and what they have to offer. This book provides you with an inside look at the array of Xcode tools from top to bottom. You’ll go beyond the basics and dive into such in-depth topics as installing the latest version of Xcode tools, customizing the look and behavior of Xcode, creating and managing projects, using the built-in class browser to model complex applications and structures, and more. With this book, you’ll be able to take full advantage of the range of tools included with Xcode.

Safari and WebKit Development for iPhone OS 3.0
ISBN: 9780470549667
With the unparalleled success of iPhone and iPod touch, iPhone OS 3.0 has emerged as a compelling platform for which vast numbers of web developers are designing and building web-based mobile applications. This book explores the Safari and WebKit development platform that is built into iPhone OS 3.0 and takes you through the process of creating an iPhone web application from the ground up. You’ll learn how to use existing open source frameworks to speed up your development time, imitate qualities of built-in Apple apps, cache data locally and even run in offline mode, and more. Whether you’re eager to build new web applications for iPhone OS 3.0 or optimize existing web sites for this platform, you have everything you need to do so within this book.
BEGINNING
MAC OS® X SNOW LEOPARD™ PROGRAMMING

INTRODUCTION ................................................................. xv

PART I MAC OS X DEVELOPER RESOURCES
CHAPTER 1 The Mac OS X Environment ........................................ 3
CHAPTER 2 Developer Tools ..................................................... 23
CHAPTER 3 Xcode ................................................................. 53
CHAPTER 4 Interface Builder ................................................... 95

PART II APPLICATION PROGRAMMING
CHAPTER 5 The Application ..................................................... 117
CHAPTER 6 The C Language ...................................................... 145
CHAPTER 7 The Objective-C Language ....................................... 229
CHAPTER 8 Introduction to Cocoa ............................................ 291
CHAPTER 9 Document-Based Cocoa Applications ...................... 313
CHAPTER 10 Core Data–Based Cocoa Applications ................... 341

PART III SCRIPT PROGRAMMING
CHAPTER 11 Overview of Scripting Languages ......................... 389
CHAPTER 12 The Bash Shell ................................................... 425
CHAPTER 13 AppleScript and AppleScriptObjC ......................... 487
CHAPTER 14 Javascript, Dashboard, and Dashcode .................. 553

INDEX ............................................................................... 633
BEGINNING

Mac OS® X Snow Leopard™ Programming

Michael Trent and Drew McCormack

Wiley Publishing, Inc.
For Angela, Katie, and Sophie

—MT

For Jennifer, Gileesa, and Rylan

—DMc
ABOUT THE AUTHORS

MICHAEL TRENT has been programming in Objective-C since 1997 and programming Macs since well before that. He is a professional computer programmer and engineering manager, a technical reviewer for numerous books and magazine articles, and an occasional dabbler in Mac OS X open source projects. Currently, he is using Objective-C and Apple Computer’s Cocoa frameworks to build professional and consumer applications for Mac OS X. Michael holds a Bachelor of Science degree in Computer Science and a Bachelor of Arts degree in Music from Beloit College in Beloit, Wisconsin. He lives in Pittsburgh, Pennsylvania, with his family.

DREW McCORMACK has a Ph.D. in Chemical Physics and has worked much of his career as a computational scientist. Recently, he founded The Mental Faculty (www.mentalfaculty.com), an independent company developing applications for the Mac and iPhone in the fields of science and education. (The Mental Faculty’s flagship product is the flashcard study application Mental Case, which can be found at www.macflashcards.com.) Drew is a board member and regular contributor to the MacResearch website (www.macresearch.org), and is a lead on the Core Plot project (www.code.google.com/p/core-plot), an undertaking with the aim to develop a complete open source graphing framework for Cocoa.
ACKNOWLEDGMENTS

I WOULD LIKE TO THANK STEVE KOCHAN for his early help with this project. Thanks also to Brett Halle, Pete Steinauer, Yaniv Gur, and many others for their encouragement. Most of all, I would like to thank my wife, Angela, and my daughters, Katie and Sophie, for putting up with all the late nights.

—Michael Trent

I WISH TO THANK MY WIFE. Jennifer, and children, Gileesa and Rylan, for suffering first hand all of my wacky schemes. Thanks also to my brother — and mentor — Cody McCormack.

—Drew McCormack
# CONTENTS

## INTRODUCTION

## PART I: MAC OS X DEVELOPER RESOURCES

### CHAPTER 1: THE MAC OS X ENVIRONMENT

- Introducing the Mac OS X
- The Core Operating System
  - The Kernel
  - The System Library
  - Unix Commands
- Graphics and Media Layers
  - The Quartz Compositor
  - Quartz 2D
  - OpenGL
  - QuickTime
  - Core Animation
- Application Frameworks and UI
  - Cocoa
  - Carbon
  - Java JDK
- Apple Human Interface Guidelines
- Summary

### CHAPTER 2: DEVELOPER TOOLS

- Installing the Developer Software
- Developer Applications
  - Build Tools
  - Performance Tools
  - Other Utilities
  - Command-Line Tools
- Developer Documentation and Examples
  - Conceptual Documentation
  - API Reference
  - Examples
  - Man Pages
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple Developer Connection</td>
<td>46</td>
</tr>
<tr>
<td>Documentation</td>
<td>47</td>
</tr>
<tr>
<td>Examples</td>
<td>48</td>
</tr>
<tr>
<td>Mailing Lists</td>
<td>49</td>
</tr>
<tr>
<td>Developer Tools</td>
<td>49</td>
</tr>
<tr>
<td>Developer Support</td>
<td>50</td>
</tr>
<tr>
<td>Summary</td>
<td>50</td>
</tr>
<tr>
<td><strong>CHAPTER 3: XCODE</strong></td>
<td>53</td>
</tr>
<tr>
<td>Starting a New Project</td>
<td>54</td>
</tr>
<tr>
<td>Working with Files</td>
<td>58</td>
</tr>
<tr>
<td>Writing Your Code</td>
<td>63</td>
</tr>
<tr>
<td>A Second Look at Indenting</td>
<td>69</td>
</tr>
<tr>
<td>Building and Running</td>
<td>71</td>
</tr>
<tr>
<td>Finding Problems with the Static Analyzer</td>
<td>75</td>
</tr>
<tr>
<td>Using the Debugger</td>
<td>78</td>
</tr>
<tr>
<td>Online Documentation</td>
<td>85</td>
</tr>
<tr>
<td>Summary</td>
<td>92</td>
</tr>
<tr>
<td><strong>CHAPTER 4: INTERFACE BUILDER</strong></td>
<td>95</td>
</tr>
<tr>
<td>Starting a New Interface</td>
<td>96</td>
</tr>
<tr>
<td>Building Menus</td>
<td>99</td>
</tr>
<tr>
<td>Building Windows</td>
<td>102</td>
</tr>
<tr>
<td>Making Windows Resizable</td>
<td>108</td>
</tr>
<tr>
<td>Summary</td>
<td>112</td>
</tr>
<tr>
<td><strong>PART II: APPLICATION PROGRAMMING</strong></td>
<td></td>
</tr>
<tr>
<td><strong>CHAPTER 5: THE APPLICATION</strong></td>
<td>117</td>
</tr>
<tr>
<td>Basic Bundle Structure</td>
<td>118</td>
</tr>
<tr>
<td>Opaque Directory</td>
<td>118</td>
</tr>
<tr>
<td>Versioned Bundles</td>
<td>119</td>
</tr>
<tr>
<td>The Info.plist File</td>
<td>120</td>
</tr>
<tr>
<td>Executable Code</td>
<td>122</td>
</tr>
<tr>
<td>Global Resources</td>
<td>122</td>
</tr>
<tr>
<td>Localized Resources</td>
<td>122</td>
</tr>
<tr>
<td>Examining Application Bundles</td>
<td>123</td>
</tr>
<tr>
<td>Building an Application Bundle</td>
<td>127</td>
</tr>
<tr>
<td>Application Preferences</td>
<td>137</td>
</tr>
<tr>
<td>Summary</td>
<td>142</td>
</tr>
</tbody>
</table>
CHAPTER 6: THE C LANGUAGE

A Little History 145
Getting Started 146
Variables and Data Types 149
Operators 152
Arrays and Pointers 156
Conditional Branching 163
Loops 168
Functions and Blocks 175
Characters and Strings 179
Input/Output 184
Data Structures 193
Memory Management 196
The Preprocessor 200
Organizing Programs 206
Summary 227

CHAPTER 7: THE OBJECTIVE-C LANGUAGE 229

Object-Oriented Programming 230
Classes and Objects 231
Methods 234
Data Hiding 240
Subclassing 242
Messaging 246
Protocols and Informal Protocols 247
Naming Conventions 252
Creating and Destroying Objects 253
Memory Management 257
Accessor Methods and Properties 261
Summary 288

CHAPTER 8: INTRODUCTION TO COCOA 291

Connecting Cocoa Interfaces 292
Connecting Interfaces Manually with Outlets and Actions 292
Working with Cocoa Bindings 302
The Model/View/Controller Design Pattern 309
Summary 310
CONTENTS

CHAPTER 12: THE BASH SHELL 425

Getting Started 425
- The Command-Line Interface 426
- Interactive versus Script 426
- The Terminal Application 427
- Editors 427
- Configuring Bash 429

Unix Basics 433
- Paths 433
- Locating and Learning Commands 434
- Running Commands and Other Executables 436
- Redirecting Input and Output 438
- Navigating the File System 443
- Working with Files and Directories 446
- File Permissions 449
- Globbing 452
- Searching for Files 455
- Working with Text 458
- Process Control 463
- Mac OS X Exclusive Commands 466
- Overview of Other Commands 467

Shell Programming 468
- Scripts 468
- Variables 469
- Quoting 473
- Conditional Branching 474
- Looping 478
- Functions 480

Summary 484

CHAPTER 13: APPELSRCRIET AND APPELSRCRIETOBJC 487

AppleScript 488
- Creating an AppleScript 488
- Scriptable and Recordable Applications 490
- Apple Events 490
- Variables and Data Types 491
- Handlers 497
- Control Statements 501
- Operators 511
- Specifying Elements 515
- Scripts and Objects 518
<table>
<thead>
<tr>
<th>CONTENTS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Classes</td>
<td>521</td>
</tr>
<tr>
<td>Working with Scripts</td>
<td>521</td>
</tr>
<tr>
<td><strong>AppleScript Applications</strong></td>
<td>531</td>
</tr>
<tr>
<td>The AppleScriptObjC Bridge</td>
<td>532</td>
</tr>
<tr>
<td>Using Xcode</td>
<td>533</td>
</tr>
<tr>
<td>Using Interface Builder</td>
<td>535</td>
</tr>
<tr>
<td><strong>Summary</strong></td>
<td>549</td>
</tr>
<tr>
<td><strong>CHAPTER 14: JAVASCRIPT, DASHBOARD, AND DASHCODE</strong></td>
<td>553</td>
</tr>
<tr>
<td><strong>JavaScript</strong></td>
<td>554</td>
</tr>
<tr>
<td>Dashboard</td>
<td>560</td>
</tr>
<tr>
<td><strong>Dashcode</strong></td>
<td>576</td>
</tr>
<tr>
<td><strong>Summary</strong></td>
<td>589</td>
</tr>
<tr>
<td><strong>Appendix A: Answers to Exercises</strong></td>
<td>591</td>
</tr>
<tr>
<td><strong>Appendix B: Developer Resources</strong></td>
<td>631</td>
</tr>
<tr>
<td><strong>INDEX</strong></td>
<td>633</td>
</tr>
</tbody>
</table>
INTRODUCTION

Mac OS X represents a union of many different operating system technologies. At its core you will find Unix, an operating system once reserved for high-end servers and workstations, now found on common desktop systems. With Unix comes a whole history of tools, computer languages, and runtime environments. At a higher level, you will find Cocoa, derived from the application toolkits found on NeXTSTEP and OpenStep — a result of Apple's merger with NeXT in 1997. In Mac OS X you will also find Carbon, a library made from elements of the original Macintosh operating system. The original Mac OS API remains for older projects, although Apple is de-emphasizing its role in Mac OS X. Other technologies have found their way into Mac OS X through the open source community, and Apple is hard at work developing new technologies unique to Mac OS X.

Although the operating system is composed of all these separate pieces, Mac OS X still looks and feels like a single piece of software. The Macintosh's legendary user interface still shines brightly today, providing a consistent look and feel for the applications on your system. When you sit down to use your computer, it just works for you. And aside from a few cosmetic differences — say using a command-line interface or a graphical interface — rarely are you aware of the differences between all these operating system technologies.

Under the covers, there are fundamental differences between these operating system technologies. For example, Carbon still provides its own special memory data types, and Cocoa requires its own object-oriented runtime. But there are no firm boundaries between these technologies; Cocoa can call Carbon API and vice versa. And though these technologies have their own history, strengths, and weaknesses, they are all still part of the same Mac OS X operating system.

With the release of Mac OS X Snow Leopard, Apple has spent two years fine-tuning these pieces. Instead of focusing on a number of new features, Apple has been working on what it has: making things faster, using fewer resources, and smoothing out the edges. As a result, Snow Leopard feels as though it is one operating system — the whole greater than the sum of its parts.

WHO THIS BOOK IS FOR

This book is for anyone who wants to get started writing programs that run on Mac OS X. Every Mac OS X system comes with everything you need: a complete set of development tools and resources. But finding the place to start can be challenging. This book provides a great starting point for programming on Mac OS X and shows you how to find more information on your own.

This book will appeal most to the hobbyist programmer who already has some exposure to a programming language. Experience with C or Objective-C, although helpful, is not required. You can learn the basics of these languages, as well as concepts such as object-oriented programming (OOP), as you go.
INTRODUCTION

If you are an experienced programmer familiar with one aspect of Mac OS X programming (such as shell scripting or Carbon), you can use this book to explore other aspects of Mac OS X programming. In addition, you learn how to incorporate these new techniques into your existing programming style.

HOW THIS BOOK IS STRUCTURED

This book takes a hands-on approach to learning the material, using the Wrox “Try It Out” format. After you learn about a particular element, you are presented with step-by-step instructions that walk you through using that element. You are encouraged to follow along on your own system by typing the examples, running them, and if necessary debugging them when they don’t work quite right. A detailed explanation of the example follows the step-by-step instructions under a “How It Works” heading.

Some examples take the form of small standalone projects, designed to illustrate one particular concept or technique. When appropriate, examples build on material presented in earlier chapters or examples, providing an opportunity to see the new material in a non-trivial context. Many of the examples focused on Mac OS X application development build on an application called Slide Master, a functional image/slideshow browser that you build piece by piece. Whenever possible, larger tasks have been broken down into smaller examples to make them easier to digest.

Each chapter includes a few exercises at the end, again illustrating the lessons presented earlier. You can work through these examples on your own, at your own pace. Complete answers to each exercise are provided in the appendix, so you can check your work or get a hint if you get stuck. Keep in mind that in programming there’s usually more than one way to do something; our solution isn’t necessarily the only correct one.

Chapters are collected into three broad categories or parts: Mac OS X Developer Resources, Application Programming, and Script Programming. This keeps related material together and helps with the flow of information from topic to topic. You are encouraged to cover the material in order so that you don’t miss anything. If you want to skip ahead for a specific topic, make a point of coming back to the earlier material at a later time.

Here’s a brief summary of what you’ll find in each chapter:

In Chapter 1, “The Mac OS X Environment,” you learn about Mac OS X’s system architecture. You also get a little hands-on experience using Mac OS X’s built-in command-line interface.

In Chapter 2, “Developer Tools,” you find out about the resources that come with your copy of Mac OS X. You also learn how to find current versions of these resources on the Internet.

In Chapter 3, “Xcode,” you explore the application used to build Mac OS X programs. A few simple examples introduce you to writing source code, building a finished product, and debugging programs one line of code at a time.

Chapter 4, “Interface Builder,” walks you through the process of designing a graphic user interface on Mac OS X. Examples in this chapter illustrate useful techniques for building an interface that conforms to Apple’s guidelines.
In Chapter 5, “The Application,” you pick apart the individual elements that make up an application on Mac OS X. You learn how application resources are stored and how applications work in multiple languages.

Chapter 6, “The C Language,” offers a brief introduction to the C programming language. If you are new to C, you will want to read this chapter before continuing on to Chapters 7, 8, 9, or 10. In addition to learning how to write programs in C, you learn how non-trivial C programs are divided among several source files.

Chapter 7, “The Objective-C Language,” builds on Chapter 6 to teach you about Objective-C, the object-oriented language used by the Cocoa application frameworks.

In Chapter 8, “Introduction to Cocoa,” you discover how to write your own Cocoa applications, from designing a user interface to writing the final code. The Cocoa application frameworks do a lot of work for you, freeing you up to concentrate on the unique aspects of your own application.

Chapter 9, “Document-Based Cocoa Applications,” explores how to use Cocoa to build a program that works with documents of user data.

Chapter 10, “Core Data-Based Cocoa Applications” describes how to use Core Data to store collections of Objective-C objects. It also focuses on using Cocoa Bindings for passing data between Core Data and document UI.

In Chapter 11, “Overview of Scripting Languages,” you examine scripting languages available on the Mac OS X system. Many of these languages extend Mac OS X’s command-line interface in one way or another. You get a sense of what sets each language apart and what tasks each language is best suited to perform.

Chapter 12, “The Bash Shell,” covers Mac OS X’s default command-line interpreter in detail. You learn how to write shell scripts that interact with command-line tools installed on your system.

In Chapter 13, “AppleScript and AppleScriptObjC,” you learn about Apple’s high-level application scripting language, AppleScript. AppleScript enables you to communicate with and automate tasks in Mac OS X applications.

Chapter 14, “JavaScript, Dashboard, and Dashcode,” focuses on the JavaScript language, a scripting language used to make Dashboard widgets and modern web applications. You will use the Dashcode developer tool to write and debug JavaScript programs.

Appendix A, “Exercise Answers,” provides the solutions to the exercises that appear at the end of each chapter throughout this book. Also be sure to check out Appendix B, “Developer Resources,” which offers valuable information that you may find helpful as you develop applications.

**WHAT YOU NEED TO USE THIS BOOK**

As we mentioned earlier, your Mac OS X installation already has everything you need to get started. You will also need an Internet connection to access Apple Computer’s developer web site. We used the Safari web browser to access these pages, but other web browsers (Internet Explorer, Firefox, and so on) should work fine.
INTRODUCTION

Examples and figures in this book were made using Mac OS X v10.6 Snow Leopard and Xcode 3. Although many code examples will still work on earlier versions of Mac OS X, a few have been updated using features new to Snow Leopard. Apple periodically makes new versions of Xcode available for download. Earlier versions of Xcode can be obtained at no charge from Apple’s web site. You can find more information on Apple’s developer web site, Apple Developer Connection, in Chapter 2.

CONVENTIONS

To help you get the most from the text and keep track of what’s happening, we’ve used a number of conventions throughout this book.

TRY IT OUT

The Try It Out is an exercise you should work through, following the text in the book.

1. It usually consists of a set of steps.

2. Each step has a number.

3. Follow the steps through with your copy of the database.

How It Works

After each Try It Out, the code you’ve typed will be explained in detail.

WARNING Boxes such as this one hold important, not-to-be-forgotten information that is directly relevant to the surrounding text.

NOTE Tips, hints, tricks, and asides to the current discussion are offset and placed in italics similar to this.

The following are styles used in the text:

We highlight important words in italics when they are introduced.

We show keyboard strokes like this: Ctrl-A.

We show file and folder names, URLs, and code within the text in a special monofont typeface, like this: persistence.properties.
INTRODUCTION

SOURCE CODE

As you work through the examples in this book, you may choose either to type in all the code manually or to use the source code files that accompany the book. All of the source code used in this book is available for download at http://www.wrox.com. Once at the site, simply locate the book’s title (either by using the Search box or by using one of the title lists) and click the Download Code link on the book’s detail page to obtain all the source code for the book.

Once you download the code, just decompress it with your favorite compression tool. Alternately, you can go to the main Wrox code download page at http://www.wrox.com/dynamic/books/download.aspx to see the code available for this book and all other Wrox books.

ERRATA

We make every effort to ensure that there are no errors in the text or in the code. However, no one is perfect, and mistakes do occur. If you find an error in one of our books, like a spelling mistake or faulty piece of code, we would be very grateful for your feedback. By sending in errata you may save another reader hours of frustration and at the same time you will be helping us provide even higher-quality information.

To find the errata page for this book, go to http://www.wrox.com and locate the title using the Search box or one of the title lists. Then, on the book details page, click the book errata link. On this page you can view all errata that has been submitted for this book and posted by Wrox editors. A complete book list including links to each book’s errata is also available at www.wrox.com/misc-pages/booklist.shtml.

If you don’t spot “your” error on the book errata page, go to www.wrox.com/contact/techsupport.shtml and complete the form there to send us the error you have found. We’ll check the information and, if appropriate, post a message to the book’s errata page and fix the problem in subsequent editions of the book.

P2P.WROX.COM

For author and peer discussion, join the P2P forums at p2p.wrox.com. The forums are a Web-based system for you to post messages relating to Wrox books and related technologies and interact with other readers and technology users. The forums offer a subscription feature that e-mails you topics of your choosing when new posts are made to the forums. Wrox authors, editors, other industry experts, and your fellow readers are present on these forums.
INTRODUCTION

At http://p2p.wrox.com you will find a number of different forums that will help you not only as you read this book, but also as you develop your own applications. To join the forums, just follow these steps:

1. Go to p2p.wrox.com and click the Register link.
2. Read the terms of use and click Agree.
3. Complete the required information to join, as well as any optional information you wish to provide, and click Submit.
4. You will receive an e-mail with information describing how to verify your account and complete the joining process.

**NOTE** You can read messages in the forums without joining P2P but in order to post your own messages, you must join.

Once you join, you can post new messages and respond to messages other users post. You can read messages at any time on the Web. If you would like to have new messages from a particular forum e-mailed to you, click the “Subscribe to this Forum” icon by the forum name in the forum listing.

For more information about how to use the Wrox P2P, be sure to read the P2P FAQs for answers to questions about how the forum software works, as well as many common questions specific to P2P and Wrox books. To read the FAQs, click the FAQ link on any P2P page.
PART I
Mac OS X Developer Resources

- CHAPTER 1: The Mac OS X Environment
- CHAPTER 2: Developer Tools
- CHAPTER 3: Xcode
- CHAPTER 4: Interface Builder
The Mac OS X Environment

WHAT YOU WILL LEARN IN THIS CHAPTER:

➤ How the Mac OS X operating system is structured, including what the major areas of the system are and how they work together
➤ How to use Mac OS X’s command-line interface
➤ How applications take advantage of the operating system services on Mac OS X
➤ How Apple encourages a common look and feel for Mac OS X applications

Welcome to the wonderful world of Mac OS X, the next-generation operating system from Apple Computer!

The Mac OS X operating system powers modern Macintosh computers. After many long years and a few scrapped attempts to modernize the older Mac OS operating system, Apple released Mac OS X in April 2001. Since then, Apple has released a steady stream of upgrades and system updates. This book was written around Mac OS X v10.6 Snow Leopard, the latest version.

To write software for Mac OS X, you need to know your way around the system. By now you may already be familiar with Mac OS X’s applications and user interface style. Those things all rest on top of a number of subsystems and services that make up the Mac OS X operating system.
INTRODUCING THE MAC OS X

What comes to mind when you think of Mac OS X? Is it the applications you use? Perhaps you recall Mac OS X’s distinctive user interface? Or maybe you think of Mac OS X’s stability? In truth, Mac OS X embodies all these things.

The Mac OS X operating system is often described as a collection of layers, as seen in Figure 1-1. You are probably already familiar with the topmost layer: the applications that run on Mac OS X (such as Mail, iTunes, Safari, and so on). These applications are all written against a collection of application frameworks. These frameworks are special libraries that provide the code and all the other resources (icons, translated strings, and so on) to perform common tasks. For example, the Cocoa framework contains a number of resources necessary to make a Cocoa application.

All Mac OS X applications use graphics to some extent, ranging from simply presenting its user interface to processing graphical data such as QuickTime movies. The system provides several specialized libraries for working with graphics and graphics files.

These layers rest on the broad shoulders of the core operating system, which at the lowest level is responsible for making your Macintosh run. For example, the core OS handles reading from and writing to your hard drive and random access memory (RAM), it manages your network connections, it powers down the computer when it falls to “sleep,” and so on. In fact, any program that talks to your hardware in any way ultimately goes through the core OS.

Throughout this book you examine Mac OS X in detail through Slide Master, an application that builds and displays photo slideshows. You will build Slide Master bit-by-bit as you learn more about how the elements of Mac OS X come together. The Slide Master application and its source code can be downloaded from Wiley’s web site; so you can check your work against our complete solution as you go.

This is a good time to take a quick tour of Slide Master. You can download Slide Master from Wiley’s web site, make a slideshow, and view your handiwork. In doing so, you touch on all the major areas of the Mac OS X operating system.

TRY IT OUT  Slide Master


2. Uncompress the MacOSXProg Chapter01.zip archive using your favorite decompression tool. (Mac OS X supports uncompressing .zip files directly in the Finder.) Inside you will find the Slide Master application, a folder of pictures called Images, and a folder of source code.
3. Run the Slide Master application by double-clicking it in Finder. The application opens an unti-tled document window.

4. Add the pictures in the Images folder to Slide Master by choosing Slide Show ➤ Add Slide. You can select all the files at once from the open panel. The images appear in a drawer to the side of the document window and the main window displays the selected image, as shown in Figure 1-2. You can use the arrow keys to change the selection.

5. Export a slideshow as a QuickTime movie by choosing File ➤ Export. Slide Master writes out a QuickTime movie and opens it with QuickTime Player.

6. Save your document by choosing File ➤ Save.

**How It Works**

Slide Master is a document-based application, which means that it provides a user interface for individual documents. In this case, documents are collections of slides that you can sift through and export as QuickTime movies. Slide Master documents can be opened, saved, and closed using the File menu. Other document-based applications also support printing, although Slide Master does not.
Much of the functionality you see here comes from Slide Master’s application framework: Cocoa. The Cocoa application framework provides the implementation for the things you see on the screen: windows, pictures, menus, buttons, and so on. Cocoa also provides support for managing the document: reading and writing document files, closing the document when its window is closed, and routing menu commands to the selected document. Finally, Cocoa provides tools for storing application data, including working with user preferences and storing lists of items in memory.

Of course Slide Master uses QuickTime to generate movie files. You are probably already familiar with QuickTime, both through QuickTime Player and through web browsers that support the display of QuickTime movies. But QuickTime also makes most, if not all, of its functionality available to applications through its framework interface.

When you save a Slide Master document, the document file contains a list of image files that are part of your slideshow, not the actual images themselves. As a result, these documents can be relatively small. Behind the scenes, Slide Master uses aliases to track these image files so that they can be found if the files are moved around on your disk. These aliases are the same aliases you can create in the Finder, although they are embedded in your document rather than saved separately to disk.

You learn more about Cocoa, QuickTime, and other technologies later in this chapter, and as you proceed through this book.

---

THE CORE OPERATING SYSTEM

The heart of Mac OS X is based on the Unix operating system. Unix was developed by AT&T in the early 1970s. In those days, computers were large and expensive, and Unix was intended as a way to share computing resources between multiple users at once. It was likely that an organization at that time could afford only one computer for all its members, and Unix provided a way for people to use that computer simultaneously without getting in each other’s way.

Over the years, Unix development has split off into many distinct “flavors” of Unix, all headed up by different groups of people, all with somewhat different goals. BSD and Linux are two such examples. Each version of Unix shares some portion of the original vision and typically implements a common set of libraries and commands.

Unix is regarded as a robust operating system whose scalability and innate networking capability make it ideal for use as a server. In fact, most of the modern-day Internet is powered by Unix servers of one version or another. It turns out that these features are also desirable in modern desktop operating systems. So it is no surprise that when Apple was seeking to modernize the original Macintosh operating system, it turned to Unix.

Mac OS X’s core operating system is a Unix flavor called Darwin. As with most Unix flavors, Darwin’s source code is freely available, allowing interested parties to see exactly how the core operating system works. Apple maintains several resources for programmers interested in Darwin, including a way for people-at-large to contribute changes and bug fixes back to Apple.
Although Mac OS X tries to hide Darwin from the average user, there are some places where the Unix command line pokes through. The most obvious example is the Terminal application, found in `/Application/Utilities`. You can use Terminal to work directly with Darwin’s command-line tools. A more subtle example includes the way you describe file locations on Mac OS X: by using a file path. A file path is a string of text that describes a file’s location.

The original Mac OS operating system abhorred file paths and tried its best to avoid them; but even so, it devised a convention for describing a path to a file. Mac OS file paths are composed of a disk volume name followed by several folder names and possibly a file, all separated by colons, as in `Macintosh HD:Applications:Utilities:Terminal.app`.

**PROGRAM, PROCESS, APPLICATION — WHAT’S THE DIFFERENCE?**

Much of the time you can use the terms *program* and *process* interchangeably to refer to something that’s *executable*. But these terms do have distinct definitions. The word *program* refers to a file on disk containing a series of computer instructions. When this file is executed (or run, launched, and so on), the computer starts processing the instructions in the file. *Process* describes the act of executing the file. To borrow an example from the kitchen, it may help to think of a program as a recipe for baking a cake, and the process as the act of baking that cake.

Ultimately, an *application* is just a program. On Mac OS X, however, programs can take many forms: simple tools typed in a command-line interface, a program you can double-click in the Finder, a plug-in file loaded by other programs, and so on. To avoid some confusion, we use the term *application* in this book to refer specifically to programs that appear in the Finder; we use the term *program* when no distinction is necessary.

Although there are places where this old convention still exists, Mac OS X mostly uses Unix’s method of describing file paths: a series of directories from the *root* directory all separated by slashes, as in `/Applications/Utilities/Terminal.app`. The root directory contains all the files and directories on a Mac OS X system and is referred to simply as `/`. The path `~/Applications` refers to a file or directory named *Applications* in the root directory. A path that begins with the root slash is called an *absolute* (or *full*) *path* because it describes a precise file location. If the root slash is not included, the path is called a *relative path* because it is relative to your current location.
NOTE If you look in /Applications/Utilities in the Finder, you might notice that there is no Terminal.app; instead there’s just a program called Terminal. By default, Finder and other applications hide file extensions such as .app and .txt from you. So the application at /Applications/Utilities/Terminal.app appears simply as Terminal. The Core OS makes no attempt to hide extensions from you; if you browse the file system using Mac OS X’s command-line interface, you can see all these extensions. You learn more about Mac OS X’s command-line interface later in this chapter.

Darwin is composed of several parts, including a kernel, a system library, and numerous commands, as illustrated in Figure 1-3.

The Kernel

The heart of a Unix operating system is its kernel. The kernel is the program that loads when the computer is first turned on and is responsible for managing all the hardware resources available to the computer. The kernel is also responsible for running the other programs on the system, scheduling process execution so that they can share the central processing unit (CPU) and other resources, and preventing one process from seeing what another process is doing. These last two responsibilities are more commonly known as preemptive multitasking and protected memory, respectively.

Because Unix prevents programs from accessing the computer hardware or other programs directly, it protects against the most common forms of system crashes. If a process misbehaves in one way or another, the system simply terminates the process and continues on its way. In other words, the misbehaving process crashes. In some operating systems, a misbehaving process can stomp all over other applications, or even break the operating system itself, before the system is able to terminate the process. As a result, poorly written programs can cause the entire computer to freeze or crash. Not so on Unix; because a process cannot modify other processes, including the kernel, there is virtually no risk of a bad process bringing down the entire operating system.

Although the kernel is responsible for accessing hardware, much of the knowledge of specific hardware details is delegated to device drivers. Device drivers are small programs that are loaded directly into the kernel. Whereas the kernel might know how to talk to hard disks, a specific device driver generally knows how to talk to specific makes and models of hard disks. This provides a way for third parties to add support for new devices without having to build it into Apple’s kernel. Mac OS X includes default drivers for talking to a wide variety of devices, so much of the time you won’t need to install separate drivers when you install new third-party hardware.