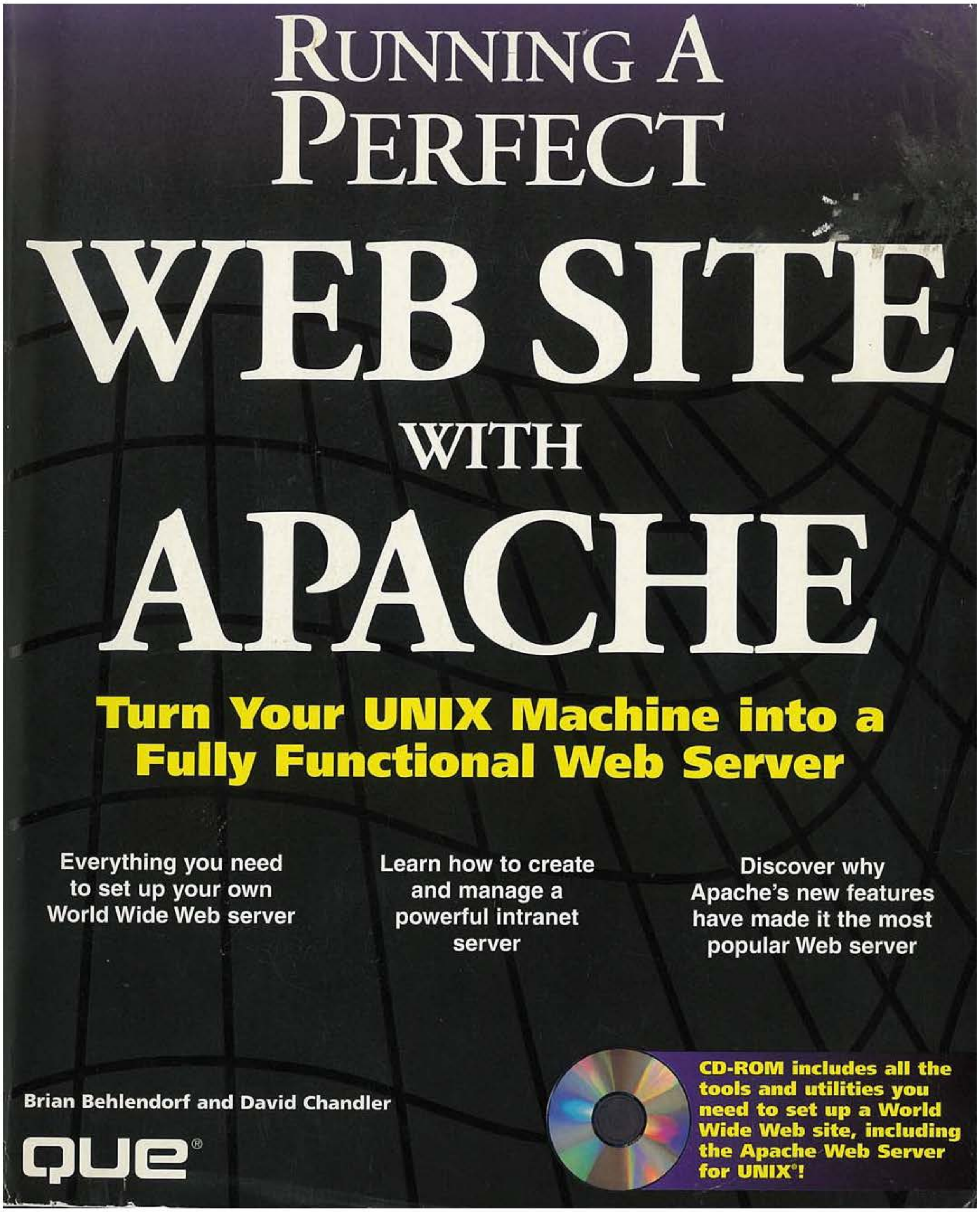


Running A Perfect Web Site with Apache

Turn Your UNIX Machine into a Fully Functional Web Server

Everything you need to set up your own World Wide Web server

Learn how to create and manage a powerful intranet server

Discover why Apache's new features have made it the most popular Web server

Brian Behlendorf and David Chandler

QUE®

CD-ROM includes all the tools and utilities you need to set up a World Wide Web site, including the Apache Web Server for UNIX®!

Here's what people are saying about *Running a Perfect Web Site with Apache...*

"*Running a Perfect Web Site with Apache* is a must-have for Webmasters of all levels of experience. It helps beginners clear the first, most difficult hurdle of installing their servers, but provides clear explanations for the more subtle points of configuration that leave even the most experienced Webmasters scratching their heads. This book is an essential piece of documentation for anyone setting up a server—from a small personal set of pages to a corporate intranet or high-powered commercial site. This book simplifies even the most cryptic aspects of the process."

Sean Welch
Webmaster
HotWired

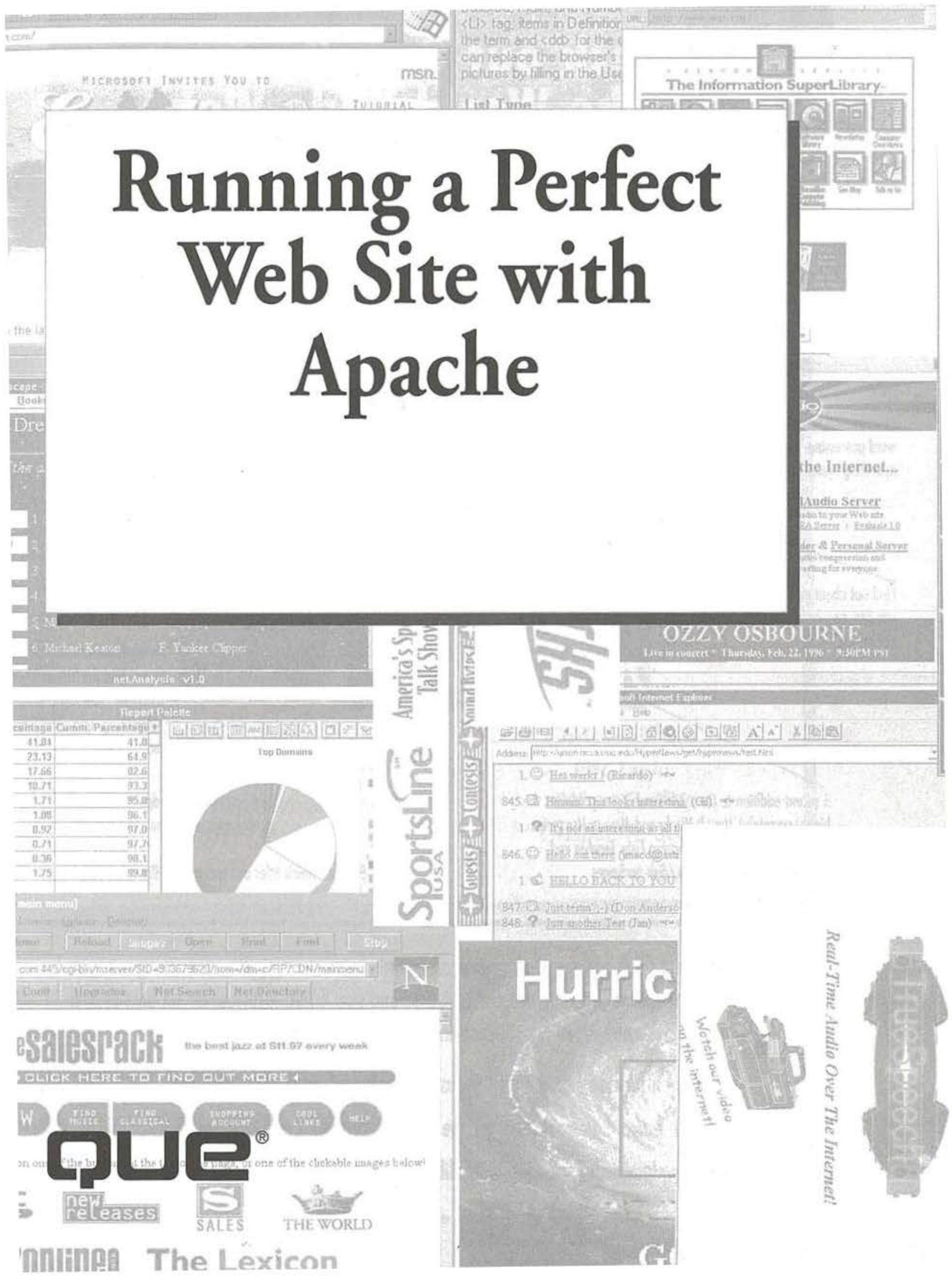
"*Running a Perfect Web Site with Apache* is a handy, no-nonsense overview for Apache users. If you're running Apache, this book deserves a spot in your toolbelt. It's the first real, comprehensive book on Apache."

Kevin Hughes
Webmaster
Enterprise Integration Technologies

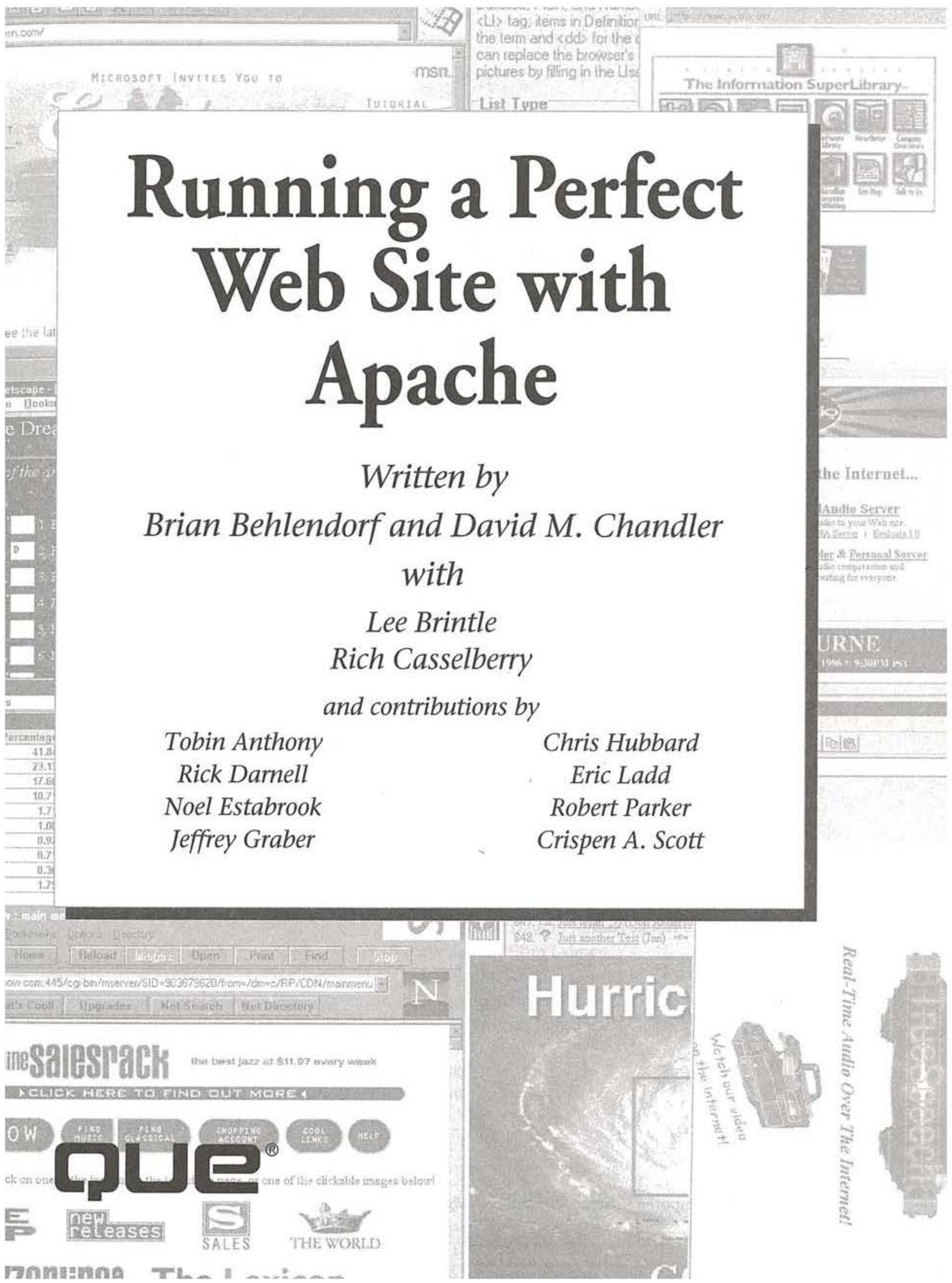
"The authors have put together an impressive collection of information for Webmasters wanting to run a perfect Web site. Everything you need, from planning to final execution and maintenance, is there. The juxtaposition of technical facts with real-world advice and the software collection on the CD makes this a valuable asset to anyone running a Web server."

Tony Sanders
Member of Technical Staff
Berkeley Software Design, Inc.

Running a Perfect Web Site with Apache


The Information SuperLibrary
OZZY OSBOURNE
SportsLine USA
America's Sports Talk Show
Hurric
the best jazz at $11.97 every week
CLICK HERE TO FIND OUT MORE
new releases
SALES
THE WORLD
The Lexicon
Real-Time Audio Over The Internet!
Watch our video on the internet!

que®



Running a Perfect Web Site with Apache

Written by

Brian Behlendorf and David M. Chandler

with

Lee Brintle
Rich Casselberry

and contributions by

Tobin Anthony	*Chris Hubbard*
Rick Darnell	*Eric Ladd*
Noel Estabrook	*Robert Parker*
Jeffrey Graber	*Crispen A. Scott*

que®

Running a Perfect Web Site with Apache

Library of Congress Catalog No.: 96-68043

ISBN: 0-7897-0745-4

98 97 96 6 5 4 3 2 1

Interpretation of the printing code: the rightmost double-digit number is the year of the book's printing; the rightmost single-digit number, the number of the book's printing. For example, a printing code of 96-1 shows that the first printing of the book occurred in 1996.

Screen reproductions in this book were created using Collage Plus from Inner Media, Inc., Hollis, NH.

Composed in *Stone Serif* and *MCPdigital* by Que Corporation

Credits

President
Roland Elgey

Publisher
Joseph B. Wikert

Editorial Services Director
Elizabeth Keaffaber

Managing Editor
Sandy Doell

Director of Marketing
Lynn E. Zingraf

Title Manager
Jim Minatel

Acquisitions Manager
Cheryl D. Willoughby

Acquisitions Editor
Doshia Stewart

Product Director
Benjamin Milstead

Production Editor
Danielle Bird

Editors
Elizabeth A. Bruns
Noelle Gasco
Patrick Kanouse
Mike La Bonne
Susan Ross Moore
Kelly Oliver
Brook Thaler

Product Marketing Manager
Kim Margolius

Technical Editors
Kyle Amon
Todd Brown
Paul Ehteridge
Faisal Jawdat
Greg Newman
Paolo Papalardo
Mark Surfas

Technical Specialist
Nadeem Muhammed

Acquisitions Coordinator
Jane Brownlow

Operations Coordinator
Patricia J. Brooks

Editorial Assistant
Andrea Duvall

Book Designer
Ruth Harvey

Cover Designer
Dan Armstrong

Production Team
Stephen Adams, Jason Carr,
Anne Dickerson, Joan Evan,
Jessica Ford, Trey Frank, Jason Hand,
Daniel Harris, Damon Jordan,
Daryl Kessler, Clint Lahnen,
Bob LaRoche, Michelle Lee,
Julie Quinn, Laura Robbins,
Bobbi Satterfield, Jody York

Indexer
Tim Tate

About the Authors

Brian Behlendorf is a technologist and entrepreneur. As Chief Technology Officer at Organic Online, he directs the investigation and implementation of new functionality for Organic clients, and plays a strong role in the Internet public standards and software development communities. Predestined for the computer age when his parents met as employees at IBM, Behlendorf studied computer science at Berkeley where his interest in the Internet was piqued in 1991. His interests in music and dance culture led to the establishment of one of the first nonacademic Web sites in early 1993, a site which has since grown into a collaborative publishing effort known as Hyperreal. In the fall of 1993 he set up a Web server for *Wired Magazine*, at first served by a 486/66 over a 14.4 connection. This served as the prototype for *HotWired*, where Behlendorf served as Chief Engineer. Behlendorf left in April 1995 to work for Organic Online, a startup he co-founded and had been moonlighting at since the fall of 1993. Behlendorf is also a co-founder and maintainer of the "www-vrml" mailing list, which served as the basis for the VRML public development effort, and he hosts, administrates, and contributes to the Apache development effort.

David M. Chandler is a World Wide Web enthusiast in Cedar Rapids, Iowa. He currently runs Internet At Work, a Midwestern consulting firm specializing in network security, Intranet development, and advanced Web applications. Chandler previously managed Web servers for the Collins Avionics & Communications Division of Rockwell International. He has programmed computers since 1982 when he received a TI-99/4A as a gift. Chandler holds a degree in Electrical Engineering from the University of Kansas. When he's not at his computer, he enjoys the mountains and flying as a private pilot. You can reach Chandler via e-mail at **chandler@iwork.net** or on the Web at **http://www.iwork.net/~chandler**.

Lee Brintle is the president and founder of Leepfrog Technologies, Inc. (**http://www.leepfrog.com**), located in Iowa City which specializes in custom database programming for the Web and providing direct access to the Internet. He has written a variety of Web browsers and servers, including a cable television Web browser. Lee has developed a trusted third-party user authentication security system for use on the Web, and also has tinkered with parallel processing and distributed databases. He was introduced to the distributed security and database arena by a too-healthy dose of the original MUDs. In that laughingly brief period he calls "free time," he plays Wally-Ball, helps administrate the ISCABBS (**telnet://bbs.isca.uiowa.edu**), enjoys role-playing, and the too-infrequent late-night card game. Lee received a BS in Computer Science from the University of Iowa, and can be reached at **lbrintle@leepfrog.com**. Drop him a line; he enjoys social e-mail.

Rich Casselberry is currently working as the Network Manager for Current Technology in Durham NH (**http://www.curtech.com/**). He lives in Southern Maine with his fiancée Kandi, two cats (Mitz and Zeb) and a miniature dachshund (Prince). Prior to working at Current Technology, Rich worked as a UNIX System Specialist for Cabletron Systems for four and a half years. It was here that he first learned about the Internet and networking. Rich graduated from New Hampshire Technical College in 1992 with an Associates degree in Computer Engineering Technology.

Tobin Anthony holds a Ph.D. in aerospace engineering, but has been tinkering with computers for over 18 years specializing in the UNIX and MacOS environments. A strict vegetarian, devout Roman Catholic, and lapsed private pilot, Anthony spends what little spare time he has with his wife Sharon and three children, Michelle, Austin, and Evan. Anthony works as a spacecraft control systems engineer at NASA's Goddard Space Flight Center in Greenbelt, MD. E-mail and Web stops are welcome at **tobin@pobox.com** and **http://pobox.com/~tobin**.

Rick Darnell is a midwest native who now lives with his wife and two daughters in Missoula, MT. He began his career in print at a small weekly newspaper after graduating from Kansas State University with a degree in broadcasting. While spending time as a freelance journalist and writer, Rick has seen the full gamut of personal computers since starting out with a Radio Shack Model I in the late 1970s. Darnell serves as a volunteer firefighter and member of a regional hazardous materials response team.

Noel Estabrook is currently a faculty member of the College of Education at Michigan State University after having obtained degrees in Psychology, Education, and Instructional Technology. He is heavily involved in delivering Internet Training and technical support to educators, professionals and laymen. In addition to writing, he also runs his own training business part-time.

Most recently, Estabrook has been involved in authoring on the Web and co-authored Que's *Using UseNet Newsgroups* and *Using FTP*. His e-mail address is **noele@msu.edu**.

Jeffrey Graber is a technical consultant for Compuware Corp. at their Washington, DC branch. There, Graber is responsible for the management and development of Internet services and business. In addition, he manages a major client Internet site at the National Science Foundation. Graber has been involved in Web development almost since it began. He has developed several sites for other government agencies. Graber has spoken on the topic of the WWW at the 2nd and 4th International WWW conferences (sponsored by the official W3.org) as well as MecklerMedia's WebDev Conference. He is also founder and chair of the DC area Internet Developers Association (**http: //www.shirenet.com/dcida/**). Over the years, Graber has taught numerous computer science courses and given presentation at numerous conferences.

Chris Hubbard is an Internet veteran and technical supervisor with Questar Microsystems, responsible for documentation, testing, and implementation of WebQuest products. His broad professional experience and a wide range of outside interests uniquely qualify him to discuss the World Wide Web in general, and WebQuest in particular. A member of the HTML Writers Guild, Hubbard has consulted and built HTML pages for numerous high-profile clients. For recreation, Hubbard surfs the Web discovering and correcting defective Web pages. You can e-mail him at **chris.hubbard@questar.com**.

Eric Ladd is a "math teacher turned Internet teacher" and currently works as Internet training coordinator for Walcoff and Associates, Inc., a communications and technology firm in Fairfax, VA. He holds B.S. and M.S. degrees in Mathematics from Rensselaer Polytechnic Institute in Troy, New York, where he also taught calculus, linear algebra, and differential equations for six years. Rensselaer also taught Ladd a thing or two about running a newspaper, engineering late-night angst radio shows, and managing a bar. Away from work and writing, he enjoys running, ice hockey, and spending far too much time playing with his new computer.

Robert Parker first caught the writing bug in the machine room of the Yale Computer Science Facility, tending mainframe systems equipped with an awesome 256K of core memory. He has crafted technical publications for such firms as Compu-Teach, DAK Industries, and most recently Quarterdeck

Corporation; scripted and narrated educational videotapes, radio theater, and commercials; and is currently on the faculty at Glendale College, where both he and his father teach courses in the same division. Parker is currently completing his doctorate in conducting, and hopes someday to retire from a successful career as a beloved professor of music.

Crispen A. Scott is an independent hardware and software engineering consultant who lists among his accomplishments such varied projects as the digital anti-skid braking system for the B-2 Stealth Bomber, various Windows drivers and applications, and embedded control systems for the medical and industrial control fields. Scott is currently developing home pages, CGI applications, and establishing Web sites for Chicago-based customers of his Commercial, Residential and Institutional Software Corporation. In addition, Scott also lectures, conducts seminars, and presents training reviews nationally. Scott is a continuing, lifelong student who barely remembers graduation from the University of Tennessee, and ardently follows his favorite sports: football and lacrosse. In his "spare" time, Scott is continuing to polish his writing skills in both the poetry and science fiction genres. Scott can currently be reached at **crisin19@starnetinc.com**, and, in the near future, at his Web site. Search for "Chicago Developments" using your favorite search engine.

We'd Like To Hear From You!

As part of our continuing effort to produce books of the highest possible quality, Que would like to hear your comments. To stay competitive, we *really* want you, as a computer book reader and user, to let us know what you like or dislike most about this book or other Que products.

You can mail comments, ideas, or suggestions for improving future editions to the address below, or send us a fax at (317) 581-4663. For the online inclined, Macmillan Computer Publishing has a forum on CompuServe (type **GO QUEBOOKS** at any prompt) through which our staff and authors are available for questions and comments. The address of our Internet site is **http://www.mcp.com** (World Wide Web).

In addition to exploring our forum, please feel free to contact me personally to discuss your opinions of this book: I'm **102121,1324** on CompuServe, and **bmilstead.que.mcp.com** on the Internet.

Thanks in advance—your comments will help us to continue publishing the best books available on computer topics in today's market.

Benjamin Milstead
Product Director
Que Corporation
201 W. 103rd Street
Indianapolis, Indiana 46290
USA

Contents at a Glance

Contents

Introduction

In the spring of 1995, I was copied on a piece of e-mail from a fellow Webmaster in France to three others who, along with myself, had spent some time modifying the Web server that had been released by the National Center for Supercomputing Applications (NCSA). We had each submitted our "patches" to NCSA, but because of a team exodus to a certain start-up software company in Mountain View, NCSA had few resources to continue development of what was then, by far, the most popular Web server out there. Thus, the five of us decided we had the time and interest to start our own Web server development effort, branching off from the development paths taken by the NCSA team. Since most of us were Webmasters for content providers or educational or research institutions, we had no desire or time to make this a commercial venture. Instead, we realized we had nothing to lose by helping each other. I suggested the name "Apache," which was a code name for my own vaporware server project, and only later realized what a great pun it made ("A patchy Web server"). And from that, the project was born.

The Apache development team quickly grew to well over 40 people, with a core development team of roughly 12 people. At one point, in the summer of 1995, the internals of the server were significantly reorganized by Rob Thau, a graduate student at the Massachusetts Institute of Technology, which produced not only a tremendous improvement in speed, but also a highly generalized API (Application Programming Interface) to the internals of the server. This API has made it possible to enhance functionality in a very modular way—not only can you add features without having to modify existing files, but third party developers can create new modules without requiring that they be "officially" rolled into the Apache releases.

Today (as of the April 1st Netcraft Survey, at **http://www.netcraft.co.uk/Survey/Reports/**) Apache is the single most-used Web server on the Internet, with more installations than all other commercial web servers combined. Public support exists on the UseNet newsgroup **comp. infosystems.www.servers.unix**, and several companies provide commercial support for Apache.

The Apache project has also spawned some parallel efforts. The Apache-SSL project is an integration of the Secure Sockets Layer specification. SSL is the security protocol that Netscape's browsers and servers implement for secure communications. It is also supported by Microsoft's Internet Explorer and numerous other browsers. Both are covered in this book.

What This Book Is

This book is designed for those who are new to setting up a Web server on a UNIX platform. The featured Web server is Apache, though many of the subjects covered are applicable to other Web servers. The basics of constructing a Web site are also covered—authoring HTML (including HTML extensions), CGI scripting, integrating new media types, installing and running a search engine, usage statistics analysis programs, database interfaces, and more. Finally, it includes a CD-ROM with the Apache server source code and binaries for a variety of platforms.

These chapters include:

- Chapter 1, "The State of the World Wide Web." This chapter examines where the Web came from, how the Web is changing today, and why you should have a presence on it.
- Chapter 2, "Introduction to Web Servers." The basic terminology of Web serving is discussed; details about the HyperText Transfer Protocol (HTTP) and the types of data one can put on a Web server are also discussed.
- Chapter 3, "Setting Up a Web Presence." In this chapter, you will learn about hosting a site on your own machine versus leasing a site on another server, how to choose a hardware and software combination for your server, and what sort of Internet connectivity you should get.
- Chapter 4, "Getting Started with Apache." This chapter takes you through the basic steps of compiling, installing, and configuring the Web server.

- Chapter 5, "Apache Configuration." This chapter covers all the aspects of complete Apache configuration, such as MIME type assignment, directory indexing, server side includes, internal imagemap handling, cookies, configurable logging, content negotiation, access control, virtual hosts, and custom error messages.
- Chapter 6, "Managing an Internet Web Server." Real nuts-and-bolts issues regarding server performance and security are covered here. Included within are tips for using Apache on very heavily loaded Web servers.
- Chapter 7, "Creating and Managing an Intranet." This chapter describes how to maximize the effectiveness of your internal server. It explains how you can manage server content, provide useful features, and protect your internal network from hostile access.
- Chapter 8, "Basic HTML: Understanding Hypertext." In this chapter, you will learn the basics of authoring HTML. This includes distinguishing between logical and presentational tags, and how to embed images and hyperlinks on a page.
- Chapter 9, "HTML 2.0, HTML 3.0, and Extensions." This chapter covers the complete HTML 2.0 specification, plus other extensions which have been proposed for standardization, and also some browser-specific extensions.
- Chapter 10, "HTML Editors and Tools." This chapter covers authoring tools for both Windows and UNIX, and includes sections on HTML filters and other types of algorithmic HTML generation, as well as HTML validation programs.
- Chapter 11, "Graphics and Imagemaps." This chapter covers the basics of using images on your Web sites, including pointers to various image manipulation tools. This chapter includes discussion of both client-side and server-side imagemaps, as well as tools to create them on Windows and UNIX.
- Chapter 12, "HTML Forms." The essentials of HTML form programming are covered, with descriptions of the different types of form widgets, an explanation of the two types of form data submission, and more.
- Chapter 13, "CGI Scripts and Server APIs." This chapter explains in detail how to use each of these three features to create dynamic, compelling Web sites.
- Chapter 14, "More Scripting Options." This chapter explores JavaScript and Visual Basic Script, two new client-side scripting languages for dramatically extending the impact of Web pages.

- Chapter 15, "Search Engines and Annotation Systems." This chapter goes over the basics of setting up simple, shareware search engines for your pages, as well as systems for Web-based collaboration and annotation.
- Chapter 16, "Usage Statistics and Maintaining HTML." This chapter tells you the best way to handle your voluminous logfile data, with pointers to logfile analysis programs. Also included are pointers to programs which help ensure the integrity of your collection of documents.
- Chapter 17, "Database Access and Applications Integration." This chapter discussed in general terms the models for integrating databases with the Web. Sample applications are given, as are lots of tips on creating such applications.
- Chapter 18, "Financial Transactions." This chapter briefly covers the history of online commerce, security protocols, digital cash, and how to become a merchant on the Web.
- Chapter 19, "Interactive and Live Applications." Discussion about integrating audio, video and virtual reality into your Web site is covered in this chapter.

What This Book Is Not

While this book covers a lot of basics regarding setting up a Web server, both from a software/configuration perspective and from a content/HTML perspective, this book is not a UNIX tutorial book. This book presumes a basic knowlege of UNIX, including UNIX terminology such as what "cron" is or how UNIX "daemons" work.

Conventions Used in This Book

Certain conventions are used in *Running a Perfect Web Site with Apache* to help you absorb the ideas easily.

Tip

Tips suggest easier or alternate methods of executing a procedure or approaching a task.

New terms are introduced in *italics* and text you type appears in **boldface**. World Wide Web URLs (essentially document addresses) are also presented in **boldface**.

Note

This paragraph format indicates additional information that might help you avoid problems, or that might be considered when using the described features.

Caution

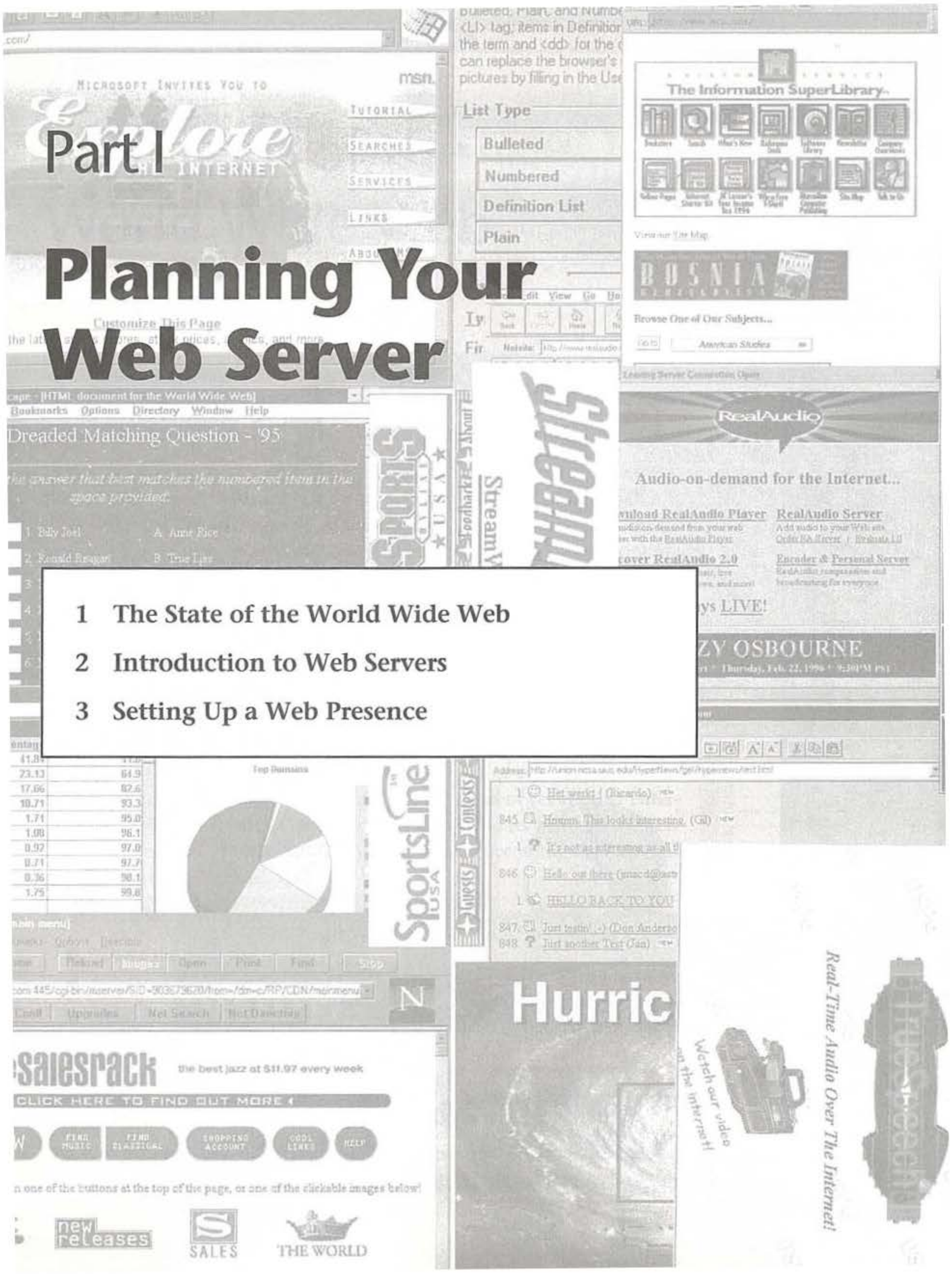
This paragraph format warns you of hazardous procedures.

▶ See "Section Title," p. xx

▶ See "Chapter Title," p. xx

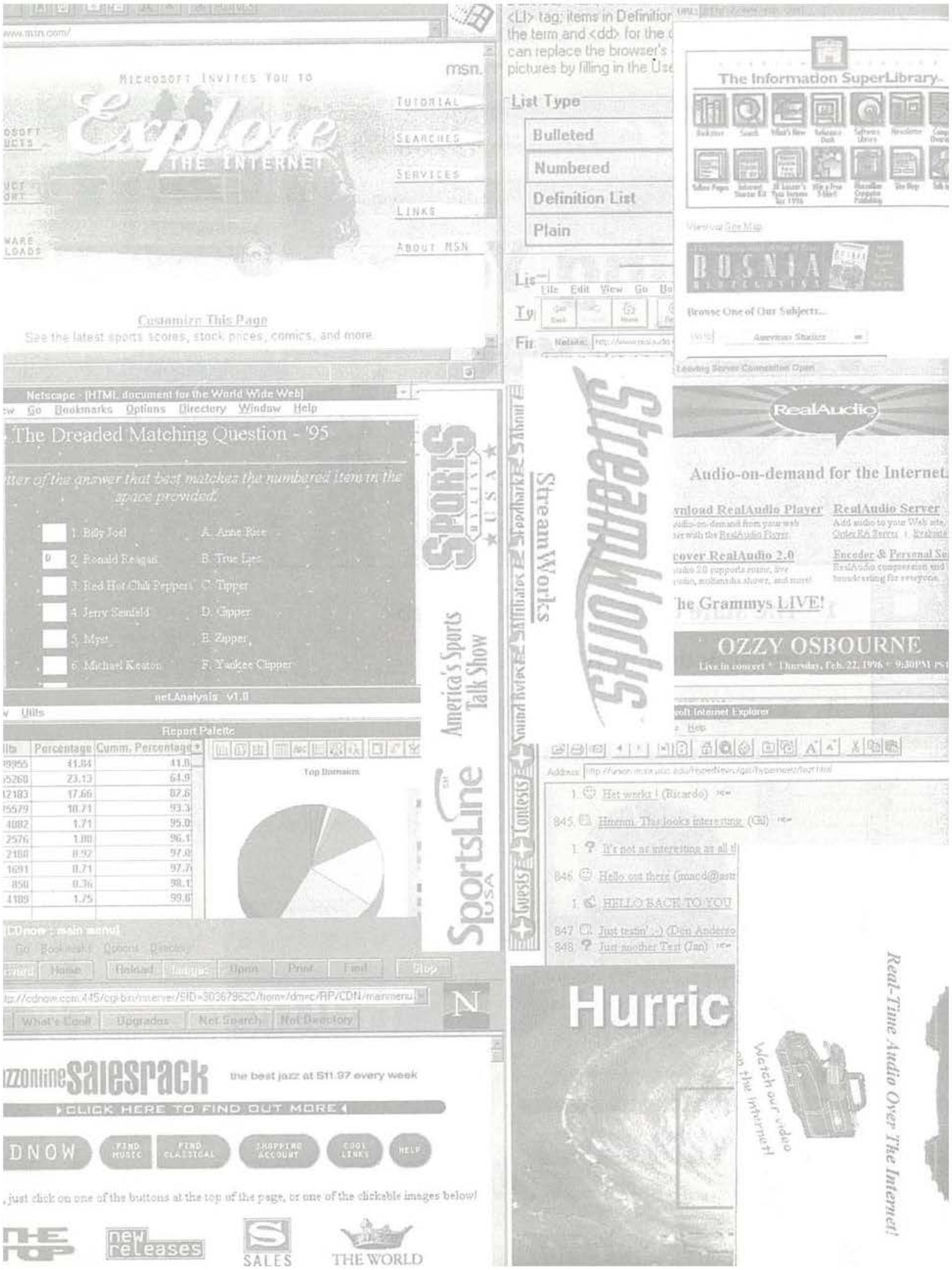
Running a Perfect Web Site with Apache uses marginal cross references so you can quickly find related information in the book. These are listed by section or chapter title and page number for convenience.

Throughout the book, you'll also see the WebmasterCD icon (shown beside this paragraph) in the margins. Where you see this icon, the text is discussing software or a document on the WebmasterCD that is included with this book.


Microsoft Invites You to Explore the Internet
Tutorial
Searches
Services
Links
List Type
Bulleted
Numbered
Definition List
Plain
The Information SuperLibrary
Bosnia
Browse One of Our Subjects...
RealAudio
Audio-on-demand for the Internet...
Dreaded Matching Question - '95
SportsLine USA
StreamWorks
Hurric
salesrack
the best jazz at $11.97 every week
CLICK HERE TO FIND OUT MORE
new releases
SALES
THE WORLD
Real-Time Audio Over The Internet!
Watch our video on the internet!

Part I

Planning Your Web Server


Microsoft Invites You To Explore The Internet
msn.
Tutorial
Searches
Services
Links
About MSN
Customize This Page
See the latest sports scores, stock prices, comics, and more.
List Type
Bulleted
Numbered
Definition List
Plain
The Information SuperLibrary
Bosnia
Browse One of Our Subjects...
Netscape - [HTML document for the World Wide Web]
The Dreaded Matching Question - '95
1. Billy Joel
2. Ronald Reagan
3. Red Hot Chili Peppers
4. Jerry Seinfeld
5. Myst
6. Michael Keaton
A. Anne Rice
B. True Lies
C. Tipper
D. Gipper
E. Zipper
F. Yankee Clipper
net.Analysis v1.0
Report Palette
Percentage
Cumm. Percentage
Top Domains
Sports Online USA
America's Sports Talk Show
SportsLine USA
StreamWorks
RealAudio
Audio-on-demand for the Internet
Download RealAudio Player
RealAudio Server
Discover RealAudio 2.0
Encoder & Personal Server
The Grammys LIVE!
OZZY OSBOURNE
Live in concert * Thursday, Feb. 22, 1996 * 9:30PM PST
Microsoft Internet Explorer
Het werkt ! (Ricardo)
Hmmm. This looks interesting (Gil)
It's not as interesting as all that
Hello out there
HELLO BACK TO YOU
Just testin'
Just another Test (Jan)
CDnow : main menu
What's Cool!
Upgrades
Net Search
Net Directory
JazzOnline SalesRack the best jazz at $11.97 every week
CLICK HERE TO FIND OUT MORE
CDNOW
FIND MUSIC
FIND CLASSICAL
SHOPPING ACCOUNT
COOL LINKS
HELP
just click on one of the buttons at the top of the page, or one of the clickable images below!
new releases
SALES
THE WORLD
Hurric
Watch our video on the internet!
Real-Time Audio Over The Internet!

CHAPTER 1

The State of the World Wide Web

In the six years since Tim Berners-Lee unleashed his graphical NeXT application, "WWW.app," upon an unsuspecting public, the World Wide Web has grown into *the* standard networked information infrastructure. (See **http://www.w3.org/pub/WWW/History.html**.) Its graphical interface and hypertext capabilities have caught the fancy of individuals and the media like no other Internet tool in history. Businesses, schools, government and non-profit organizations, and millions of individuals are flocking to the Web to promote themselves and their products in front of an audience spanning the entire planet. Millions more are using the Web on a daily basis as a tool to conduct business, get informed, be entertained, and even form virtual communities.

It's difficult to watch a sporting event, a commercial, or even the news without seeing that increasingly familiar *http://*— telling us of yet another enterprise on the Web. Because of the Web's popularity and its cost-effectiveness as a marketing tool, the World Wide Web is quickly becoming the electronic marketplace of the decade.

In this chapter, you learn:

- Where the Web has been
- How Web usage is changing
- How you might do business on the Web
- Where the Web is going

The Scope of the World Wide Web

The Web is now accessible in over 200 countries on all seven continents, and its information and services range from the esoteric to the absurd. As of this writing, the Alta Vista search engine reports that its robot has indexed 21 million Web pages, and the Netcraft Web Server Survey reports 135,000 different Web servers in its database. Web sites are maintained by universities, companies, public institutions, states, cities, and even high schools. Even McMurdo Station in Antarctica has a Web site. A number of powerful search engines (like Alta Vista) and catalog sites (like Yahoo) allow rapid information location and retrieval, making the Web the ultimate tool for research, interactive entertainment, and even advertising. For more information, see **http://www.w3.org/pub/WWW/History.html**, **http://www.altavista.digital.com**, **http://www.netcraft.co.uk/Survey/Reports**, and **http://www.mcmurdo.gov/**.

One of several reasons why the World Wide Web rose to such prominence was because the underlying technology, HyperText Transfer Protocol (HTTP) and the Hypertext Markup Language (HTML), were "free." (See **http://www.w3.org/pub/WWW/Protocols/** and **http://www.w3.org/pub/WWW/MarkUp/**.) Anyone could write an HTML viewer or HTTP application without having to pay anyone royalties or licensing fees. This also made it easy for the Web to be platform independent—a Microsoft Windows Web browser has no problem talking to a UNIX Web server, and a Windows Web browser can display the HTML pages exactly the same as a Macintosh Web browser or UNIX Web browser does.

Furthermore, because HTML describes documents at a structural level rather than a "pretty picture" level, HTML is extremely portable between platforms of different capabilities. In other words, it is possible to write a well-formed HTML page that looks equally attractive on a graphical Web browser like Mosaic as it does on a text-only browser like Lynx. In fact, some companies are building audio-only Web browsers for the visually impaired, and HTML's structural markup makes this not only possible, but quite elegant.

Note

The Web is not free from forward compatibility problems—HTML has lacked a formal evolutionary strategy, and the one in HTTP (content negotiation) has not been widely implemented. Thus, many companies use new HTML tags, some of which cause older browsers to act inelegantly. This means you will often see sites that say "You must use Browser X to view these pages," which is more often a statement on the page author's capabilities than your browser's capabilities, since a site designed with care can be elegant for all browsers. Web standards coordination is the responsibility of the World Wide Web Consortium. See **http://www.w3.org/**.

Note

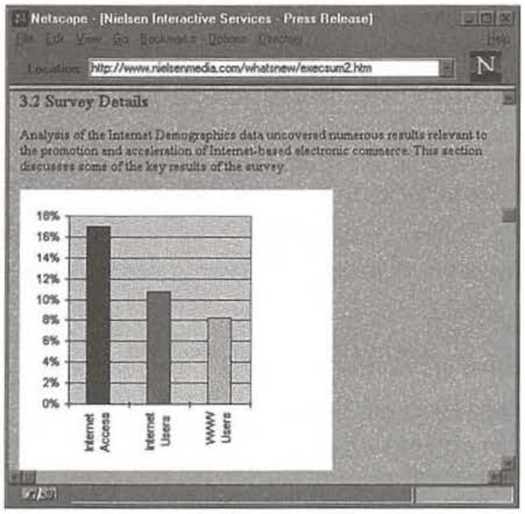
There are many good sources for Internet usage statistics. One is located at Matrix Information and Directory Service (**http://www.mids.org/**). There are still statistics available at NSFNet (**http://nis.nsf.net/nsfnet/statistics/**), though the project was dismantled in April 1995.

The Web's Phenomenal Growth

In January 1993, there were only 50 known Web servers in existence. Today, the Web has become the largest source of traffic on the Internet. Table 1.1 and figure 1.1 show the growth of the Web relative to other Internet services on the Internet. You can find more details about the data shown in figure 1.1 at **http://www.nielsenmedia.com/demo.htm**. As was mentioned earlier, Web servers are in almost every developed country in the world.

Note

If you're interested, a list of all registered servers is available from **http://www.w3.org/hypertext/DataSources/WWW/Servers.html**.



Netscape - [Nielsen Interactive Services - Press Release]

Location: http://www.nielsenmedia.com/whatsnew/execsum2.htm

3.2 Survey Details

Analysis of the Internet Demographics data uncovered numerous results relevant to the promotion and acceleration of Internet-based electronic commerce. This section discusses some of the key results of the survey.

Fig. 1.1
This data was taken by a Nielsen survey conducted on Web usage.

Table 1.1 Growth of World Wide Web Traffic Percentage of Total Byte Traffic Change on the NSF Backbone in a Four-Month Period

Service Name	Port	Rank %	Pkts	Rank %	Bytes
Beginning of Four-Month Period					
ftp-data	20	1	18.758	1	30.251
www	80	2	13.122	2	17.693
telnet	23	3	10.357	6	3.715
End of Four-Month Period					
www	80	1	21.443	1	26.250
ftp-data	20	2	14.023	2	21.535
nntp	119	3	8.119	3	8.657

As table 1.1 illustrates, the World Wide Web already comprises more traffic than any other Internet function. Despite the fact that the Web has been in operation for several years now, it is still able to grow at a rate of almost 20 percent a year.

Note

The World Wide Web traffic in table 1.1 reflects only connections to World Wide Web servers. Web browsers can also connect to FTP (File Transfer Protocol), Gopher, and other types of servers.

But do we know anything else about who is actually on the Web? The Nielsen study mentioned in figure 1.1 tells us quite a bit about who is on the Net. Among the findings:

- 56 percent of WWW users were between 25-44 years old.
- 64.5 percent of users were male.
- 88 percent had at least some college education.

The Proliferation of Web Server Software

One of the barometers of the growth of the Web is the incredible number of different products out there, particularly in the Web server field. Two years ago, there were roughly half a dozen Web servers, all products of research groups or experimentation. Now there are dozens and dozens of different

Web server products out there. Many are commercial, but the free Web servers (like Apache) continue to be developed, supported, and very widely used. The chart available from WebCompare shows over 45 actively supported Web servers. For more information, see **http://www.webcompare.com/server-main.html**.

Why You Want To Be on the WWW

There is little doubt that there are some huge benefits to being on the Web today. In business, numbers speak volumes. There is no doubt that the Web has them. Web users are generally educated, professional, middle to upper-middle class people who want to use the Web for information, research, fun, and even for purchasing products.

Like every new major medium that has come before, the Web has distinct, inherent, and unique advantages to other media. Instant access to information resources—also known as the "pull" model compared to television's "push" model—is one of the most significant advantages. Many commercial sites report thousands of visitors within the first days of operation. Electronic malls are appearing everywhere, and financial transactions are becoming safer all the time.

The best thing about the Web, of course, is that it isn't going to go away. It's only going to get bigger and bigger. Connections will get faster, computers will get better, programming will get slicker, and access will get better. Most importantly, more people will getting online.

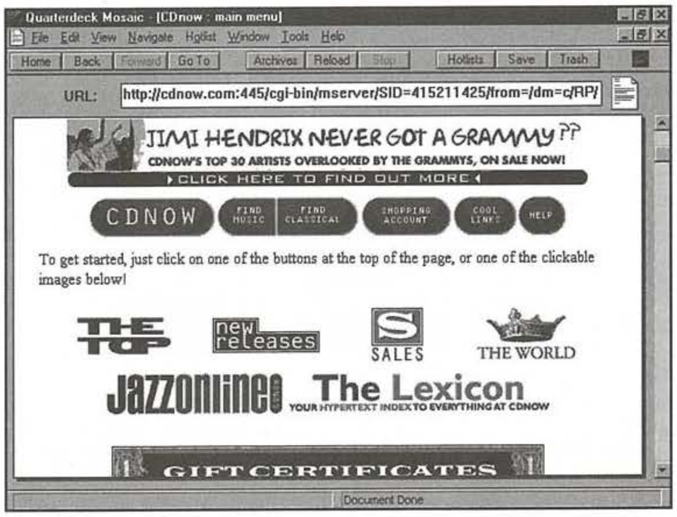
So, you're convinced. The Web is the greatest thing since tail fins, right? Well, almost. There are definitely a lot of advantages to doing business on the Net (as well as some pitfalls), and it will definitely be helpful to know about some of them. Who's out there? What are they like? Are they ready to buy your product? Who's doing business on the Web?

Some of those questions are easy to answer. We know that there are a lot of educated professionals on the Internet. We also know that many of them are involved in education, research, and industry. It's time to dig a little deeper and find out a little bit about how the Web can serve businesses and consumers of all kinds.

More Than a High-Tech Billboard: Your Name on the Web

The Web has proven that people will come—in droves—to the Internet if it's easy to use and accessible. For those in business for profit, being on the Web, referred to as "having a presence on the Web," usually serves two main purposes.

One reason many business get on the Web is to sell their product. Many companies, such as CD-Now, are focused on marketing a specific product or class of products (see fig. 1.2). They are not as concerned about establishing a brand identity or giving information away. They offer a product, and hope people will buy it. For more information about CD-Now, see **http://www.cdnow.com/**.


Quarterdeck Mosaic - [CDnow : main menu]
File Edit View Navigate Hotlist Window Tools Help
Home Back Forward Go To Archives Reload Stop Hotlists Save Trash
URL: http://cdnow.com:445/cgi-bin/mserver/SID=415211425/from=/dm=c/RP/
JIMI HENDRIX NEVER GOT A GRAMMY ??
CDNOW'S TOP 30 ARTISTS OVERLOOKED BY THE GRAMMYS, ON SALE NOW!
CLICK HERE TO FIND OUT MORE
CDNOW
FIND MUSIC
FIND CLASSICAL
SHOPPING ACCOUNT
COOL LINKS
HELP
THE TOP
new releases
SALES
THE WORLD
Jazzonline
The Lexicon
YOUR HYPERTEXT INDEX TO EVERYTHING AT CDNOW
GIFT CERTIFICATES
Document Done

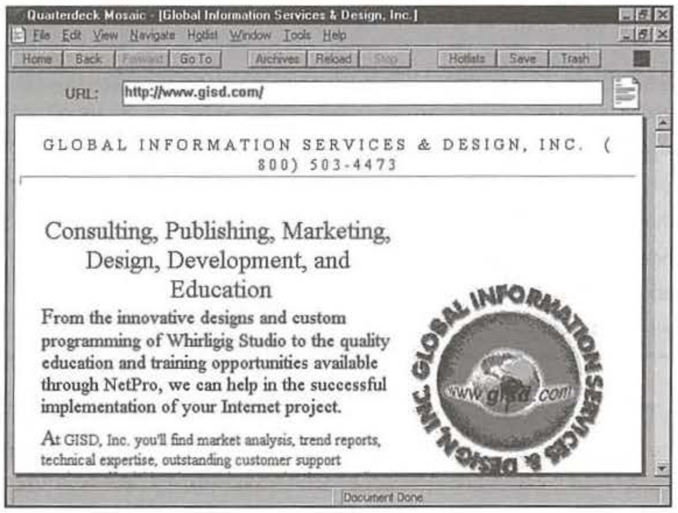
To get started, just click on one of the buttons at the top of the page, or one of the clickable images below!

Fig. 1.2
CD-Now is a Web-based company that sells music on the Internet.

Another big motivation for businesses to get on the Net is for advertising purposes. These companies want to further the equity they have established with their name brand, whether it be local, statewide, or national (see figures 1.3 and 1.4).

Companies who use the Web for this purpose are often service-based businesses, such as Global Information Services & Design in Michigan. Still others offer products that are just very difficult to sell over the Web and for whom product familiarity is of utmost importance. Again, these types of companies are generally national in nature or are service providers of some sort.

Table 1.2 shows what businesses reported when asked what they used the Web for. As you can see, many of the functions already being employed through other media are being utilized on the Web even today.


Quarterdeck Mosaic - [Global Information Services & Design, Inc.]
File Edit View Navigate Hotlist Window Tools Help
Home Back Forward Go To Archives Reload Stop Hotlists Save Trash
URL: http://www.gisd.com/
Document Done
GLOBAL INFORMATION SERVICES & DESIGN, INC.
www.gisd.com

GLOBAL INFORMATION SERVICES & DESIGN, INC. (800) 503-4473

Consulting, Publishing, Marketing, Design, Development, and Education

From the innovative designs and custom programming of Whirligig Studio to the quality education and training opportunities available through NetPro, we can help in the successful implementation of your Internet project.

At GISD, Inc. you'll find market analysis, trend reports, technical expertise, outstanding customer support

Fig. 1.3
GISD is a Michigan company that provides Internet training and other services.

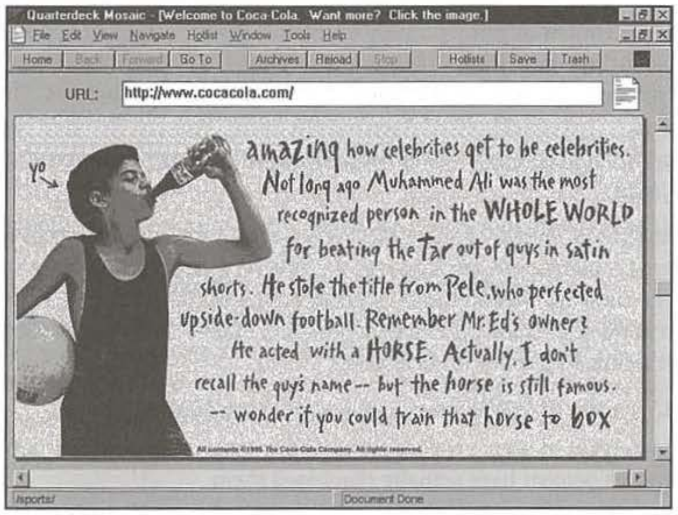

Quarterdeck Mosaic - [Welcome to Coca Cola. Want more? Click the image.]
File Edit View Navigate Hotlist Window Tools Help
Home Back Forward Go To Archives Reload Stop Hotlists Save Trash
URL: http://www.cocacola.com/
yo
amazing how celebrities get to be celebrities. Not long ago Muhammed Ali was the most recognized person in the WHOLE WORLD for beating the tar out of guys in satin shorts. He stole the title from Pele, who perfected upside-down football. Remember Mr. Ed's owner? He acted with a HORSE. Actually, I don't recall the guy's name -- but the horse is still famous. -- wonder if you could train that horse to box
All contents ©1995 The Coca-Cola Company. All rights reserved.
/sports/
Document Done

Fig. 1.4
Even Coca-Cola, known the world around, advertises on the Web.

Table 1.2 Business Usage of the Web

Percent of WWW Business Users Who Have Used It to...	
Collaborate with others	54%
Publish information	33%

(continues)

Table 1.2 Continued

Percent of WWW Business Users Who Have Used It to...	
Gather information	77%
Research competitors	46%
Sell products or services	13%
Purchase products or services	23%
Provide customer service and support	38%
Communicate internally	44%
Provide vendor support and communications	50%

Web Demographics: Who Is Your Audience?

It's time to talk about a few specifics about who exactly is on the Internet, and whether they actually buy what a business has to sell. We're going to return to the Nielsen survey mentioned in figure 1.1 for some more statistics that were gathered from 280,000 telephone interviews nationwide.

If you're interested in getting a copy of the full report, The Final Report is available for purchase from CommerceNet (phone: 415-617-8790; e-mail: **survey@commerce.net**) and Nielsen Media Research (phone: 813-738-3125; e-mail: **interactive@nielsenmedia.com**). You can also find summary information on the Web at **http://www.nielsenmedia.com/demo.htm**.

So what did the Nielsen survey find? Well, over 2.5 million Americans have purchased products and services over the WWW. Again, as with all other numbers, these too will continue to grow. Earlier in the chapter, you were given a glimpse of some general demographics of users. The survey showed specific results important for businesses.

- 25 percent of WWW users had incomes over $80,000 a year.
- 55 percent of users have used the Web to research products or services and 14 percent have actually purchased them.
- There was a user base of 18 million Web users in the United States and Canada
- Total time spent on the Internet in the U.S. and Canada was actually equivalent to the total time spent watching rented video-tapes!

Cautionary Note

As rosy a picture as the Web paints, there are some downsides. The biggest is that any Internet survey or usage statistics fail to take into account the still large majority of people who do not access the Internet. Even with 18 million users using the Web on a regular basis according to the Nielsen report, that still leaves over 250 million people in the U.S. and Canada who still aren't on, not to mention the billions of people around the world who have yet to get online, where the rate of Internet penetration is even less. See **http://www.census.gov/** and **http://www.statcan.ca/Documents/English/Faq/Pop/pop.htm**.

The Internet is not yet (nor will it likely ever be) a panacea for everyone's advertising and marketing woes. It's another tool that can, and should, be utilized along with other more traditional media.

What Will the Web Be Like Tomorrow? Next Week?

Now that you have a better idea of where the Web has been and where it is, wouldn't you like to know where it's going? Wouldn't we all? A popular TV commercial shows all sorts of fanciful futuristic gadgets as being "the future." The commercial ends with the conclusion that each possibility is likely and it's sheer guess-work as to what the future will actually hold. To an extent, that commercial is right, but we can make some educated guesses.

We know that many advances are being made in technology that are now used on the cutting edge. Although we can't know exactly what everything will be like later, we can attempt to point out some directions the Web appears to be moving in, and what in particular you should be thinking about.

Problems with Today's Web Technology

There's no doubt the Web's popularity has benefited in no small part from increased public awareness and the availability of dial-up Internet connections. But, let's face it, if any of you have tried to look at a complicated Web site using a 28.8 kbps modem, you know that we've still got a long way to go.

Not only are there problems with access speed, but, as was mentioned in the last section, a large segment of the population remains untapped. The culture of the Internet is also changing—as more people get online, the demographics shift from those primarily in the computer and academic industry, to something that reflects more of the mainstream American and worldwide culture. This is on the whole a very good thing, but it can lead to some transitional problems, as we will later see.

Finally, a big roadblock to online commerce is the lack, or perceived lack, of security on the Internet. This problem is partially technological and partially psychological. There are protocols that can encrypt and validate transactions, such as Netscape's "Secure Sockets Layer" protocol or TERISA system's "S-HTTP" protocol. (See **http://home.netscape.com/newsref/std/SSL.html** and **http://www.terisa.com/shttp/intro.html**.) But many users are concerned about giving, say, their credit card to an entity they only know about through the Internet. A few well-publicized Internet hacking incidents have also discouraged trust. This will be solved, but not by technology alone.

Breaking the Speed Limits

In the past, getting a full connection to the Internet required a high-speed leased telephone line and expensive networking hardware. As a result, only businesses and large institutions could afford Internet access. This limited the Internet's usefulness for commercial purposes. However, the introduction of high-speed modems and dial-up Internet Service Providers (ISPs) has made WWW access from home both possible and practical.

The Serial Line Internet Protocol and Point-to-Point Protocol (SLIP and PPP) are two commonly used schemes for transferring Internet data to a home computer over the regular phone system. These protocols allow home users to obtain full Internet connections without having to purchase a leased line or expensive connecting hardware. This means that SLIP and PPP users can do everything that users with faster leased-line connections can do, albeit quite a bit slower.

Note

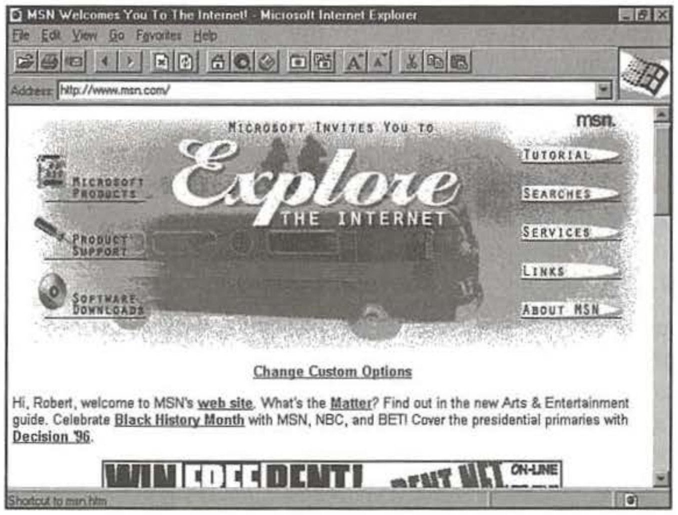
"SL/IP" is equivalent to "SLIP." Both refer to Serial Line Internet Protocol; this book uses SLIP.

But as services get bigger and more complicated, even high-speed modems often don't do the job. The use of ISDN (Integrated Services Digital Network) lines has recently become more popular, but even this solution brings up the problem of needing specialized add-on cards and protocols. ISDN is also quite expensive and is not available in many areas. As an example, a typical ISDN line in North Carolina now costs over $200 for installation and will cost an additional $75 per month to maintain (for more on ISDN, go to **http://alumni.caltech.edu/~dank/isdn/**).

Two areas that seem to hold a lot of promise for solving the bandwidth problems are cable modem access and satellite delivery. Satellite delivery is probably farther away, but some cable companies in the United States are already offering Internet access through the same line through which you receive your TV stations. One example is TCI in East Lansing, which already offers 10Mbps/sec (Ethernet speed) Internet connections for under $50 a month. It's expected that these types of connections will only get cheaper and more widespread in the future.

As more and more schools, libraries, community colleges and other public institutions get connected, those who use these facilities will also become Internet users. In addition, ISP rates will continue to fall and, as Internet Service becomes available through more accessible and accepted means (such as cable modems), people's fear of technology will also continue to decrease.

One of the last factors involved in increased usage will stem from a not-so-obvious source. In the past, if you wanted Internet service, you had to contact the provider, install the software, make the connection, and basically go through a lot of trouble to get online. However, with the breakout of Windows 95, OS/2 Warp, and other "Internet-Ready" operating systems, the Internet is now built-in (see fig. 1.5). When Internet access becomes as easy as buying your computer, plugging it in, and getting online, a large barrier to access will be removed.



Hi, Robert, welcome to MSN's web site. What's the Matter? Find out in the new Arts & Entertainment guide. Celebrate Black History Month with MSN, NBC, and BET! Cover the presidential primaries with Decision '96.

Fig. 1.5 Microsoft's Internet Explorer incorporates the same functionality as Netscape right out of the box.

New Technologies

Web site developers and Web content creators are now getting faced with a dizzying array of new technologies: Java, VRML, Shockwave, MPEG audio and video, and more. (See **http://java.sun.com/**, **http://www.vrml.org/**, and **http://www.macromedia.com/**.) The Web has always been multimedia, but until recently that has been limited to inlined GIF and JPEG images, and externally-played sound and movie files. As the state of browser technology has advanced, so too have the types of media that can be supported. Newer browsers that support a plug-in architecture can now support an arbitrary number of new media types, so we can expect to see an explosion of new types as companies start building these plug-ins.

Java and VRML are not covered in this book too deeply; they are both complex enough and powerful enough to merit their own books, and there are plenty of those out there. But well-prepared Webmasters should be aware of their existence, when they are appropriate, and how to integrate them into the server.

What Looks Good About the Web's Future

The future really is bright. We've already looked at many of the things that are available or soon will be that will make using the Web more efficient, profitable, and sensible. Perhaps one of the biggest benefits of all these changes is in the opportunity presented to small organizations without a lot of computer expertise or a lot of money to establish a presence on the Web.

Running a Web server, as this book will hopefully prove, can be not only a pretty rewarding experience, but also a pretty inexpensive endeavor. The software provided with this book is free. The software and hardware for a UNIX operating system can be pretty cheap if you purchase a 486 or Pentium and install Linux on it, and bandwidth is getting less expensive all the time. One reason for the success of the Web has been that it has been very easy to set up and add content to a server, and thus there were no restrictions as to who could do it or what they could say. Even as "the big boys" come to play in this sandbox, that liberating capability is not likely to disappear.

As an example of this, the Windows 95 Web site (developed by a company other than Microsoft, by the way), at **http://www.windows95.com/**, runs BSDI on a Pentium with Apache (the same software provided with this book) and handles approximately 2 million hits on a busy weekday. The total hardware and software cost is somewhere around $5000, so don't let anyone fool you into thinking you need big expensive iron to put out a "real" Web site.

The Web has experienced terrific growth in the first several years of existence. Fueled by applications in business, government, education, and research, and turbo-charged by dramatic improvements in browser and server technology, the Web is poised to become *the* electronic marketplace and information source of the century. ❖

CHAPTER 5

Apache Configuration

By this point you should have a running, minimal Web server. In this chapter, you learn about most of the functionality that comes bundled with the server. This chapter is organized as a series of tutorials, so that new users can get up-to-speed. Toward the end of the chapter, you dive into some experimental Apache modules as well.

By the time you read this chapter, given the rapid pace of development, there will be some significantly new functionality implemented and released. However, the existing functionality is not likely to change much. The Apache Group has had a strong ethic toward backward compatibility.

In this chapter, you will learn how to:

- Configure the MIME types of objects on the server
- Use those MIME types to trigger special actions
- Redirect and alias requests for different parts of your site
- Configure directory indexing
- Set up and use server side includes
- Set up internal imagemap handling
- Use "cookies" to track user sessions
- Set up configurable logging
- Turn on and use content negotiation
- Configure access control based on hostnames and IP numbers, or passwords
- Configure "virtual" hosts
- Customize the server's error messages

In short, this chapter covers most of the major functionality of Apache 1.0.

Configuration Basics

The srm.conf (also known as the `ResourceConfig` file, which is a directive that can be set in httpd.conf) and access.conf (also known as the `AccessConfig` file, also a directive in httpd.conf) files are where most of the configuration related to the actual objects on the server takes place. The names are mostly historical—at one point, when the server was still NCSA, the only thing access.conf was good for was setting permissions, restrictions, authentication, and so forth. Then, when directory indexing was added, the cry went out for the capability to control certain characteristics on a directory-by-directory basis. The access.conf file was the only one that had any kind of structure for that: the pseudo-HTML `<Directory>` container.

With Apache's revamped configuration file parsing routines, most directives can literally appear anywhere. For example, within `<Directory>` containers in access.conf, within `<VirtualHost>` containers in httpd.conf, and so on. However, for sanity's sake, you should keep some structure to the configuration files. You should put server-processing-level configuration options in httpd.conf (like `Port`, `<VirtualHost>` containers, etc.), put generic server resource information in srm.conf (like `Redirect`, `AddType`, directory indexing information, etc.), and per-directory configurations in access.conf.

In addition to the `<Directory>` container, there is the `<Limit>` container, which is used within `<Directory>` containers to specify certain HTTP methods to which particular directives apply. Examples will be given later in this chapter.

Per-Directory Configuration Files

Before you get too deep into the long list of features, take a look at a mechanism that controls most of those features on a directory-by-directory basis by using a file in that directory itself. You can already control subdirectory options in access.conf, as outlined in the previous chapter. However, for a number of reasons, you may want to allow these configurations to be maintained by people other than those who have the power to restart the server (such as people maintaining their home pages), and for that purpose the `AccessFileName` directive was invented.

The default `AccessFileName` is .htaccess. If you want to use something else, for example, `.acc`, you would say the following in the srm.conf file:

```
AccessFileName .acc
```

If looking for this file is enabled, and a request comes in that translates to the file /www/htdocs/path/path2/file, the server will look for /.acc, /www/.acc, /www/htdocs/.acc, /www/htdocs/path/.acc, and /www/htdocs/path/path2/.acc, in that order. Also, it will parse the file if it finds it to see what configuration options apply. Remember that this parsing has to happen with each hit, separately, so this can be a big performance hit. If you turn it off by setting the following in your access config file:

```
<Directory />
AllowOverride None
</Directory>
```

For the sake of brevity and clarity, let's call these files .htaccess files. What options can these files affect? The range of available options is controlled by the `AllowOverride` directive within the `<Directory>` container in the AccessConfig file, as mentioned previously. The exact arguments to `AllowOverride` are as follows:

Argument	Result
AuthConfig	When listed, .htaccess can specify their own authentication directives, such as `AuthUserFile`, `AuthName`, `AuthType`, `require`, and so on.
FileInfo	When listed, .htaccess can override any settings for metainformation about files, using directives such as `AddType`, `AddEncoding`, `AddLanguage`, and so forth.
Indexes	When listed, .htaccess files can locally set directives that control the rendering of the directory indexing, as implemented in the module `mod_dir.c`. For example, `FancyIndexing`, `AddIcon`, `AddDescription`, and the like.
Limit	Allow the use of the directives that limit access based on hostname or host IP number (`allow`, `deny`, and `order`).
Options	Allow the use of the `Options` directive.
All	Allow all of the above to be true.

`AllowOverride` options are not merged, which means that if the configuration for /path/ is different than the configuration for /, the /path/ one will take precedence because it's deeper.

MIME Types: *AddType* and *AddEncoding*

A fundamental element of the HTTP protocol, and the reason why the Web was so natural as a home for multiple media formats, is that every data object transferred through HTTP had an associated MIME type. What does this mean?

Note

MIME stands for *Multipurpose Internet Mail Extensions,* and its origins lie in an effort to standardize the transmission of documents of multiple media through e-mail. Part of the MIME specification was that e-mail messages could contain meta-information in the headers—information *about* the information being sent. One type of MIME header is Content-Type, which states the format or data type the object is in. For example, HTML is given the label "text/html," and JPEG images are given the label "image/jpeg". There is a registry of MIME types maintained by the Internet Assigned Numbers Authority at **http://www.isi.edu/div7/iana/**.

When a browser asks a server for an object, the server gives that object to the browser and states what its "Content-Type" is, and the browser can make an intelligent decision about how to render the document. For example, it can send it to an image program, to a postscript viewer, or to a VRML viewer.

What this means to the server maintainer is that every object being served out must have the right MIME type associated with it. Fortunately, there has been a convention of expressing data type through two-, three-, or four-letter suffixes to file name—i.e., foobar.gif is most likely to be a GIF image.

What the server needs is a file to map the suffix to the MIME content type. Fortunately, Apache comes with such a file in its config directory, a file called mime.types. You'll see that the format of this file is simple. The format consists of one record per line, where a record is a MIME type and a list of acceptable suffixes. This is because, while more than one suffix may map to a particular MIME type, you can't have more than one MIME type per suffix. You can use the `TypesConfig` directive to specify an alternative location for the file.

The Internet is evolving so quickly that it would be hard to keep that file completely up-to-date. To overcome that, you can use a special directive called `AddType`, which can be put in an `srm.conf` file like the following:

```
AddType x-world/x-vrml wrl
```

Now, whenever the server is asked to serve a file that ends with ".wrl," it knows to also send a header like the following:

```
Content-type: x-world/x-vrml
```

Thus, you don't have to worry about reconciling future distributions of the `mime.types` file with your private installations and configuration.

As you'll see in future pages, however, `AddType` is also used to specify "special" files that get magically handled by certain features within the server.

A sister to `AddType` is `AddEncoding`. Just as the MIME header `Content-Type` can specify the data format of the object, the header `Content-Encoding` specifies the *encoding* of the object. An encoding is an attribute of the object as it is being transferred or stored; semantically, the browser should know that is has to "decode" whatever it gets based upon the listed encoding. The most common use is with compressed files. For example, if you have

```
AddEncoding x-gzip gz
```

and if you then access a file called "myworld.wrl.gz," the MIME headers sent in response will look like the following:

```
Content-Type: x-world/x-vrml
Content-Encoding: x-gzip
```

And any browser worth its two cents will know "Oh, I have to uncompress the file before handing it off to the VRML viewer."

Alias, ScriptAlias, and *Redirect*

These three directives, all denizens of srm.conf, and all three implemented by the module `mod_alias.c`, allow you to have some flexibility with the mapping between "URL-Space" on your server and the actual layout of your file system.

If that last statement sounded cryptic, don't worry. What it basically means is that any URL that looks like "http://myhost.com/x/y/z" does not have to necessarily map to a file named "x/y/z" under the document root of the server:

```
Alias /path/ /some/other/path/
```

The preceding directive will take a request for an object from the mythical subdirectory `/path` under the document root and map it to another directory somewhere else entirely. For example, a request for

```
http://myhost.com/statistics/
```

might normally go to `document root /statistics`, except that for whatever reason you wanted it to point somewhere else outside of the document root. Say /usr/local/statistics. For that you'd have the following:

```
Alias /statistics/ /usr/local/statistics/
```

To the outside user this would be completely transparent. If you use `Alias`, it's wise not to alias to somewhere else inside of document root. Furthermore, a request like

```
http://myhost.com/statistics/graph.gif
```

would get translated into a request for the file

```
/usr/local/statistics/graph.gif
```

`ScriptAlias` is just like `Alias`, with the side-effect of making everything in the subdirectory by default a CGI script. This might sound a bit bizarre, but the early model for building Web sites had all the CGI functionality separated into a directory by itself, and referenced through the Web server as shown in the following:

```
http://myhost.com/cgi-bin/script
```

If you have in your srm.conf

```
ScriptAlias /cgi-bin/ /usr/local/etc/httpd/cgi-bin/
```

then the preceding URL points to the script at "/usr/local/etc/httpd/cgi-bin/script." As you'll see in a page or two, there is another way to specify that a file is a CGI script to be executed.

`Redirect` does just that—it redirects the request to another resource. That resource could be on the same machine, or somewhere else on the Net. Also, the match will be a substring match, starting from the beginning. For example, if you did:

```
Redirect /newyork http://myhost.com/maps/states/newyork
```

then a request for

```
http://myhost.com/newyork/index.html
```

will get redirected to

```
http://myhost.com/maps/states/newyork/index.html
```

Of course, the second argument to `Redirect` can be a URL at some other site. Just make sure that you know what you're doing. Also, be wary of creating loops accidentally. For example,

```
Redirect /newyork http://myhost.com/newyork/newyork
```

can have particularly deleterious effects on the server!

A Better Way To Activate CGI Scripts

You read earlier that there is a more elegant way of activating CGI scripts than using `ScriptAlias`. You can use the `AddType` directive and a "magic" MIME type, like so:

```
AddType application/x-httpd-cgi cgi
```

When the server gets a request for a CGI file, it maps to that MIME type, and then catches itself and says "Aha! I need to execute this instead of just dish it out like regular files." Thus, you can have CGI files in the same directories as your HTML and GIF and all your other files.

A later chapter will go into more detail about the implementation of CGI in Apache.

Directory Indexing

When Apache is given a URL to a directory, instead of to a particular file, for example

```
http://myhost.com/statistics/
```

Apache first looks for a file specified by the `DirectoryIndex` directive in srm.conf. In the default configs, this is index.html. You can set a list of files to search for, or even an absolute path to a page or CGI script:

```
DirectoryIndex index.cgi index.html /cgi-bin/go-away
```

The preceding directive says to look for an "index.cgi" in the directory first. If that can't be found, then look for an "index.html" in the directory. If neither can be found, then redirect the request to "/cgi-bin/go-away."

If it all fails to find a match, then Apache will create, completely on-the-fly, an HTML listing of all the files available in the directory:

```
<Give a figure here of the directory listing output>
```

There are quite a few ways to customize the output of the directory indexing functionality. First, you need to ask yourself if you care about seeing things like icons or last-modified times in the reports. If you do, then you want to turn to

```
FancyIndexing On
```

otherwise, you'll just get a simple menu of the available files, which you may want for security or performance reasons.

With that going on, you must ask whether you need to customize it further, and how. The default settings for the directory indexing functionality are already pretty elaborate.

The `AddIcon`, `AddIconByEncoding`, and `AddIconByType` directives customize the selection of icons next to files. `AddIcon` matches icons at the file name level by using the pattern

```
AddIcon iconfile filename [filename] [filename]...
```

Thus, for example,

```
AddIcon /icons/binary.gif .bin .exe
```

means that any file that ends in `.bin` or `.exe` should get the `binary.gif` icon attached. The file names can also be a wildcard expression, a complete file name, or even one of two "special" names: `^^DIRECTORY^^` for directories and `^^BLANKICON^^` for blank lines. So you can see lines like

```
AddIcon /icons/dir.gif ^^DIRECTORY^^
AddIcon /icons/old.gif *~
```

Finally, the "iconfile" can actually also be a string containing both the iconfile's name and the alternate text to put into the ALT attribute. So, your examples should really be

```
AddIcon (BIN,/icons/binary.gif) .bin .exe
AddIcon (DIR,/icons/dir.gif) ^^DIRECTORY^^
```

The `AddIconByType` directive is actually a little bit more flexible and probably comes more highly recommended in terms of actual use. Instead of tying icons to file name patterns, it ties icons to the MIME type associated with the files. The syntax is very roughly the same:

```
AddIconByType iconfile mime-type [mime-type]…
```

`mime-type` can be either the exact MIME type matching what you have assigned a file, or it can be a pattern match. Thus, you see entries in the default configuration files like the following:

```
AddIconByType (SND,/icons/sound2,gif) audio/*
```

This is a lot more robust than trying to match against file name suffixes.

`AddIconByEncoding` is used mostly to distinguish compressed files from the others. This makes sense only if used in conjunction with `AddEncoding` directives in your srm.conf file. The default srm.conf has these entries:

```
AddEncoding x-gzip gz
AddEncoding x-compress Z
AddIconByEncoding (CMP,/icons/compressed.gif) x-compress x-gzip
```

This will set the icon next to compressed files appropriately.

The `DefaultIcon` directive specifies the icon to use when none of the patterns match a given file when the directory index is generated.

```
DefaultIcon /icons/unknown.gif
```

It is possible to add text to the top and the bottom of the directory index listing. This capability is very useful as it turns the directory indexing capabilities from just a UNIX-like interface into a real dynamic document interface. There are two directives to control this: `HeaderName` and `ReadmeName`, which specify the file names for the content at the top and bottom of the listing, respectively. Thus, as shown in the default srm.conf file:

```
HeaderName HEADER
ReadmeName README
```

When the directory index is being built, Apache will look for "HEADER.html." If it finds it, it'll throw the content into the top of the directory index. If it fails to find that file, it'll look for just "HEADER," and if it finds that it will presume the file is plain text and do things like escape characters such as "<" to "&lt;", and then insert it into the top of the directory index. The same process happens for the file "README," except that the resulting text goes into the bottom of the generated directory index.

In many cases, be it for consistency or just plain old security reasons, you will want to have the directory indexing engine just ignore certain types of files, like Emacs backup files or files beginning with a ".". The `IndexIgnore` directive addresses this; the default setting is

```
IndexIgnore */.??* *~ *# */HEADER* */README* */RCS
```

This line might look cryptic, but it's basically a space-separated list of patterns. The first pattern matches against any "." file that is longer than three characters. This is so that the link to the higher-up directory (`..`) can still work. The second (`*~`) and third (`*#`) are common patterns for matching old emacs backup files. The next ones are to avoid listing the same files used for `HeaderName` and `ReadmeName` as in the preceding. The last (`*/RCS`) is given because many sites out there use `RCS`, a software package for revision control maintenance, which stores its extra (rather sensitive) information in `RCS` directories.

Finally you get to two really interesting directives for controlling the last set of options regarding directory indexing. The first is `AddDescription`, which works similarly to `AddIcon`.

```
AddDescription description filename [filename]...
```

That is

```
AddDescription "My cat" /private/cat.gif
```

As elsewhere, `filename` can actually be a pattern, so you can have

```
AddDescription "An MPEG Movie Just For You!" *.mpg
```

Finally, you have the granddaddy of all options-setting directives, `IndexOptions`. This is the smorgasbord of functionality control. The syntax is simple:

```
IndexOptions option [option]...
```

The list of available options are listed in the following table:

Option	Explanation
FancyIndexing	This is the same as the separate `FancyIndexing` directive. Sorry to confuse everyone, but backward compatibility demands bizarre things sometimes!
IconsAreLinks	If this is set the icon will be clickable as a link to whatever resource the entry it is associated with links to. In other words, the icon becomes part of the hyperlink.
ScanHTMLTitles	When given a listing for an HTML file, the server will open the HTML file and parse it to obtain the value of the <TITLE> field in the HTML document, if it exists. This can put a pretty heavy load on the server, since it's a lot of disk accessing and some amount of CPU to extract the title from the HTML, so it's not recommended unless you know you have the capacity.
SuppressDescription, SuppressLastModified, SuppressSize	These will suppress their respective fields in the directory indexing output. Normally each of those (`Description`, `Last Modified`, Size) is a field in the output listings.

By default none of these are turned on. The options do not *merge*, which means that when you are setting these on a per-directory basis by using either access.conf or .htaccess files, setting the options for a more specific directory requires resetting the complete options listing. For example, envision the following in your access configuration file:

```
<Directory /pub/docs/>
IndexOptions ScanHTMLTitles
</Directory>
<Directory /pub/docs/others/>
IndexOptions IconsAreLinks
</Directory>
```

Directory listings done in or below the second directory, /pub/docs/others/, would not have `ScanHTMLTitles` set. Why? Well, you figured administrators would need to be able to disable an option they had set globally in a specific directory, and this was simpler than writing "NOT" logic into the options listings.

If you run into problems getting directory indexing to work, make sure that the settings you have for the `Options` directive in the access config files allow for directory indexing in that directory. Specifically, the `Options` directive must include `Indexing`. Furthermore, if you are using .htaccess files to set things like `AddDescription` or `AddIcon`, the `AllowOverride` directive must include in its list of options `FileInfo`. This is covered in more depth later in this chapter.

User Directories

Sites with many users sometimes prefer to be able to give their users access to managing their own parts of the Web tree in their own directories, using the URL semantics of

```
http://myhost.com/~user/
```

Where "~user" is actually an alias to a directory in the user's home directory. This is different from the `Alias` directive, which could only map a particular pseudo-directory into an actual directory. In this case, you want "~user" to map to something like "/home/user/public_html," and because the number of "users" can be very high, some sort of macro is useful here. That macro is the directive `UserDir`.

With `UserDir` you specify the subdirectory within the users' home directory where they can put content, which is mapped to the "~user" URL. So in other words, the default

```
UserDir public_html
```

will cause a request for

```
http://myhost.com/~eric/index.html
```

to cause a lookup for the UNIX file

```
/home/eric/public_html/index.html
```

presuming that /home/eric is eric's home directory. The default of public_html is a historical artifact more than anything else. There's no reason why you can't make it "Web_stuff" or something like that.

Note

Apache 1.1's user-directory module will have even more functionality, but at press time the feature set has not been nailed down.

Special Modules

Most of the functionality that distinguishes Apache from the competition has been implemented as modules to the Apache API. This has been extremely useful in allowing functionality to evolve separately from the rest of the server, and for allowing for performance tuning. This section will cover that extra functionality in detail.

Server Side Includes

Server side includes are best described as a preprocessing language for HTML. The "processing" takes place on the server side, such that visitors to your site never need know that you use server side includes, and thus requires no special client software. The format of these includes looks something like the following:

```
<!--#directive attribute="value" -->
```

Sometimes a given "directive" can have more than one attribute at the same time. The funky syntax is due to the desire to hide this functionality within an SGML comment—that way your regular HTML validation tools will work without having to learn new tags or anything. The syntax is important; leaving off the final "--," for example, will result in errors.

#include

This directive is probably the most commonly used directive. It is used to insert another file into the HTML document. The allowed attributes for this directive are `virtual` and `file`. The functionality of the `file` attribute is a subset of that provided by the `virtual` attribute, and it exists mostly for backward compatibility, so its use is not recommended.

The `virtual` attribute instructs the server to treat the value of the attribute as a request for a relative link—meaning that you can use "../" to locate objects above the directory, and that other transforms like `Alias` will apply.

For example:

```
<!--#include virtual="quote.txt" -->
<!--#include virtual="/toolbar/footer.html" -->
<!--#include virtual="../footer.html" -->
```

#exec

This directive is used to run a script on the server side and insert its output into the SSI document being processed. There are two choices: executing a CGI script by using the `cgi` attribute, or executing a shell command by using the `cmd` attribute.

For example:

```
<!--#exec cgi="counter.cgi" -->
```

would take the output of the CGI program `counter.cgi` and insert it into the document. Note that the CGI output still has to include the "text/html" content type header or an error will occur.

Likewise,

```
<!--#exec cmd="ls -l" -->
```

would take the output of a call to `ls -l` in the document's directory and insert it. Like the `file` attribute to the `#include` directive, this is mostly for backward compatibility, because it is something of a security hole in an untrusted environment.

Note

There are definitely security concerns with allowing users access to CGI functionality and even greater concerns with `#exec cmd`, such as `cmd="cat/etc/passwd"`. If the site administrator wishes to allow people to use server side includes, but not to use the `#exec` directive, then they can set `IncludesNOEXEC` as an option for the directory in the access configurations.

#echo

This directive has one attribute, `var`, whose value is any CGI environment variable as well as a small list of other variables:

Attribute	Defintion
DATE_GMT	The current date in Greenwich Mean Time.
DATE_LOCAL	The current date in the local time zone.
DOCUMENT_NAME	The file system name of the SSI document, not including the directories below it.
DOCUMENT_URI	In a URL of the format "http://host/path/file." This is the "/path/file" part.
LAST_MODIFIED	The date the SSI document was modified.

Example:

```
<!--#echo var="DATE_LOCAL" -->
```

This will insert something like `Wednesday, 06-Mar-96 10:44:54 GMT` into the document.

#fsize, #flastmod

These two directives print out the size and the last-modified date, respectively, of any object given by the URI listed in the `file` or `virtual` attribute, as in the `#include` directive. For example

```
<!--#fsize file="index.html" -->
```

would return the size of the index.html file in that directory.

#config

You can modify the rendering of certain SSI directives by using this directive.

The `sizefmt` attribute controls the rendering of the `#fsize` directive with values of `bytes` or `abbrev`. The exact number of bytes is printed when `bytes` is given, whereas an abbreviated version of the size (either in `K` for kilobytes or `M` for megabytes) is given when `abbrev` is set.

Thus, for example, a snippet of SSI HTML like

```
<!--#config sizefmt="bytes" -->
The index.html file is <!--#fsize virtual="index.html" --> bytes
```

would return `The index.html file is 4,522 bytes`. Meanwhile, if

```
<!--#config sizefmt="abbrev" -->
```

was used, "The index.html file is 4K bytes" would be returned. The default is `abbrev`.

The `timefmt` directive controls the rendering of the date in the `DATE_LOCAL`, `DATE_GMT`, and `LAST_MODIFIED` values for the `#echo` directive. It uses the same format as the `strftime` call (In fact, that's what the server does. It calls `strftime`.) This format consists of variables that begin with `%`. For example, `%H` is the hour of the day, in 24-hour format. The list of variables is best found by consulting your system's "man" page by typing **man strftime** for directions as to how to construct a `strftime`-format date string.

An example might be:

```
<!--#config timefmt="%Y/%m/%d-%H:%M:%S" -->
```

and the resulting date string for Jan. 2, 1996 at 12:30 in the afternoon would thus be

```
1996/01/02-12:30:00
```

Finally, the last attribute the `config` directive can take is `errmsg`, which is simply the error to print out if there are any problems parsing the document. For example, the right default is:

```
<!--#config errmsg="An error occurred while processing this
directive" -->
```

Internal Imagemap Capabilities

The default imagemap module supplied with Apache allows you to reference imagemaps without using or needing any CGI programs. This functionality is contained in the `mod_imap` module. First, you add to your srm.conf yet another magic `AddType` directive:

```
AddType application/x-httpd-imap map
```

This now means that any file ending with ".map" will be recognized as an imagemap file. After restarting the server to pick up the change, one can make reference to a .map file directly.

Look at an example: the following document, index.html, has an imagemap on it, where the image is `usa.jpg` and the mapfile is `usa-map.map`. The HTML to build that imagemap would look like:

```
<A HREF="usa-map.map"><IMG SRC="usa.jpg" ISMAP></A>
```

Imagemaps are covered in more detail in a later chapter—the only important thing from a configuration standpoint is that the magic content type is activated.

Cookies

HTTP *cookies* are a method for maintaining statefulness in a stateless protocol. What does this mean? In HTTP, a session between a client and a server typically spans many separate actual TCP connections, thus making it difficult to tie together accesses into an application that requires state, such as a shopping cart application. Cookies are a solution to that problem. As implemented by Netscape in their browser and subsequently by many others, servers can assign clients a *cookie*, meaning some sort of opaque string whose meaning is significant only to the server itself, and then the client can give that cookie back to the server on subsequent requests.

The module *mod_cookies* nicely handles the details of assigning unique cookies to every visitor, based on their hostname and a random number. This cookie can be accessed from the CGI environment as the HTTP_COOKIE environment variable, for the same reason that all HTTP headers are accessible to CGI applications. The CGI scripts can use this as a key in a session tracking database, or it can be logged and tallied up to get a good, if undercounted, estimate of the total number of users that visited a site, not just the number of hits or even number of unique domains.

Happily, there are no configuration issues here—simply compile with `mod_cookies` and away you go. Couldn't be easier.

Configurable Logging

For most folks, the default logfile format (also known as *Common Logfile Format*, or CLF) does not provide enough information when it comes to doing a serious analysis of the efficacy of your Web site. It provides basic numbers in terms of raw hits, pages accessed, hosts accessing, timestamps, etc., but it fails to capture the "referring" URL, the browser being used, and any cookies being used. So, there are two ways to get more data for your logfiles: by using the NCSA-compatibility directives for logging certain bits of info to separate browsers, or using Apache's own totally configurable logfile format.

NCSA Compatibility

For compatibility with the NCSA 1.4 Web server, two modules were added. These modules log the `User-Agent` and `Referer` headers from the HTTP request stream.

`User-Agent` is the header most browsers send that identifies what software the browser is using. Logging of this header can be activated by an `AgentLog` directive in the srm.conf file, or in a virtualhost-specific section. This directive takes one argument, the name of the file to which the user-agents are logged. For example:

```
AgentLog logs/agent_log
```

To use this, you need to ensure that the `mod_log_agent` module has been compiled and linked to the server.

Similarly, the `Referer` header is sent by the browser to indicate the tail end of a link—in other words, when you are on a page with a URL of "A," and there is a link on that page with a URL of "B," and you follow that link, the request for page "B" includes a `Referer` header with the URL of "A." This is very useful for finding what sites out there link to your site, and what proportion of traffic they account for.

The logging of this header is activated by a `RefererLog` directive, which points to the file to which the referers get logged.

```
RefererLog logs/referer_log
```

One other option the `Referer` logging module provides is `RefererIgnore`, a directive that allows you to ignore `Referer` headers, which contain some string. This is useful for weeding out the referers from your own site, if all you are interested in is links to you from other sites. For example, if your site is "www.myhost.com," you might want to use the following:

```
RefererIgnore www.myhost.com
```

Remember that logging of the `Referer` header requires compiling and linking in `mod_log_referer`.

Totally Configurable Logging

The previous modules were provided, like many Apache features, for backward compatibility. They have some problems, though. Because they don't contain any other information about the request they are logging from, it's nearly impossible to tell which `Referer` fields went to which specific objects on your site. Ideally all the information about a transaction with the server can be logged into one file, extending the `common logfile format` or replacing it altogether. Well, such a beast exists, in the `mod_log_config` module.

This module implements the `LogFormat` directive, which takes as its argument a string, with variables beginning with `%` to indicate different pieces of data from the request. The variables are:

Variable	Definition
%h	Remote host.
%l	Remote `identd` identification.
%u	Remote user, as determined by any user authentication that may take place. Note that if the user was not authenticated, and the status of the request is a 401, this field may be bogus.
%t	The common logfile format for time.
%r	First line of request.
%s	Status. For requests that got internally redirected, this is status of the original request; `%>s` will give the last.
%b	Bytes sent.
%{Foobar}i	The contents of Foobar: header line(s) in the request from the client to the server.
%{Foobar}o	The contents of Foobar: header line(s) in the response from the server to the client.

So, for example, if you wanted to capture in your log just the remote hostname, the object they requested, and the timestamp, you would do the following:

```
LogFormat "%h \"%r\" %t"
```

And that would log things that looked like

```
host.outsider.com "GET / HTTP/1.0" [06/Mar/1996:10:15:17]
```

Note that you really have to use a quote around the request variable—the configurable logging module does not escape the values of the variables. But use a slash-quote, `\"`, to distinguish that from the end of the string.

Say you want to add logging of the `User-Agent` string to that as well—in this case, your log format would become:

```
LogFormat "%h \"%r\" %t \"%{User-Agent}i\""
```

Because the `User-Agent` field typically has spaces in it, it too should be quoted. Say you want to capture the `Referer` field:

```
LogFormat "%h \"%r\" %t %{Referer}i"
```

You don't need the escaping quotes because `Referer` headers, since they are URL's, don't have spaces in them. However, if you are building a mission-critical application you might as well quote it as well, because the `Referer` header is supplied by the client and thus there are no guarantees about its format.

The default format is the Common Logfile Format (CLF), which in this syntax is expressed as

```
LogFormat "%h %l %u %t \"%r\" %s %b"
```

In fact, most existing logfile analysis tools for CLF will ignore extra fields tacked onto the end, so to capture the most important extra information and yet still be parseable by those tools, you might want to use the format:

```
LogFormat "%h %l %u %t \"%r\" %s %b %{Referer}i \"%{User-
➥Agent}i\""
```

Power users take note: If you want even more control over what gets logged, you can use the configurable logging module to implement a simple conditional test for variables. This way, you can configure it to only log variables when a particular status code is returned, or not returned. The format for this is to insert a comma-separate list of those codes between the `%` and the letter of the variable, like so:

```
%404,403{Referer}i
```

This means that the `Referer` header will only be logged if the status returned by the server is a `404 Not Found`, or a `403 Access Denied`. All other times just a "-" is logged. This would be useful if all you cared about using `Referer` for was to find out old links that point to resources no longer available.

The negation of that conditional is to put a `!` at the beginning of the list of status codes; so for example,

```
%!401u
```

will log the user in any user authentication transaction, unless the authentication failed, in which case you probably don't want to see the name of the bogus user anyway.

Remember that, like many functions, this can be configured per virtual host. Thus, if you want all logs from all virtual hosts on the same server to go to the same log, you might want to do something like

```
LogFormat "hosta ...."
```

in the `<virtualhost>` sections for hosta and

```
LogFormat "hostb ...."
```

in the `<virtualhost>` sections for hostb. More details about virtual hosts will appear later in this chapter.

A key note: You have to compile in `mod_log_config` for this functionality. You must also make sure that the default logging module, `mod_log_common`, is not compiled in, or the server will get confused.

Content Negotiation

Content negotiation is the mechanism by which a Web client can express to the server what data types it knows how to render, and based on that information, the server can give the client the "optimal" version of the resource requested. Content negotiation can happen on a number of different characteristics—the content type of the data (also called the *media type*), the human language the data is in (English, French, etc.), the character set of the document, and its encodings.

Content Type Negotiation

For example, say you want to use inlined JPEG images on your pages. You don't want to alienate people using older browsers, which don't know how to inline JPEG images, so you also make a GIF version of that image. Even though the GIF might be larger or only 8-bit, that's still better than giving the browser something it can't handle, causing a broken link. So, the browser and the server *negotiate* for which data format the server sends to the client.

The specifications for content negotiation have been a part of HTTP since the beginning. Unfortunately, it can't be relied upon as extensively as one would like. For example, current browsers that implement plug-ins, by and large do not express in the connection headers which media types they have plug-ins for. Thus, content-negotiation can't be used to decide whether to send someone a ShockWave file or its Java equivalent, currently. The only safe place to use it currently is to distinguish between inlined JPEG or GIF images on a page. Enough browsers in use today implement content negotiation closely enough to get this functionality.

The `mod_negotiation.c` in Apache 1.0 implements the content negotiation specifications in an older version of the HTTP/1.0 IETF draft, which at the time of this writing is on its way to informational RFC status. It was removed because the specification was not entirely complete, and a document describing it could not be labeled "Best Current Practice," which is what the HTTP/1.0 specification became. Content negotiation is getting significantly enhanced for HTTP/1.1. However, this doesn't mean it can't be safely used now for inlined image selection.

To activate it, you must include the module `mod_negotiation.c` into the server. There are actually two ways to configure content negotiation:

- Using a type-map file describing all the variants of a negotiable resource with specific preference values and content characteristics
- Setting an `Options` value called `MultiViews`.

Since your focus is pragmatic, you will go only into the "MultiViews" functionality. If you are interested in the type-map functionality, the Apache Web site has documentation on it.

In your access.conf file, find the line that sets the options for the part of the site you wish to enable content negotiation within. This may be the whole site, but that's fine. If `MultiViews` is not present in that line, it must be. The `All` value does not, ironically enough, include `MultiViews`. This is again for backward compatibility. So, you might have a line that looks like:

```
Options Indexes Includes Multiviews
```

or

```
Options All MultiViews
```

Once this change is made, restart your server to pick up the new configuration.

With this turned on, you can do the following: place a JPEG image in a directory, say /path/, and call it image.jpg. Now, make an equivalent GIF format image, and place it in the same directory, as image.gif. The URLs for these two objects are

```
http://host/path/image.jpg
```

and

```
http://host/path/image.gif
```

respectively. Now, if you ask your Web browser to fetch,

```
http://host/path/image
```

the server will go into the /path/ directory, see the two image files, and then determine which one to send based on what the client states it can support. In the case where the client says it can accept either JPEG images or GIF images equally, the server will choose the version that is the smallest, and send that to the client. Usually, JPEG images are much smaller than GIF images.

So, if you made your HTML look something like the following:

```
<HTML><HEAD>
<TITLE>Welcome to the Gizmo Home Page!</TITLE>
</HEAD><BODY>
<IMG SRC="/header" ALT="GIZMO Logo">
Welcome to Gizmo!
<IMG SRC="/products" ALT="Products">
<IMG SRC="/services" ALT="Services">
```

then you can have separate GIF and JPEG files for `header`, `products`, and `services`, and the clients will for the most part get what they claim they can support.

Note that, if you have a file called "image" and a file called "image.gif," the file called "image" will be requested no matter if a request is made for just "image." Likewise, a request specifically for "image.gif" would never return "image.jpg" even if the client knew how to render JPEG images.

Human Language Negotiation

If `MultiViews` is enabled, you can also distinguish resources by the language they are in, such as French, English, and Japanese. This is done by adding more entries to the file suffix namespace that map to the languages the server wishes to use, and then giving them a ranking that ties can be broken. Specifically, in the "srm.conf" file, go two new directives, `AddLanguage` and `LanguagePriority`. The formats are as follows:

```
AddLanguage en .en
AddLanguage it .it
AddLanguage fr .fr
AddLanguage jp .jp
LanguagePriority en fr jp it
```

Say you want to use this to negotiate on the file "index.html," which you had available in English, French, Italian, and Japanese. You would create an "index.html.en," "index.html.fr," "index.html.it," and "index.html.jp," respectively, and then reference the document as "index.html." When a multilingual client connects, it should indicate in one of the request headers (`Accept-Language`, to be specific) which languages it prefers, and it expresses that in standard two-letter notation. The server sees what the clients can accept, and gives them "the best one." `LanguagePriority` is what organizes that decision of "the best one." If English is unacceptible to the client, try French, otherwise try Japanese, otherwise try Italian. `LanguagePriority` also states which one should be served if there is no `Accept-Language` header.

Because the language mapping suffixes and the content-type suffixes share the same namespace, you can mix them around. "index.fr.html" is the same as "index.html.f.," Just make sure that you reference it with the correct negotiable resource.

As-Is Files

Often, you might like to request specific HTTP headers in your documents, such as Expires:, but you don't want to make the page a CGI script. The easiest way is to use the `httpd/send-as-is` magic MIME type.

```
AddType httpd/send-as-is asis
```

This means that any file that ends in ".asis" can include its own MIME headers. However, it *must* include *two* carriage returns before the actual body of the content. Actually, it should include two carriage return / line feed combinations, but Apache is forgiving and will insert that for you. So, if you wanted to send a document with a special unique custom MIME type you didn't want registered with the server, you can send:

```
Content-type: text/foobar

This is text in a very special "foobar" MIME type.
```

The most significant application I've run across for this is as an extremely efficient mechanism for doing server-push objects without CGI scripts. The reason a CGI script is needed to create a server-push usually is that the Content-type usually includes the multipart separator (since a server-push is actually a MIME multipart message). For example,

```
Content-type: multipart/x-mixed-replace;boundary=XXXXXXXX

--XXXXXXXX
     Content-type: image/gif

....(GIF data)....
--XXXXXXXX
Content-type: image/gif

....(GIF data)....
--XXXXXXXX
....
```

By making this stream of data a simple file instead of a CGI script, you save yourself potentially a lot of overhead. Just about the only thing you lose is the ability to do timed pushes. For many people, slow internet connection acts as a sufficient time valve.

If you have MultiViews turned on, you can add an ".asis" to the end of a file name and none of your links need to be renamed. For example, "foobar.html" can easily become "foobar.html.asis," while still being able to call it "foobar.html."

One last compelling application of "asis" is being able to do HTTP redirection without needing access to server config files. For example, the following .asis file will redirect people to another location:

```
Status 302 Moved
Location: http://some.other.place.com/path/
Content-type: text/html

<HTML>
<HEAD><TITLE>We've Moved!</TITLE></HEAD>
<BODY>
<H1>We used to be here, but now we're
<A HREF="http://some.other.place.com/path/">over there. </A>
</H1>
</BODY></HTML>
```

The HTML body is there simply for clients who don't understand the 302 response.

Advanced Functionality

Host-Based Access Control

One can control access to the server, or even a subdirectory of the server, based on the hostname, domain, or IP number of the client's machine. This is done by using the directives `allow` and `deny`, which can be used together at the same time by using `order`. `allow` and `deny` can take multiple hosts:

```
deny from badguys.com otherbadguys.com
```

Typically, you want to do one of two things: you want to deny access to your server from everyone but a few other machines, or you want to grant access to everyone except a few hosts. The first case is handled as follows:

```
order allow,deny
allow from mydomain.com
deny from all
```

This means, "only grant access to hosts in the domain 'mydomain.com'." This could include "host1.mydomain.com," "ppp.mydomain.com," and "the-boss.mydomain.com."

The `order` directive above tells the server to evaluate the `allow` conditions before the `deny` conditions when determining whether to grant access. Likewise, the "only exclude a couple of sites" case described above can be handled by using:

```
order deny,allow
deny from badguys.com
allow from all
```

`order` is needed because, again mostly for historical reasons, the order in which directives appear is not significant. Thus, the server needs to know which rule to apply first. The default for `order` is `deny,allow`.

There is a third argument to order, called mutual-failure, in which a condition has to pass both the allow and deny rules in order to succeed. In other words, it has to appear on the allow list, and it must not appear on the deny list. For example,

```
order mutual-failure
allow from mydomain.com
deny from the-boss.mydomain.com
```

In this example, the-boss.mydomain.com is prevented from accessing this resource, but every other machine at mydomain.com can access it.

It should be mentioned at this point that protecting resources by hostname is dangerous. It is relatively easy for a determined person who control the reverse-DNS mapping for their IP number to spoof any hostname they want. Thus, it is strongly recommended that you use IP numbers to protect anything sensitive. In the same way you can simply list the domain to refer to any machine in that domain, you can also give fragments of IP numbers:

```
allow from 204.62.129
```

This will only allow hosts whose IP numbers match that, such as 204.62.129.1 or 204.62.129.130.

Typically, these directives are used within a <Limit> container, and even that within a <Directory> container, usually in an access.conf configuration file. The following example is a good template for most protections; it protects the directory /www/htdocs/private from any host except those in the 204.62.129 IP space.

```
<Directory /www/htdocs/private>
Options Includes
AllowOverride None
<Limit GET POST>
order allow,deny
deny from all
allow from 204.62.129
</Limit>
</Directory>
```

User Authentication

When you place a resource under *user authentication*, you restrict access to it by requiring a name and password. This name and password is kept in a database on the server. This database can take many forms; Apache modules have been written to access flat file databases, DBM file databases, Msql databases (a freeware database), Oracle and Sybase databases, and more. This book covers only the flat-file and DBM-format databases.

First, some basic configuration directives. The `AuthName` directive sets the authentication "Realm" for the password-protected pages. The "Realm" is what gets presented to clients when prompted for authentication—"Please enter your name and password for the realm   ."

The `AuthType` directive sets the authentication type for the area. In HTTP/1.0 there is only one authentication type, and that is `Basic`. HTTP/1.1 will have a few more, such as `MD5`.

The `AuthUserFile` directive specifies the file thT contains a list of names and passwords, one pair per line, where the passwords are encrypted by using the simple UNIX `crypt()` routines. For example,

```
joe:D.W2yvlfjaJoo
mark:21slfoUYGksIe
```

The `AuthGroupFile` directive specifies the file which contains a list of groups, and members of those groups, separated by spaces. For example:

```
managers: joe mark
production: mark shelley paul
```

Finally, the `require` directive specifies what conditions need to be met for access to be granted. It can list only a specified list of users who may connect, it can specify a group or list of groups of users who may connect, or it can say any valid user in the database is automatically granted access. For example:

```
require user mark paul
(Only mark and paul may access.)

require group managers
(Only people in group managers may access.)

require valid-user
(Anyone in the AuthUserFile database may access.)
```

The configuration file ends up looking something like this:

```
<Directory /www/htdocs/protected/>
AuthName Protected
AuthType basic
AuthUserFile /usr/local/etc/httpd/conf/users
<Limit GET POST>
require valid-user
</Limit>
</Directory>
```

If you want to protect it to a particular group, the configuration file looks something like the following:

```
<Directory /www/htdocs/protected/>
AuthName Protected
```

```
AuthType basic
AuthUserFile /usr/local/etc/httpd/conf/users
AuthGroupFile /usr/local/etc/httpd/conf/group
<Limit GET POST>
require group managers
</Limit>
</Directory>
```

DBM Authentication

Apache can be configured to also use DBM files for faster password and group-membership lookups. To use this, you must have the `mod_auth_dbm` module compiled into the server.

DBM files are UNIX file types that implement a fast hashtable lookup, making them ideal for handling large user/password databases. The flat-file systems requires parsing the password file for every access until a match is found, potentially going through the entire file before returning a `can't find that user` error. Hash tables, on the other hand, know instantly whether a "key" exists in the database, and what its value is.

Some systems use the `ndbm` libraries; some use the `berkeley db` libraries. However, the interface through Apache is exactly the same.

To use a DBM file for the database instead of a regular flat file, you use a different directive, `AuthDBMUserFile` instead of `AuthUserFile`. Likewise for the group file—`AuthDBMGroupFile` instead of `AuthGroupFile` is used.

Take a look at creating the DBM files. There is a program supplied in the support subdirectory of the Web site called "dbmmanage." It is a file for creating and managing DBM files. The basic syntax is as follows:

```
dbmmanage dbmfile command key [value]
```

`command` can be one of: `add`, `adduser`, `view`, `delete`

So, to add a `value` to a DBM file called "users," one would say:

```
dbmmanage users add joe joespassword
```

You have just added a record to the DBM file, with `joe` as the key and `joespassword` as the value. To see this you say:

```
dbmmanage users view joe
```

or if you want to see the whole database,

```
dbmmanage users view
```

However, you want to store encrypted passwords, because that's what the server uses for authentication. For that you use the `adduser` command:

```
dbmmanage users adduser joe joespassword
```

Now, if you do a `view` to look at it, `joespassword` will be replaced by a lot of what looks like junk. Don't worry, that's the encrypted password.

Groups are done a little bit differently in DBM files. Instead of making the key of the database the group, the key is the user and the value is a comma-separated list of the groups that user is in. For example:

```
dbmmanage group adduser joe managers,production
```

Wait, you say, there's no file called "users." Why do I see a "users.pag" and "users.dir"?

Well, DBM files are pretty weird. They aren't like regular files; they can't be looked at. Some systems implement the hash table by keeping the index separate from the data, as in this example with the .pag and .dir files. On BSD systems, where Berkeley DB is implemented, DBM files are saved with a ".db" appendix. So, one should get used to the idea that the "name" of a DBM file is actually its file name without the suffix.

The configuration file snippet now looks something like:

```
<Directory /www/htdocs/protected/>
AuthName Protected
AuthType basic
AuthDBMUserFile /usr/local/etc/httpd/conf/users
AuthDBMGroupFile /usr/local/etc/httpd/conf/group
<Limit GET POST>
require group managers
</Limit>
</Directory>
```

Note that `users` and `groups` must be the "name" of the DBM file, as described in the preceding. Pointing to

```
AuthDBMUserFile /usr/local/etc/httpd/conf/users.db
```

would not work.

Note

Make sure that you don't put the user and group databases in the public Web tree, *ever*. Several Web search engines out there have proven themselves to be efficient sources for /etc/passwd files unintentionally put on the site. Don't take that risk.

Virtual Hosts

Apache implements a very clean way of handling *virtual hosts*, which is the name for the mechanism for being able to serve more than one host on a particular machine. Due to a limitation in HTTP, this is accomplished currently by assigning more than one IP number to a machine, and then having Apache bind differently to those different IP numbers. For example, a UNIX box might have 204.122.133.1, 204.122.133.2, and 204.122.133.3 pointing to it, with www.host1.com bound to the first, www.host2.com bound to the second, and www.host3.com bound to the third.

This book will not go into how to configure additional IP addresses for your machine, since that varies completely from platform to platform. Your user manual for the operating system should contain information about configuring additional numbers—this is a standard capability on just about all systems these days.

Virtual hosts are configured using a container in httpd.conf. They look something like this:

```
<VirtualHost www.host1.com>
DocumentRoot /www/htdocs/host1/
TransferLog logs/access.host1
ErrorLog logs/error.host1
</VirtualHost>
```

The attribute in the `VirtualHost` tag is the hostname, which the server looks up to get an IP address. Note that if there is any chance that `www.host1.com` can return more than one number, or if the Web server might have trouble resolving that to an IP number at any point, you might want to use the IP number instead.

Any directives put within the `VirtualHost` container pertain only to requests made to that hostname. The `DocumentRoot` points to a directory which (presumably) contains content specifically for `www.host1.com`.

Each virtual host can have its own access log, its own error log, its own derivative of the other logs out there, its own `Redirect` and `Alias` directives, its own `ServerName` and `ServerAdmin` directives, and more. In fact, the only things it cannot support, out of the core set of directives, are:

> `ServerType`, `UserId`, `GroupId`, `StartServers`, `MaxSpareServers`, `MinSpareServers`, `MaxRequestsPerChild`, `BindAddress`, `PidFile`, `TypesConfig`, and `ServerRoot`.

If you plan on running Apache with a large number of virtual hosts, you need to be careful to watch the process limits; for example, some UNIX platforms only allow processes to open 64 file descriptors at once. An Apache child will consume one file descriptor per logfile per virtual host, so 32 virtual hosts each with its own transfer and error log would quickly cross that limit. You will notice if you are running into problems of this kind if your error logs start reporting errors like `unable to fork()`, or your access logs aren't getting written to at all. Apache does try and call `setrlimit()` to handle this problem on its own, but the system sometimes prevents it from doing so successfully.

Customized Error Messages

Apache can give customized responses in the event of an error. This is controlled using the `ErrorDocument` directive. The syntax is:

```
ErrorDocument <HTTP response code> <action>
```

Where `HTTP response code` is the event which triggers the `action`. The `action` can be:

- A local URI to which the server is internally redirected.
- An external URL to which the client is redirected.
- A text string, which starts with a `'"'`, and where the `%s` variable contains any extra information if available.

For example:

```
ErrorDocument 500 "Ack! We have a problem here: %s.
ErrorDocument 500 /errors/500.cgi
ErrorDocument 500 http://backup.myhost.com/
ErrorDocument 401 /subscribe.html
ErrorDocument 404 /debug/record-broken-links.cgi
```

Two extra CGI variables will be passed to any redirected resource: `REDIRECT_URL` will contain the original URL requested, and `REDIRECT_STATUS` will give the original status that caused the redirection. This will help the script if its job is to try and figure out what caused the error response.

Assorted httpd.conf Settings

There are a couple last configuration options that fell through the cracks.

BindAddress

At startup, Apache will bind to the port it is specified to bind to, for all IP numbers which the box has available. The `BindAddress` directive can be used to specify only a specific IP address to bind to. Using this, one can run multiple copies of Apache, each serving different virtual hosts, instead of having one daemon which can handle all virtual hosts. This is useful if you want to run two web servers with different system user-id's for security and access control reasons.

For example, let's say you have three IP addresses (1.1.1.1, 1.1.1.2, and 1.1.1.3, with 1.1.1.1 being the primary address for the machine), and you want to run three Web servers, yet you want one of them to run as a different user ID than the other two. One would have two sets of configuration files; one would say something like

```
User web3
BindAddress 1.1.1.3
ServerName www.company3.com
DocumentRoot /www/company3/
```

And the other would have

```
User web1
ServerName www.company1.com
DocumentRoot /www/company1/
<VirtualHost 1.1.1.2>
ServerName www.company2.com
DocumenbtRoot /www/company2/
</VirtualHost>
```

If you launch the first, it will only bind to IP address `1.1.1.3`. The second one, since it has no `BindAddress` directive, will bind to the port on all IP addresses. So, you want to launch a server with the first set of config files, then launch another copy of the server with the second set. There would essentially be two servers running.

PidFile
This is the location of the file containing the process-ID for Apache. This file is useful for being able to automate the shutdown or restart of the web server. By default, this is `logs/httpd.pid`. For example, one could shut down the server by saying:

```
cat /usr/local/etc/httpd/logs/httpd.pid ¦ xargs kill -15
```

You might want to move this out of the logs directory and into something like /var, but it's not necessary.

Timeout
This directive specifies the amount of time that the server will wait in-between packets sent before considering the connection "lost." For example, `1200`, the default, means that the server will wait for 20 minutes after sending a packet before it considers the connection dead if no response comes back. Busy servers may wish to turn this down, at the cost of reduced service to low-bandwidth customers. ❖

RUNNING A PERFECT WEB SITE WITH APACHE

This ultimate reference for building and maintaining Internet and intranet Web sites with Apache provides all the software and documentation you need to turn a UNIX PC into a fully functional Web site. Step-by-step instructions show you how to install and configure all aspects of Web servers for the Internet and intranets. And, expert advice guides you through the complete maintenance and securing of your server. You'll also find special coverage on how an intranet can be used as an information center and distributor, a bulletin board, a discussion forum, a workgroup server, or even a place to keep common business forms.

With Que's *Running a Perfect Web Site with Apache*, everything you need to build a Web server is right at your fingertips!

- Master the ins and outs of Apache HTTP Server—the most popular and fastest growing Web server
- Troubleshoot configuration and security problems
- Integrate databases, search engines, SSL, and other powerful advanced features in your Web site

CD-ROM includes:

- The Apache Web Server version 1.1 This is the replacement for the popular NCSA 1.3 Server with the following additional features:
  - Faster and more efficient than NCSA's Web Server
  - DBM Databases for authentication
  - Allows multiple aliases and multi-homed servers so you can run more than one site from the machine or run a site that requires several different machines
  - Fixes many bugs and security holes in NCSA
- CGI Tools: Perl 5 and PerFormCGI are both vital tools for Web administrators running interactive forms and CGI on Web servers
- Over 300 pages of additional reference information about HTML, CGI, and JavaScript for Webmasters and Internet programmers. This includes five chapters from each of *Special Edition Using HTML, Second Edition; Special Edition Using CGI; and Special Edition Using JavaScript.*

User Level

New Casual Accomplished Expert

$49.99 USA / $67.99 CAN / £46.99 Net UK (inc of VAT)

Category: Communications/Online—Internet
Covers: Version 1.1 for UNIX

QUE®

Visit us on the Internet at: http://www.mcp.com/que

0 29236 07454 8

ISBN 0-7897-0745-4

94999

9 780789 707451