

Building a Linux Internet Server

.....

George Eckel
Chris Hare

**New
Riders**

New Riders Publishing, Indianapolis, Indiana

mc

INDIANA UNIVERSITY
LIBRARIES
BLOOMINGTON

QA
76.9
.C55
H36
1995

Building a Linux Internet Server

By George Eckel and Chris Hare

Published by:

New Riders Publishing

201 West 103rd Street

Indianapolis, IN 46290 USA

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without written permission from the publisher, except for the inclusion of brief quotations in a review.

Copyright © 1995 by New Riders Publishing

Printed in the United States of America 1 2 3 4 5 6 7 8 9 0

Library of Congress Cataloging-in-Publication Data

Hare, R. Christopher E., 1962-

Building a Linux Internet server / R. Christopher E. haare, George Eckel.

p. cm.

Includes index.

ISBN 1-56205-525-9

1. Client/server computing. 2. Linux. 3. Internet (Computer network) I. Eckel, George, 1954- . II. Title.

QA76.9.C55H36 1995

004.6'7--dc20

95-38424

CIP

Warning and Disclaimer

This book is designed to provide information about the Internet. Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied.

The information is provided on an "as is" basis. The authors and New Riders Publishing shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or from the use of the disks or programs that may accompany it.

Publisher	Don Fowley
Associate Publisher	Tim Huddleston
Marketing Manager	Ray Robinson
Acquisitions Manager	Jim LeValley
Managing Editor	Tad Ringo

Product Development Specialist

Emmett Dulaney

Software Specialist

Steve Weiss

Production Editor

Amy Bezek

Copy Editor

Phil Worthington

Associate Marketing Manager

Tamara Apple

Acquisitions Coordinator

Tracy Turgeson

Publisher's Assistant

Karen Opal

Cover Designer

Karen Ruggles

Cover Illustrator

Jerry Blank

Book Designer

Sandra Schroeder

Production Team Supervisor

Laurie Casey

Graphic Image Specialists

Dennis Clay Hager

Jason Hand

Clint Lahnen

Dennis Sheehan

Production Analysts

Angela Bannan

Bobbi Satterfield

Mary Beth Wakefield

Production Team

Kim Cofer

Kevin Foltz

Aleata Howard

Erika Millen

Regina Rexrode

Erich J. Richter

Christine Tyner

Karen Walsh

Robert Wolf

Indexer

Brad Herriman

reck
LAGK6271

About the Authors

George Eckel has worked on a number of books for Macmillan Publishing, including *Memory Management for All of Us*, *Inside Windows NT*, and *Inside UNIX*. As a professional writer, George has worked as a consultant for Hewlett-Packard, Sun Microsystems, Informix, AT&T, Time-Warner, and Silicon Graphics, Inc. Presently, he is a part of the core team that developed and demonstrated the world's first interactive television system. George is now working on the Japanese (NTT) implementation of interactive television, and running a consulting business for companies setting up services on the Internet. You can reach him at geckel@warp.engr.sgi.com, or 510-820-5243.

George is the father to three children—Madeline, Nathalie, and Genevieve—and happy husband of Shirlee, whose patience and support helped make this book possible.

Chris Hare is the Operations Manager for a Canadian national Internet service provider, i*internet. He started working in computer-based technology in 1986, after studying Health Sciences. Since that time, he has worked in programming, system administration, quality assurance, training, network management, consulting, and technical management positions.

Chris became the first SCO Authorized Training instructor in Canada in 1988, and has taught Unix courses all over the world for his previous employers and for SCO. As a professional writer, Chris has authored almost 20 articles for *Sys Admin* magazine and has coauthored several books for New Riders Publishing, including *Inside UNIX*, *Internet Firewalls and Network Security*, and the yet-to-be-released *Unix Professional Reference*.

Chris lives in Ottawa, Canada with his wife, Terri, and their children, Meagan and Matthew.

Trademark Acknowledgments

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. New Riders Publishing cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Dedication

In memorium for Nancy M. Eckel...
For all the lovely memories...
All the lovely times...
Until we are together again...

Acknowledgments

Chris Hare would like to take the opportunity to thank a few people. First off, to George Eckel for such a wonderful manuscript to work with. Next, to Emmett Dulaney and the staff at New Riders for their encouragement. They are wonderful people as always.

A heartfelt thank you goes to Sohail Khan of Computers and More in Kanata, Ontario for assistance in providing some much-needed computing and network hardware to duplicate and test the various components of this book on the Linux Operating System.

Finally, thanks to wife Terri for supporting me in yet another authoring project. The 3 a.m. mornings are almost becoming habit for her (and me). Nevertheless, these projects wouldn't be possible but for her encouragement.

Contents at a Glance

Introduction	1
Part I Using the Internet for Business	
1 What Is the Internet?	9
2 Business and the Internet	17
3 Advertising on the Internet	35
Part II Getting Connected to the Internet	
4 Understanding Basic Technology	45
5 Tools for Accessing Internet Information	57
6 Connecting to the Internet	69
7 Legal Considerations	85
8 Making Your LAN Secure	91
Part III Setting Up and Managing Internet Services	
9 Understanding Linux	101
10 Setting Up an FTP Service	119
11 Setting Up a More Secure FTP Service	131
12 Managing an FTP Service	141
13 Setting Up a freeWAIS Service	161
14 Managing a freeWAIS Service	177
15 Setting Up and Managing ZDIST	193
16 Setting Up a Gopher Service	201
17 Managing a Gopher Service	211
18 Setting Up a WWW Service	243
19 Managing a WWW Service	261
Part IV Appendixes	
A Gopher Services	289
B Free Software on the Internet	313
Glossary	317
Bibliography	321
Index	323

Table of Contents

Introduction

How This Book Is Organized	4
New Riders Publishing	6

Part I: Using The Internet for Business **7**

1 What Is the Internet?	9
A Revolution in Communications	10
A Short History of the Internet	10
Internationalization of the Internet	13
Who Governs the Internet?	14
Who Pays for the Internet?	15
What You Find on the Internet	15
What Can You Do on the Internet?	16
Summary	16
2 Business and the Internet	17
Who Is Using the Internet?	18
Profile of PC and Online Services' Penetration into Homes	18
Profile of Internet Usage	19
Profiling the Typical Internet User	19
Looking at Internet Trends	21
Business Uses of the Internet	22
Internal Uses of the Internet	22
Offering External Internet Services	24
A New Principle in Customer Relations	25
Telecommuting	26
Keeping Up with the World	26
Reducing Costs	26
What Businesses Are on the Internet	26
Internet Shopping Network	28
Federal Express	29
Hewlett Packard	29
CommerceNet	30
Paramount Pictures	30
Canadian Airlines	31
Other Business Examples	31
Summary	33
3 Advertising on the Internet	35
How Not to Advertise on the Internet	36
How to Advertise on the Internet	36

Three Advertising Principles	36
Throwing Away Old Advertising Principles	37
Multimedia Internet Interfaces Help Motivate Users	37
Where Business Fits In	39
Appropriate Ways to Advertise on the Internet	39
GNN	39
Awards	40
CUI	40
EINet	40
Special Interest Connections	40
WWW Library	40
WebCrawler	40
WWW Worm	40
Home Pages Broker	40
CommerceNet	40
Shopping Network	41
Commercial Services	41
Internet Info	41
Internet Business Center	41
Interesting Business Sites	41
Become Active in the Internet	41
Newsgroups with Which to Advertise	41
Create Your Own Newsgroup	42
Send Your Service Announcement to Catalog Maintainers	42
Summary	42

Part II: Getting Connected to the Internet

43

4 Understanding Basic Technology

45

Internet Connection Technology Primer	46
Using High-Speed Leased Lines	46
Dialing Up the Internet	48
Running TCP/IP on Your Personal Computer	48
A Brief Introduction to TCP/IP	49
IP	50
TCP	51
UDP	52
Naming Addresses	52
Big Brother: inetd	53
Using Daemons	54
How Service Requests and Replies Happen	54
Summary	55

5 Tools for Accessing Internet Information	57
Using finger	58
Using telnet	59
Getting and Sending E-mail	59
Offering an FTP Service	59
Offering an Archie Service	62
Offering a Gopher Service	62
Offering a WAIS Service	64
Offering a WWW Service	65
Deciding What Service to Offer	67
Summary	67
6 Connecting to the Internet	69
An Overview of Internet Service Categories	70
Categories of Service	71
Defining Your Company's Needs	72
Understanding Bandwidth Terminology	73
Using a Dedicated or Shared Telephone Line	73
Considering Connection Costs	74
System and Personnel Requirements	75
Determining Hardware Requirements	75
Manning the Post	77
Making the Decision	77
Determining Hardware Requirements for Dial Up and Proxy Connections	78
Using the Dial Up Connection	78
Looking at Internet Providers for Dial Up Connections	79
Using SLIP and PPP	80
Using ISDN	81
Using Switched 56k	81
Using a Leased Line	82
Using Service Providers	83
Getting an Internet Address	83
Summary	84
7 Legal Considerations	85
Protecting Your Materials	86
Trademarks	86
Patented Designs	86
Trade Secrets	87
Copyrighted Materials	87
Avoiding Legal Problems	88
Seek Permission	88
Avoid Libel	88

Export Problems	88
Avoid Pornography	88
Pirated Software	88
Checking Uploaded Material	89
Summary	89

8 Making Your LAN Secure 91

Guarding the Gateway	92
Kinds of Firewalls	92
Inline Host	92
Monitoring Host	92
Getting Firewall Software	93
Using the Firewall Toolkit	93
Using the CERN WWW Service in Proxy Mode	94
Using SOCKS	95
Placing the Gateway Computer	95
Inline Configuration	96
Monitored Configuration	97
A Grab Bag of Security Addenda	97
Summary	98

Part III: Setting Up and Managing Internet Services 99

9 Understanding Linux 101

A Brief History of Linux	102
Supporting Hardware	102
Processor	103
Memory Requirements	103
Hard Disk Requirements	103
Video Requirements	104
Mice and Pointing Devices	104
CD-ROM Drives	105
Tape Drives	105
Printer Support	105
Network Interface Cards	105
Sound Devices	105
Online Documentation	105
Installing Linux	106
Creating a Boot and Root Disk	106
Creating Hard Disk Partitions	108
Installing the Software	112
Restarting	116
Summary	117

10 Setting Up an FTP Service	119
Naming Your Service	120
Configuring an FTP Service	121
Setting Up the Anonymous User	122
Reorganizing Your Files	123
Login Directory Permissions	124
Naming the System Administrator	126
Trying Out Your FTP Archive	127
What Your Session Should Look Like	127
What Might Go Wrong	128
Summary	129
11 Setting Up a More Secure FTP Service	131
Getting the Source Code	132
Installing the Service	133
Installation Notes	137
New Directories	137
New Command-Line Options	139
Summary	139
12 Managing an FTP Service	141
Organizing Your Archive Files	142
Providing Navigational Aids	143
Using README Files	143
Using Aliases	144
Defining Paths	145
Using Symbolic Links	145
Using Is-IR Files	146
Using Archie	146
Checking syslog Files	147
Checking ftpadmin E-Mail	148
Configuring the WU Version of ftpd	148
Restricting User Access	149
Restricting User Access Based on CPU Load	150
Restricting Access for Security Reasons	151
Restricting Users' Actions	153
Creating and Managing Groups of Users	153
Sending Messages to FTP Users	154
Logging User Events	157
Restricting Uploading	158
Using the Administrative Tools	159
Using ftpshut	159
Using ftpwho	160
Using ftpcount	160
Summary	160

13 Setting Up a freeWAIS Service	161
Understanding the Parts of freeWAIS	162
An Overview of freeWAIS	164
Getting the Software	164
Understanding Files in the src Directory	167
Editing the Makefile	167
Editing the Directory Paths	168
Setting Library Variables	168
Setting Compiler Options	168
Building X and Motif Clients	168
Building the Service	172
Testing Your freeWAIS Service	172
Indexing Files	172
Using freeWAIS Clients to Find Documents	173
Starting and Checking Your Service	173
Starting waissserver with inetd	173
Starting waissserver without inetd	174
Testing waissserver	174
Letting the World Know about Your Service	175
Altering the Service and Clients	175
Summary	175
14 Managing a freeWAIS Service	177
Making waissserver Secure	178
Changing the User	178
Restricting Service Access	178
Controlling Access to Sources	179
A Security Caveat	180
Providing Content	180
Understanding What waisindex Does	181
WAIS Index Files	182
Determining the Relevance Ranking	184
Indexing Different Data File Types	185
Indexing Undefined Data File Types	187
Making Data Files Available to WWW Browsers	187
Indexing Multiple Files for WWW Browsers	188
Linking Files	188
Using Synonyms	188
Specifying Source Options	189
Logging Messages During Indexing	190
Indexing Controls	190
Relative Position of Words	190
Freeing Memory and Disk Space	190
Summary	191

15 Setting Up and Managing ZDIST	193
ZDIST Distribution Software	194
ZServer	194
Overview of the ZServer Installation	194
Configuring ZServer	195
Currently Supported Search Engines	198
Starting ZServer	198
Testing ZServer	198
Summary	199
16 Setting Up a Gopher Service	201
Versions of Gopher	203
Introducing Gopher+	203
Understanding Gopher and Gopher+ File Attributes	204
Using Gopher+ Forms	206
Choosing the Service Software	206
Licensing Gopher+	207
Small Business Fee Structure	207
Standard Fee Structure	207
Getting the Service Software	207
Gazing into the Future of Gopher Services	210
The Future of GopherVR	210
Summary	210
17 Managing a Gopher Service	211
Editing the gopherd.conf File	212
Creating an Alias for the Service	212
Setting Cache Times	212
Checking File Decoders	212
Limiting Concurrent Sessions	212
Setting Maximum Number of Clients	213
Figuring Out File Contents	213
Hiding Files Based on File Types	215
Using Character Patterns to Hide Files	215
Splitting Files into Multiple Parts	215
Editing gopherdlocal.conf	216
Identifying System Administrators	216
Adding an Abstract	216
Including Gopher Site Information	216
Defining User Messages	217
Defining Access Permission	217
Authenticating Users	218
Configuring the Compile-Time Options	218
Editing Makefile.config	219
Editing conf.h	222

International Options	224
Installing Programs for the Gopher Client	224
Compiling the Gopher Service and Client	225
Installing the Gopher Client and Service	225
Starting Your Gopher+ Service	225
Using Other Command-Line Options	226
Starting gopherd with inetd	227
Testing Your Service	229
Using telnet	229
Using gopherls	229
Publicizing Your Gopher Service	230
Using Log Files	230
Getting the Big Picture	232
Updating Log Files	233
Constructing Your Menu System	233
Creating Directories and Files	233
Naming Files and Directories	234
Using .names Files	234
Creating Links to Other Files and Directories	235
Using .cap Files	236
Linking with Other Services	236
Linking to telnet Services	236
Linking to Other Services	237
Linking to WAIS Services	237
Using Links to Execute Scripts	237
Searching Indexes	238
Structuring Your File System	238
Using Files Common to Other Services	239
Using FTP Files	239
Creating Forms	239
Summary	241

18 Setting Up a WWW Service 243

Getting the Client Software	244
Getting Service Software	246
Setting Up the NCSA WWW Service	247
Recompiling the Linux WWW Service Binary	248
Configuring the WWW Service	249
Setting Up httpd.conf-dist	249
Setting Up srm.conf	253
Setting Up access.conf-dist	255
Installing httpd	257
Starting Your WWW Service	257

Running httpd under inetd	258
Running httpd as a Daemon	258
Testing httpd	259
Letting Others Know	259
Summary	260
19 Managing a WWW Service	261
Constructing URLs	262
Creating a Home Page	262
Writing Your Home Page	263
Managing Log Files	264
Using Log Analysis Tools	264
Archiving Log Files	267
Adding New Drives	268
Advanced Web Management	268
Handling More File Types	269
Enabling Automatic Directory Indexing	270
Extending WWW Resources	272
Using Gateways in cgi-bin	272
Service and CGI Script Interaction	273
Getting Input to the Gateway Script	273
Using CGI Test Scripts	275
Getting Information from the Gateway Script	275
Guidelines for Writing Gateway Scripts	275
Gateway Security	276
Consider the eval Statement	276
Eliminate Server-Side Includes	276
Avoid popen() and system()	276
Free Gateways on the Internet	276
Genera	276
Hytelnet	277
man2html	277
Oracle	277
Making a WAIS Service Available to WWW Browsers	277
The Do Nothing Option	277
Using WAIS-to-WWW Gateway Freeware	278
Creating Your Own Gateway	278
Buying Gateway Software	278
Choosing a Gateway	278
Managing WWW Security	279
Restricting Access by Domain Name	279
Requiring Passwords	282
Compromising Security	284
Summary	285

A Gopher Services	289
Gopher Sites	290
Gopher Sites by Technical Area	291
Agriculture	291
Astronomy and Astrophysics	292
Biology and Biosciences	293
Botany	296
Chemistry	296
Computers	297
Environment	298
Geology and Oceanography	299
Medical	300
Physics	302
Nontechnical Categories	303
Art	303
Books and Magazines	304
Education and Research	304
Employment Opportunities and Resume Postings	305
Federal Government	306
Federal Laboratories	306
General Reference Resources	307
Internet/Cyberspace	308
Legal or Law-Related	309
Music	310
Technical Reports, Publications, Journals, and Newsletters	310
B Free Software on the Internet	313
Glossary	317
Bibliography	321
Index	323

c h a p t e r

9

Understanding Linux

As the most sought after and widely used public implementation of Unix, Linux is a complete and ever-growing entity with a worldwide following all its own.

This chapter introduces Linux, provides a brief history of Linux, and explains how to install the Slackware version of Linux provided on the CD-ROM accompanying this book.

A Brief History of Linux

Linux traditionally has been associated with the Unix geniuses who sit in front of their computers night and day. Linux has gained increasingly widespread use, however, in business, academia, and personal productivity.

Linux (pronounced with a short i, as in LIH-nucks) is a clone of the Unix operating system, and it runs on Intel 80386 and 80486 computers. It supports a wide range of software, from TeX to X Windows to the GNU C/C++ compiler to TCP/IP. It's a versatile, bona fide implementation of Unix, freely distributed according to the terms of the Free Software Foundation's General Public License.

Linux can turn any 386 or 486 PC into a workstation. It gives you the full power of Unix at your fingertips. More businesses implement Linux every day, particularly in the realm of financial services, hospital administration, and distributed computing.

What makes Linux so different—and so appealing to everyone—is that it's a free implementation of Unix. It was, and continues to be, developed by a group of volunteers, primarily on the Internet, exchanging code, reporting bugs, and fixing problems in an open-ended environment. Given this type of environment, why do people bet their businesses on it? Because they have access to the internal operating system. Patches and updates for any number of different things ranging from additional hardware support to security are available daily, whereas a company might have to wait weeks, months, or even years for the same patches from a commercial vendor.

Unix is one of the most popular operating systems in the world because of its large support base and

distribution. Originally developed as a multitasking system for minicomputers and mainframes in the mid-70s, it has since grown to become one of the most widely used operating systems anywhere, despite its sometimes confusing interface and lack of central standardization.

Versions of Unix exist for many systems, ranging from personal computers to supercomputers such as the Cray Y-MP. Most versions of Unix for personal computers are quite expensive and cumbersome. Commercial implementations range from \$500 to thousands of dollars.

Linux is a freely distributed version of Unix developed primarily by Linus Torvalds at the University of Helsinki in Finland. The first official Linux release was in October 1991. Although relatively young compared to "real Unix," probably as much, if not more, development effort exists for Linux than for most implementations of Unix.

Today, Linux is a complete Unix implementation that offers X Windows, TCP/IP, and most of the freely available software packages. Many Internet service providers use it as a low cost way to provide their users with facilities, and you commonly see INN, C-News, and UUCP also running on it.

One of the most startling facts about Linux is that many people have executed benchmarks on 80486 Linux systems and found them comparable with mid-range workstations from Sun Microsystems and Digital Equipment Corporation.

Supporting Hardware

Before installing the software, you need to be aware of Linux's hardware requirements. If you already

have hardware, chances are it's supported, because Linux supports more hardware and peripherals than some of the commercial implementations do.

Processor

Linux currently supports systems that have an Intel 80386 or 80486 CPU, including all variations of this CPU type, such as 386SX, 486SX, 486DX, and 486DX2. As the Intel Pentium chip supports the 386/486 instructions set, Linux should install and run with little or no extra difficulty. Linux works with non-Intel "clones," such as AMD and Cyrix processors, as well.

If you have a 80386 or 80486SX, you also might want to use a math coprocessor, although you don't have to have one (the Linux kernel can do FPU emulation if you don't). Linux supports all standard FPU couplings, such as IIT, Cyrix FasMath, and Intel coprocessors.

The system motherboard must use ISA or EISA bus architecture. These terms define how the system interfaces with peripherals and other components on the main bus. Most systems sold today use ISA or EISA bus. Linux doesn't currently support IBM's MicroChannel (MCA) bus, found on machines such as the IBM PS/2. Linux also supports systems that use a local bus architecture for faster video and disk access. You should have a standard local bus architecture, such as the VESA Local Bus ("VLB").

Memory Requirements

Compared to other implementations of Unix, Linux requires very little memory. You need at the very least 2 MB of RAM, although 4 MB would be better and is recommended. The more memory you

install on your system, the faster Linux runs. In addition, Linux utilizes all the RAM installed in your system. Linux runs happily with only 4 MB of RAM, including all the bells and whistles such as X Windows, Emacs, and so on. However, more memory is almost as important as a faster processor. Although 8 MB is more than enough for personal use, you might need 16 MB or more if you expect a heavy user load on the system.

Most Linux users allocate a portion of their hard drive as swap space, which acts as virtual RAM. Even if you have a great deal of physical RAM in your machine, you might want to use swap space. Although swap space certainly doesn't replace for actual physical RAM, it can let your system run larger applications by swapping out inactive portions of code to disk.

Hard Disk Requirements

You can run Linux from a floppy disk, if the words slow and cumbersome don't bother you. Considering the almost daily dip in hard disk storage cost, your system probably has some kind of hard disk.

The amount of disk space you need before you can install Linux depends very much on the distribution you install. Some distributions are larger than others. Some are minimal implementations, whereas others are full-blown releases.

You must have an AT-standard (16-bit) controller. The kernel contains support for XT-standard (8-bit) controllers; however, most controllers today are AT-standard. Linux should support all MFM, RLL, ESDI, and IDE controllers. The general rule for non-SCSI hard drive and floppy controllers is that if you can access the drive from MS-DOS or another operating system, you should be able to access it from Linux.

Linux also supports a number of popular SCSI drive controllers, although support for SCSI is more limited because of the wide range of controller interface standards. Supported SCSI controllers include the following:

- Adaptec AHA1542B
- AHA1742A
- AHA1522
- AHA1740
- AHA1740
- Future Domain 1680
- TMC-850
- TMC-950
- Seagate ST-02
- UltraStor SCSI
- Western Digital WD7000FASST.

To install Linux, you must have some free space available on your disk. You can install Linux across multiple disks. Linux can share the disk space with MS-DOS and OS/2 with the Linux Loader. The amount of hard drive space you need depends greatly on your specific needs and the amount of software you want to install.

Compared to other commercial and publicly available Unix implementations that require hundreds of megabytes of disk space, Linux can run in as little as 10 to 20 MB. However, the amount of software and applications you can use is limited by this amount. The documentation that comes with the given distribution should provide the information you need for decision making. If you want to install all of the software from the CD-ROM, you need approximately 700 MB of disk space. If you want

to run a small WWW site, for example, you need a disk partition of approximately 100 MB to install Linux, the HTTPD server, and some HTML files.

Remember to figure in your disk space and some additional space for a swap area. Should Linux not have enough physical memory in which to load all your running applications, some of the contents of RAM are written to the swap space.

The swap space is used as virtual RAM, or memory. As each program loads, memory is consumed. When Linux runs out of memory, it writes some of this memory to the swap area so it can be used by other programs. For example, if you have 16 MB of RAM and a 16 MB swap space, Linux thinks you have 32 MB of RAM. This enables you to execute programs that need a lot of memory.

Video Requirements

As do most operating systems available today, Linux supports all standard Hercules, CGA, EGA, VGA, and Super VGA video cards and monitors for the default text-based interface. In general, if the video card and monitor coupling works under another operating system such as MS-DOS, it should work fine with Linux.

Graphical environments, such as X Window, have their own video hardware requirements. You should refer to the Xfree86 server information included with your distribution for these requirements. Once you have Linux installed, you can also check the online documentation for the requirements of each server.

Mice and Pointing Devices

Generally you use a mouse only in graphical environments. You can associate several Linux applications, however, with a mouse. Linux supports all

standard serial mice, including Logitech, MM series, Mouseman, Microsoft (2-button), and Mouse Systems (3-button). Linux also supports Microsoft, Logitech, and ATIXL busmice, as well as the PS/2 mouse interface. In addition, Linux supports trackball pointing devices that use these interfaces.

CD-ROM Drives

CD-ROM drivers have several different interface types, which has compounded the problem of making them available for Linux. If you have a SCSI host adapter and a SCSI CD-ROM, you shouldn't have any problems with Linux. The same is true for IDE interface drives. Linux also provides support for some proprietary interfaces, including the NEC CDR-74, Sony CDU-541, and Texel DM-3024. Linux supports the Sony internal CDU-31a and the Mistsumi CD-ROM drives as well.

Tape Drives

Like the CD-ROM drive, several versions of tape drive are available. Any SCSI-based tape should operate properly with Linux. Some have been explicitly verified, such as the Sanyo CP150SE; Tandberg 3600, Wangtek 5525ES, 5150ES, and 5099EN with the PC36 adapter. You also could add a Floppy tape module to the system to support the Colorado and other QIC-40 and QIC-80 tape drives that work off the floppy disk controller.

Printer Support

Linux supports the complete range of parallel printers. If you can access your printer under MS-DOS, you most likely could under Linux as well. If your printer requires special software under DOS to work, then it will not work under Linux. The Linux

printing software consists of the Unix standard `lp` and `lpr` software. This enables you to print remotely to printers elsewhere on the network.

Network Interface Cards

Linux supports many popular Ethernet and LAN adapter cards, including 3Com's 3c503, 3c503/16, and 3C509; Novell's NE1000 and NE2000; Western Digital's WD8003 and WD8013; and Hewlett Packard's HP27245, HP27247, and HP27250. Other network interface cards reportedly also work, and Linux developers regularly add support for additional hardware.

Sound Devices

For audiophiles, Linux supports various sound cards and related software, such as CDplayer (a program that can control a CD-ROM drive as a conventional CD player, surprisingly enough), MIDI sequencers and editors (which enable you to compose music for playback through a synthesizer or other MIDI-controlled instrument), and various sound editors for digitized sounds.

Online Documentation

Several forms of documentation are available for Linux, depending on what you have available. On the CD-ROM are three directories: `/guide`, `/howto`, and `/viewer`. If you have MS-DOS and Windows running with a CD-ROM, you should install the Multimedia viewer, in which all the files in the `/howto` directory are prepared as hypertext in a Windows help file format. The Multimedia viewer makes finding information regarding specific questions you might have about Linux very easy. Figure 9.1 illustrates one page in the Multimedia browser.

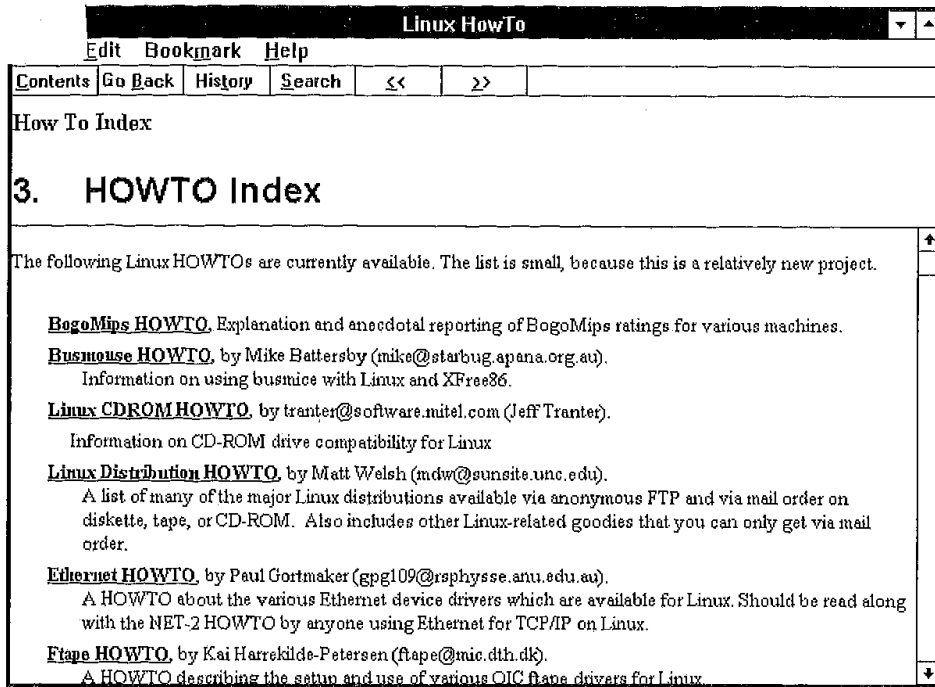


Figure 9.1

*The Multimedia
HOWTO Browser.*

Alternatively, if you don't have MS-Windows, you can find the same information in a text-based configuration in the /howto directory. The /guide directory also contains a wonderful document entitled install.txt, a generic reference manual that covers installing Linux and discusses the different utilities and programs, its history, future direction, development work, and how to operate it.

After you install the system, you have full access to the standard online manual pages using the traditional Unix man command.

Installing Linux

This discussion of how to install Linux focuses on the Linux distribution included with this book. Although it doesn't pretend to be the definitive reference, it does introduce readers to the process of installing Linux on their computers.

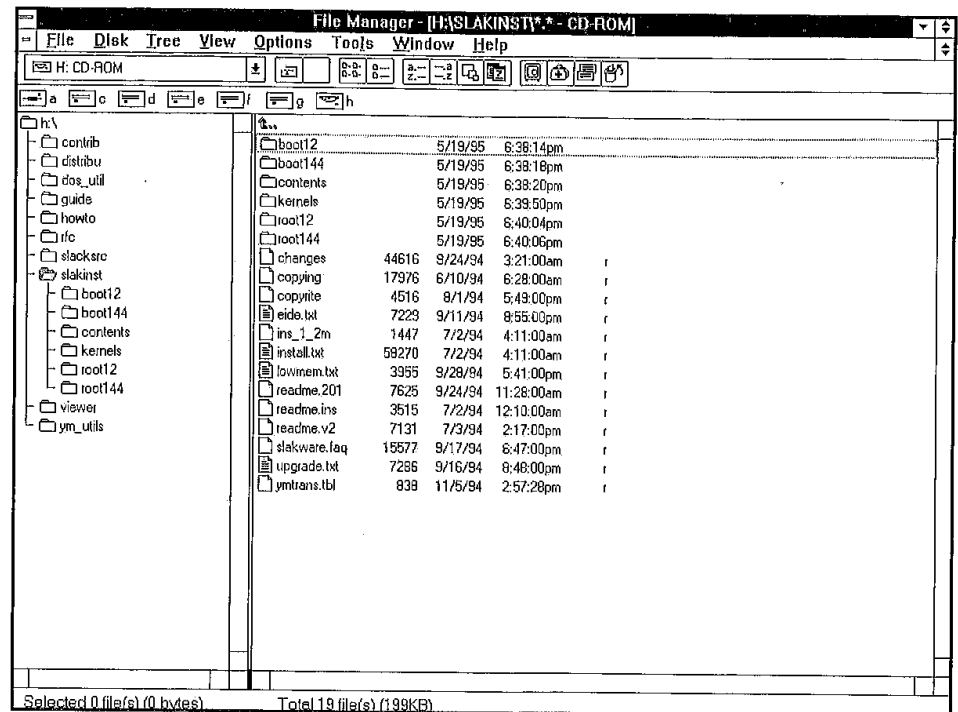
Because Linux can easily exist with DOS, and even share DOS partitions, installing Linux from a DOS environment is easy. If you don't have DOS installed on the computer on which you want to install Linux, you need access to a DOS-based system that has a CD-ROM driver you can access from DOS.

Creating a Boot and Root Disk

This discussion assumes that you have access to a machine that runs MS-DOS and that you can access a CD-ROM drive on this system. The Linux CD-ROM that comes with this book has a directory called /slakinst (see fig. 9.2). The /slakinst directory contains a number of other directories, including /boot12, /boot144, /root12, and /root144. The /bootXX directories include the kernel and boot code for both 5.25- and 3.5-inch floppies.

Figure 9.2

The Linux CD-ROM directory structure.



You should try to use the modern kernel because it contains most of the device drivers. If you cannot boot or install using that kernel, you need to be more specific about the kernel you use. If you have a SCSI device such as your CD-ROM and a network card, then you should try the scsinet kernel. If the selected kernel does not match your system, it may not boot. If that happens, then you should select another kernel and try the boot process again.

To create a boot disk, you must run the DOS utility program rawrite.exe, which you can find in the CD-ROM's /dos_util directory. This program copies the disk image from the CD-ROM to the floppy disk. Figure 9.3 illustrates the execution of the RAWRITE program.

When you use the rawrite command, it accepts no command line arguments, but prompts for the names of the source file and the destination disk drive.

You use the same procedure to create the root file system disk. To create the two disks using the modern kernel and a scsinet root file system, for example, you would use the following command sequence. This command sequence assumes that the CD-ROM is drive H:, and the target disk drive is drive A:.

```
c:\> H:\dos_utils\rawrite
```

```
RaWrite 1.2 Write disk file to raw floppy diskette
```

```
Enter source file name: h:\slakinst\boot144\modern
```

```
Enter destination drive: a:
```

```
Please insert a formatted diskette into drive A:
```

```
➡and press -ENTER-
```

```
Number of sectors per track for this disk is 18
```

```
Verifying image to drive A:. Press ^C to abort.
```

```
Track: 01 \Head: 0 Sector: 13
```

```
c:\> H:\dos_utils\rawrite
```

```
RaWrite 1.2 Write disk file to raw floppy diskette
```

```

MS-DOS Prompt

Type EXIT and press ENTER to quit this MS-DOS prompt and
return to Windows.
Press ALT+TAB to switch to Windows or another application.
Press ALT+ENTER to switch this MS-DOS Prompt between a
window and full screen.

Microsoft(R) MS-DOS(R) Version 6.20
(C)Copyright Microsoft Corp 1981-1993.

C:\WINDOWS>h:
H:\SLAKINST>cd ..\dos_utils
H:\DOS_UTIL>rawrite
Rawrite 1.2 - Write disk file to raw floppy diskette

Enter source file name: h:\slakinst\boot144\modern
Enter destination drive: a:
Please insert a formatted diskette into drive A: and press -ENTER- :
Number of sectors per track for this disk is 18
Writing image to drive A:. Press ^C to abort.
Track: 01 Head: 0 Sector: 13

```

Figure 9.3

*Creating a boot disk
with rawrite.exe.*

```

Enter source file name: h:\slakinst\root144\scsinet
Enter destination drive: a:
Please insert a formatted diskette into drive A:
and press -ENTER-
Number of sectors per track for this disk is 18
Verifying image to drive A:. Press ^C to abort.
Track: 01 \Head: 0 Sector: 13
c:\>

```

Make sure that you clearly label the disks so you don't have problems later during the install. After you create the two disks you need here, you can proceed to boot the Linux system and begin installing.

When you boot from the Linux boot disk, you see a Boot: prompt. Read the screen carefully. If you have no arguments to pass to the boot program, simply press Enter. *Do not change the floppy diskettes yet.*

Pressing Enter at this point loads and prepares the kernel. The modern kernel distribution tries to recognize the different pieces of hardware configured in the system. When the system is ready for you to insert the root file system disk, it prints a message saying so on-screen and waits.

After you insert the root file system floppy, press Enter to load Linux. After Linux loads, a login prompt appears. Log in as root; you don't have or need a password yet.

Creating Hard Disk Partitions

At this point, you have a floppy disk-based system. But you want to install Linux and its application environment on the hard disk, so you need to create some partitions—like MS-DOS and other PC-based operating systems, you must create disk partitions on which to install Linux. The following illustrates running fdisk to create several disk partitions for Linux.

```

# fdisk
Using /dev/hda as default device!

Command (m for help): p

Disk /dev/hda: 16 heads, 63 sectors, 827 cylinders
Units = cylinders of 1008 * 512 bytes

Command (m for help):

```

If you want to create Linux partitions on more than one hard disk, remember to run `fdisk` on each hard disk, because you must set up each partition separately.

A variety of options are available for `fdisk`, the most commonly used one being to specify the name of the hard disk you want to set up. If you have only one disk, `fdisk` uses that device.

Table 9.1 lists a number of commands that `fdisk` offers to enable you to manipulate the partition table. You can obtain a listing of these same commands and descriptions by typing `m` (the `m` command) at the `Command (m for help):` prompt.

You use the `n` command to create a new partition. You probably don't need to worry about most of the other options. You use the `q` command to quit `fdisk` without saving any changes, but to quit `fdisk` and write the changes to the partition table to disk, you use the `w` command.

The first thing you should do now is display your current partition table and write the information down for later reference. Use the `p` command to do this, as shown on the next page:

Table 9.3 `fdisk` Commands

Command	Action
a	Toggles a bootable flag
d	Deletes a partition
l	Lists known partition types
m	Generates a list of these commands and descriptions
n	Adds a new partition
p	Prints the partition table
q	Quits without saving changes
t	Changes a partition's system id
u	Changes display/entry units
v	Verifies the partition table
w	Writes table to disk and exit
x	Extra functionality (experts only)



Command (m for help): p

Disk /dev/hda: 16 heads, 63 sectors, 827 cylinders
Units = cylinders of 1008 * 512 bytes
Command (m for help): p

Disk /dev/hda: 16 heads, 63 sectors, 827 cylinders
Units = cylinders of 1008 * 512 bytes

Device	Boot	Begin	Start	End	Blocks	Id	System
/dev/hda1	1	1	300	151168+	83	DOS-32	bit

Command (m for help):

In the preceding example, a single MS-DOS partition starts at cylinder 1 and ends at cylinder 300, leaving 527 of the total 827 cylinders available for disk partitions.

You need to create two partitions in this disk space: one partition for the root file system, to contain all the Linux utilities (you don't need to have more than one file system for data, but it does give you better control over where files install on your system), and the other partition for the swap space. The following illustrates creating the swap space as the second partition.

Command (m for help): n
Command action
e extended
p primary partition (1-4)

p
Partition number (1-4): 2
First cylinder (301-827): 301
Last cylinder or +size or +sizeM or +sizeK (204-683): 351

Command (m for help):

Now you've created your swap partition. The fdisk program asked you to enter the number for the partition you wanted to create. Because partition 1 already existed, you created a partition

numbered 2. The first cylinder corresponds to the partition's starting point. Leaving a gap between the partitions is pointless.

When fdisk prompts for the last cylinder value, you have several options. You can provide an explicit last cylinder value, or you can respond with several other answers. If you don't know how many cylinders to make the partition but you know you want an 80 MB partition, for example, respond to the prompt with +80M. From this value, fdisk will calculate the number of cylinders.

For your swap space, though, you don't need such a large partition. Even with the values given in the following example, you have a 25 MB swap partition! You can calculate this easily using fdisk's p command's output, as shown in the following:

Command (m for help): p

Disk /dev/hda: 16 heads, 63 sectors, 827 cylinders
Units = cylinders of 1008 * 512 bytes

Device	Boot	Begin	Start	End	Blocks	Id	System
/dev/hda1	1	1	300	151168+	83	DOS-32	bit

The preceding shows that the units for this disk are cylinders of 1008 × 512 bytes (which is 516,096 bytes), and that times 51 cylinders (351–300) gives a size of 26,320,896 bytes (26 MB).

Because this is your swap partition, many distributions of Linux require that you specially identify it for the installation program, for which you use fdisk's t command to set the label on the partition. You must use the distribution that accompanies this book, shown in the following:

Command (m for help): t
Partition number (1-4): 2
Hex code (type L to list codes): L

```

0 Empty 8 AIX 75 PC/IX b7 BSDI fs
1 DOS 12-bit FAT 9 AIX bootable 80 Old MINIX b8
↳BSDI swap
2 XENIX root a OS/2 Boot Manag 81 Linux/MINIX c7
↳Syrinx
3 XENIX usr 40 Venix 80286 82 Linux swap db CP/M
4 DOS 16-bit <32M 51 Novell? 83 Linux native e1
↳DOS access
5 Extended 52 Microport 93 Amoeba e3 DOS R/O
6 DOS 16-bit >=32 63 GNU HURD 94 Amoeba BBT f2 DOS
↳secondary
7 OS/2 HPFS 64 Novell a5 BSD/386 ff BBT
Hex code (type L to list codes):82
Command (m for help): t

```

You use the same procedure to create the root file system. The following illustrates the entire procedure.

```
Command (m for help): p
```

```
Disk /dev/hda: 16 heads, 63 sectors, 827 cylinders
Units = cylinders of 1008 * 512 bytes
```

```

Device Boot Begin Start End Blocks Id System
/dev/hda1 1 1 300 151168+ 83 DOS-32 bit
/dev/hda2 301 301 351 25704 82 Linux swap

```

```
Command (m for help): n
```

```
Command action
```

```
e extended
```

```
p primary partition (1-4)
```

```
p
```

```
Partition number (1-4): 3
```

```
First cylinder (301-827): 352
```

```
Last cylinder or +size or +sizeM or +sizeK (204-
↳683): 827
```

```
Command (m for help): a
```

```
Partition number (1-4): 3
```

```
Command (m for help): p
```

```
Disk /dev/hda: 16 heads, 63 sectors, 827 cylinders
Units = cylinders of 1008 * 512 bytes
```

```

Device Boot Begin Start End Blocks Id System
/dev/hda1 1 1 300 151168+ 83 Linux native
/dev/hda2 301 301 351 25704 82 Linux swap
/dev/hda3 * 352 352 827 239904 83 Linux native

```

```
Command (m for help):w
```

```
#
```

This example shows you the fdisk commands a and w in action for the first time. You use the a command to toggle the Boot flag. You want Linux to be bootable, so select it as the boot partition. You use w to write the partition table back out to the disk and exit fdisk.



Before you continue to the next installation step, write down the information from the partition table. You need it later to complete the installation, and it's essential should you ever need to resolve a problem in the future.

Before you can continue, you should prepare the swap space. Although you can do this during the installation, you really should do so before you begin the installation, especially if you have less than 4 MB of RAM.

Preparing swap partitions consists of two tasks: establishing a file system for the swap, and turning swapping on. To perform the first task, use the mkswap command as follows:

```
# mkswap -c /dev/hda2 25704
```

You use the mkswap command to set up disk partitions or disk files to use as swap space. The -c option instructs mkswap to check the space for bad blocks first. The device name /dev/hda2 is the name

you gave your swap partition. The final argument, 25704, is the size of the partition you created earlier.

Swap space isn't limited to a separate disk partition, although doing it that way is the most common practice. You also can place swap in a file, much like the MS-Windows and OS/2 approach to swapping. To do so, use the commands shown in the following:

```
# dd if=/dev/zero of=swapfile bs=1024 count=8192
# mkswap swapfile 8192
# sync
```

The commands in the preceding example create a file of 8,192 bytes in length, called swapfile, in the current directory. The count= argument for the dd command controls the size of the swap file. If you want more space, increase this value. As when you create the partition, you need to use the mkswap command to create the structure for the swap file. The final sync command causes the information to be written to the disk and clears the disk buffers of any information pending for disk writes.

The second command you use with swapping—the swapon command—turns swapping on for that device or swap file and accepts on the name of the device for which you want to enable swapping. You also can use the swapon command if you want Linux to use additional swap partitions or swap files once it's up and running. The following illustrates use of the swapon command.

```
# swapon swapfile
# swapon /dev/hda2
```

You now have completed the initial work to install the Slackware distribution of Linux on your system.

Installing the Software

Installing the Slackware version of Linux is quite simple, actually much less difficult than setting up the disk partitions. You do need to follow some specific instructions here, however, to install successfully from the CD-ROM.

To begin installing, run the setup command, which results in a screen filled with the information shown in the following.

```
+---Slackware Linux Setup (version HD-2.0.1) --+
| |
| | Welcome to Slackware Linux Setup. |
| |
| | Hint: If you have trouble using the arrow keys
| | on your keyboard, |
| | you can use '+', '-', and TAB instead. Which
| | option would you like? | | |
| | +-----+ |
| | | HELP Read the Slackware Setup HELP file | |
| | | KEYMAP Remap your keyboard | |
| | | QUICK Choose QUICK or VERBOSE install mode
| | [now: VERBOSE] | | |
| | | MAKE TAGS Tagfile customization program | |
| | | TARGET Select target directory [now: /] | |
| | | SOURCE Select source media | |
| | | DISK SETS Decide which disk sets you wish to
| | install | | |
| | | INSTALL Install selected disk sets | |
| | | CONFIGURE Reconfigure your Linux system | |
| | | PKGTOOL Install or remove packages with
| | Pkgtool | | |
| | | EXIT Exit Slackware Linux Setup | |
| | +-----+ |
| | +-----+ |
| | < OK > <Cancel> |
| | +-----+ |
```

You really should read all the HELP files you can regarding installation, but the setup program is configured so that if you start at the top of the list and

work through all the steps, establishing a functional Linux system should pose little or no difficulty.

The HELP option provides information on how to move around the options in the setup program, and what each of the sections do. After you read the Help section, you don't need to change anything for KEYMAP (unless you aren't using a US-English based keyboard), QUICK, MAKE TAGS, or TARGET. The current target is the root file system of your Linux partition, which typically is where you install the distribution.

Selecting the Source Media

The first option you must configure is the SOURCE media, which establishes where you will install from. The following shows the SOURCE menu.

```
+----- SOURCE MEDIA SELECTION -----+
| |
| Where do you plan to install Slackware Linux
| from? |
| |
| +-----+ |
| | 1 Install from a hard drive partition | |
| | 2 Install from floppy disks | |
| | 3 Install via NFS | |
| | 4 Install from a pre-mounted directory | |
| | 5 Install from CD-ROM | |
| +-----+ |
+-----+
| < OK > <Cancel> |
+-----+
```

If you have installed Linux on a different partition, such as under an MS-DOS partition, you can use the first option to install it from that location. Should you want to install this distribution from floppy disk, be prepared: it takes about 75 1.44 floppy disks to install all the components from the Slackware distribution, and that doesn't include any of the

additional utilities and programs included on the CD-ROM.

When you first install, you can't install through NFS (the Network File System) because you won't have networking configured or available yet. You can install from a premounted directory if the Linux distribution files are installed on another Linux disk or the CD-ROM currently is mounted.

You select the last option from the source media selection screen to install from the included CD-ROM. The following shows the CD-ROM selection menu.

```
+----- INSTALLING FROM CD-ROM -----+
| |
| In order to install Slackware Linux from a CD-
| ROM, it must |
| contain the distribution arranged beneath a
| source directory |
| in the same way as if you were to install it
| from a hard |
| drive or NFS. The source directory must contain
| subdirectories |
| for each floppy disk. Your CD-ROM should be
| compatible with |
| this format if it contains a mirror of the
| Slackware FTP site. |
| |
| What type of CD-ROM drive do you have? |
| +-----+ |
| | 1 SCSI [ /dev/scd0 or /dev/scd1 ] | |
| | 2 Sony CDU31A [ /dev/sonycd ] | |
| | 3 Sony 535 [ /dev/cdu535 ] | |
| | 4 Mitsumi [ /dev/mcd ] | |
| | 5 Sound Blaster Pro (Panasonic) [ /dev/sbpcd ]
| |
| +-----+ |
+-----+
| < OK > <Cancel> |
+-----+
```

Remember that Linux doesn't support every CD-ROM drive on the market. If you don't have one that it does support, you might be out of luck. You're best bet aside from buying a new CD-ROM drive is to find someone who can loan you one or make you install floppy disks. Here, you install from a SCSI CD-ROM.

If you choose SCSI CD-ROM, you must select the CD-ROM device from which you will install, as shown in the following:

```
+----- SELECT SCSI DEVICE -----+
| Which SCSI CD-ROM are you using? (If you're not
| sure, |
| select /dev/sr0) |
| +-----+ |
| | /dev/scd0 SCSI cd 0 | |
| | /dev/scd1 SCSI cd 1 | |
| +-----+ |
| < OK > <Cancel> |
+-----+
```

If you aren't sure which one to use, try /dev/scd0, not /dev/sr0 as suggested. The informational prompt in the setup program in the preceding example is incorrect.

Selecting the Linux Distribution

After you select the CD-ROM from which you want to load the distribution, you must indicate which Linux distribution you are loading (use the next selection screen, shown for all CD-ROM install types and illustrated in the following).

```
+----- SELECT SOURCE DIRECTORY-----+
| Now we need to know which directory on the CD
| contains the Slackware |
| sources. This location may vary depending on the
| cd you have. There |
```

```
| are default selections for the Slackware
| Professional CD (including |
| an option to run mostly from the CD), InfoMagic
| CD, TransAmeritech |
| CD, and the Linux Quarterly CD-ROM. There may be
| other directories |
| containing other versions - enter a custom
| directory name if you |
| like. Which option would you like? |
| +-----+ |
| | slakware Slackware Pro: install to HD | |
| | slackpro Slackware Pro: run from CD | |
| | slackware.111 TransAmeritech CD | |
| | packages/slackware Linux Quarterly CD-ROM | |
| | distributions/slackware InfoMagic CD-ROM | |
| | Custom Enter your own directory name | |
| +-----+ |
| < OK > <Cancel> |
+-----+
```

The distribution included with this book is the Slackware 2.0.1 release, provided by InfoMagic. So here, the logical choice is the InfoMagic CD-ROM selection. However, the CD-ROM has been prepared with DOS style short file names to be DOS-readable as well. Names like "distribution" are saved as "distribu" on the CD-ROM. Some CD-ROMS won't be able to find this distribution, and will print a message like the one shown in the following:

```
+----- ERROR: DIRECTORY NOT FOUND -----+
| The Slackware source directory could not be
| found. |
| You'll have to reconfigure your source media
| before |
| you can install. |
| |
| +-----+ |
| < OK > |
+-----+
```

If you're up against such a situation, work through these screens again and from the selection menu and choose "Custom" from the SOURCE menu, then enter the following path:

```
/distrib/slackwar
```

Using this path causes the installation program to look in the directory /distrib/slackwar, which it can find, and then to start the installation.

Selecting the Disk Sets

The next phase is to select the different disk sets you want to install. You shouldn't just install them all, because the CD-ROM easily contains more than 500 MB of uncompressed utilities and programs. The following shows the disk set windows.

```
+-----+
| |
| |
| Use the spacebar to select the disk sets you
| wish to install. |
| You can use the UP/DOWN arrows to see all the
| possible choices. |
| Press the ENTER key when you are finished. If
| you need to |
| install a disk set that is not listed here,
| check the box for |
| custom additional disk sets. |
| |
| +-----+
| | [ ] CUS Also prompt for CUSTOM disk sets | |
| | [X] A Base Linux system | |
| | [ ] AP Various Applications that don't need X
| |
| | [ ] D Program Development (C, C++, Lisp, Perl,
| | etc.) | |
| | [ ] E GNU Emacs | |
| | [ ] F FAQ lists, HOWTO documentation | |
| | [ ] I Info files readable with info, JED, or
| | Emacs | |
```

```
| +-----v(+)-----+ |
| +-----+ |
| | < OK > <Cancel> | |
| +-----+ |
```

The CD-ROM contains 17 disk sets, not including any customs sets. To select the one you want to install, highlight the option you want to select and press the spacebar.

Installing the Disk Sets

After you choose the disk sets you want to install, select INSTALL from the setup menu to start the installation. The program installs all disk sets and applications marked as required by Linux without user intervention. The installation program prompts you for each optional package as to whether you want to select the option. Choosing yes installs the package; choosing otherwise doesn't.

Configuring your Linux System

With the disk sets installed, you must configure the system and some components to ensure that it reboots after you finish. The steps in this configuration include the following:

- Creating a Linux Boot disk
- Modem configuration
- Mouse configuration
- Network configuration (if installed)
- Linux Loader configuration

You definitely should go through the configuration options. Configuring the Linux Loader (LILO) and making a boot disk are musts. If you don't have a modem or a mouse, however, you don't need to install those components.

At this point, you can exit the setup command, because installation is now complete.

Restarting

Now that the installation is complete, you must reboot your system so that you can boot off your hard disk (see the following for the necessary commands).

```
# sync
# reboot
```

These commands flush the disk buffers, and reboot the system. Any currently running processes stop and the system restarts. As the system restarts, you see messages that resemble those in the following, depending upon your system configuration:

```
Kernel logging (proc) started.
Console: colour EGA+ 80x25, 8 virtual consoles
Serial driver version 4.00 with no serial options
➔ena
tty00 at 0x03f8 (irq = 4) is a 16450
tty01 at 0x02f8 (irq = 3) is a 16450
lp_init: lp1 exists, using polling driver
ftape: allocated 3 buffers alligned at: 001F8000
Calibrating delay loop.. ok - 33.22 BogoMips
aha152x: BIOS test: passed, porttest: ok, auto
➔config
aha152x: vital data: PORTBASE=0x340, IRQ=11, SCSI
➔ID=
scsi0 : Adaptec 152x SCSI driver; $Revision: 1.0 $
scsi : 1 hosts.
Vendor: SONY Model: CD-ROM CDU-55S Rev: 1.0
Type: CD-ROM ANSI SCS
Detected scsi CD-ROM sr0 at scsi0, id 3, lun 0
scsi : detected 0 SCSI disks 0 tapes 1 CD-ROM drive
➔t
Scd sectorsize = 2048 bytes
```

```
Memory: 14508k/16384k available (780k kernel code,
➔38
This processor honours the WP bit even when in
➔superv
Floppy drive(s): fd0 is 1.44M
Swansea University Computer Society NET3.014
IP Protocols: ICMP, UDP, TCP
PPP: version 0.2.7 (4 channels) NEW_TTY_DRIVERS
➔OPTIM
TCP compression code copyright 1989 Regents of the
➔Un
PPP line discipline registered.
SLIP: version 0.7.5-NET3.014-NEWTTY (4 channels)
CSLIP: code copyright 1989 Regents of the Univer-
➔sity
eth0: 3c509 at 0x300 tag 1, 10baseT port, address
➔00
3c509.c:pl15k 3/5/94 becker@super.org
Checking 386/387 coupling... Ok, fpu using
excepti➔on
Linux version 1.1.18 (root@fuzzy) #4 Tue Sep 13
18:35
Partition check:
hda: hda1 hda2 hda3
VFS: Mounted root (ext2 filesystem) readonly.
Adding Swap: 25700k swap-space
```

```
Welcome to Linux 1.1.18
login:
```

The installation of your system is complete. You can now log in as root to perform any other necessary configuration tasks. You should create a user for yourself rather than use root all the time. To add a user, use the adduser command. You can obtain instructions on adduser by typing the following command:

```
man adduser
```

The process of adding a user is important because you should not log in and work regularly as the root user. The adduser command prompts for you for

the information that it requires. The process of adding a user is demonstrated in the following:

```
pc login: root
Last login: Sun Aug 13 23:59:44 on tty1
Linux 1.0.9. (POSIX).
You have mail.
pc:~# adduser
```

Adding a new user. The username should be not
➔ exceed 8 characters
in length, or you may run into problems later.

Enter login name for new account (^C to quit):
chrish

Editing information for new user [chrish]

Full Name: Chris Hare
GID [100]:

Checking for an available UID after 500

First unused uid is 501

UID [501]:

Home Directory [/home/chrish]:

Shell [/bin/bash]:

Password [chrish]: aaaaa

Information for new user [chrish]:
Home directory: [/home/chrish] Shell: [/bin/bash]
Password: [aaaaa] uid: [501] gid: [100]

Is this correct? [y/N]: y

Adding login [chrish] and making directory [/home/
chrish]

Adding the files from the /etc/skel directory:

```
./.kermrc -> /home/chrish/./.kermrc
./.less -> /home/chrish/./.less
./.lessrc -> /home/chrish/./.lessrc
./.term -> /home/chrish/./.term
./.term/termrc -> /home/chrish/./.term/termrc
./.emacs -> /home/chrish/./.emacs
```

pc:~#

Because your user is now created, you can perform any other system administrator tasks, or you can log out and log back in using your new user name.

Summary

This chapter presented Linux, its general hardware requirements, and how to install the Slackware distribution included with this book. Anyone who has been around Unix for any period of time will be impressed by Linux. New users to the Unix environment will wish they had some of this functionality at their office.

If you experience difficulties or have questions regarding the installation, your best source of reference is the HOWTO documentation in the Multimedia Help viewer for Windows, which is included on the CD-ROM.

c h a p t e r

10

Setting Up an FTP Service

Finally, you're at a point where you can actually make a service work. Congratulations! If you're reading this chapter, you believe your company can benefit from an FTP (File Transfer Protocol) service. FTP is, in fact, one of the most widely used services on the Internet. Nearly every host has an FTP client or service—not too surprising when you consider that the FTP service, `ftpd` (or `in.ftpd`), comes with most standard flavors of Unix. You can usually find `ftpd` in the `/usr/etc` (or `/etc`) directory.



If you do not want to offer an FTP service on your server, skip to the chapters that discuss the services your company wants to offer.

ftpd is great for services that do not require strict security and do not handle large archives. If either issue matters for you, however, you probably would benefit from implementing the WU version of ftpd. (WU stands for Washington University in St. Louis, MO.) The next chapter discusses this service. Read on, however, because many things in the standard implementation of ftpd are also true for the WU ftpd daemon.

You can offer two kinds of FTP service: *user oriented* and *anonymous*. In the first case, users use their user IDs to log in to an FTP service; in the second case, they use the generic user ID, *anonymous*.

You really don't want to implement a user-oriented service because of the danger of security breaches. User IDs are broadcast over the Internet in plain text, which makes them vulnerable to mischievous hackers. If they can find a user's ID and password, they can use the system with the permissions of the user whose ID was detected.

You can restrict the permissions of specific users. There is, however, a significant administration task of establishing permissions for multiple user IDs. You might very well establish such permissions for people within your company, or even for people in a company working on a joint project with your company, but certainly not for people in the Internet community.

With anonymous FTP, you can restrict the kinds of actions users can take by defining the permissions of the anonymous user ID accordingly, and you can

restrict users' access to your file system. Although you might give some of your employees user-oriented access to your FTP service, we will not even consider offering anything but an anonymous FTP service to the Internet community.

Naming Your Service

There is a de facto standard for naming FTP services. By no means is it required, but it is the name users will try first and find easiest to remember. The format is

`ftp.domain.type`

in which *domain* and *type* are variables. *Domain* represents the alias domain name of your FTP server (often your company's name). *Type* represents the type of organization yours is; for example, *edu* is used for educational institutions, *com* is used for businesses. The following FTP service (of the company supporting the WAIS service) is a good example of an FTP name:

`ftp.wais.com`

Naming your FTP service `www.domain.type`, or `abc.domain.type`, or `zzz.domain.type` is not illegal, but it is not a good idea. You also do not want to use the real name of your host server in the name of your FTP service. If you change your FTP service from one server to another, you do not want to have to tell everyone accustomed to the old name to switch to the name of the new FTP server (which might change again in the future). You could easily lose your standard user base. Instead, you should use an alias for your server. To set up an alias for a server, edit the domain name services (DNS) database files by adding a line similar to the following:

```
ftp IN ftp.domain.type real.domain.type.
```

Substitute your company's domain name for *domain*. Make sure that *real.domain.name* is the real domain name of the FTP server. Note the period that ends the line.

If you change your server, there is a lag time on the Internet determined by the time-to-live defined by your domain name service. The time-to-live variable is an expiration date for Internet servers. When it expires, other name servers must release data about your zone from their cache. If you plan to switch servers, reduce the time-to-live variable accordingly.

Configuring an FTP Service

inetd, the big brother of services, invokes ftpd. It watches port 21 for TCP packets. For every service request—TCP packet—that arrives at port 21, inetd creates a new copy of ftpd to service the request.



For more information about inetd, refer to Chapter 4, "Understanding Basic Technology."

Many implementations of Unix already have an FTP service, ftpd, set up in inetd's configuration file, inetd.config. You often find inetd.config in the /etc directory. Some Unix implementations have a sample line in inetd.config (commented out) that sets up ftpd. If your configuration file has such a sample line, uncomment it. If you do not have any configuration information for ftpd in your inetd.config file, add a line in the file similar to the following:

```
ftp stream tcp nowait root /usr/etc/ftpd ftpd -l
```

Chapter 4 includes a thorough explanation of the syntax of this line. (And soon we will look at the command-line options for ftpd.) Suffice it to say that this entry prescribes that TCP is responsible for reordering the FTP packets when they arrive at the client, that inetd should spawn a new service for each FTP request, that /usr/etc is the directory where ftpd lives, and that you want to start ftpd with the -l (logging) option. The entry in your inetd.config file might have a different location for ftpd, and might use different command line options.

Depending on the computer platform, ftpd has several other options, as explained in the following table:

<i>ftpd Option</i>	<i>Description</i>
-l	Turns on logging.
-d	Writes verbose debugging information in syslog.
-t <i>seconds</i>	Sets the timeout period in seconds. The default is 15 minutes. In other words, if the FTP session is inactive for 15 minutes, ftpd terminates it.
-T <i>seconds</i>	Sets the maximum timeout period, in seconds, that a FTP client can request. By default, the maximum timeout period an FTP client can request is 2 hours. If that is too long, use this option to shorten it.

<i>ftpd Option</i>	<i>Description</i>
<code>-u permission</code>	Sets the umask value, on some flavors of Unix, for files uploaded to a server using an FTP service. The default <i>permission</i> is 022: group- and world-readable. Clients can request a different mask.

The most common option is the logging option, `-l`. Having this option on is nice because it enables you to monitor FTP connections and check for foul play. This option logs to syslog every failed and successful FTP session. If you specify the option twice, the get, put, append, delete, make directory, remove directory, and rename operations and their file-name arguments are also logged. If you specify this option three times, log entries for get and put include the number of bytes transferred.

The `-d` option logs all the commands and arguments sent by the client, except for arguments from the PASS command.

After you set the configuration file, the system must reread the file before it can take effect. The best way to do that is to use the kill command to pass a hang-up signal (SIGHUP):

```
% killall -HUP pid
```

in which *pid* is the process ID of inetd. But how do you get the pid? If you are using Unix System V, you use

```
% ps -ef | grep inetd
```

The equivalent for the BSD distribution of Unix is

```
% ps ax | grep inetd
```

The following example shows you the output of the ps command:

```
furnace 5% ps -ef | grep inetd
      root   246      1  0   Apr 25 ?        0:45
➔ /usr/etc/inetd
      geckel 6099  5796  2 10:19:19 pts/11   0:00
➔ grep inetd
```

The second field in each line is the process ID.

Setting Up the Anonymous User

Now that you've configured the service, you must create a user that has the appropriate set of permissions and access to the file system. You set up the anonymous user in the server's `/etc/passwd` file. Before doing anything else, ftpd checks this file for whether a user exists. If you fail to create the anonymous user, ftpd prevents a user who logs in as anonymous from accessing any FTP archives.

You do not want to match any user or group IDs for the anonymous user to any system-level IDs. Hackers can use those IDs to access the entire file system through telnet or rlogin. You can prevent such a security breach by putting an asterisk (*) in the password field for all user IDs except anonymous. The asterisk prevents any entry from working in the password field. If there are utilities on your system that do not require a password to log in, you should set the login shell to `/bin/false` so that the asterisk safeguard is not overridden.



Many man pages might suggest that ftpd ignores the login shell. Don't believe it; many versions of ftpd use it.

You might, for example, construct the anonymous user in the following way:

```
ftp:*:500:25:anonymous FTP user:/usr/ftp:/bin
➡/false
```

In this example, `/usr/ftp` is the user login directory, which is what you use as the root directory for anonymous FTP clients. If the login directory is the client's root, that user does not have access to anything higher in the file system than `/usr/ftp`; he or she has access only to directories and files lower in the file system. These provisions give your system the level of security it needs by restricting the users' access to file system.

The root of the anonymous user file system does not have to be under `/usr`. If you are offering a lot of data in your FTP service, you might allocate a separate disk, file system, or both for anonymous FTP use.

You restrict file access by using the change root command, `chroot()`. If `ftpd` calls this function and uses the anonymous user login directory as its argument, all absolute references begin at the login directory. For example, if a user were to change directory to root (`cd /`), he or she would only get to `/usr/ftp`. Or, for example, changing to `/bin` really moves the user to `/usr/ftp/bin`.

`ftpd` calls `chroot()` only if the FTP user is anonymous. Therefore, users who use user IDs to log in are not restricted to the file system specified by their login directory. So, not allowing user ID access to FTP archives confines FTP users to the restricted file system, thereby protecting files outside that file system.

Reorganizing Your Files

One ramification of using `chroot()` to restrict users' access to your system is that you need to move all

the files that `ftpd` needs into the restricted login directory. To do that, you need to set up a new directory structure that places several standard directories under the login directory. The following table shows the directories you need to copy.

Directory	Description
bin	Contains the program <code>ls</code> , which supports the list commands.
dev	Use <code>mknod</code> to copy <code>/dev/zero</code> into <code>~ftp/dev</code> with the same major and minor device numbers. <code>Rld</code> uses this file.
etc	Contains <code>passwd</code> and <code>group</code> . These files enable the <code>ls</code> command to display owner names instead of numbers. The password field in <code>passwd</code> is unused; it should not contain real passwords.
lib	Copy the files <code>/lib/rld</code> and <code>/lib/libc.so.1</code> into <code>lib</code> . <code>ls</code> needs them in order to run.
pub	Contains the archive files available to FTP clients.

The important executable you want to copy to `/usr/ftp/bin` is `ls`.



Do not copy the `/etc/passwd` or `/etc/group` files into `/usr/ftp/etc`, because it would allow anonymous FTP users to copy the files with encrypted passwords to their machines for their own devilish purposes. With this information, third parties might be able to crack the encryption. You should put a minimal amount of information into the `passwd` and `group` files. For example, if the FTP archive contains files created by `ftp` and `bin`, your password file should look similar to the following:

```
ftp:*:500:1:anonymous user:/users/ftp:/bin/false
bin:*:2:2:/:/bin/false
```

If FTP archives are given group access to *user* and *other*, your group file might look like the following:

```
user:*:1:
other:*:3:
```

Login Directory Permissions

So you don't want hackers to upload computer vermin into your service? Using the set of permissions listed in table 10.1 for the subdirectories under the login directory can help.

Table 10.1 Permissions for Subdirectories

Directory	Owner	Permission	Comments
/	root	555	Make the home directory unwritable.
bin	root	555	Make this directory owned by the super-user and unwritable by anyone.
dev	root	555	Make this directory owned by the super-user and unwritable.
etc	root	555	Make this directory owned by the super-user and unwritable.
lib	root	555	Make this directory owned by the super-user and unwritable.
pub	root FTP administrator	555	Users can read and execute the archive files.
upload	root other	777 733	Users can upload files, delete and replace existing files.

Directory	Owner	Permission	Comments
dist	root FTP administrator	555	Users can read and execute the special archive files.

All directories can be accessed by group *other*.

All users must be able to read and execute (that is, search through) files in the login directory, or you don't have much of a service.

You make the `/bin` directory executable, because it contains only the `ls` binary, so users can use it, but unwritable so that nefarious users cannot replace your `ls` binary with one of their Trojan Horses that suddenly brings your system to its knees.

The `/etc` directory contains valuable information about user permissions and group membership. The last thing you want is users resetting their system permissions and group memberships. Consequently, you need to make the directory unwritable.

The `/pub` directory contains your FTP archive files. These files must be readable and executable so that users can search through and download interesting files. On the other hand, you do not want to give users write permission to the directory, because then users could replace or delete archive files that you strenuously maintain. Users cannot upload files in this directory.

Creating the `/upload` directory is optional. It gives users a place to upload files to the server running the FTP service. Separating these files into their own directory is a form of *damage control*. If someone wheels in a Trojan Horse, hopefully its damage is confined to the `/upload` directory. If you see no

reason to permit uploads, don't include the `/upload` directory.

The permission set for the `/upload` directory permits users to write and execute in the directory, but not read from it. This permission set is somewhat inconvenient because it prevents users from looking at files others upload to the server. On the other hand, it prevents users from perverting or deleting those same files.



A byproduct of allowing users to read one another's uploaded files is that they can post files on your server completely unrelated to your service. These uploaded files can be bland or blasphemous. For the sake of your company's image, you don't want to inadvertently support the greatest porn or WaReZ (pirated software) archive in the country.

If you feel that protecting `/upload` files from in-house users of your system (not FTP users) who have access to this file is necessary, you can set the sticky bit on the permissions set by changing the permission to 1733.

The `/dist` directory also is optional; it's really nothing more than a regular FTP archive that you might find in `/pub` except that the files in `/dist` are documents or binaries that your company supports. They should not be programs that your company sells. They might be patches to programs, utility add-ons, driver updates, or tools that give added value to your products. Because these binaries are not

Table 10.2 Permissions for Programs and Files

File	Owner	Permissions	Comments
/bin/l ^s	root	111	Users can execute ls to search the FTP archive files.
/etc/passwd	root	444	Users can only read file ownership.
/etc/group	root	444	Users can only read about group membership.

generic programs, you need not fear people copying, renaming, and publishing these programs under their names.

Again, you run into the same danger of someone uploading something into /dist somehow. You can combat such raiders by making the media that contains this directory read-only (by putting the files on a CD-ROM drive, for example). You might put as many of your files as you can on such media if the files and directories are not likely to change much.

Inside the /bin and /etc directories, you also must use chmod to set the permissions for the standard programs and files, as shown in table 10.2.

Naming the System Administrator

Everyone likes a helping hand. Part of the job of the Internet system administrator is to provide that service to the Internet community. You should set

up two mailboxes for yourself, one that handles your normal e-mail, and another that handles Internet system administration requests. To make it easier for users to reach you, you should use a common alias rather than your e-mail address; for example, you might use ftp-system-admin.

You can point the ftp-system-admin alias at your user ID. This directs all FTP system administration concerns directly to your mailbox. Your aliases file is probably under /etc . To set up the alias, use the following syntax:

```
ftp-system-admin real-user-ID@domain
```

Then again, you may choose not to make the ftp-system-admin user ID an alias for your ID because you might like to keep separate your Internet FTP mail and your other mail.

If you set up an alias, it doesn't take effect until the system rereads the dbm-format database, which you can do by executing new aliases.

Trying Out Your FTP Archive

Before you announce to the world that you have a great FTP service, you really should test it. If you can, let as many people as possible bang on the service to see if you can catch any bugs, inconsistencies, or anything that would frustrate real users to the point of not using the service. Also, you should try to break into restricted areas of the file system and compromise files that should not allow such nonsense. Write the hackers guide to your service—then fix it before anyone else does.

If you have problems, you can look at the entries in the syslog file—assuming you turn on logging (–l) for `ftpd` in the `inetd.config` file. If you did not start `ftpd` with logging enabled, change the `inetd.config` file by adding the `–l` option at the end of the FTP configuration line, then make `inetd` reread the configuration file by executing the following command, where *pid* is the process ID of the `inetd` service:

```
% killall -HUP pid
```

The `/etc/syslog.conf` file should specify the location of the log files, usually `/var/log`. Keep a window open with the syslog file in it. To watch the syslog file, use the `tail` command (`tail –f`) while you use an FTP client to interact with your FTP service. Change directories, list the files in the directories, and download a several files to make sure all is working as planned.

Do a long listing of all the files and directories to make sure the permissions, owners, and groups are correct. Use the tables earlier in this chapter for guidance.

What Your Session Should Look Like

If everything is working okay, when you use your FTP client to `ftp` to your service, you should get a set of responses similar to the following (user input is in italics):

```
% ftp ftp.company.com
```

```
Connected to ftp.company.com
220 sizzle.asd.sgi.com FTP server (Version
➔4.2.321.2 Wed Feb 15 03:24:32 GMT 1995) ready.
Name (ftp.company.com:clinton): anonymous
331 Guest login ok. Use email address as password
Password: clinton@washington.com
230 Guest login ok, access restrictions apply.
ftp> cd /hot_topics
250 CWD successful
ftp> binary --- specify binary to download
➔binary or non-ASCIIfiles
200 Type set to I.
ftp> get one_great_program.tar.Z
200 PORT command successful.
150 Opening BINARY mode data connection for
➔one_great_program.tar.Z
226 Transfer complete.
ftp> quit
221 Goodbye.
```

```
%
```

To actually use the program, you would do the following:

```
% uncompress one_great_program.tar.Z
% tar xvf one_great_program.tar
```

The beginning of the syslog file should look similar to the following:

```
Feb 15 03:24:32 sizzle inetd[121]: ftp/tcp:
↳Connection from sizzle (12.3.2.4) at Wed Feb 15
↳03:24:32 1995
```

```
Feb 15 03:24:32 sizzle ftpd[121]: Connection from
↳sizzle at Wed Feb 15 03:24:32 1995
```

What Might Go Wrong

If the path to the service in the `inetd.config` file is wrong, you might get a message in `syslog` similar to the following:

```
Feb 15 03:24:32 sizzle inetd[121]: ftp/tcp:
↳Unknown service
```

If a directory should be executable but isn't, you get a message similar to the following:

```
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
. not found
226 Transfer complete.
ftp>
```

If `ftpd` is not on the host machine, you might get an error similar to the following:

```
ftp: connect: Connection refused
```

If you try to log in as anonymous, but have not set up the anonymous user correctly, you get a message similar to the following:

```
520 Guest login not permitted.
Login failed.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

What files? It did not really transfer any files. You're not even logged in, which you can confirm by trying to `ls`.

When you try to download an archive file, it should be readable. If not, you get an error message similar to the following:

```
ftp> get download_file /tmp/mine
200 PORT command successful.
550 download_file: Permission denied.
ftp>
```

If you made a mistake with the permission set for a directory, and a directory that should be readable is not, you get a message similar to the following:

```
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
. unreadable
total 2
226 Transfer complete.
ftp>
```

If you cannot upload a file, make sure the `/upload` directory is writable.

If you cannot change to one of the directories under the login directory, check to see that they are executable.

This is hardly a complete list of what can go wrong. But it at least gives you a flavor of problems you might see and what you need to do about them. After banging away on your service for a good while, open it to the people in your company. If they cannot break the service, pop the champagne corks and open the service to the eyes of the Internet world.

Summary

In this chapter you learned how to set up the configuration file for `inetd`, `inetd.config`. You added, populated, and set appropriate permissions for all of the directories under the login directory. You also set appropriate permissions for some of the files in those directories. All along you've tried not to create a hacker's paradise by being lax about file access and directory and file permissions.

You created a user, named `anonymous`, to give FTP clients access to your service, and, at the same time, to `chroot()` them to a restricted set of directories in your file structure. You also learned that you could set up additional users and passwords to give users access to additional directories under the login directory.

After you got everything set up, you had the chance to bang on your service long and hard to make sure that it was bullet-proof. You watched the `syslog` file using `tail` to debug errors that you found. And then you gave the Internet community a valuable new resource by giving your blessing to your company's new FTP service.

Now that your service is up and running, you might wonder if any upkeep is required. Are you kidding? Whenever did files and directories take care of themselves? Chapter 12 discusses how to properly manage your service.

Setting Up a WWW Service

All the command-line interfaces covered so far in this book are powerful and fast, but have a learning curve and are text based. You can make text look only so good. WWW services present a multimedia interface. If you work at all in a windowing environment, the GUI front end flattens out the curve for learning how to use a WWW browser. The color, sound, images, and video clips provide the flash and excitement that makes "surfing the Internet" so easy and fun.

The real tool that WWW services use to make interfacing with the Internet so easy is *hyperlinks*. To look at a different document, just click on a highlighted word, button, or image. Rather than read the README files or look through reams of files that only might match your search criteria, WWW clients enable you to navigate the Internet with point-and-click ease (see fig 18.1).

I present the WWW service last because it's the newest of the services, and because WWW clients can use many of the services previously presented.

For example, a WWW client can access Gopher, FTP, and (for some clients) WAIS services, in addition to WWW services.

Getting the Client Software

The various WWW browsers all are free, and some will soon be reborn as commercial products. Table 18.1 describes the browsers and tells you where you can get them.

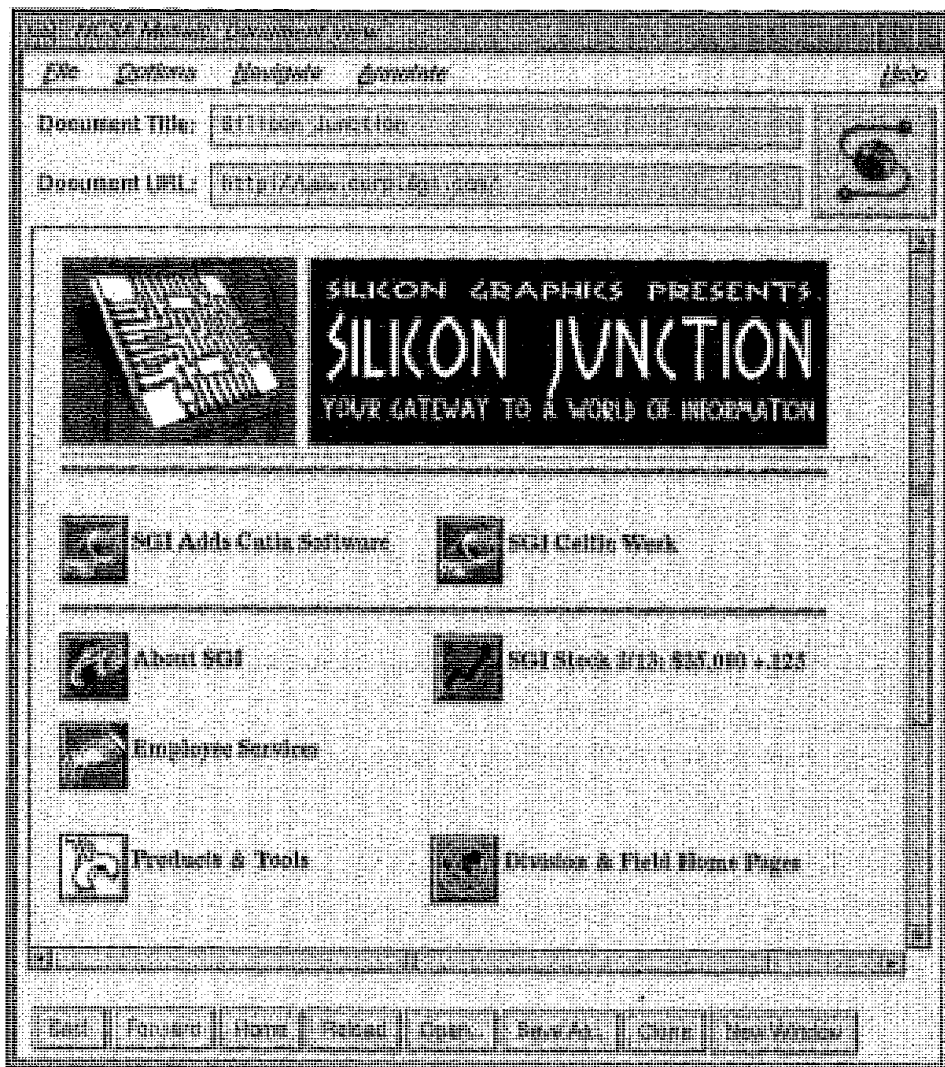


Figure 18.1

A World Wide Web page.

If you haven't already used a WWW browser, no description can substitute for getting your hands on and playing with a browser. Try it out first on the following URL:

<http://www.paramount.com>

Paramount's home page includes most of the tools you are likely to see in Web pages, including hypertext, buttons, and movie clips.

Table 18.1 WWW Browsers

Name	URL and Description
Lynx	ftp://ftp2.cc.ukans.edu/pub/lynx/ A full-featured WWW browser developed at the University of Kansas that best displays character-based pages. The Lynx client compiles without difficulty on Linux.
MidasWWW	http://www-midas.slac.stanford.edu/midasv22/introduction.html A full-featured WWW browser developed at Stanford Linear Accelerator Center (SLAC) for X Window systems. Similar to Mosaic, but provides better support for inline graphics, including .fig, .jpeg, and .tiff files without having to spawn an external viewer.
Mosaic	ftp://ftp.ncsa.uiuc.edu/Mosaic/ The most celebrated WWW browser, developed at the National Center for Supercomputer Applications (NCSA) at the University of Illinois, with versions for most Unix platforms, Apple Macintosh, and Microsoft Windows. MPEG movies, JPED images, and sound files automatically spawn external viewers, because Mosaic can only handle GIF or XBM image formats.
Netscape	http://home.netscape.com A full-featured, graphical browser created by Netscape Communications Corporation.
Emacs	ftp://moose.cs.indiana.edu/pub/elisp/w3/ Compatible with Mosaic. Especially good for Mosaic users who also must dial in to the Internet, perhaps from home, on a slower line.
NeXT	ftp://info.cern.ch/pub/www/bin/next/ Available for NeXT computers only.
perlWWW	ftp://archive.cis.ohio-state.edu/pub/w3browser A character-based WWW browser, written in PERL, developed at Ohio State University.

Getting Service Software

Three major WWW services have been written for the Unix platform. This book discusses installing the NCSA version of the Web service. Table 18.2, however, shows you where you can get the binaries for the other services, too.

This book discusses the NCSA Web service for the following reasons:

- It is small and fast.
- It can serve many documents simultaneously because the service draws so little on system resources.
- It can run as a standalone daemon, or under inetd.
- The service understands both the HTTP/1.0 and HTTP/0.9 protocols.
- You can limit user actions and connections. You can prevent users from specific hosts from connecting to your service. You can also require user authentication to gain access to your

service, if the WWW client supports user authentication.

- If a browser supports user authentication, you can authenticate browsers with a user name and a password.
- It enables you to create HTML catalogs of your directories. Catalogs describe what is in the directories.
- You can write your own service scripts that generate documents on the fly.
- You can move document files to different directories and different servers without having to rewrite the HTML, and without having to advise users of the move.
- You can write scripts to handle form requests. Forms are documents that have one or more fill-in text fields that are often used for entering search criteria.

The NCSA Web service has a couple of features not found in the CERN service. Both services, however, are excellent.

Table 18.2 Binaries for Other WWW Services

Service	URL and Description
CERN	<code>ftp://info.cern.ch/pub/www/bin</code> From the birthplace of the WWW, comes the public-domain CERN Web service, supporting clickable images, forms, and access authorization.
NCSA	<code>http://boohoo.ncsa.uiuc.edu</code> or <code>ftp.ncsa.edu</code> Like the CERN service, the NCSA service is in the public-domain and written in C. This small, fast service supports clickable images, searches, HTML forms, and access restrictions.
Plexus	<code>ftp://austin.bsd1.com/plexus/2.2.1/dist/Plexus.html</code> This service is written in PERL, so is easy to modify, but heavyweight.

Setting Up the NCSA WWW Service

To get the service, you can either use a WWW client and point your URL at `http://boohoo.ncsa.uiuc.edu`, or use FTP to connect to `ftp.ncsa.uiuc.edu`.

After logging in as an anonymous user, change to the `httpd_1.4` directory. The directory path is `/Web/httpd/Unix/ncsa_httpd/httpd_1.4`.



This book discusses the 1.4 version of the service. If a later version of the service exists when you connect to NCSA, you might choose to install it instead. If you do, make sure you check the README file and the installation documentation.

When you list the contents of the service directory, you find that there are two forms of the binary that you can download:

- Compiled
- Uncompiled

If your machine is on the following list, you're lucky: you can download a compiled version of the service. If not, you must download `httpd_1.4_source.tar.z` and compile it after uncompressing and dearchiving it. NCSA offers the following compiled versions of their WWW service:

- `httpd_1.4.2_aix3.2.5.Z`
- `httpd_1.4.2_aix3.2.5.tar.Z`
- `httpd_1.4.2_hpux9.0.5.Z`
- `httpd_1.4.2_hpux9.0.5.tar.Z`
- `httpd_1.4.2_irix5.2.Z`
- `httpd_1.4.2_irix5.2.tar.Z`

- `httpd_1.4.2_linux.Z`
- `httpd_1.4.2_linux.tar.Z`
- `httpd_1.4.2_osf3.0.Z`
- `httpd_1.4.2_osf3.0.tar.Z`
- `httpd_1.4.2_solaris2.4.Z`
- `httpd_1.4.2_solaris2.4.tar.Z`
- `httpd_1.4.2_sunos4.1.3.Z`
- `httpd_1.4.2_sunos4.1.3.tar.Z`

All binaries come compressed, so you must uncompress and untar them—but first make sure you move yours to the directory in which you want it to reside, for example, `/usr/bin/X11`. The files that contain the word `tar` are the entire distribution, such as all the configuration and support files. The files that do not contain the word `tar` contain only the binary for the HTTPD server. Then use the following command:

```
% zcat httpd_1.4.2_operatingSystem.tar.Z | tar xvf-
```

in which *operatingSystem* is one of the operating systems listed above, depending on your operating system, or `source`, if your operating system is not listed in the `http_1.4.2` directory.

Executing the preceding command creates an `httpd_1.4.2` directory, which contains a README file and the following subdirectories.

Subdirectory	Contains
cgi-bin	(Common gateway interface) Sample gateway binaries and scripts; where you put custom gateway binaries of your own

continues

<i>Subdirectory</i>	<i>Contains</i>
Makefile	Rules and dependencies that build the service
conf	Not surprisingly, service configuration files
icons	Icons for directory indexing
src	All of the .c executables
support	Applications that the passwords users need to access the service

Installation notes are included in a separate, compressed file, `Install.txt.z`. You must download it separately.

```
ftp> get Install.txt.z
```

To uncompress it, use `zcat`.

```
% zcat Install.txt.z | more
```

The `cgi-bin` directory contains the following directories and files:

```
warp 188% cd cgi-bin
warp 189% ls
archie      date      fortune    nph-test-cgi
test-cgi.tcl uptime
calendar    finger    mail       test-cgi
test-env     wais.pl
```

These programs allow your service to connect to other Internet services.

The `conf` directory contains the following directories and files:

```
warp 243% cd support
warp 244% ls
Makefile      change-passwd.readme
```

```
➔inc2shtml.c
auth          htpasswd.c
➔unescape.c
```

You use these programs to establish the passwords that users need to access the system.

Most likely, you will never have to work at all with the other directories, `cgi-src`, `src`, and `icons`.

Recompiling the Linux WWW Service Binary

If you have decided to reconfigure the default file locations for configuration files, then you must edit the associated files and rebuild the HTTPD service. The process of rebuilding the binary is simplified, however, because you already have the required components, and the `Makefile` is customized to Linux.

The HTTPD service binary is configured to use a directory under `/usr/local/etc/httpd`. The `httpd` binary is installed as `httpd` in this directory, along with the `cgi-bin`, `support`, `logs`, `htdocs`, `conf`, and `icons` directories.

If you wish to reconfigure where the files are located in the file system, then you must edit the file `src/httpd.h`. The following is the relevant section that must be changed:

```
/* ..... config dir ..... */

/* Define this to be the default server home dir.
➔Anything later in this file with a relative
➔pathname will have this added.
*/
#define HTTPD_ROOT "/usr/local/etc/httpd"
```

```

/* Root of server */
#define DOCUMENT_LOCATION "/usr/local/etc/httpd/
↳htdocs"

```

Let's assume you want to put your HTTPD service files in /home/httpd. You would change the preceding lines to read:

```

/* ..... config dir ..... */

/* Define this to be the default server home dir.
↳Anything later in this * file with a relative
↳pathname will have this added.
*/
#define HTTPD_ROOT "/home/httpd"

```

```

/* Root of server */
#define DOCUMENT_LOCATION "/home/httpd/htdocs"

```

Save the files and run a make to build the new binary. The make is started by running the following command:

```
make linux
```

Once the build is done, copy the new binary and support files into your new directory structure. You now are ready to configure your server.

Configuring the WWW Service

This section describes the minimal number of steps you must perform to configure your WWW service to your platform. The process involves editing three configuration files in the conf directory:

- httpd.conf-dist
- srm.conf-dist
- access.conf-dist

You should leave these configuration files as they are and make copies, perhaps saving without the -dist extension.



In all conf files, extra whitespace is ignored and only one directive can go on each line.

Whereas all of the basic configuration information is presented in the next several sections, advanced configuration administration is covered in the next chapter.

Setting Up httpd.conf-dist

The httpd.conf configuration file controls how the service, httpd, runs. Make a copy of httpd.conf-dist, call it httpd.conf, and use it to configure the service. Keep httpd.conf-dist as a reference copy.

Before you can edit the configuration file, you must decide whether you want the service to run under inetd or as a standalone daemon, and whether you want to restrict access to the service. In this section, you set the service to run as a standalone daemon that everyone can access. A later section discusses more involved configuration setups.

To set up httpd.conf, set the values of the directives in the following table.

<i>Directive</i>	<i>Description</i>
<i>AccessConfig fileName</i>	<i>fileName</i> is either an absolute pathname or a partial pathname relative to <i>ServerRoot</i> that specifies the location of the <i>access.conf</i> configuration file. The default is <i>AccessConfig conf/access.conf</i> .
<i>AgentLog file</i>	<i>file</i> is the file where you want to keep record of the client agent software. This directive is for statistical purposes and the tracing of protocol violations. You can only use this directive in standalone mode. The default is <i>AgentLog logs/agent_log</i> .
<i>ErrorLog fileName</i>	<i>fileName</i> is either an absolute pathname or a partial pathname relative to <i>ServerRoot</i> that specifies the location of the error log file. <i>httpd</i> includes information such as segmentation violations, bus errors, bad scripts, timed out clients, <i>.htaccess</i> files that attempt to defeat <i>access.conf</i> directives. The default is <i>ErrorLog logs/error_log</i> .
<i>Group [groupName groupNumber]</i>	Specifies the group ID the copies of the service run as when answering client requests. This directive is only pertinent if the <i>ServerType</i> is set to standalone. The group ID can either be a name or user number. If you use a number, you must precede it with a pound sign (#). The default is <i>Group #-1</i> .
<i>IdentityCheck [on off]</i>	Determines if the remote user is logged in as himself. This directive only works if the client application is running an RFC 931-compliant Identity daemon, such as <i>identd</i> . The default is <i>IdentityCheck off</i> .
<i>MaxServers number</i>	<i>number</i> is the maximum number of children for the <i>hunt</i> group.
<i>PidFile fileName</i>	<i>fileName</i> is a partial pathname relative to <i>ServerRoot</i> where you want <i>httpd</i> to record the process ID of each running copy of <i>httpd</i> . You can only use this directive in standalone mode. The default is <i>PidFile logs/httpd.pid</i> .
<i>Port portNumber</i>	<i>portNumber</i> specifies the port number <i>httpd</i> listens to. Ports below 1024 are reserved by the system. Port numbers cannot be greater than 65536. Unless you do

not want to make the service available to the Internet community, set the port number to 80. The default is Port 80.

ResourceConfig *fileName*

fileName is either an absolute pathname or a partial pathname relative to ServerRoot that specifies the location of the srm.conf configuration file. The default is ResourceConfig conf/srm.conf.

ServerAdmin *emailAddress*

emailAddress specifies the Internet System Administrators address. It is a good idea to use an alias, for example,

ServerAdmin sysAdmin

so that you can change system administrators without changing the e-mail address. This directive does not have a default.

ServerName *hostName*

hostName specifies the the domain name of your server or a DNS alias, for example

ServerName www.companyName.com

ServerRoot *pathName*

pathName defines the absolute path of the root of your service above which users cannot trespass. The default is ServerRoot /usr.local.etc.httpd.

ServerType [inetd | standalone]

Specifies whether httpd is running under inetd or in standalone mode. The default is ServerType standalone.

StartServers *number*

number is the number of service processes you want to run concurrently.

TimeOut *seconds*

seconds defines the maximum amount of time (in seconds) the service waits for the client to submit a request once it has been connected, and the maximum amount of time the service should wait for the client to accept a request. The default is TimeOut 1800.

TransferLog *fileName*

fileName is either an absolute pathname or a partial pathname relative to ServerRoot that specifies the location of the log that records data, such as host, date, and file name, of service requests. These entries are described in greater detail following this table. The default is TransferLog logs/access_log.

continues

<i>Directive</i>	<i>Description</i>
TypesConfig <i>fileName</i>	<i>fileName</i> is either an absolute pathname or a partial pathname relative to ServerRoot that specifies the location of the MIME configuration file. The default is TypesConfig conf/mime.types.
User [<i>userName</i> <i>userNumber</i>]	<i>userName</i> or <i>userNumber</i> defines the user ID the copies of the service run as when answering client requests. This directive is only pertinent if the ServerType is set to standalone. The user ID can either be a name or user number. If you use a number, you must precede it by a pound sign (#). By default, the copies run as user number one:

User #-1

Normally, you should run the service in standalone mode (ServerType standalone). If you run it under inetd, each time a service request comes in, inetd must fork off a new process, load the httpd binary, and load and parse all three configuration files. When running in standalone mode, httpd_1.4 just copies itself and the copy handles the new request. As you can see, system performance is improved using the standalone mode. The only exception to this rule is if the service is not often used. Then

you have the CPU serving the httpd process even though there are not requests for it to do anything.

Entries in the access_log file, or the file specified by the TransferLog directive, have the following form:

```
host rfc931 authuser [DD/Mon/YYYY:hh:mm:ss]
➔ "request"
   ddd bbbb
```

The elements of the entry are explained in the following table.

<i>Element</i>	<i>Description</i>
host	Either the IP address or the DNS name of the client making the request
rfc931	Any information returned by identd for the client
authuser	The user name that was sent by the client for the purposes of authentication
DD/Mon/YYYY	The day, month, and year the request was received
hh:mm:ss	The hours minutes and seconds of when the request was made
request	First line of the request
ddd	Status code returned by the server
bbbb	The total number of bytes sent by the server

Setting Up srm.conf

The srm.conf-dist (server resource management) configuration file specifies the location in which the service finds your scripts and documents. Make a copy of srm.conf-dist, call it srm.conf, and use it to configure the service. Keep srm.conf-dist as a reference copy.

To set up the configuration file, set the values for the directives listed in the following table.

You set the UserDir directive to the partial path, relative to the user's home directory (as given in /etc/passwd), that leads httpd to personal files. The

service adds this partial path to the requested path name to find the user's document. For example, if this directive were set to its default

```
UserDir public_html
```

httpd would translate a request for /~images/first.gif to ~images/public_html/first.gif. You then define the path, public_html.

Because this directive can lead to security problems, you have the option of turning off this directive by using the keyword DISABLED.

```
UserDir DISABLED
```

<i>Directive</i>	<i>Description</i>
AccessFileName <i>fileName</i>	<i>fileName</i> specifies the name of the file that you can include in any directory that specifies access permissions for that directory. The default is AccessFileName .htaccess.
AddDescription <i>text fileID</i>	Associates descriptive <i>text</i> with a type of file defined by extensions, a file name, an absolute path name, or a file name using wild cards (for example AddDescription "image file" *.gif).
AddEncoding <i>kind ext</i>	Specifies that files with <i>ext</i> are of type <i>kind</i> so that appropriate actions can be taken. For example, if the file is compressed, the browser can automatically uncompress it: AddEncoding compress Z.
AddIcon <i>path name1 name2 ...</i>	Specifies the icon to display with a kind of file; used when browsers display FTP menus.
AddIconbyEncoding <i>path name1 name2 ...</i>	Performs the same task as Icon except that the encoded information determines the icon used.
AddIconType <i>path type1 type2 ...</i>	Performs the same task as Icon except that the MIME type determines the icon used.
AddType <i>kind ext</i>	Supersedes MIME definitions for the specified extensions (<i>ext</i>) found in the mime.types file.

continues

<i>Directive</i>	<i>Description</i>
Alias <i>name path</i>	Substitutes <i>path</i> for <i>name</i> in path names. For example, if Alias books /usr/resources then books/apples is equivalent to /usr/resources/apples.
DefaultType <i>type</i>	Specifies the default MIME type. The default is DefaultType text/html.
DefaultIcon <i>pathName</i>	<i>pathName</i> specifies the default icon to use when FancyIndexing is on. The default is DefaultIcon /icons/unknown.xbm.
DirectoryIndex <i>fileName</i>	Specifies the <i>fileName</i> to return when the URL request is just your service, for example, http://www.sgi.com. The default is DirectoryIndex index.html.
DocumentRoot <i>path</i>	<i>path</i> specifies the absolute path to the directory from which httpd retrieves documents. The default is DocumentRoot /usr/local/etc/httpd/htdocs. If you have to serve documents outside of this directory, you can provide symbolic links or an alias from this directory.
FancyIndexing [on off]	Adds icons, file name data, headers, and footers to lists of files automatically indexed; necessary only for backward compatibility with HTTP V1.0. The default is FancyIndexing on.
HeaderName <i>fileName</i>	Specifies the file name to be used at the top of a list of files automatically indexed. The default is HeaderName HEADER.
IndexIgnore <i>kind1 kind2 ...</i>	Specifies kinds of files to be ignored during file processing. The default is IndexIgnore */.*? *~ *# */HEADER* */README.
IndexOptions <i>option1 option2 ...</i>	Specifies a variety of indexing parameters, including FancyIndexing, IconsAreLinks, ScanHTMLTitles, SuppressLastModified, SuppressSize, and SuppressDescription.
OldScriptAlias <i>name path</i>	Performs the same task as Alias; but provides for backward compatibility to HTTP V1.0.
ReadmeName <i>fileName</i>	<i>fileName</i> specifies the footer information to attach to automatic directory indexes. The default is ReadmeName README.
Redirect <i>pathname URL</i>	Remaps <i>pathname</i> of document to new <i>URL</i> . There is no default for this directive.

ScriptAlias *name path*

Is similar to Alias, but used for scripts. This directive substitutes *path* for *name* in path names, for example

```
Alias collection /usr/cgi-bin/
```

so that `collection/scripts` is equivalent to `/usr/cgi-bin/scripts`. If you move `cgi-bin` from its default location, `/usr/local/etc/httpd/cgi-bin`, you might provide an alias path to it with ScriptAlias.

UserDir [path | DISABLED]

Specifies the directory users can make available for httpd access. The default is `UserDir public_html`.

Setting Up access.conf-dist

The `access.conf-dist` configuration file defines what service features are available to all WWW browsers. The default is to make everything available to all browsers. Make a copy of `access.conf-dist`, call it `access.conf`, and use it to configure the service. Keep `access.conf-dist` as a reference copy.



The next chapter describes security measures you can take to protect your WWW service, so I put off an extensive discussion about security until then.

Many of the configuration directives in the `access.conf-dist` file are sectioning directives. They stand out because they use angle brackets.

Sectioning directives have a beginning and ending delimiter, for example,

```
<Directory>
...
(/Directory>
```

Any directives between the delimiters apply to the listing following the first delimiter. For example,

```
<Limit GET>
order allow, deny
allow from all
</Limit>
```

In this example, the sectioning directive, `Limit`, determines who can retrieve information from the service, which, in this case, is “allow from all.”

To set up the `access.conf` file, perform the following tasks:

- Set the first Directory sectioning directive to the path of your `cgi-bin` directory. The default is

```
<Directory /usr/local/etc/httpd/cgi-bin>
```
- Remove the `Indexes` option from the `Options` directive (in the first Directory sectioning directive) so that users cannot browse through the `httpd` directory. The default is

`Options Indexes FollowSymLinks`

The possible values for the `Options` directive include the following:

- **All.** All features are enabled for the directory.
- **ExecCGI.** `cgi` scripts can be executed in this directory.
- **FollowSymLinks.** `httpd` follows symbolic links.
- **Includes.** Server-side include files are enabled in this directory.

- **IncludesNoExec.** Enables server-side includes, but disables the exec option.
- **Indexes.** httpd allows users to retrieve service-generated indexes of this directory. Precompiled indexes in the directory are always available.
- **None.** No features are enabled for the directory.
- **SymLinksIfOwnerMatch.** httpd only follows symbolic links if the target file or directory is owned by the same user ID as the link.
- Set the second Directory sectioning directive to the path DocumentRoot defines in the srm.conf file. The default is

```
<Directory /usr/local/etc/httpd/htdocs>
```

In this case, allowing the Indexs option in the Option directive (within the second Directory sectioning directive) is fine, because you want people to browse your documents.

- Set the AllowOverride variable to None to prevent others from changing the settings in this file. This directive controls which access control directives can be overridden in a directory by the .htaccess file. The default is

```
AllowOverride All
```

Other values include the following:

- **All.** Access control files are unrestricted in this directory.
- **AuthConfig.** Enables the use of AuthName, AuthType, AuthUserFile, and AuthGroupFile directives.
- **FileInfo.** Enables the use of AddType and AddEncoding directives.
- **Limit.** Enables the use of the Limit sectioning directive.

- **None.** No access control files are allowed in this directory.
- **Options.** Enables the use of the Options directive.

The directives enabled by AuthConfig are defined as the following:

- **AuthName.** Sets the authorization name of the directory, for example,


```
AuthName CompProject
```
- **AuthType.** Sets the authorization type of this directory. Currently, there is only one type: Basic.

- **AuthUserFile.** Specifies the file to use that contains the list of users and passwords used in user authentication, for example,

```
AuthUserFile /usr/local/etc/httpd/conf/.htpasswd
```

- **AuthGroupFile.** Specifies the file that lists user groups for user authentication, for example,

```
AuthGroupFile /usr/local/etc/httpd/conf/.htgroup
```

- Set the Limit sectioning directive to the appropriate values. The directives that can come in Limit sectioning directive include the following:

- **allow *hostName*.** Enables specified hosts from accessing the service.
- **deny *hostName*.** Prevents specified hosts from accessing the service.
- **order *ordering*.** Determines the order in which the allow and deny directives are evaluated. Customary values are "deny,allow" and "allow,deny."

→ **require entity1 entity2 ...** Entity values can be user, group, or valid-user. These are the authenticated users or groups that can access the system. valid-user are users identified by AuthUserFile.

The only method that can currently follow the first Limit directive is GET, which allows clients to retrieve documents and execute scripts.

The default for the Limit sectioning directive is

```
<Limit GET>
order allow, deny
allow from all
</Limit>
```

In the default condition, the order directive defines the order in which allow and deny are evaluated, and everyone is allowed to retrieve documents.

Installing httpd

After you make basic changes to the configuration files, you can move httpd to the correct location in your file system.



Remember that the HTTPD binary assumes it will find things in the default location of /usr/local/etc/httpd. If you put the files somewhere

else, the binary still expects to look in the default directory for its configuration files. This can be altered by changing the HTTPD service configuration files or through a command line option.

Use the mkdir command to create a directory—call it httpd—in a location specified by ServerRoot. Then copy httpd and all the subdirectories, conf, logs, icons, and cgi-bin to the new directory, httpd, using the following command line:

```
% cp -r httpd logs conf icons cgi-bin <pathName>/
➤httpd
```

Use chown to make the logs directory writeable by the user ID under which the service runs.

Starting Your WWW Service

Your service is ready to start! You now need to decide whether you want httpd to run under inetd or as a standalone daemon. If your service gets little use (I hope not!) or you are testing your service, you can minimize the impact of httpd on your system by starting your WWW service under inetd. However, if your service is accessed regularly, you should run httpd as a standalone daemon, because every time inetd starts up and shuts down the service (which it does for each service request) causes a delay.

The following three command-line options, however, are common to both run-time environments:

Option	Description
-d directoryName	Specifies the absolute path to the httpd binary if it's not in the default location; this path matches the path specified by ServerRoot in httpd.conf
-f file	Name Specifies a configuration file to read instead of httpd.conf
-v	Displays the version number of the service

Running httpd Under inetd

inetd starts and stops services according to the requests the server receives: each service request starts an instance of the service, which terminates after sending a reply to the request. Each request specifies a port number; port numbers correspond to services running on the server. The standard port number for a WWW service is 80. You can change the Port variable in httpd.conf if you want the service to run on a different port. You would do that only to run a different version of the service for internal personnel, otherwise, all WWW services should run on port 80.

To make httpd run under inetd, complete the following tasks:

- Add a line to /etc/services similar to the following:

```
http portNumber/tcp
```

http is the name of the service and *portNumber* is normally 80.

- Add a line to /etc/inetd.conf similar to the following:

```
http stream tcp nowait nobody /pathName/httpd  
httpd
```

where *pathName* is the pathname to the httpd binary. You can add any necessary command-line options after httpd.

- Use the following command to restart inetd to make it read the configuration file, inetd.conf:

```
% kill -HUP <pid>
```

pid is the process ID of the service. The WWW service then starts whenever a request comes to port 80.

To stop the service, comment out the line you just added to inetd.conf.

Running httpd as a Daemon

The downside of running httpd as a standalone daemon is that it is always running. Of course, if your service is well liked, your service is going to run all the time anyway. The service runs faster in standalone mode because it does not need to repeatedly start, stop, and read the configuration files.

To start the service daemon using the defaults, type the following:

```
# httpd &
```

The ampersand (&) makes the service run in the background. You start the service as root so that the service can bind to port 80. After reading the configuration files, it changes the user ID and group ID to that specified in the httpd.conf configuration file.

Remember that, for example, if you have moved httpd from its default location, to start the executable with options.

You probably do not want to do this every time you restart the server. To automatically start the service during server startup, edit the /etc/rc0 file, for example:

```
if [ -x /usr/bin/X11/httpd ]
```

```
then
```

```
/usr/bin/X11/httpd
```

```
fi
```

When you need to restart the service, find the pid of httpd. If httpd has more than one, which it probably does, be sure to kill the parent by finding the

process that has the lowest ID and a parent process ID of 1.

You can rest assured that the service is working well if the httpd server error log (logs/error_log, by default) says the following:

```
httpd: successful restart.
```

To restart httpd, execute the following on the command line:

```
# kill -1 'cat pidfile'
```

in which `pidfile` is the setting of `PidFile` for your service.

Alternatively, you can use the `ps` command to `grep` for httpd's process ID and send it a HUP signal.

To stop the service, execute the following on the command line:

```
# kill 'cat pidfile'
```

Testing httpd

Even if you have not yet populated your resources/directory with data files, you at least can use a WWW browser to access the service adding the following to the URL field:

```
http://serverName
```

If you receive the contents of `ServerRoot/index.html` or a directory index, hooray!—your service works. If you don't, look at the access and error logs for some clues about what is wrong.

Another way to determine whether the service is running is telneting into it, as the following example demonstrates:

```
warp 106% telnet www.corp.sgi.com 80
Trying 192.26.51.29...
Connected to palladium.corp.sgi.com.
Escape character is ^].
HEAD / HTTP/1.0
HTTP/1.0 200 OK
Server: Netsite-Communications/1.0
Date: Saturday, 11-Mar-95 02:25:58 GMT
Content-type: text/html
Connection closed by foreign host.
warp 107%
```

The preceding display shows the name of the server and the version of the service software.

Letting Others Know

All this effort goes for naught unless you let the Internet community know about your service. Chapter 3, "Advertising on the Internet," provides a comprehensive guide for getting out the word. There are, however, some WWW-specific avenues of announcing your service, including sending a message to the following addresses:

- www-request@info.cern.ch
- www-announce@www0.cern.ch
- whats-new@ncsa.uiuc.edu



Make doubly sure that your announcement contains correct information. Correcting incorrect information after you send it out is difficult. Make sure, for example, that you set your URL off by itself. If you make it part of a sentence, users can confuse sentence punctuation with the URL; for example, an ending period.

Summary

It's time to open the gates and let the outside world in on the riches of your WWW service. In this chapter you completed a variety of tasks, including downloading the software, building and installing it, configuring the service, and editing the Makefile. After testing the service and creating a home page, you are at the jump point: if you send the announcements, they will come.

But wait! The security measures discussed in this chapter were very basic. If your service requires more sophisticated security, you need to read the next chapter, "Managing a WWW Service," before opening the flood gates. And, as the title of the next chapter implies, your job as Internet System Administrator is not complete. Although WWW services are easy to maintain, there are important chores that require your attention.